An approach for automated descriptive answer verification.

Abesh Jha
Computer and Electronics Engineering Department
Kantipur Engineering College,
(Tribhuvan University)
Kathmandu, Nepal
magicianabesh123@gmail.com

Nikhil Mishra
Department of Computer Engineering
Delhi Technological University (DTU),
Delhi, India
nikhilmishra10@gmail.com

***Abstract*—Descriptive answer verification has been a modern-day requirement. It is undoubtedly a challenge for analysis and assessment of subjective answers along with their autonomous grading. Although objective answer evaluation has been in use for a long time, the analysis of subjective answers still possesses great complexity and there is no resolute or precise method for grading answers. Many methods have been implied and various metrics are currently being used for its evaluation with each one having its benefits and drawbacks. Thus, this research aims in tracking the feasibility of the metrics being used and providing an alternative approach for an automated system having relatively higher accuracy.**

***Index Terms*—Natural Language Processing(NLP), Text-rank algorithm, Cosine Similarity, Fuzzy Logic.**

I. Introduction

The concept of education and examination go hand in hand. It is essential to test the degree of knowledge absorbed by the learner to apply it practically. Since examinations are conducted periodically at a large scale, correcting examination papers can prove to be a soporific task. Analyzing hundreds of papers that contain monotonic answers has an element of declined enthusiasm attached to it, in addition to being time-consuming and unproductive. Hence, alternative approaches have been implemented with the purpose of computerization of evaluation but have been largely focused on objective answers which are relatively easy. However, objective examination solely cannot accommodate all types of questions and does not provide complete assurance in terms of the level of understanding of the subject. This research aims to automate the evaluation system for descriptive answers to make it fast processing and independent of manpower. It also guarantees the elimination of emotion from the whole checking procedure. Also, it intends to simplify the process of record-keeping and extraction. The proposed system will use a standard answer as a reference and utilize various metrics for comparison and subsequent allocation of marks through an unbiased checking procedure. Moreover, these kinds of automated systems not only lead to an efficient and unbiased checking system but are also time and cost-effective.

II. Literature Review

In March 2018, A. Shinde, C. Nirbhavane, S. Mahajan, V. Katkar, and S. Chaudhary presented a paper in which the proposed system constituted of three main steps: keywords and synonyms extraction, matching of keywords and synonyms, weighting the keywords, and generating score.[1] This evaluation system graded the answers depending upon the number of keywords matched. The major drawback of this system is that the keywords need to be provided in capital letters. For long sentences, this proves to be largely inconvenient. Also, the system only looks for those keywords that have been provided in the standard answer rather than extracting those from the students answer which may contain a few different words. Our system aims to overcome this by implementing the text rank algorithm to auto-extract keywords from both the answers separately. Similarly, another paper published in November 2014 by P. Sheeba provided another approach towards subjective answer verification. In this system, the evaluation process consists of four main steps: Pre-Processing, Part-of-Speech Tagging, Grading System, and Parse Tree. While the proposed system provided an element of semantic and syntactic analysis, it can only efficiently evaluate single sentences and an upgrade of the algorithm was required as explicitly mentioned in the paper. [2] Our system overcomes these hurdles as it efficiently evaluates multi-sentenced descriptive answers along with proper grammatical analysis. Another paper publishes in August 2013 by A. Kaur, M. Sasikumar, S. Nema, and S. Pawar proposes a graphical approach. The methodology used for the development of the algorithm is to match learners' single sentence answers with the standard single sentence answers by converting both answers into a graphical form and then intelligently matching their nodes and labels.[3] In the proposed approach, the graphical representation of descriptive answers to present complex information clearly and to represent knowledge in a machine-interpretable form has been considered. However, this approach fails when used for multiple sentences and consequently cannot be implemented in large-scale examinations. In our system, we

use various similarity matching metrics such as percent match, Levestanian distance, and cosine similarity to examine the degree of similarity between two answers. This method is better suited to evaluate long answers containing multiple sentences.

III. Scope Of Work

1) Implementation and analysis of text-rank algorithm for keyword extraction.
2) Automated grading of manual answers using five different metrics where syntactic analysis is not a requirement.

IV. Methodology

A. Proposed System

Our proposed system primarily consists of three major components: information extraction module, weighting module and score generation module as shown in Figure 1.

