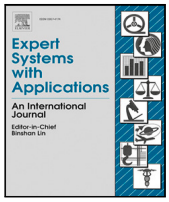
Expert Systems With Applications 239 (2024) 122470

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Revolutionizing subjective assessments: A three-pronged comprehensive approach with NLP and deep learning

Raghav Agrawal [a], Harshit Mishra [a], Ilanthenral Kandasamy [a,*], Shrishail Ravi Terni [b], Vasantha W.B. [a]

[a] School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Katpadi, Vellore, Tamil Nadu, 632014, India
[b] School of Engineering and Applied Sciences(SEAS), The George Washington University, 1918 F Street, NW, Washington, DC, 20052, United States

ARTICLE INFO

Keywords:
Deep Neural Networks (DNN)
Natural Language Processing (NLP)
Question answering
Yet Another Keyword Extractor (YAKE)
KeyBERT
Simple Contrastive Sentence Embedding Framework (simCSE)
Camembert
Sentence Bidirectional Encoder Representations from Transformers (SBERT)

ABSTRACT

The enhanced answer evaluation system is a cutting-edge automated tool that evaluates subjective answers in various contexts, such as educational assessments, surveys, and feedback forms. The proposed system leverages Natural Language Processing (NLP) and deep learning techniques to analyse subjective answers and provide evaluation scores with precision. Students' answers are evaluated based on various criteria, such as keywords, context, relevance, coherence, and similarity. This paper introduces an architecture for a subjective answer evaluator using three main aspects: detection of keywords, similarity matrix, and presence of named entities. It combines the three aspects and provides a final score. It provides a standardized mechanism to score a given user answer compared to the particular model answer without human prejudice.

This research aims to transcend traditional methodologies that predominantly utilize keyword or keyphrase scoring (text-based similarity) to determine the final score of an answer without delving into its technical intricacies. The semantic similarity (vector-based) employs vector data representations for score calculation. This approach necessitates partitioning data into multiple vectors for a comprehensive analysis. While text similarity is effective for short answers, its efficacy diminishes as the length of the answer increases. Therefore, this study emphasizes the critical role of similarity scoring and Named Entity Recognition (NER) scoring in evaluating more extended responses based on the stsb-en-main dataset (short answers) and a custom dataset with 190 records.

This research reveals its remarkable performance, which excels through a dynamic three-pronged approach: keyword scoring, semantic similarity, and NER scoring with models like Yet Another Keyword Extractor (YAKE), SimCSE and Camembert. These three independent components synergize to produce unmatched results, establishing a new standard in the field. This enhancement led to Root Mean Square Error (RMSE) scores of 0.031 (optimized error rate) and an impressive 71%+ accuracy for our comprehensive system. This achievement surpasses existing works, which typically reached accuracies ranging between 40%–60% for long answers.

1. Introduction

Subjective answer evaluation is crucial in assessing a candidate's merit and competence during the examination process. Inaccurate evaluations can negatively impact students and undermine the fairness of the assessment. Traditional manual evaluation methods are prone to workforce errors, biases, and challenges such as expert unavailability and delayed results, which can further hinder the process. Until recently, automated evaluation was limited to multiple-choice questions, highlighting the need for a comprehensive solution to address these issues and deliver an efficient, high-quality, subjective answer evaluation system. Current methodologies, including Latent Semantic Analysis (LSA) (Kherwa & Bansal, 2017), Information Retrieval (IR), and keyword recognition, exhibit limitations that hinder their accuracy and efficiency. LSA struggles to differentiate between essential and

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: raghav.agrawal2019@vitstudent.ac.in (R. Agrawal), harshit.mishra2019@vitstudent.ac.in (H. Mishra), ilanthenral.k@vit.ac.in (I. Kandasamy), shrishail.terni@gwmail.gwu.edu (S.R. Terni), vasantha.wb@vit.ac.in (Vasantha W.B.).

https://doi.org/10.1016/j.eswa.2023.122470
Received 5 August 2023; Received in revised form 31 October 2023; Accepted 4 November 2023
Available online 7 November 2023

superfluous keyword repetitions and neglects the grammatical structure of responses. IR techniques need to be more comprehensive in comprehending the semantic intricacies of answers. At the same time, keyword recognition alone is insufficient for evaluation, as it overlooks the semantic context and the overall framework in which answers are composed. A detailed analysis of the existing and recent research work is discussed in Section 2, and the research gaps identified are given.

1.1. Significant contribution and research objectives

This research paper aims to develop a comprehensive solution for efficient, effective, and high-quality subjective answer evaluation to address the challenges and shortcomings discussed in the previous section and in detail in Section 2. We propose a three-pronged approach for a subjective answer evaluator: keyword detection, similarity matrix, and the presence of named entities. Our proposed system targets the research gaps in this domain by incorporating the following objectives:

- Keyword Module: Leverage Yet Another Keyword Extractor (YAKE) (Campos et al., 2020), and KeyBERT (Grootendorst, 2020) models to enhance keyword extraction capabilities.
- Semantics Module: Utilize SimCSE (Gao, Yao, & Chen, 2021), SBERT (Reimers & Gurevych, 2019), and paraphrase models for in-depth semantic analysis.
- NER module: Implement SpaCy (Shelar, Kaur, Heda, & Agrawal, 2020), Camembert (Martin et al., 2020) and NLTK (Shelar et al., 2020) models for robust NER.
- Final DNN module: Employ deep learning models to predict final scores in the evaluation process accurately.

By achieving these objectives, our research endeavours to bridge the gaps in current methodologies and establish a more reliable and advanced subjective answer evaluation system, as we have performed thorough research by implementing and comparing all the algorithms and presenting an error comparison table in the latter half of the paper. We have developed a hybrid model which considered the mentioned limitations in phases.

This paper is organized in the following manner: Section 1 is introductory, and Section 2 details the literature review of different studies on answer evaluation and highlights the research gaps. The methodology is described in section three, along with the dataset and the proposed model's overall architecture, consisting of three modules, i.e., keyword scoring, similarity scoring, and NER scoring. Section 4 covers the experimental setup, while Section 5 presents the comparisons, results and discussions along with the implications of the results. Finally, the study's conclusions are given in the last section.

2. Literature survey

This section deals with the state of the art of evaluating subjective answers; it offers insights into the existing approaches and their limitation and highlights the need for a more comprehensive solution. There is a broad scope of experimenting with methods of evaluating subjective answers since it is a recent and upcoming area of research. According to a recent study, Johri, Dedhia, Bohra, Chandak, and Adhikari (2021), the researchers presented the cosine similarity between two extracts and keyword scoring as potential evaluation techniques. While the cosine similarity method seems promising, the keyword scoring method requires manual implementation. This means that the teacher needs to provide a set of keywords that will be matched for presence in user answers. The final score is calculated using a hard-coded formula that does not consider keyword dominance and treats all keywords equally. The paper could not provide a fully automated keyword grading system, as the exam setter needs to input all the keywords independently. Finally, there is no validation of the final scoring formula provided, as there was an absence of derivation of weighted constants.

The various similarity analysis domains and techniques have been explored in Mittal and Syamala Devi (2016). The work implements domains of LSA, bilingual evaluation understudy (BLEU), and fuzzy logic. BLEU is added to cover up for the lost semantics of LSA and adds to the model's efficiency. Ontology-based mapping is also proposed with a synonym replacement from WordNet. WordNet, based on the English lexical database, is an NLTK corpus reader. It is utilized to search the words' meaning, synonyms or antonyms. The proposal of synonym replacement is a significant breakthrough, enabling multiple forms of the same answers. Using multiple algorithms for similarity analysis can affect the model's performance as it might act as a bottleneck while evaluating large sets of answers. The fuzzy logic mentioned in Mittal and Syamala Devi (2016) is ambiguous because it does not use a member function; instead, it relies on a set of if and else statements.

The architecture proposed in Hu, Yang, Li, Sun, and Yang (2023) is a question-based answering model named ADRAV (A-Lite-BERT (ALBERT) with Dynamic Routing and Answer Voting). It proposes a 3-stage architecture. The first stage is dynamic routing and has two components: the first component is encoding, and the second is dynamic routing for encoded input. The next stage is answer voting, which defines a pre-output layer and an algorithm to find the final score. Finally, the third stage is pre-fine-tuning, done using the SQuAD dataset[1] over multiple iterations. The architecture avoids using pre-trained models due to the partial knowledge of hidden layers in the existing and relevant models which makes hyper-tuning inefficient.

The two-phased grading architecture is proposed in Sakhapara et al. (2019) using LSA, a cosine similarity matrix, and information gain. The model uses synonym replacement from WordNet. Other techniques used for testing include K-Nearest Neighbour (K-NN), enhanced NLP, and generalized LSA. A Kaggle dataset of 1200 biology questions was used on pre-graded answers and their actual graded scores, with marks ranging from 0–3. There is no mention of keyword analysis. Essay-type answers are prone to lose their scoring due to overall semantic analysis and sentence-by-sentence semantics. Additionally, the architecture needs to include the use of feedback reports.

The proposed work in Jain, Seeja, and Jindal (2019) focuses on identifying textual information based on text mining and NLP. It provides ontology-based similarity measures using a knowledge base. The case also presents many ontology techniques, including single, cross, and fuzzy. The architecture tests its performance on multiple benchmarked datasets, which include Miller and Charles, Rubenstein and Good Enough's, and Word Similarity-353 Test collection.[2]

The study by Alqaryouti, Khwileh, Farouk, Nabhan, and Shaalan (2018) centres on analysing keywords and their frequency in the graphlets pattern proposed for each abstract and its corresponding class. The Naïve Bayes classifier assigns the probability to each word. The probability then describes if a word is a keyword or not. The work includes the use of supervised as well as unsupervised learning approaches. The comparisons in supervised learning were between Support Vector Machine (SVM) and Naïve Bayes. The unsupervised learning approach focuses on Term Frequency–Inverse Document Frequency (TF-IDF). It also mentions other scoring mechanisms for keywords, including random-walk algorithms to score a webpage according to its significance, which is driven by its interlinks, dividing the abstract based on keyword and key-phrases candidates through stop words and phrase delimiters. The performance of a supervised learning model is highly dependent on the domain it was trained on. In the event of a domain change, the model would require retraining to maintain accuracy. The unsupervised modules failed to work with professional domains like health and medicine. One major architectural flaw is the random sought of 10 000 samples out of 205 000 available in the corpus, inducing a population bias.

[1] https://huggingface.co/datasets/squad.
[2] https://gabrilovich.com/resources/data/wordsim353.

