Computer Assisted Short Answer Grading with Rubrics using Active Learning

Ganesamanian Kolappan

Publisher: Dean Prof. Dr. Sascha Alda

Hochschule Bonn-Rhein-Sieg – University of Applied Sciences, Department of Computer Science

Sankt Augustin, Germany

August 2023

Technical Report 01-2023

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272 ISBN 978-3-96043-107-7

This work was supervised by	Paul G. Plöger Manfred Kaul Tim Metzler

Digital Object Identifier https://doi.org/10.18418/978-3-96043-107-7

Abstract

This thesis investigates the benefit of rubrics for grading short answers using an active learning mechanism. Automating short answer grading using Natural Language Processing (NLP) is one of the active research areas in the education domain. This could save time for the evaluator and invest more time in preparing for the lecture. Most of the research on short answer grading was treated as a similarity task between reference and student answers. However, grading based on reference answers does not account for partial grades and does not provide feedback. Also, the grading is automatic that tries to replace the evaluator. Hence, using rubrics for short answer grading with active learning eliminates the drawbacks mentioned earlier.

Initially, the proposed approach is evaluated on the Mohler dataset, popularly used to benchmark the methodology. This phase is used to determine the parameters for the proposed approach. Therefore, the approach with the selected parameter exceeds the performance of current State-Of-The-Art (SOTA) methods resulting in the Pearson correlation value of 0.63 and Root Mean Square Error (RMSE) of 0.85. The proposed approach has surpassed the SOTA methods by almost 4%.

Finally, the benchmarked approach is used to grade the short answer based on rubrics instead of reference answers. The proposed approach evaluates short answers from Autonomous Mobile Robot (AMR) dataset to provide scores and feedback (formative assessment) based on the rubrics. The average performance of the dataset results in the Pearson correlation value of 0.61 and RMSE of 0.83. Thus, this research has proven that rubrics-based grading achieves formative assessment without compromising performance. In addition, the rubrics have the advantage of generalizability to all answers.

Acknowledgements

First, I thank God for keeping me healthy and strong during the thesis period. I thank my first supervisor, Prof. Dr. Paul G. Plöger, for allowing me to work on this thesis topic. I am grateful to have Prof. Dr. Manfred Kaul as my second supervisor. Also, I thank my advisor, Mr. Tim Metzler, for being the backbone of this project and guiding me throughout the project. I extend my gratitude to Mrs. Iman Awaad for her strong support throughout my journey in this master's course.

A special thanks to Mrs. Jill Yates-Wolff, for her valuable input on literature writing to compose this report. I appreciate my colleagues, Deepan Chakravarthi Padmanabhan, Santosh Reddy, and Meghana Sambrani, for their constant support, constructive criticism, and motivation.

Lastly, I extend my love and dedicate this work to my late father, Mr. Kolappan Perumal, and my grandmother Mrs. Kangathanu Perumal.

Contents

List of Abbreviations

AMR	Autonomous Mobile Robot
ASAG	Automatic Short Answer Grading
Bi-LSTM	Bidirectional Long Short Term Memory
BLEU	Bilingual Evaluation Understudy
BOW	Bag of Words
CNN	Convolutional Neural Network
ELMo	Embeddings from Language Model
GPT	Generative Pre-trained Transformer
GUI	Graphical User Interface
LSAP	Linear Sum Assignment Problem
ML	Machine Learning
NLP	Natural Language Processing
PC	Personal Computer
POS	Part-of-Speech
RBF	Radial Basis Function
RF	Random Forest
RFR	Random Forest Regressor
RMSE	Root Mean Square Error
S-BERT	Sentence-Bidirectional Embedding Representation from Transformers
SOTA	State-Of-The-Art

SVM	Support Vector Machine
SVR	Support Vector Regression
TF-IDF	Term Frequency-Inverse Document Frequency