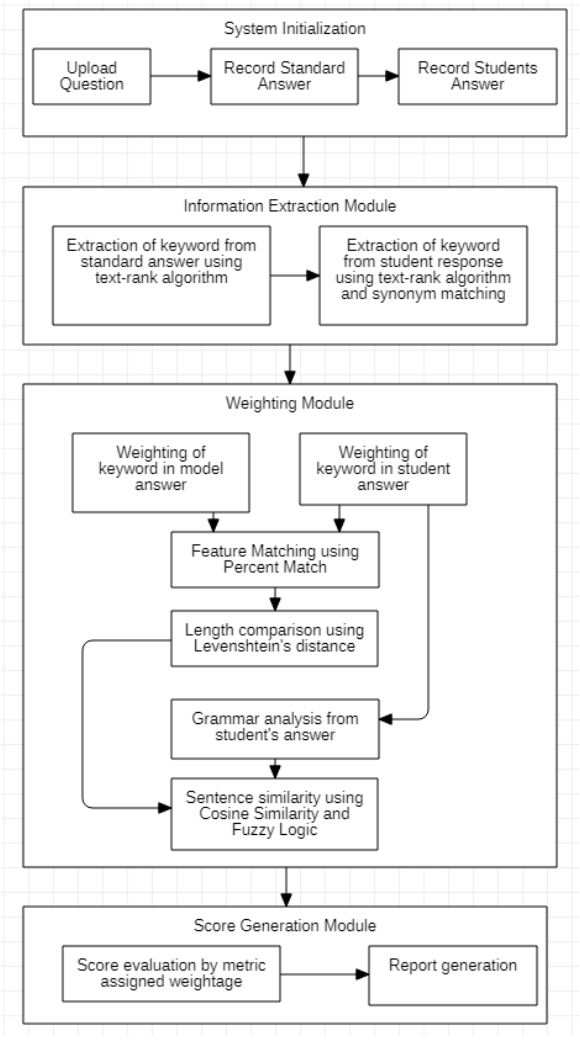

Fig. 1. System Diagram.

1) Initially, the system extracts keywords from the stored standard answer as well as the answer submitted by the user. Along with the keywords, the synonyms of the keywords submitted by the user will also be extracted and equated. The keywords that get repeated very often in a document are given less importance, while the keywords which occur rarely in a document will be assigned a higher value. The synonyms of the keywords will be extracted from the dictionary imported from a python library named PyDictonary.

2) The next step comprises of Information Extraction Module. The keywords are extracted and prioritized by the text-rank algorithm which updates the weights of every keyword according to the importance and frequency of words. Thus, only the most important keywords in a given paragraph are listed. Subsequently, keywords are extracted from the students answer as well and compared with the keywords from the standard answer. The weight of every matched keyword is added to obtain the final total.

3) After keywords are matched, we check if the length of the answer is close to the standard answer using Levenshtein distance. Successively, the percent match checks the accuracy of the percentage matched with the standard answer.

4) Further, cosine similarity is calculated to measure the similarity between two sentences by measuring the cosine of the angle between the two sentences by converting them into their respective vectors by implementing the word to vector process. Additionally, fuzzy string matching helps to find matching strings even if the string is not entered exactly.

5) Finally, the evaluation of the score is done based on marks predicted by the score generation module. This module provides the result based on the weight assigned to various metrics. Appropriate percentage is distributed according to the priority of the metrics.

B. Information Extraction Module

Text Rank Algorithm for Keywords Extraction

The task of the keyword extraction algorithm is to automatically identify a set of terms that best describe the document. Such keywords may constitute useful entries for building an automatic index for a document collection, can be used to classify a text or may serve as a concise summary for a given document. The Text Rank keyword extraction algorithm is fully unsupervised. The authors of the paper [4], introduce TextRank, a graph-based ranking model for text processing, and show how this model can be successfully used in natural language applications. First, the text is tokenized and annotated

with part of speech tags a preprocessing step required to enable the application of syntactic filters. To avoid excessive growth of the graph size by adding all possible combinations of sequences consisting of more than one lexical unit (ngrams), we consider only single words as candidates for addition to the graph, with multi-word keywords being eventually reconstructed in the post-processing phase. Next, all lexical units that pass the syntactic filter are added to the graph, and an edge is added between those lexical units that co-occur within a window of words. This is used to construct an undirected unweighted graph. After the graph is constructed, the score associated with each vertex is set to an initial value of 1 and the text-rank algorithm is run on the graph for several iterations until it converges usually for 20-30 iterations, at a threshold of 0.0001. Once a final score is obtained for each vertex in the graph, vertices are sorted in reversed order of their score and the top T vertices in the ranking are retained for post-processing.