In Firoozeh, Nazarenko, Alizon, and Daille (2020), the authors define various keyword extraction methods, assessing and testing methods, and the specificity of methods. The architecture helps analyse different domains and dimensions of keyword analysis and extraction. It also defines a property named 'keyness', which implies the targeting of a token from a text extract. It also proposes using metadata enrichment and document summarization as alternates to the keyword extraction process using standard NLP techniques. Furthermore, it has defined three keyness properties: informational, linguistic, and domain-based. The testing mechanisms for various keyword extraction processes can be brought to use. This includes intrinsic evaluation done a posteriori, human judgments, precision, recall, author-assigned, and reader-assigned keywords. The evaluation metrics use recall and precision scores after simple classification, which might not be the best way to test the model's performance.

The architecture analyses various aspects of YAKE (Campos et al., 2020). The paper delves into the inner workings of YAKE, an unsupervised and lightweight automatic keyword extraction method. YAKE utilizes statistical text features extracted from a single document to identify the most relevant keywords. By leveraging advanced statistical techniques, YAKE can accurately and efficiently extract keywords from text, making it a valuable tool for various NLP applications. The unsupervised nature of YAKE means that it does not require any prior training data, making it a highly flexible and adaptable solution. The methods described for evaluation include corpus analysis, domain, and language analysis, interior stopwords analysis, scaling factor, and term frequency analysis. The approach also lists various unsupervised and supervised mechanisms for keyword extraction. The algorithm is not domain-dependent and cannot be used in specific analyses like medicine or physics. The research work in Bashir, Arshad, Javed, Kryvinska, and Band (2021) directly compares the accuracies by measuring the cosine similarity and the difference using the Word Movers Distance (WMD). The technique involves collecting data using a web application and then pre-processing it using an NLP pipeline. The final result prediction module calculates the answers' score, and it is at this stage comparisons between similarity score calculations are made. It proposes a pre-trained word2vec model from Google consisting of 300 dimensions of around 100 billion vocabulary words for similarity scoring. The corpus is split into a training set and a test set in a ratio of 80%(train):20%(test). The training data is used to calculate initial scores from the scores of the score prediction modules. The system is given testing data one at a time, which then updates the machine learning model. The outcomes are achieved by combining cosine similarity, word mover's distance, and a multinomial Naïve Bayes model. The overall performance of WMD is substantially better than cosine similarity's performance. The word embeddings done are not paid much attention in the architecture.

The model employs pre-trained multilingual language models for zero-shot cross-lingual keyword extraction used in Koloski, Pollak, Škrlj, and Martinc (2022). The research examines pre-trained language models capable of detecting keywords in multiple languages. These models have been fine-tuned for a specific set of languages, and their effectiveness is evaluated when applied to a new language that was not included in previous tests. The work is done on the Slovenian dataset for keyword extraction. The model evaluation is done on six different datasets from new domains. This includes Russian, Croatian, Latvian, and Estonian news articles with labelled keywords from the Koloski dataset repository. The model's performance could have been improved concerning previous works.

An AI-based answer verifier was proposed in Jagadamba and Shree (2020) to automate the evaluator's task for objective and subjective answers. The system stores standard answers in a database, along with their descriptions and keywords, similar to a knowledge base. The AI then evaluates each answer by matching its keywords or synonyms with the standard answer. For short to medium-length answers, the system's efficiency is approximately 80%. As the length of the answers increases, the importance of the keywords also increases. Therefore, the system's evaluation is more efficient when fewer extra words are added to the answer, making it more suitable for point-to-point answers. However, the system's architecture only focuses on keywords or key phrases and does not consider other measures, such as similarity and context, in the answers. Incorporating these measures would add overhead due to using Grammarly API's sentence checks.

The framework proposed in Yang, Li, Gao, and Zhang (2021) for accurately calculating the similarity of short texts by combining semantic and syntactic information. The framework employs a dynamic semantic vector model that models short text based on term similarity using knowledge and corpora to represent a term's meaning, thereby solving the polysemy problem. The semantic vector is dense, compact, and more suitable for short, sparse texts. The framework also analyses the internal structure of the short text using a Constituency Parse Tree (CPT) to exploit the syntactic information fully. The model applies word similarity measures by combining knowledge base and corpus-based methods to calculate the similarity between terms. Semantic composition obtains semantic vectors of Multi-Word Expressions (MWEs). Firstly, all possible segmentations are generated recursively, and secondly, terms that do not have segmentation ambiguity from all text segmentation cases are chosen. Third, the semantic similarity between the segment and each reference term is calculated, and the highest score is preserved. Finally, the segment with the maximum score is selected as the best segmentation. A matrix is constructed based on the number of terms in two short texts, and then the similarity of each term pair is computed based on the rules. The paper computes the syntactic similarity of short texts based on a CPT. However, the module requires more computing resources. Additionally, the proposed methods perform less satisfactorily for unique items such as phone numbers and email addresses.

The paper (Gao et al., 2021) is the base paper on Simple Contrastive Learning of Sentence Embeddings (SimCSE). The paper has proposed using both supervised and unsupervised, un-labelled dataset versions. The model achieved an accuracy of 76.3%, a 4.2% improvement compared to the previous best results. The work is done on a standard Natural Language Interface (NLI) dataset. Supervised NLI datasets prove to be efficient in learning sentence embeddings by determining if the connection between two sentences is entailment, neutral, or contradiction. The testing is done on multiple datasets such as QQP[3]: Quora question pairs, Flickr30k, ParaNMT, SNLI, and MNLI.

In Shashavali et al. (2019), a method is presented that calculates sentence similarity scores using N-gram and sliding window techniques, along with FastText word embeddings, which surpass current state-of-the-art results in sentence similarity. The proposed approaches employ word embeddings and cosine similarity techniques for word representation and similarity score computation. This method incorporates two distinct strategies: sliding windows with average weighted word vectors and weighted N-gram vectors. After applying the sliding window, a list of substrings is generated. To obtain the vector representation of each window, the method iterates through the substrings and calculates the weighted average of word embeddings, ultimately determining the cosine similarity. In their review article, Hao, Li, He, Wang, and Qu (2022) emphasize the advancements made in applying deep learning techniques to question answering. The paper covers all aspects of subjective question answering, including question classification, answer extraction, question–answer matching, knowledge-based question answering, question generation, and evaluation metrics for question-answering systems. It briefly discusses the significant contributions in this field and their impact on future research. Question classification employs the deepCNN model to identify the target types of posted questions or the expected answer types, enhancing question Answering system accuracy by determining answer types and

[3] https://www.kaggle.com/datasets/aphelionr/qqpdataset.

narrowing the search scope. Answer extraction involves testing processes with MemN2N (Sukhbaatar, Szlam, Weston, & Fergus, 2015), GLU (Madasu & Rao, 2019), MemN2N-GL, CMemN2N, and Dynamic Memory Network (DMN) algorithms, each yielding different accuracies. For question–answer matching, a common approach projects questions and answers into a consistent vector space where similarity can be measured. A Kernel-based Deep Architecture (KDA) was proposed, integrating kernel methods and neural networks into a unique structure. The hybrid method combining LR and IR yielded the best results. A hybrid attention-based DNN, UIA-LSTM-CNN, was developed for answer selection in community question answering (cQA). The paper provides essential insights, results on various algorithms like RNN and ANN, and a summary of the datasets used to train the models. It also discusses the evaluation metrics employed and their relevance. According to the survey, Bi-LSTM performed slightly better than RNN. CNN was more accurate in scoring answers than the other algorithms, while some research papers reported that hybrid models also performed comparatively well.

In Condor, Litster, and Pardos (2021), Automatic Short Answer Grading (ASAG) was evaluated using three unique conceptual representation approaches. These included a count-based technique that extracts the specific vocabulary from the data (bag of words), a basic neural approach employing pre-trained word vectors (word2vec), and a cutting-edge, contextual neural method (Sentence-BERT). A brief overview of each representation type was provided. During the experiment, eight diverse content combinations were vectorized and merged with the response vectors prior to training the classification models. Regarding question holdout, the model can be adapted to out-of-sample questions with only slight enhancement over the majority class when using advanced representation techniques like SBERT.

The research presented in Contreras, Hilles, and Abubakar (2018) is divided into two main stages. The first phase involves extracting domain ontology from a text corpus, with OntoGen assisting educators in identifying concept domains and essay similarities. OntoGen uses the SVM method to discover keywords. Based on the chosen documents, it proposes concepts. The second phase focuses on feature extraction using NLTK tools for essay-scoring purposes. NLTK libraries, such as part of speech count and orthography, are employed for text processing, followed by scoring through a linear regression model. The average similarity score for supervised algorithms is 47.16%, while unsupervised algorithms achieve 34%.

Keyword matching, followed by concept matching for semantics, has been used as the model's architecture in Das, Sharma, Rautaray, and Pandey (2019). The model retrieved the answer and the user's answer. It also used TF-IDF for encoding the answers. The model does not apply to long answers, and there is an absence of semantic scoring in the architecture.

A combination of NLP, semantic analysis, and ontology is used in Rokade, Patil, Rajani, Revandkar, and Shedge (2018). The model included keyword matching, concept matching, and sequence of points grading. The architecture also involves concept maps for comparison of answers. The model lacks applicability for mathematical answers, algorithms, and flowcharts.

The work in the paper (Kadam, Tarachandani, Vetal, & Nehete, 2020) evolves around grading the student's answer by these seven major methodologies, namely spacy similarity, term frequency inverse, document frequency, Difflib similarity, Jaccard similarity, grammar check, cosine similarity and WMD distance. The final score was generated by allotting various weights to each methodology. Their proposed methodology has an accuracy of 40%.

Using LSTM cells was proposed for the first time in Bahel and Thomas (2021). The data required for processing is to be imputed by the examiner as the ideal answer, the number of words, keywords, and then text matching is applied.

In the paper (Kusumaningrum, Hanifah, Khadijah, Endah, & Sasongko, 2023), the authors suggest employing the LSTM model to efficiently harness long-range sequential context data for the purpose of answer selection tasks. In addition, the model uses various hyper-parameters of word2vec and LSTM, such as dimension, context window, dropout, hidden unit, learning rate, and margin. The model used two datasets: a Question Answering dataset and a health article dataset. Additionally, the model needs to perform better over medium-sized answers and answers with more or no sentence interdependencies.

A combined approach using the Jaccard similarity method for checking sentence similarity and then using the BERT model for checking semantic similarity was proposed in Singh et al. (2021). There was a lack of grammar checks and no description of the performance of the semantic similarity method.