List of Figures

List of Tables

1 Introduction

The examination is a practice of assessing student's knowledge which is considered necessary in their learning process [72]. This assessment will determine the student's capability to grasp the information provided by the teachers. The examination might be written, oral, practical, or computer-based (higher usage recently) [58]. Irrespective of the type of exam, the possible question types are fill-in-the-blanks, multiple-choice, short answer, essay, reading comprehension, or others that include math formula, coding, and matching [16, p. 5]. Comparatively, the short answer has gained more interest [32] and is used to evoke student knowledge about the concepts in the subject [89]. Since it has a unique combination of three criteria [16], 1. students have to recall or think from their knowledge instead of recognizing/choosing, 2. the answer is limited to one or two sentences or max one paragraph, and 3. it is close-ended where the content is preferred instead style of writing. Essay grading focuses on the style of writing where the flow of content/keywords and organization of information is favored [54]. The classical way of assessing the short answer involves the evaluator grading student's answers individually. This consumes much time and energy of the evaluator. Additionally, the evaluator cannot grade all the answers in a limited time, which causes a delay for students in receiving their assessment results [56]. In contrast, computer-based short answer grading is faster and more consistent than the classical method. Computer-based grading of short answers is called Automatic Short Answer Grading (ASAG). The ASAG eases the evaluator's workload, which provides more time for self-study and preparation for the lectures [45]. Sometimes, ASAG requires a human grader to assist in grading in this situation; it is semi-automatic or computer-assisted, which is the focus of this research [41]. Thus automating the grading system benefits both the student and the evaluator.

ASAG is considered as a similarity task where the Machine Learning (ML) model assigns a grade based on the degree of similarity of the student's response to the reference answer for the given question. The major challenge in ASAG is replicating the evaluator's performance in interpreting the student's answers for grading. Evaluators are often capable of grading a student's answer even if it is paraphrased or does not match precisely with the reference answer [58]. Figure 1.1a illustrates that the student response partially matches the reference answer, which gained half of the total score. However, the grade is biased toward the evaluator. Also, this provides only a score that does not contain the justification for it (summative assessment). Therefore, the reference answer needs rich information to grade the short answer.

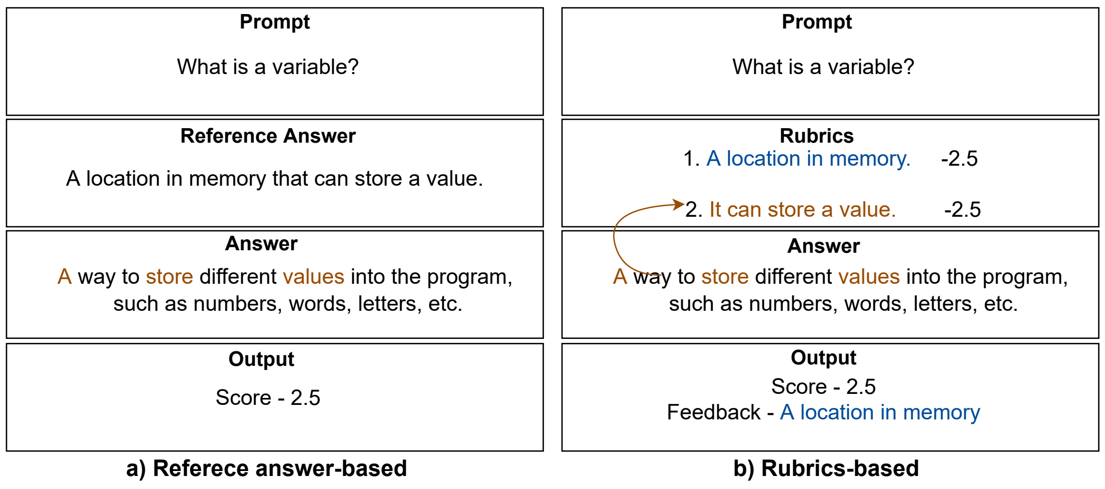

Prompt
What is a variable?
Reference Answer
A location in memory that can store a value.
Answer
A way to store different values into the program, such as numbers, words, letters, etc.
Output
Score - 2.5
a) Referece answer-based
Prompt
What is a variable?
Rubrics
1. A location in memory. -2.5
2. It can store a value. -2.5
Answer
A way to store different values into the program, such as numbers, words, letters, etc.
Output
Score - 2.5
Feedback - A location in memory
b) Rubrics-based

Figure 1.1: Example of short answer grading using a) reference answer and b) rubrics. This example is taken from the Mohler and Mihalcea [63]. Despite no rubrics in the dataset, as shown in (a), it is made manually to highlight the importance of rubrics as presented in (b). The color highlights the key elements similar to the student response and reference/rubrics.