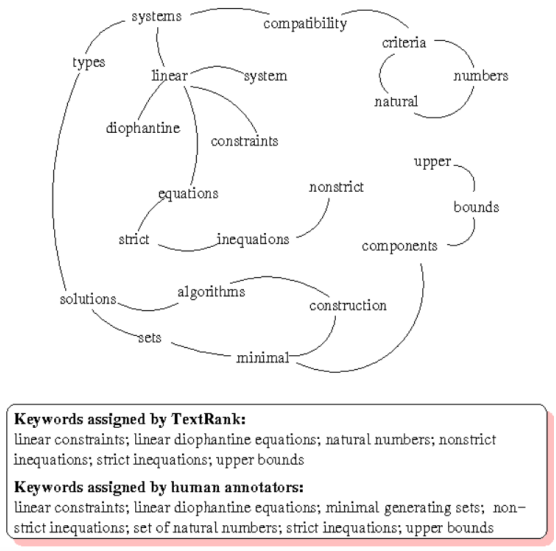

Fig. 2. Sample Graph for keyword extraction using Text Rank algorithm.[4]

Figure 2 shows a sample graph built for an abstract from our test collection. While the size of the abstracts ranges from 50 to 350 words, with an average size of 120 words, we have deliberately selected a very small abstract for illustration. For In this example, the lexical units found to have higher importance by the TextRank algorithm is (with the TextRank score indicated in parenthesis): numbers (1.46), inequations (1.45), linear (1.29), Diophantine (1.28), upper (0.99), bounds (0.99), strict (0.77). This ranking is different from the one rendered by simple word frequencies. For the same text, a frequency the approach provides the following top-ranked lexical units: systems (4), types (3), solutions (3), minimal (3), linear (2), inequations (2), algorithms (2). All other lexical units have a frequency of 1, and therefore cannot be ranked, but only listed.

It can be observed from Figure 2 that the text-rank algorithm prioritizes the words in terms of ranking which implies that if a student uses a keyword having a high rank, he receives a higher score as compared to the words which are given low priority by the algorithm.

C. Weighting Module (Evaluation of Metrics Used)

I)Levenshtein Distance

The difference in length between the two sentences is equally important. For example, a student might write an error-less sentence, however, be very limited in words. Thus marks must be deducted for the decreased length. Hence to calculate the length difference from the desired answer, Levenshteins distance is calculated from the standard answer. The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions, or substitutions) required to change one word into the other as shown in fig.3.

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Fig. 3. Mathematical representation for calculation of Levestian Distance. [5]

EXAMPLE : The Levenshtein distance between FLOMAX and VOLMAX is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits.

II) Percentage match

Percentage match is used in the system to determine how closely two values match each other by calculating the Character Edit Distance between two String values, and also taking into account the length of the longer or shorter of the two values, by character count. In mathematical terms, the Character Match Percentage comparison uses the following formula to calculate its results as shown in equation 1 where:
In mathematical terms, the Character Match Percentage comparison uses the following formula to calculate its results as shown in equation 1 where:

CMP = Character Match Percentage
MCL = Maximum Character Length of the two values being compared, in characters
CED = the Character Edit Distance between two String values
CL = either the Maximum or Minimum Character

Length, depending on the setting of the Relate to shorter input option. If Relate to shorter input is set to No (as by default),the Maximum Character Length is used. If Relate to shorter input is set to Yes, the Minimum Character Length is used (that is, the number of characters in the shorter of the two values by character count).

$$CMP = \frac{MCL - CED}{CL} * 100 \quad (1)$$

III) Grammar

Grammar is one of the primary parameters used for a detailed evaluation of subjective answers since a sentence might contain just the important keywords but no correlation or grammatical equivalence between them which makes the answer highly equivocal. Hence, evaluation of grammar is inevitably a high priority to attain greater accuracy. The system uses grammar as one of the prominent evaluation metrics. We have used the Python library "Pydictionary" for analysis purposes. Our system uses three major processes for grammar.

Tokenization: It is used to create tokens by breaking up a sequence of strings into pieces such as words, keywords, phrases, symbols, and other elements for further processing.
Stemming: It is used to reduce the word to its word stem by removing the affixes or conversion to its respective root word.
Lemmatization: It is a normalization technique with benefits same as that of stemming. However, the major difference between them is that lemmatization is used to reduce the words to base form as opposed to stemming which just removes affixes.