The model proposed in Mandge and Thalor (2021) used TF-IDF and cosine similarity. The model involves pre-processing, information extraction, and score generation. There were no relevant findings on the dataset, and the accuracy was around 70%.

The suggested framework in Ou, Chuang, Wang, and Wang (2022) employs Euclidean distance to determine the distance between words and the words found in the NLTK library. This proposed system is divided into three components. The initial part involves text generation, processing the video into text. The second module, answer extraction, focuses on extracting entities and nouns from the text as potential answers. This module serves as a named entity recognition component to enhance objectivity in the evaluation process. The final module is a Bidirectional Auto-Regressive Transformers (BART)-based question generation module, as referenced in Lewis et al. (2020). By inputting sentences containing answers, this module generates relevant questions.

The paper (Das, Majumder, Phadikar, & Sekh, 2021) is a comprehensive survey of the past ten years of automated question generation and assessment research. It includes a review article collection covering related datasets and their wide applications. The paper discusses the datasets used for automated question generation and assessment, including their characteristics and limitations. It then moves on to objective answer assessment, which involves evaluating the correctness of answers generated by automated systems. The paper discusses various algorithms used for objective answer assessment, such as LSA, LDA, MBLEU, and GLSA, and their performances. The paper discusses the algorithms used for visual question–answer generation, such as LSTM networks and AutoEncoders, and their performances. The challenges are in question generation and answer assessment, including issues related to data quality, algorithmic complexity, and human evaluation.

In Annamoradnejad, Fazli, and Habibi (2020), they employed BERT and a pre-processing BERT model for evaluation. The model had an MSE value of 0.046 and was less accurate. Since it is a crowd-sourced dataset, the model had a population bias.

The work in paper (Martin et al., 2020) focuses on analysing the performance of Camembert in a non-English language, specifically French in this case. The work is investigated on the grounds of part-of-speech (POS) tagging, named entity recognition, dependency parsing, and natural language inference tasks. The model's performance on NLI is tested on the French part of the XNLI dataset. The model also uses the CCNet dataset, extracted from a common crawl with a different filtering process than for OSCAR. The model's performance measure was impossible for NER due to the limited availability of annotated corpora. The model's performance was compared with the only available baselines set by Dupont (2017).

3. Methodology

The methodology of the research work is presented in the section. The dataset is first described along with the system architecture. Each module is described in detail as separate subsections. The keyword scoring module is presented in Section 3.3, the similarity scoring module is presented in Section 3.4, and the NER module is given in Section 3.5. The last Section 3.6, deals with the final scoring module of the proposed model.

3.1. Dataset description

The research utilizes two primary datasets: the Semantic Textual Similarity (STS) Benchmark (stsb-en-main) dataset and a custom dataset, both structured similarly.

The stsb-en-main dataset is a compilation of English datasets used in the STS tasks conducted during SemEval from 2012 to 2017 available online.[4] It encompasses a variety of text sources, including image captions, news headlines, and user forums. The dataset comprises approximately 8630 records, each containing answer pairs that have been manually scored for similarity on a scale of 0 to 5. The records typically consist of single-line text pieces.

There have been numerous research studies conducted on the (stsb-en-main) dataset such as "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks" (Reimers & Gurevych, 2019) where the authors demonstrated the effectiveness of their approach on the STSB dataset, "Universal Sentence Encoder" (Cer et al., 2018) where the work introduced the Universal Sentence Encoder which encodes text into high dimensional vectors for tasks including semantic similarity which was evaluated on STSB dataset and "Learning Cross-Lingual Sentence Representations via a Multi-task Dual-Encoder Model" (Chidambaram et al., 2018) proposed a multi-task dual-encoder model for learning cross-lingual sentence representations and evaluated its performance on STSB dataset.

The custom dataset was explicitly created to train the semantic scoring module, necessitating more extensive texts for effective vectorization. This dataset contains question-and-answer pairs manually scored by subject matter experts based on the degree of correctness, using a scale of 0 to 10. It includes around 190 records originating from the core computer science domain. This dataset was designed to evaluate the lowest possible accuracies and is available online.[5]

3.2. System architecture

The proposed framework of our research has three primary modules: keyword scoring, similarity scoring, and NER scoring. All these modules will accept two inputs at the very base. The first is a model answer, and the second is a user/candidate answer. Each module then pre-processes both the answers and generates module-wise scores, which are then used by the final score module. The final score calculation and the overall system architecture are given in Fig. 4. Each module has a descriptive diagram that represents the flow of each module. The idea here is to capture single dominant keywords, the semantic similarity between the user and model answers, and identify common entities in both answers.

3.3. Keyword scoring module

Fig. 1 represents the keyword scoring module. The module aims at catching essential words in the model and user answers. This falls in the standard guideline to capture significant objective results of multiple questions in a single question. The module also handles synonym occurrence in user answers using WordNet's synonym dictionary. Furthermore, the module handles non-noun replacement using POS tagging using standard NLTK libraries. The module takes the user and model answers as input and captures keywords in each using the keyword extraction model. It generates two sets of keywords: user keywords and model keywords. The user keywords are then tagged using POS tagging, and each word is replaced by a set of synonyms using WordNet.

The keyword extraction model also gives us prominence (cardinal values) of each keyword in the model answer based on its semantics to the text. This prominence is used as the dominance of keywords and is used mathematically to generate the keyword score. Each keyword has its value, and all keywords are not treated equally. The presence P is calculated using Eq. (1). Hence, using the presence and dominance of a keyword or its synonym, we can find the matching ratio when divided by the sum of dominance, i.e., $\sum dominance$ of all keywords of the model answer. This matching ratio is the final score for this module. Finally, the scoring mechanism is in accordance with Eq. (2).

$$P = \begin{cases} 0 & \text{if not present} \\ 1 & \text{if present} \end{cases} \tag{1}$$

$$\textbf{\textit{Keyword score}} = \frac{\sum ((dominance) \times P)}{\sum dominance} \tag{2}$$

3.4. Similarity scoring module

Fig. 2 represents the similarity scoring module. It accepts user answers and the model answer as input. The user answers are prone to grammatical inaccuracies; hence, firstly, we correct those. User answers are then summarized using abstractive summarization to deal with the sentence conjugation issues. The user's answer is then appended with its summarization and is treated as a single-user answer. Then, the user and model answers are pre-processed using the same pipeline, which involves stemming, sentence tokenization, lemmatization, and vectorization. At this stage, we get embeddings: one is user answer embeddings, and the other is model answer embeddings. Now, model answer vectors are compared with user answer vectors. The comparison is made with a minimum threshold value. Suppose a given user answer vector and model answer vector have a cosine similarity value greater than this threshold value; in that case, it is counted as a hit for this model sentence.

The final similarity score is given by the formula in Eq. (3):

$$\textbf{\textit{Similarity score}} = \frac{N_{hits} \times M_{marks}}{TN_{sentence}} \tag{3}$$

where

$$\begin{cases} N_{hits} & = \text{number of hits for model sentences} \\ M_{marks} & = \text{maximum marks} \\ TN_{sentence} & = \text{Total number of model sentences} \end{cases} \tag{4}$$

The third module, namely NER, is described in the following subsection.

3.5. Named Entity Recognition (NER) module

Fig. 3 represents the NER module. The module aims to capture objective answers not captured in the keyword scoring module. The module takes the user answer and model answer as input and generates two entities: user answer and model answer entities. It then compares both the user entities and model entities to generate the NER score using the formula given in Eq. (5):

$$\textbf{\textit{NER score}} = \frac{N_{hits} \times M_{Marks}}{TN_{entities}} \tag{5}$$

where

$$\begin{cases} N_{hits} & = \text{number of hit model entities} \\ M_{Marks} & = \text{maximum marks} \\ TN_{entities} & = \text{total number of model entities} \end{cases} \tag{6}$$

The final scoring module that combines all three modules and uses a DNN to predict the score is described in the following subsection.

[4] STS Benchmark Dataset.
[5] Online at Code Ocean.

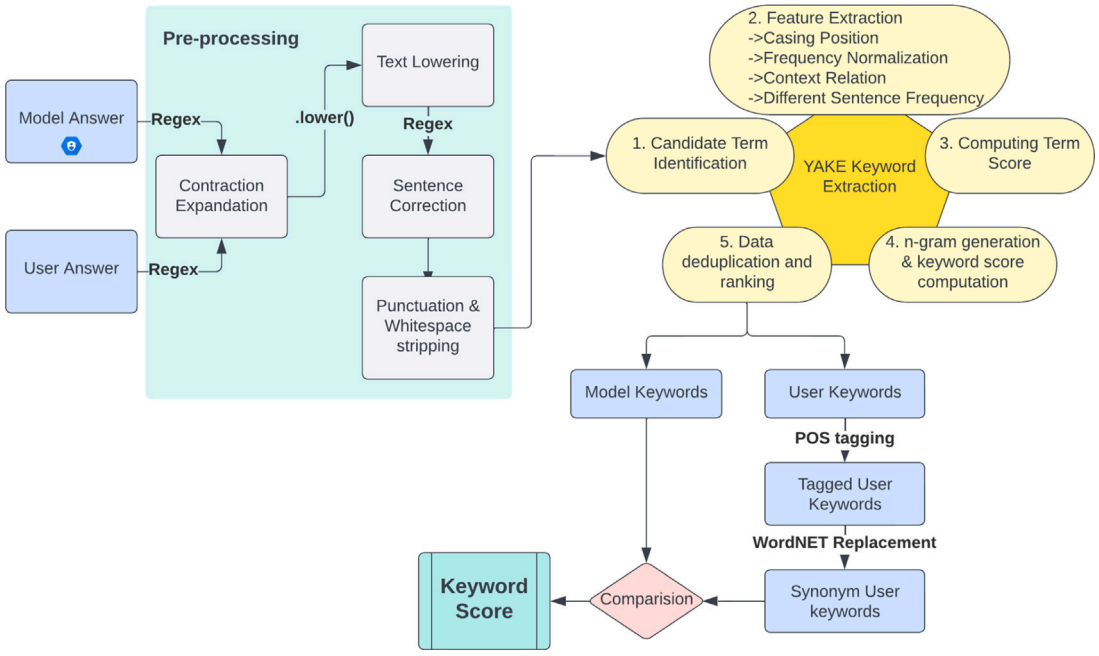


Fig. 1. Keyword scoring module.