Rubrics are a common assessment technique followed by graders to evaluate student's answers consistently and provide feedback in the classical method. Assessments based on rubrics highlight the area or topic the student has to improve [82]. Rubrics state the key elements that need to be present in the answer with their corresponding scores as depicted in Figure 1.1b. This is of two types: positive rubrics; when the key elements are mentioned in the answers, their corresponding scores add up to a total score for that particular answer [94, p. 1]. Another is negative rubrics when the key element is missing; their respective scores are added, and the difference from that particular answer's total score is calculated. Figure 1.1b depicts the negative rubrics of two key elements, each of score -2.5, where the second rubric is presented in the answer. Hence, the first rubric score adds up to the total score of 5 to yield a final score of 2.5 (= 5 + (-2.5)). Thus, the first rubric that needs to be presented in the student answer is provided as feedback (formative assessment) along with the grade.

ASAG eliminates the evaluator from the grading task. In real-world scenarios, ASAG struggles to perform similarly to the evaluator [16]. Relatively simple Natural Language Processing (NLP) methods should assist the manual grading. So, the evaluator will be in the grading cycle to support the ML modALel predicting the grade achieved by active learning. Active learning is a wrapper that can be placed above any model [35]. Active learning allows the model to query a human grader/evaluator/annotator to label the data during training [84]. The fundamental intention of active learning is to allow the model to choose the data to learn. This approach helps train the model with few labeled data. The unlabeled data are annotated, with the knowledge gained from labeled data. If the model cannot label the data, it queries the human grader. Thus, active learning helps annotate a large amount of data inexpensively. This process is called semi-supervised learning. Therefore, this research uses active learning with negative rubrics to grade the short answer.

1.1 Motivation

The classical way of grading does not scale to a large group. As the schools switching to online assessments, grading also becomes automatic to minimize the effort and maximize the number of evaluations [70]. ASAG could be a better fit for this scenario which has been an active research area since 2001 [16] that assures minimum effort and equity. ASAG might be effortless for True/False and multiple-choice questions. In contrast, automatic essay and short answer scoring are fundamental challenges as there is no possibility of waiving unlikely answers. Grading essay answers does not require reference since the automatic essay scoring accounts for spelling correction, sentence coherence, grammar, and similarity to the topic [72]. At the same time, grading short answer considers understanding a specific concept expressed in one to three sentences [34]. ASAG has been developed across various domains, namely citizenship exams, foreign language learning, classroom exams, entrance exams, and general tests [41]. It has several benefits, such as

- Grades are available faster, and there is no longer a waiting time for students. Additionally, teachers can invest less time in grading where they need to supervise the evaluation [40].
- Grading is consistent, whereas human graders may tend to be wrong sometimes due to fatigue, stress, bias, or the effects of ordering [16, 40, 24].

- Grading can be provided for small to large groups of students [24].
- Scores, as well as feedback, is available that combines both summative and formative assessments [58, 16, 24].
- Grading style of the evaluator can be integrated.

ASAG applies to any course ranging from science to computer engineering [94, 63] across different languages, including English, German, Chinese, and French [16, 40]. The idea of ASAG can be extended to similar domains requiring grading, such as an interview or competitive test [80], entrance and certification exams, quiz competitions, or similar. Hence, developing a better ASAG model will benefit the education domain and other domains assessing the individual's performance through examination.