In our system, tokenization occurs first which breaks the strings and sentences followed by stemming and lemmatization. A combination of stemming and lemmatization is required for better results. Since grammatical analysis tends to be very vague and complex, all these operations are performed using existing inbuilt libraries in our proposed system for faster and accurate evaluation.

IV) Fuzzy Logic

Approximate matching of strings is reviewed with the aim of surveying techniques suitable for finding an item in a database when there may be a spelling mistake or other error in the keyword. The methods found are classified as either equivalence or similarity problems.[6]
Fuzzy systems offer numerous methods that prove helpful in solving application problems, where the governing parameters bear fuzziness in nature. In particular, many variants of fuzzy decision-making models have been successfully explored as vocational guidance systems to assist decision-makers in recognition and guidance processes, especially in the Medical diagnostic realm.[8]
Fuzzy string search can be used in various applications such as a spell checker or a typos corrector. For example, when a user types "Missisaga" into Google, a list of hits is returned along with Showing results for "Mississauga". This implies that a search query returns results even if the user input contains additional or missing characters or other types of spelling errors. Fuzzy logic has been used in the proposed system in a similar context. It aims at providing some marks to the student's answers even if they are somewhat similar and doesnt require answers to be identical to the standard answer. Fuzzy logic is used in the system by importing the python library named fuzzywuzzy which takes two strings as input and returns a fuzzy value between them.

V) Cosine Similarity

Cosine similarity is used as a method for approximating how similar two words/sentence vectors are to each other. The intuition behind cosine similarity is relatively calculating the cosine of the angle between the two vectors to quantify how similar the two documents are.[7] The process of conversion of word to vector is done by a process called Bag Of Words. For example:
Sentence 1: AI is our friend and it has been friendly
Sentence 2: AI and humans have always been friendly

Step 1: We calculated Term Frequency using Bag of Words as shown in Fig.4:

Term Frequencies:

Sentence	AI	IS	FRIEND	HUMAN	ALWAYS	AND	BEEN	OUR	IT	HAS
1	1	1	2	0	0	1	1	1	1	1
2	1	0	1	1	1	1	1	0	0	1

Fig. 4. Calculation of Term Frequency by word count

Step 2: The main issue with term frequency counts shown above is that it favours the documents or sentences that are longer. One way to solve this issue is to normalize the term frequencies with the respective magnitudes or L2 norms. Summing up squares of each frequency and taking a square root, L2 norm of Sentence 1 is 3.3166 and Sentence 2 is 2.6458. Dividing above term frequencies with these norms, we get results as depicted in Fig 5.

Document	AI	IS	FRIEND	HUMAN	ALWAYS	AND	BEEN	OUR	IT	HAS
1	0.302	0.302	0.603	0	0	0.302	0.302	0.302	0.302	0.302
2	0.378	0	0.378	0.378	0.378	0.378	0.378	0	0	0.378

Fig. 5. Values after Normalization of Term Frequencies

Step 3: As we have already normalized the two

vectors to have a length of 1, we can calculate the cosine similarity with a dot product:
Cosine Similarity = (0.302*0.378) + (0.603*0.378) + (0.302*0.378) + (0.302*0.378) + (0.302*0.378) = 0.684
Therefore, cosine similarity of the given two sentences is 0.684.However, one major drawback of cosine similarity calculation using Bag of Words is semantics.While bag of words is intuitive and easy to implement, there is nothing in the document-term matrix which is capable of capturing the semantic similarity between words like hi and hey. Thus to overcome this, another feasible approach called word embeddings can be used which are basically vector representations of words which model semantic similarity through each words proximity to other words in the vector space.

D. Score Generation Module

The score is generated by the combination of all the metrics listed above. Appropriate weight is assigned to each of the metrics to obtain a final predicted score.
According to a published research paper entitled AI Answer Verifier, the score for marks is generated depending on the various factors and the total marks will be the sum of all the marks obtained from, individual, sections. Depending upon the priority, keywords are given 40 percent priority followed by grammar 30 and then synonyms 20 respectively and finally, the length of the answer is given 10 percent. The marks for length are dependent on the percentage of keywords matching and a making scheme is tabulated for giving marks to the length.[1] Thus, if the students write a completely different answer with no keywords matching, marks would not be allocated. Our proposed system also assigns similar distribution of percentage to the metrics, however, with some necessary changes. The problem with the distribution of weights mentioned above is that an answer may be grammatically correct but convey no actual meaning or be relevant to the question asked. Thus, to overcome this drawback, our proposed system only considers the evaluation of grammar only if at least 30 percent of the keyword is matched with the standard answer. Also, while grammar is an important factor to be taken into consideration, other parameters seem to be equally accountable for a proper evaluation. Hence, Grammar accounts for 20 percent of overall marks including synonym matching. Further, our proposed system takes cosine similarity and fuzzy logic into account equating to 20 percent of the total marks. Finally, the Levenshtein distance and percent match between the two answers: standard and students, accounts for ten percent each.