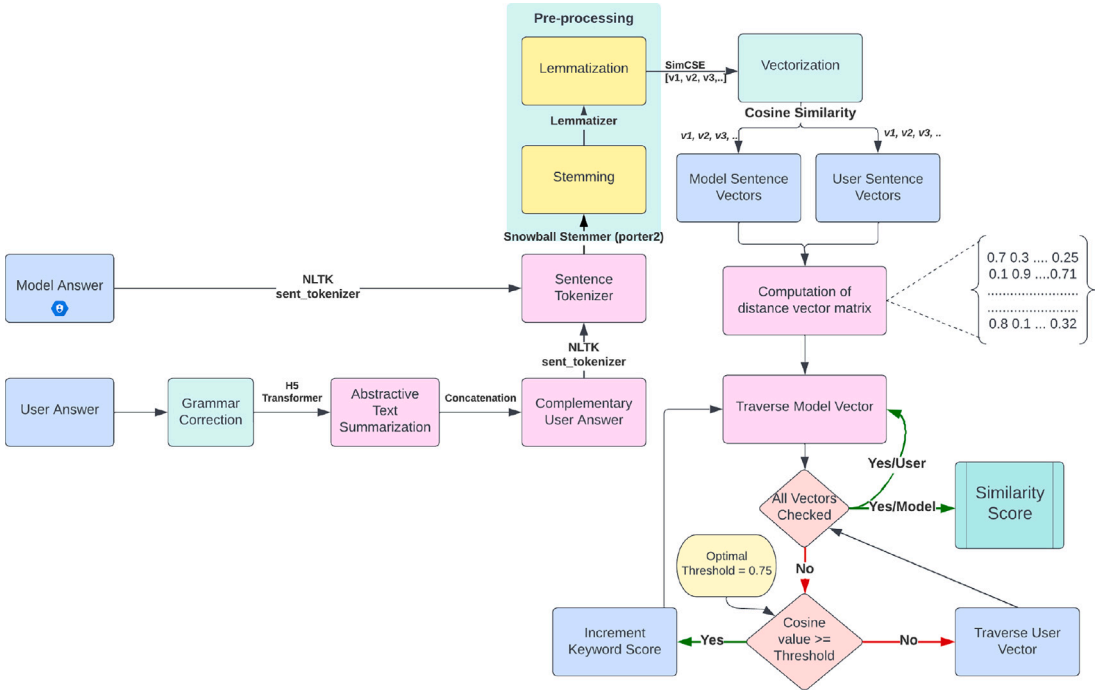


Fig. 2. Similarity scoring module.

3.6. Final scoring module

The final scoring module is represented in Fig. 4 takes the similarity, keyword, and NER scores as input and gives the final score as output. This is the training part where a custom dataset is used. The dataset is a self-designed set of model answers and secondary answers from the computer science domain containing questions and answers from various subjects. It contains around 190 questions and answers and their scores from manual evaluation. In this system, the problem is a standard regression problem where we have three variables, namely the keyword score, similarity score and the NER score, and only one dependent, i.e., the final score or the final marks. This final score module used DNN (with four layers) to predict overall marks for a particular question.

4. Experimental setup

The preprocessing of data, the experimental setup and complexity of all the modules and the computational resources required are discussed in this section.

This paper proposes a novel NLP and deep learning framework to create an architecture with three independent modules, using keyword scoring and similarity scoring (including semantic scores within the similarity matrix) and NER scoring modules. These modules form the

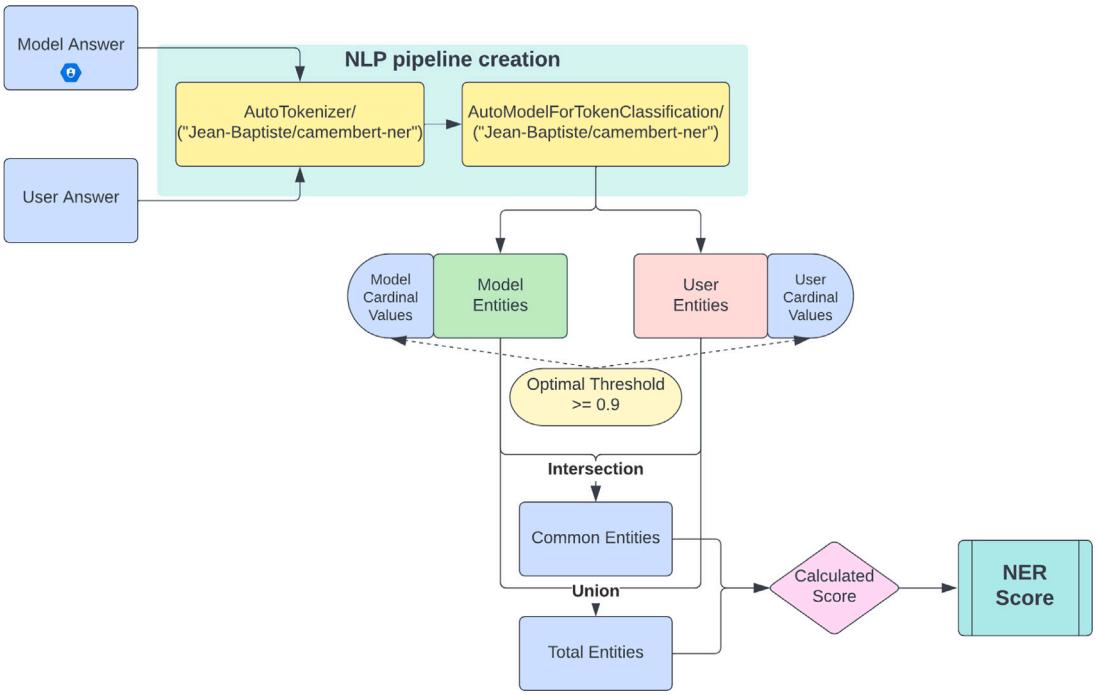


Fig. 3. NER scoring module.

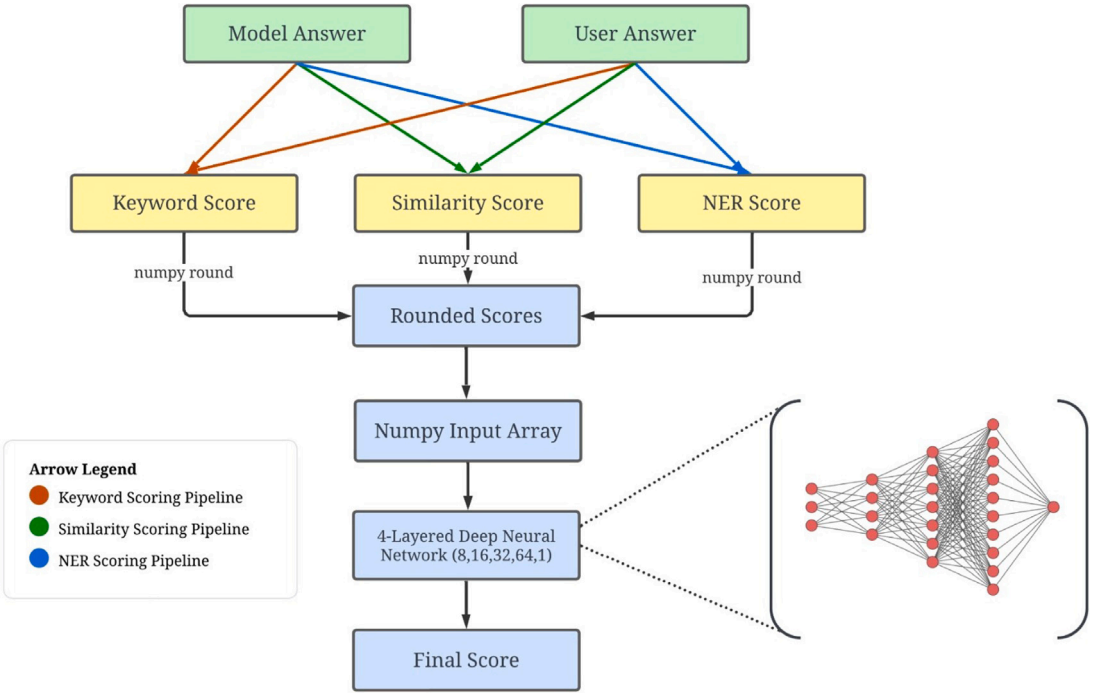


Fig. 4. Final scoring module.

core of our research; hence, the Key Performance Indicators (KPIs) of the final scoring module, a DNN containing neurons for the parametric/constant calculation, directly affect the scores predicted by the module.

Keywords are words crucial to the answer and cover the essence of the answer; with the keywords and their synonyms, the frequency check of the keyword can be done to count the number of matches. Semantic means the graph of the knowledge where each sentence is webbed into a word network, which generates the true essence of a sentence. This semantic scoring is the crux of the paper. The similarity is a crucial aspect which is interconnected with semantics. In this context, semantic scores play a vital role in determining the degree of similarity between sentences. It focuses on substituting the bigrams and multigrams with semantic scoring and calculating the similarity score using another unique matrix called similarity matrix.

The process has been split into three different pipelines, which form the parameters for the evaluation/actual grading of the answers entered by the user. The answers are divided into three categories, i.e., 1-word answers (less accurate with our system), short answers (2 or more words, but restricted to single sentences), and long paragraph answers

(more than one sentence). The models used in each pipeline are trained on a different dataset based on functionality, and the parameters are hyper-tuned (for the deep learning transformer models). The dataset is collected so that the process works on different domains independently.

4.1. Dataset and preprocessing of data

The paper uses two datasets for the entire subjective answer evaluation system. The initial stsb-en-main dataset (benchmarked) (Fikri, Oflazer, & Yanikoglu, 2021; Jiang et al., 2020) contains single-sentence answers, preview available at.[6] Keyword similarity scores of general contexts were used to prepare the keyword scoring module and black box testing of other modules. However, it was not sufficient for the similarity module, hence a new custom dataset with questions and answers (both model answer and user answer) with manually evaluated scores by genuine faculty of the respective computer science subjects. It contains 190 question–answers, available at.[7]

Before we applied our keyword and semantic algorithms to the data, we needed to clean the answers as long answers generally have many issues that need to be rectified before they can be passed to any algorithm. Once the preprocessing is done, the answers can be compared by passing them to these algorithms and then calculating their final score based on the inter-modular score.