For example: A sample of standard answer and an answer written by student is taken as an input in the system as shown in fig.6. Further, fig.7 shows the

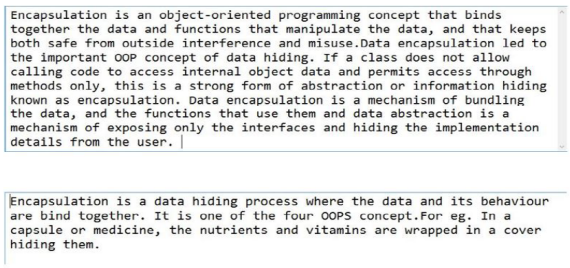

Fig. 6. Sample model of a standard answer and a manually written answer from the dataset.

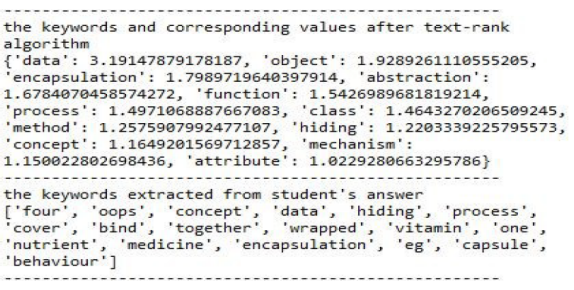

Fig. 7. Keyword extraction using Text rank algorithm for both standard and manual answers respectively.

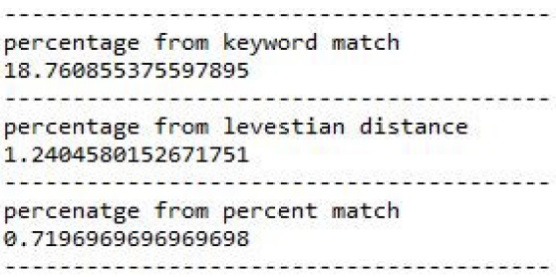

Fig. 8. Scores obtained for various metrics.

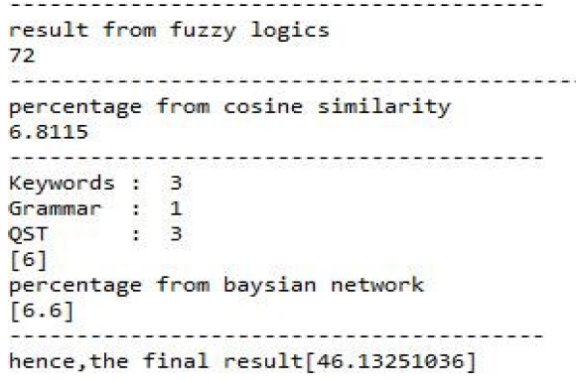

Fig. 9. Final score obtained after weight assignment and calculation.

important keywords extracted and their corresponding values after the implementation of Text Rank algorithm. Consequently, the marks obtained after the amount of keywords matched, levenshteins distnce , percent match and grammar is calculated individually as demonstrated in fig.8 . Finally,the respective weight assigned to each metrics is calculated to obtain the final score as shown in fig.9.

Result And Discussion

To check the reliability and the accuracy of the proposed system, we compared the results obtained from the system with the standard answers graded by some real evaluators. The dataset comprised of a standard answer and a readers commentary for a correct answer. Also numerous manually hand written answers by students for the same question was given and score was provided along with evaluation of mistakes.For experimental purpose, we took around 10 questions from the official GRE website where some prompt was given and students' answers were graded by real teachers. A score of 6 implied that it was a perfect response whereas a score of 1 was the lowest. We took the score 6 response as our standard answer and then inserted all the other responses having scores 5,4,3,2 and 1 respectively in our system. Respective scores obtained for each response were noted and then compared with the actual scores received for all the 10 responses. The scores were converted to a scale of 100. The results obtained can be demonstrated in a line chart as shown in Figure 10.