In the keyword scoring module, we needed to replace the contractions with their standard widened form so that the keyword frequency density can quickly figure out the keywords to find its overall weightage in the answer. Striping the extra white spaces and bringing the text to the same case, i.e., lower case, were other important preprocessing required for this module. In the similarity scoring module, striping of extra white space is required for the summarization module (encoder) to work correctly, as the other algorithms are capable of self-preprocessing. Since the text semantics are not required to be changed in the NER module, the SpaCy algorithm works best when sentence formations are correct; hence, only extra white spaces and numbers must be removed.

4.2. Keyword scoring module

The first module involves keyword extraction and scoring, where the stsb-en-main benchmark dataset is utilized to evaluate the performance of standard keyword extraction models like KeyBERT and YAKE. KeyBERT is a simple keyword extraction technique that employs the transformer library and takes advantage of the BERT language model. KeyBERT extracts keywords through the following process:

1. A pre-trained BERT model is integrated into the input document. Available from the transformers library, each BERT model converts a text segment into a fixed-size vector designed to represent the document's semantics.
2. N-grams and keywords are obtained from the same page using bag of words methods (such as TfidfVectorizer or CountVectorizer).
3. Each keyword is converted into a fixed-size vector using the same method for embedding the content.
4. With the document and its keywords represented in the same space, KeyBERT calculates their cosine similarity. Subsequently, the terms with the highest cosine similarity scores are extracted as the most relevant.

The idea is relatively simple: It is an enhanced version of a conventional keyword extraction method, with the BERT language model contributing its semantic capabilities. Ultimately, KeyBERT employs two techniques to introduce variation to the final keywords.

1. **Maximum Sum Similarity (MSS):** The first step in using this method is to set the top n parameter to a value, such as 10. The next step extracts $2 \times$ top n keywords from the document. The pairwise similarities of these keywords are calculated. The algorithm then extracts the most relevant terms that are least similar to each other.
2. **MMR (Maximum Marginal Relevance):** This approach is similar to the first one, incorporating a diversity factor. MMR aims to reduce repetition and increase the diversity of outputs in text summarization tasks. The initial step is to select the keywords most closely related to the document. The process then iteratively selects new candidates that are both relevant to the document and dissimilar to the keywords already chosen. There is an option to choose a low-diversity threshold or a high one.

With texts of varying sizes, domains, and languages, YAKE is an original feature-based method used for multilingual keyword extraction. YAKE does not use Thesaurus or dictionaries, and neither are trained against any corpora. Instead, it uses an unsupervised method that relies on textual characteristics, making it applicable to articles written in various languages without needing extra data. This can be useful for many jobs and situations when access to training corpora is limited or restricted. The scoring logic then compares the overall keyword matching and synonym matching using grammatical roots. The values are then compared with the actual scores to get the performance metrics.

The time complexity for the keyword scoring is $O(m^2) + O(n^2)$, where m is the length of the user answer and n is the length of the model answer, due to deduplication algorithms used by YAKE, such as 'dedupFunc' which boost the accuracy. At the same time, the space complexity of the algorithm is $O(n^2)$, where n is the maximum number of keywords extracted.

4.3. Similarity scoring module

The second is the similarity scoring module based on the principle of semantic matching. As in the dataset, we have two kinds of answers: the "Model Answers" (extracted from the infinite knowledge base — AI technique) and the "User Answers" (Manually collected response answers). Semantic matching needs to be done between the answers. Firstly (after pre-processing), we used models like SimCSE, SBERT, and all-MiniLM-L6-v2 (Face, 2023) (a hugging face deep learning transformer model). Cross encoders like stsb-roberta-base (Reimers & Gurevych, 2019) are used to generate the word embeddings from the transformer models and get a multi-dimensional vector. The technique called cosine similarity is used in which the pairwise dot products of vectors are done with the standard documents to generate the semantic score for a particular sentence.

We have incorporated a grammar correction algorithm based on a fine-tuned version of the t5_base model (Raffel et al., 2020) (Text-To-Text Transfer Transformer-5). This algorithm utilizes transformers on tokenized text segments to enhance the accuracy and effectiveness of grammar correction to eliminate the inaccuracies of user answers.

A text summarizer was created to deal with the conjugated sentences, i.e., the sentences with a conjunction between them so that the sentences can clear the minimum threshold required by the pipeline for scoring. The maximum length for which the summarizer can work is 512 words.

all-MiniLM-L6-v2 is a sentence transformer model: It maps sentences and paragraphs to a 384-dimensional dense vector space and can be used for tasks like clustering or semantic search. It has been trained on 1B (one billion) training data to get such a sparse matrix by transformation. Regardless of the size of the documents, cosine similarity is a statistic used to determine how similar they are. Cosine similarity is a mathematical measure of the angle formed by two vectors when they are projected onto a multi-dimensional space. In our instance, these

[6] https://huggingface.co/datasets/kmyoo/stsb-en-tiny.
[7] Online at Code Ocean.

User Answers - Tokenized + Summarized	Model Answer - Tokenized					
	MA-S1	**...**	**...**	**...**	**MA-Sn-1**	**MA-Sn**
UA-S1	0.1	0.6	0.8	0.2	0	0.3
UA-S2	0.5	0.1	0.1	0.3	0.7	0.9
...	...	...	...	...	...	...
...	...	...	...	...	...	...
UA-Sn-1	0.6	0.6	0.7	0.8	0.1	0
UA-Sn	0.2	0.1	0.54	0.23	0.5	0.87
UA-SA-S1	0.1	0.1	0.45	0.2	0.04	0.008
UA-SA-Sn	1	0.5	0.4	0.1	0.2	0.9

Fig. 5. Similarity matrix.

vectors represent word embeddings of two different texts. The cosine similarity captures the direction (the angle) but not the amplitude of the documents when displayed in a multi-dimensional space where each dimension represents a word in the text. The resemblance increases with decreasing angle.

After getting the semantic scores from the vectorizer followed by the cosine similarity metric, we can create a matrix for calculating the similarity scores from this matrix of semantic scores with a threshold value in place. This is the final score from the second pipeline. The similarity matrix can be created as given in Fig. 5.

The space and time complexity for the similarity scoring is $O(m * n)$, where m is the number of sentences in user answers and n is the number of sentences in model answer. The grammar correction is an individual pre-processing which takes $O(t^2)$, where t is the number of terms in each tokenized vector.

4.4. Named Entity Recognition (NER) scoring module

The third pipeline is the NER scoring module, where entities are recognized from the answers. The entities, like names of persons, places, animals, and others, are sometimes the central part of the answer and need to be correctly identified by the evaluator. Thus, this module uses SpaCy and NLTK libraries to find the entities and their respective POS tags. The other module to be used is the Camembert model, which tags these entities. The final score/grading is done based on the number of matches of entities. This module is used to reduce the bias of the overall pipeline by using the tags. NLTK provides a function *nltk.ne_chunk()* that is already a pre-trained classifier to recognize named entities using the POS tag as input. In the output, we can see that the classifier has added category labels such as PERSON, ORGANIZATION, and GPE (geographical-physical location) wherever it is named.

The space and time complexity for the NER Scoring is $O(m * n)$, where m is the number of sentences in user answers and n is the number of sentences in model answers.

4.5. Final score calculation

The scores from all these modules are stored along with the actual graded scores or the marks awarded in the dataset (in both the training and testing). The final scores are then put into $score1\_keyword$, $score2\_similarity$, and $score3\_ner$ along with the actual scores. Then, the machine learning (multiple linear regression model) and the deep learning (DNN) models are used to find the parameters/constants for the domain answers. Then, the new set of real-time answers can be quickly graded by the system belonging to the same domain.

The DNN created consists of 4 layers: an input layer with three inputs, eight neurons sharing the inputs, and the 'relu' activation function. The 16-neuron second layer has the same activation; the 32-neuron third layer has the 'tanh' activation function. The fourth layer has 64 neurons where the 'relu' activation function is used. The final output layer has a normal kernel initializer with a single output score ranging from 0 to 10, rounded to one decimal place.

4.6. Computational resources

The research system was tested with an Intel i5 4vCore logical processor and 8 GB RAM for asynchronous execution, scaling down to 2vCore and 4 GB RAM for synchronous execution of individual modules. The similarity scoring modules require the maximum CPU time, while the NER module demands the least computational resources. These were the resources to load the existing models and then hyper-tuning them, including adding an additional similarity matrix with semantic scores and a final scoring module. To train the DNN model, the minimum requirement of GPU was 2vCore and 2 GB of RAM, without any additional overhead.

5. Results, comparison and discussion

This section presents the results of the proposed model, the comparison, and the study's implications. The comparison with other existing models is dealt with in the theoretical implications, and the practical implications are also discussed.

The paper used a novel architecture based mainly on three modules (containing many parameters) which are independent of each other. These modules are tested individually and on the final dataset, which contained around 190 question–answer pairs (containing the model answers and user answers), which the faculty manually evaluated as manual evaluation is mandatory for DNN to be trained on to find the constants.

The keyword scoring module was individually tested on the answers of the stsb-en-main-test dataset, whose values were compared with the keyword score present for the single-line answers. The score was calculated using two different algorithms, KeyBERT and YAKE, which performed effectively by getting the respective RMSE scores of 0.2261 and 0.2459, compared with the same expected keyword scores from the dataset. This is because each word is treated as a unique word, and along with it, its synonyms are also given equal weightage using NLTK corpora so that the entire vocabulary is covered with the scope of the answer. The critical fact is that the weightage of each word is summed up to the frequency of the entire keyword dictionary. Fig. 6 depicts

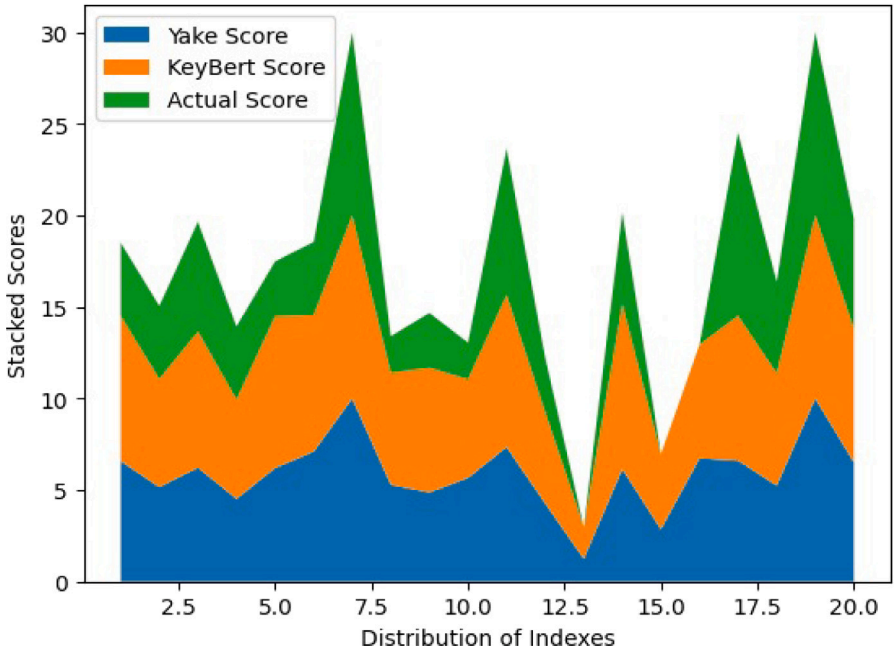


Fig. 6. Keyword scores and affection with actual score.