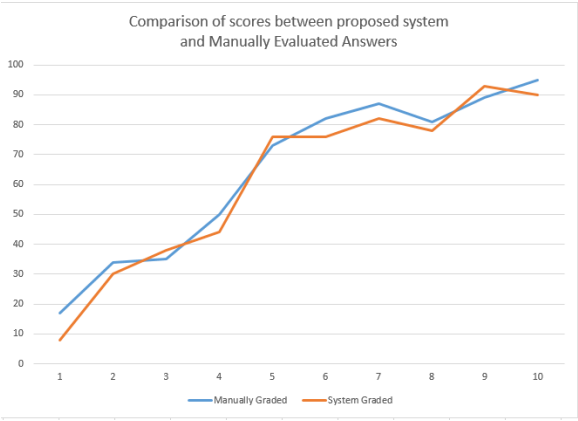

Fig. 10. Line Chart of manually evaluated scores and system evaluated scores for same answers.

It can be observed that the result is mostly similar in most of the answers apart from few outliers. This huge discrepancy in some cases can be credited to the fact that the standard answer may or may not include every possible answer. Since this system has a major limitation of syntactic analysis, an answer may be equally correct without possibly matching the standard answer. Therefore, in this case, the teachers grade it high but the system fails to do so. Such outliers were not taken into consideration or recorded in the graph shown. Also, the limited number of test results was due to the lack of availability of a dataset that mirrors our proposed system. To overcome these limitations, the system should be tested in a compromised environment where the data (students' answers) are more or less similar to that of the standard answer provided. Also, the question should be based on factual knowledge and keyword similarity rather than semantic analysis and cognitive reasoning.

Limitation

1) The students' answers should be more or less comparable to the standard answer stored. An entirely different variant of the answer, although correct, may not be graded properly by the system.
2) Lack of syntactic analysis. The system doesnt check for the meaning of words or how well an argument is supported or contradicted.

Future Enhancement

1) A Reinforcement learning approach will be used for better emulation of the manual checking process.
2) Currently the system is suited for only a single question. However, verification of multiple answer choices at the same time will be used.
3) The accuracy of the system will be checked on a large dataset comprising of manually graded answer sheets evaluated by real teachers.
4) Possible technologies such as deep learning will be incorporated in the system for higher accuracy.

Conclusion

Hence it can be seen that various metrics have their advantages and disadvantages. A trade-off between metrics is required to build a system having greater accuracy. Still, without the inclusion of syntactic analysis, the evaluation cannot be fully accurate. However, for grading descriptive answers by comparison with a standard answer, a combination of diverse metrics can be merged to obtain relatively higher precision. Although this inclusion accounts only for empirical and non-semantic answers, it nonetheless makes the evaluation process a lot easier for human accessors. Moreover, a Machine Learning approach is required for semantic analysis of answers and completely automated checking of descriptive answers.

References

[1] A. Shinde, C. Nirbhavane, S. Mahajan, V. Katkar, and S. Chaudhary, Ai answer verifier, International Journal of Engineering Research in Computer Science and Engineering, vol. 5, no. 3, pp. 143145, 2018.
[2] P. Sheeba, An approach to evaluate subjective questions for online examination system, International Journal of Innovative Research in Computer and Communication Engineering, vol. 2, no. 11, pp. 64106413, 2014.

[3] A. Kaur, M. Sasikumar, S. Nema, and S. Pawar, Algorithm for automatic evaluation of single sentence descriptive answer, International Journal of Inventive Engineering and Science, vol. 1, no. 9, pp. 112121, 2013.
[4] R. Mihalcea and P. Tarau, Textrank: Bringing order into text, in Proceedings of the 2004 conference on empirical methods in natural language processing, 2004,pp. 404411.
[5] R. Haldar and D. Mukhopadhyay, Levenshtein distance technique in dictionary lookup methods: An improved approach, arXiv preprint arXiv:1101.1232, 2011.
[6] P. A. Hall and G. R. Dowling, Approximate string matching, ACM computing surveys (CSUR), vol. 12, no. 4, pp. 381402, 1980.
[7] J. Ye, Cosine similarity measures for intuitionistic fuzzy sets and their applications, Mathematical and computer modelling, vol. 53, no. 1-2, pp. 9197, 2011.
[8] Thomas, A. et al. Intelligent Fuzzy Decision Making for Subjective Answer Evaluation Using Utility Functions. 2008 First International Conference on Emerging Trends in Engineering and Technology (2008): 587-591.