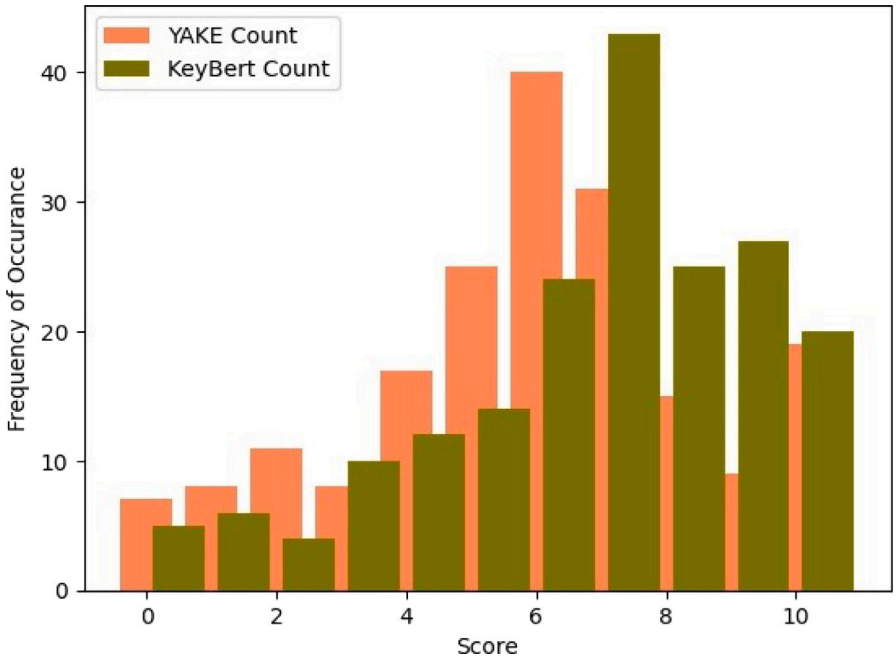


Fig. 7. Frequency of occurrence of keyword scores.

the stacked scores (up to 30 marks as ten marks for each algorithm and actual score) with the distribution of the indexed (compared on 20 random data points), which indicates how the keyword scores pattern for the two algorithms and its affection with actual keyword score. Irrespective of the deviation in the values, YAKE follows the actual keyword score pattern more precisely than KeyBERT. The graph in Fig. 7 tells the frequency of occurrence of particular scores. The values are slightly left-skewed as the scores were high, which depicts that for an answer, keywords play an essential role in score calculation; hence, biases need to be prevented.

The similarity scoring module was tested against the benchmarked paraphrase model like all-MiniLM-L6-v2 from hugging face (hf score), which is a very advanced model to compare the results. Our research tested it with two algorithms used in this module: SBERT and SimCSE. These algorithms were compared against the benchmark algorithms, and the results showed that SimCSE performed better for the short answers (one line/ single sentence). However, on the other hand, SBERT performed better on the large subjective answers (multiple sentences). The SBERT algorithm formed a normal distribution curve. Hence, the values were uniformly distributed. In this module, the results were enhanced with the use of a summary generator using cross encoders, which resolved the issue when the answers could clear the threshold of 0.75 (since our dataset was more technical, we chose the threshold to be strict; otherwise, we can reduce the threshold if the questions are generic), which helps the semantic matching of conjugated sentences. Both the algorithms performed well, but for a technical set of

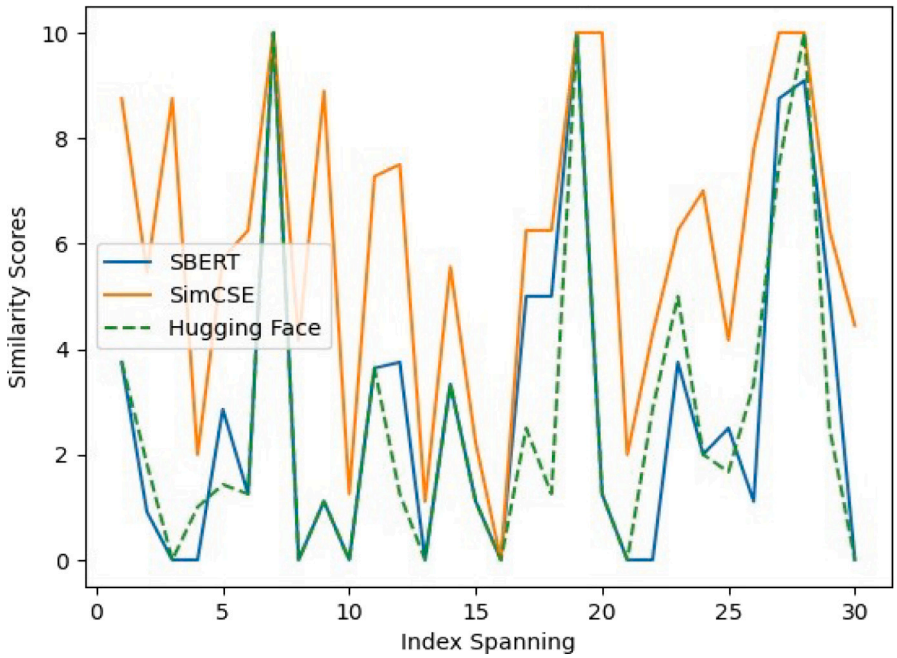


Fig. 8. Distribution of similarity scores.

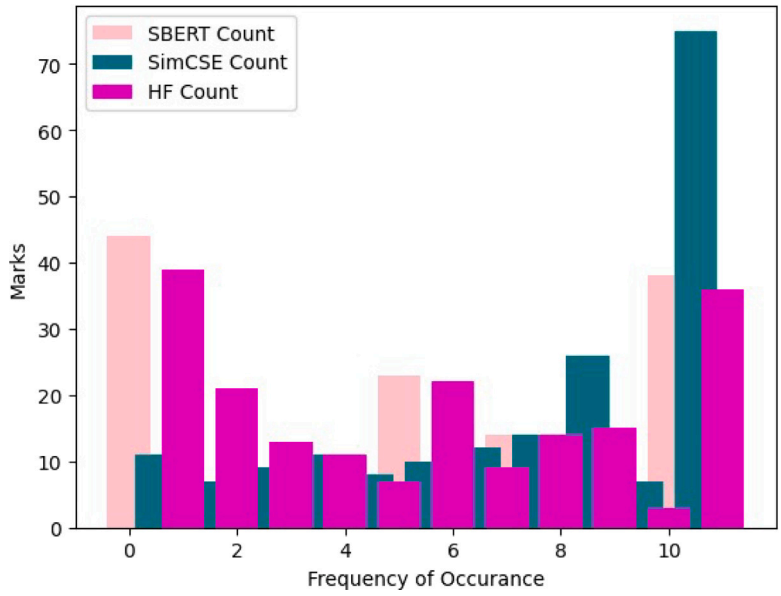


Fig. 9. Frequency of occurrence of similarity scores.

question–answers, SimCSE marginally outperformed SBERT but vice-versa happened for a generic set of question–answers. Through the various trials and Fig. 8, it is evident that the SBERT model tried to overfit the benchmarked scores, indicating high variance while using it; hence, SimCSE perfectly followed the trend. Fig. 9 depicts the frequency of occurrence of similarity scores for all the semantic scoring algorithms used in this paper.

The NER scoring module is indispensable for the named entities handling, which is an essential part of getting the nouns and noun phrases, which forms another important aspect for grading the student's answer. A third benchmarked algorithm, Camembert, was used for finding the named entities. Two algorithms that were used were the SpaCy and NLTK NER tagging algorithms. In most cases, SpaCy performs better, but generally, NLTK spots all the grammatical POS tagging, which finds all the grammatical vocabulary. Hence, this module becomes a meaningful part of reducing bias. The respective RMSE scores for the algorithms are 0.0576 and 0.0439; the lower the RMSE value, the better the algorithm's efficiency. Hence, both models perform their task better than other algorithms in the subjective answer evaluation domain. The Camembert algorithm significantly improved over its counterparts, primarily due to its optimal value of 0.9 used for the cardinal values associated with the named entities. This value effectively addressed the limitations associated with the NLTK NER's non-proper noun tagging and SpaCy's heavy reliance on NER semantic tagging. By optimizing the algorithm, the Camembert approach mitigated the issues arising from NLTK NER's misclassification of non-proper nouns and SpaCy's constrained dictionary size, which hindered the calculation of more accurate NER scores. Consequently, the Camembert algorithm emerged as a superior solution for enhancing performance in NER tasks. As a standalone, NLTK gave uniform score distribution, as in Fig. 10, but

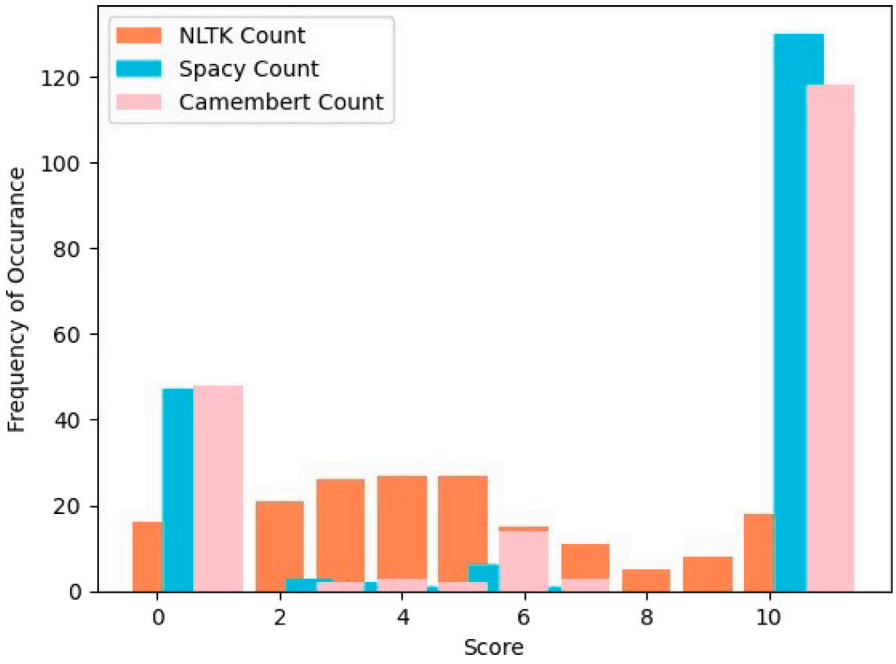


Fig. 10. Frequency of occurrence of NER scores.

Table 1
Comparison table of different scores (keyword, similarity, NER).

Keyword	Similarity	NER	testError/DNN	trainError/DNN	totalError/DNN	testError/LR	trainError/LR	totalError/LR
Keybert	SBERT	NLTK	0.029542765	0.042750988	0.040109343	0.030590132	0.042403618	0.040040921
Keybert	SBERT	SpaCy	0.036037498	0.045166119	0.043340395	0.032740132	0.043254934	0.041151974
Keybert	SBERT	Camembert	0.031342763	0.039664802	0.038000394	0.031650658	0.043388487	0.041040921
Keybert	SimCSE	NLTK	0.030411185	0.040277962	0.038304606	0.029745395	0.041778618	0.039371974
Keybert	SimCSE	SpaCy	0.032476975	0.044636513	0.042204606	0.035019079	0.044023355	0.0422225
Keybert	SimCSE	Camembert	0.035708551	0.044188485	0.042492498	0.034403289	0.044108224	0.042167237
Keybert	HF	NLTK	0.034674341	0.04395625	0.042099868	0.032419079	0.041985855	0.0400725
Keybert	HF	SpaCy	0.034882237	0.041993751	0.040571448	0.0350375	0.042116776	0.040700921
Keybert	HF	Camembert	0.037348027	0.044929275	0.043413025	0.033863816	0.042432566	0.040718816
YAKE	SBERT	NLTK	0.033108554	0.046091118	0.043494605	0.032226974	0.042735855	0.040634079
YAKE	SBERT	SpaCy	0.037174342	0.041348355	0.040513552	0.033926974	0.04301875	0.041200395
YAKE	SBERT	Camembert	0.036895396	0.046383882	0.044486185	0.034292763	0.042980592	0.041243026
YAKE	SimCSE	NLTK	0.029929606	0.042425987	0.039926711	0.0308375	0.041202961	0.039129868
YAKE	SimCSE	SpaCy	0.034361183	0.042237829	0.0406625	0.033405921	0.041595724	0.039957763
YAKE	SimCSE	Camembert	0.031950657	0.038386512	0.037099341	0.033605921	0.041489145	0.0399125
YAKE	HF	NLTK	0.033029606	0.042732566	0.040791974	0.035011184	0.041727303	0.040384079
YAKE	HF	SpaCy	0.036545393	0.041914145	0.040840395	0.036313816	0.041697039	0.040620395
YAKE	HF	Camembert	0.037998025	0.04119375	0.040554605	0.0359875	0.041685197	0.040545658

it was due to selecting non-proper nouns and other noise as entities. On the other hand, on generic values, Camembert outperformed others; SpaCy values depend on the text's semantic structure, which hampers the scoring. Hence, even though NLTK is more uniform, precisely better in distribution is Camembert or SpaCy.

The final scoring module contains all the algorithms for their peer-to-peer comparison. Camembert performed in sync with the DNN model. The regression problem (predicting final scores) is solved using two techniques of machine learning and deep learning, i.e., multiple linear regression and DNN, respectively, due to the most accurate prediction by these algorithms. Table 1 explains the errors in calculating the correct scores for the different combination of the algorithms for which the testError (error on testing data), trainError (error on training data) and totalError (error on all data points) was calculated. The results were encoded in the reverse heatmap with the numerical values. Using Table 1, we can find that DNN understood the data better from each of the modules and produced a relatively better prediction with 0.031 as the RMSE score for test data (testing error), 0.038 as the RMSE score for the training data (training error), and 0.037 as the RMSE score for the overall data (total error) with the keyword scoring algorithm as YAKE, similarity/semantic scoring algorithm as SBERT, and NER scoring algorithm as SpaCy. The DNN likely predicted approximately correct scores initially rounded to one decimal place. The errors, which were scores beyond ten, were kept as it is and were not assumed to affect the accuracy or the error; otherwise, the error was 0.025 for those edge cases. In the case of multiple linear regression, it predicted the answers with 0.033 (approx.) as the RMSE score for test data (testing error), 0.041 as the RMSE value for the training data (training error), and 0.039 as the RMSE score for the overall data (total error) with the keyword scoring algorithm as YAKE, similarity/semantic scoring algorithm as SimCSE, and NER scoring algorithm as Camembert (highlighted by all blue in Table 1) which is the best combination of algorithms in this case.

5.1. Implications of the results

Theoretical Implications of the Study

The proposed system uses a deep learning algorithm to evaluate subjective answers and generate scores. The scores are generated after

passing through three pipelines: keyword matching, semantic similarity, and NER module to the final score calculation module. Researchers have used these three methods individually but not as a combined approach with deep learning.

From the discussion and comparison session, it is clear that the proposed system is better than the existing one. The results are relatively better than just the keyword-based scoring of the answers as proposed by the Firoozeh et al. (2020) where the different keyword extraction techniques were used for the unigram, bigram and multigram (more than two words) to match the similarity of the keywords ignoring the synonyms of the words which may differ the tense and semantics.

In the approach used by Witten, Paynter, Frank, Gutwin, and Nevill-Manning (1999), the KEA algorithm's efficiency depended on the type of TF-IDF vectorization efficiency, which eventually degraded as the lengths of the multigram increased. In contrast, compared to our approach, the semantics as an independent module ensures that the matches are more semantic than just checking the similarity/hit of the keyword match.

In another paper (Johri et al., 2021), the answers were categorized into five groups: excellent (0–1), good (2–3), bad (4–6), very bad (7–8) and worst (9–10). The overall difference, delta, i.e., change in the category, was kept as the parameter was testing accuracy. The actual marks were not considered for the answer evaluation, which is not a correct measure to check the accuracy of the algorithmic model. In our case, we covered the tests by getting their actual values. We calculated the standard deviation of results from the actual manually graded scores by the teacher, which outperforms the used technique. The parameter for similarity matching was kept as 0.75, making it a magic number and a manually fed score in contrast to the deep learning/DNN technique followed by us for calculating the parameters. The overall accuracy of the approach is calculated by mark deviation, which may lie between 40%–50%, which is too low for these examinations.

Using SBERT in Condor et al. (2021) contained many permutations of how the answers and their actual responses are evaluated in a particular domain. Using the Bag-of-words technique for vectorization along with the SBERT model gave an accuracy of 0.621, which is still too high as the 60% accurate model that too for the short answers; compared to our long answers (up to 500 words), the quality degraded as the sentence tokenization as well as the classification technique ignores all other scores and focusses on calculating the F1 score for the evaluation metrics. This is less than our model being roughly 71%+ accurate and independent in the domains, just by training the DNN with all the domains.

For creating this impacted procedure for the answer evaluation system, we get the light on all the other modules, which are essential for the grading system to evaluate subjective answers with transparency and a precise weightage ratio. The similarity matrix using the semantic scores opens a new area of research where the thresholds can be optimized rather than focusing on the domain. Only the last step, where the DNNs are trained to get the domain constants, can be recurrently trained on the other domains, which can be a real-time parameter calculation, other things being constants. Hence, in the worst scenario where the NER is of almost no use as the subjective answers did not contain many named entities, the model will perform even better when the answers are more general and contain some entities. Hence, the proposed approach and the models will always produce accuracies above 71%

Practical Implications of the Study

Human evaluation is regarded as the benchmark for evaluating subjective answers or generally the standard with student paper evaluation. Nevertheless, it has the prospect of being tricky due to the need for more consistency in how human evaluation is carried out.

The results can have much variability, which has implications for the validity of human evaluation results due to the influence of evaluation design preferences. However, transparency issues are generally overlooked, and the potential for biases influencing results is high.

There are several practical implications of this research: Since the evaluation is done by AI, human bias and human error can be reduced to a large extent. Also, since the proposed model is not domain-specific, it can aid in subjective answer evaluation of any domain.

6. Conclusions

We have developed a system using a deep learning algorithm to evaluate subjective answers and generate scores. The scores are generated after passing through three pipelines: keyword matching, semantic similarity, and NER module. We have used enhanced keyword matching techniques, which identify potential keywords and match them with the user's answer. Semantic similarity is the unique functionality of our system, which checks the semantics of the model answer and the user answer. NER helps to determine the context of the user's answer concerning the model answer. Finally, the DNN algorithm generates a weighted score after obtaining the scores from the above three pipelines. The methodology used is robust and precise and has a wide use case. We have demonstrated the effectiveness of the KeyBERT and YAKE algorithms for keyword scoring, with YAKE showing a closer alignment to actual keyword scores. We have also compared the SimCSE and SBERT algorithms for similarity scoring, with SimCSE performing better for technical question–answers and SBERT for generic ones. In the NER scoring module, Camembert outperformed SpaCy and NLTK. The final scoring module utilized DNN and multiple linear regression, with DNN producing better predictions with an RMSE of 0.031 and accuracy of scores 71%+. Overall, the proposed architecture offers a promising approach for automated answer evaluation. This system has the benefit of being almost finished, performing better, and serving a sizable audience. The system aims to surpass or compete with manual evaluators' deviating efficiency, generally between 40 to 60 per cent.

Limitations and Future work

As the AI revolution accelerates, most future examinations will transition to online formats. Our current model effectively handles subjective responses but needs to be improved when addressing answers involving formulas and diagrams. Future work should incorporate these elements to enhance the model's capabilities. Leveraging advancements in Large Language Models (LLMs) like GPT, we aim to address this shortcoming and improve the model's language processing capabilities for future research. GPT has shown remarkable progress in NLP tasks, and it can significantly enhance our system's performance. Lastly, we are committed to maintaining the highest security standards for our model by continuously integrating the latest security features and best practices, including data encryption for answer transmission, strict access controls, and authentication mechanisms. This ensures that only authorized users, such as admins for paper setting and students for exam attempts, can interact with the system. Additionally, our microservice architecture ensures secure endpoints for each service. This ongoing effort ensures the integrity and reliability of our AI-driven examination system. In conclusion, our system significantly advances automated answer evaluation and can revolutionize how examinations are conducted.

CRediT authorship contribution statement

Raghav Agrawal: Formal analysis, Visualization, Investigation, Methodology, Software. **Harshit Mishra:** Formal analysis, Visualization, Investigation, Methodology, Software. **Ilanthenral Kandasamy:** Formal analysis, Data curation, Supervision, Writing – original draft, Writing – review & editing. **Shrishail Ravi Terni:** Formal analysis, Visualization, Investigation, Methodology, Software. **Vasantha W.B.:** Conceptualization, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is available online.

References

Alqaryouti, O., Khwileh, H., Farouk, T., Nabhan, A., & Shaalan, K. (2018). Graph-based keyword extraction. In K. Shaalan, A. E. Hassanien, & F. Tolba (Eds.), *Intelligent natural language processing: trends and applications* (pp. 159–172). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-67056-0-9.

Annamoradnejad, I., Fazli, M., & Habibi, J. (2020). Predicting subjective features from questions on QA websites using BERT. In *2020 6th international conference on web research (ICWR)* (pp. 240–244). IEEE.

Bahel, V., & Thomas, A. (2021). Text similarity analysis for evaluation of descriptive answers. arXiv preprint arXiv:2105.02935.

Bashir, M. F., Arshad, H., Javed, A. R., Kryvinska, N., & Band, S. S. (2021). Subjective answers evaluation using machine learning and natural language processing. *IEEE Access*, *9*, 158972–158983. http://dx.doi.org/10.1109/ACCESS.2021.3130902.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., & Jatowt, A. (2020). YAKE! keyword extraction from single documents using multiple local features. *Information Sciences*, *509*, 257–289. http://dx.doi.org/10.1016/j.ins.2019.09.013.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., et al. (2018). Universal sentence encoder. arXiv preprint arXiv:1803.11175.

Chidambaram, M., Yang, Y., Cer, D., Yuan, S., Sung, Y.-H., Strope, B., et al. (2018). Learning cross-lingual sentence representations via a multi-task dual-encoder model. arXiv preprint arXiv:1810.12836.

Condor, A., Litster, M., & Pardos, Z. (2021). Automatic short answer grading with SBERT on out-of-sample questions. *International Educational Data Mining Society*.

Contreras, J. O., Hilles, S., & Abubakar, Z. B. (2018). Automated essay scoring with ontology based on text mining and nltk tools. In *2018 international conference on smart computing and electronic enterprise (ICSCEE)* (pp. 1–6). IEEE.

Das, B., Majumder, M., Phadikar, S., & Sekh, A. A. (2021). Automatic question generation and answer assessment: a survey. *Research and Practice in Technology Enhanced Learning*, *16*(1), 5. http://dx.doi.org/10.1186/s41039-021-00151-1.

Das, I., Sharma, B., Rautaray, S. S., & Pandey, M. (2019). An examination system automation using natural language processing. In *2019 international conference on communication and electronics systems (ICCES)* (pp. 1064–1069). IEEE.

Dupont, Y. (2017). Exploration de traits pour la reconnaissance d'entités nommées du français par apprentissage automatique (feature exploration for french named entity recognition with machine learning). In *Actes des 24ème conférence sur le traitement automatique des langues naturelles. 19es rencontres jeunes chercheurs en informatique pour le TAL (RECITAL 2017)* (pp. 42–55).

Face, H. (2023). Sentence-transformers/all-minilm-L6-v2.

Fikri, F. B., Oflazer, K., & Yanikoglu, B. (2021). Semantic similarity based evaluation for abstractive news summarization. In *Proceedings of the 1st workshop on natural language generation, evaluation, and metrics (GEM 2021)* (pp. 24–33).

Firoozeh, N., Nazarenko, A., Alizon, F., & Daille, B. (2020). Keyword extraction: Issues and methods. *Natural Language Engineering*, *26*(3), 259–291. http://dx.doi.org/10.1017/S1351324919000457.

Gao, T., Yao, X., & Chen, D. (2021). SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing* (pp. 6894–6910). Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2021.emnlp-main.552.

Grootendorst, M. (2020). KeyBERT: Minimal keyword extraction with BERT. http://dx.doi.org/10.5281/zenodo.4461265.

Hao, T., Li, X., He, Y., Wang, F. L., & Qu, Y. (2022). Recent progress in leveraging deep learning methods for question answering. *Neural Computing and Applications*, 1–19. http://dx.doi.org/10.1007/s00521-021-06748-3.

Hu, Z., Yang, P., Li, B., Sun, Y., & Yang, B. (2023). Biomedical extractive question answering based on dynamic routing and answer voting. *Information Processing & Management*, *60*(4), Article 103367. http://dx.doi.org/10.1016/j.ipm.2023.103367.

Jagadamba, G., & Shree, G. C. (2020). Online subjective answer verifying system using artificial intelligence. In *2020 fourth international conference on I-SMAC (IoT in social, mobile, analytics and cloud) (I-SMAC)* (pp. 1023–1027). http://dx.doi.org/10.1109/I-SMAC49090.2020.9243601.

Jain, S., Seeja, K., & Jindal, R. (2019). Identification of new parameters for ontology based semantic similarity measures. *EAI Endorsed Transactions on Scalable Information Systems*, *6*(20), 1–14.

Jiang, H., He, P., Chen, W., Liu, X., Gao, J., & Zhao, T. (2020). SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 2177–2190). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.acl-main.197, URL: https://aclanthology.org/2020.acl-main.197.

Johri, E., Dedhia, N., Bohra, K., Chandak, P., & Adhikari, H. (2021). ASSESS-automated subjective answer evaluation using semantic learning. In *Proceedings of the 4th international conference on advances in science & technology (ICAST2021)*.

Kadam, S., Tarachandani, P., Vetal, P., & Nehete, C. (2020). *AI based E-assessment system: Technical report*, EasyChair.

Kherwa, P., & Bansal, P. (2017). Latent semantic analysis: An approach to understand semantic of text. In *2017 international conference on current trends in computer, electrical, electronics and communication (CTCEEC)* (pp. 870–874). http://dx.doi.org/10.1109/CTCEEC.2017.8455018.

Koloski, B., Pollak, S., Škrlj, B., & Martinc, M. (2022). Out of thin air: Is zero-shot cross-lingual keyword detection better than unsupervised? In *Proceedings of the thirteenth language resources and evaluation conference* (pp. 400–409). Marseille, France: European Language Resources Association, URL: https://aclanthology.org/2022.lrec-1.42.

Kusumaningrum, R., Hanifah, A. F., Khadijah, K., Endah, S. N., & Sasongko, P. S. (2023). Long short-term memory for non-factoid answer selection in Indonesian question answering system for health information. *International Journal of Advanced Computer Science and Applications*, *14*(2).

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., et al. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7871–7880). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.acl-main.703.

Madasu, A., & Rao, V. A. (2019). Gated convolutional neural networks for domain adaptation. In *Natural language processing and information systems: 24th international conference on applications of natural language to information systems, NLDB 2019, Salford, UK, June 26–28, 2019, proceedings 24* (pp. 118–130). Springer.

Mandge, V. A., & Thalor, M. A. (2021). Revolutionize cosine answer matching technique for question answering system. In *2021 international conference on emerging smart computing and informatics (ESCI)* (pp. 335–339). IEEE.

Martin, L., Muller, B., Ortiz Suárez, P. J., Dupont, Y., Romary, L., de la Clergerie, É., et al. (2020). CamemBERT: a tasty french language model. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7203–7219). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.acl-main.645, URL: https://aclanthology.org/2020.acl-main.645.

Mittal, H., & Syamala Devi, M. (2016). Computerized evaluation of subjective answers using hybrid technique. In H. S. Saini, R. Sayal, & S. S. Rawat (Eds.), *Innovations in computer science and engineering* (pp. 295–303). Singapore: Springer Singapore.

Ou, Y.-Y., Chuang, S.-W., Wang, W.-C., & Wang, J.-F. (2022). Automatic multimedia-based question-answer pairs generation in computer assisted healthy education system. In *2022 10th international conference on orange technology (ICOT)* (pp. 01–04). IEEE.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(1), 5485–5551.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (pp. 3982–3992). Hong Kong, China: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/D19-1410, URL: https://aclanthology.org/D19-1410.

Rokade, A., Patil, B., Rajani, S., Revandkar, S., & Shedge, R. (2018). Automated grading system using natural language processing. In *2018 second international conference on inventive communication and computational technologies (ICICCT)* (pp. 1123–1127). IEEE.

Sakhapara, A., Pawade, D., Chaudhari, B., Gada, R., Mishra, A., & Bhanushali, S. (2019). Subjective answer grader system based on machine learning. In J. Wang, G. R. M. Reddy, V. K. Prasad, & V. S. Reddy (Eds.), *Soft computing and signal processing* (pp. 347–355). Singapore: Springer Singapore.

Shashavali, D., Vishwjeet, V., Kumar, R., Mathur, G., Nihal, N., Mukherjee, S., et al. (2019). Sentence similarity techniques for short vs variable length text using word embeddings. *Computación y Sistemas*, *23*(3), 999–1004.

Shelar, H., Kaur, G., Heda, N., & Agrawal, P. (2020). Named entity recognition approaches and their comparison for custom NER model. *Science & Technology Libraries*, *39*(3), 324–337. http://dx.doi.org/10.1080/0194262X.2020.1759479.

Singh, S., Manchekar, O., Patwardhan, A., Rote, U., Jagtap, S., & Chavan, H. (2021). Tool for evaluating subjective answers using AI (TESA). In *2021 international conference on communication information and computing technology (ICCICT)* (pp. 1–6). IEEE.

Sukhbaatar, S., Szlam, A., Weston, J., & Fergus, R. (2015). End-to-end memory networks. In *Proceedings of the 28th international conference on neural information processing systems-Volume 2* (pp. 2440–2448).

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (1999). KEA: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on digital libraries* (pp. 254–255).

Yang, J., Li, Y., Gao, C., & Zhang, Y. (2021). Measuring the short text similarity based on semantic and syntactic information. *Future Generation Computer Systems*, *114*, 169–180. http://dx.doi.org/10.1016/j.future.2020.07.043.