1.2 Problem Statement

There is a need for ASAG for consistent assessment as new questions, and different responses are generated regularly [16]. The current ASAG systems are based on a supervised learning method that requires labeled data [41]. Additionally, these model grades are based on the reference answers provided by the grader or automatically selected by the model using clustering [16, 58]. These approaches induce difficulties such as:

- Manually annotating the data which is expensive and time-consuming. Sometimes, these data are required in large amounts if the model is deep learning-based supervised learning [24, 94].
- Having reference answers for each question requires one or two human graders/expert's authorization which is not cost-effective [24].
- Deep learning requires a large amount of data and is not fast enough to grade as it requires more computational time.
- Assuming the data consists of a balanced distribution of classes [98].
- No generalization of all ways to correctly answer a particular question [58].
- Most of the ASAG does not provide feedback (formative assessment).
- Partial grades for the student's response are not in consideration.
- Sometimes, the supervision of the evaluator is ignored, or ASAG is treated as a replacement for a human grader [16].

Research by Marvaniya et al. [58], Wang et al. [94], and Hasanah et al. [40] has proven that including rubrics in ASAG has improved performance compared to reference answers. Current approaches pave much less attention to the rubrics, which is significant in a real-world situation for evaluating student's answers [94]. Hence, having rubrics using active learning could address the difficulties mentioned above induced by the present models for ASAG. The Research Questions (RQ) to be answered in this work are as follows:

RQ1 What are the available methods for ASAG?

RQ2 Does the rubrics aid in providing proper and helpful feedback with grades?

RQ3 Is the model able to generalise?

RQ4 Which models are suitable for active learning to grade fast with effectiveness?

1.3 Objective

The primary purpose of this research is:

1. To provide feedback along with scores - formative and summative assessment.
2. To have the evaluator in the grading process.

Negative rubrics are incorporated to provide feedback and score on grading the short answer. The approach extracts NLP-based features from each key element in the rubrics and student's responses to compute their similarity. The similarities help to calculate the score, and the key element that is not in the student's answer is provided as feedback. In this process, active learning includes the evaluator teaching the rubrics for the queried student's response from an optimal minimum quantity of the total answers. Therefore, the research hypothesis is that active learning with the ML model using negative rubrics aids to provide feedback along with the score having the minimal intervention of the evaluator for grading the short answer.

1.4 Report Outline

This report comprises six chapters; Chapter 1 provides the introduction followed by the motivation, an overview of the research proceedings, and a report outline. Chapter 2 provides various State-Of-The-Art (SOTA) methodologies for ASAG and its deficits. Chapter 3 enlightens the necessary knowledge about the concepts used throughout this research work. Chapter 4 states the proposed method for the short answer grading using negative rubrics. Chapter 5 presents the working and results of the proposed methodology on two datasets from different domains. Chapter 6 includes this research work's summary, contribution, limitations, and future direction.

2 Related Work

The significance of NLP is that it can analyze textual data, untie human perception and opinion about a service, product, or another person. So, NLP is being utilized for diverse applications in the education field. The research work by Fonseca et al. [30] automatically classified the student's programming assignments based on the subject-based context using NLP. Thaker et al. [92] recommended remedial readings based on the student's knowledge state using the textual similarity method. The research by Arthurs and Alvero [2] uses word embedding to obtain the ground truth from the student's college admission essay to remove the evaluator's bias. Also, Xiao et al. [96] used transfer learning and active learning for problem detection between subjects in peer assessments, d'Aquin and Venant [25] evaluated text complexity in essays written by English learners using a concept graph, which is a vectorization method in graph analysis. Chen et al. [22] predicted student satisfaction with the online tutorial using various textual analysis methods. The research work by Prokhorov and Safronov [73] is fascinating as it uses NLP methods to obtain relevant articles in NLP for the researchers. Xing et al. [97] evaluated the student's education in engineering college for their sustainable development using the Term Frequency-Inverse Document Frequency (TF-IDF). Rodriguez-Ruiz et al. [79] used various features such as inference, chunking, text, and word analysis to estimate the student's digital literacy skills. The research work by Romadon et al. [80], Yusuf and Lhaksmana [100] states that the NLP has been used to evaluate candidate answers during the interview assessment, such as aptitude and oral examination. Shaik et al. [85] did a literature review about incorporating NLP methods to analyze the student's feedback about the course or professor. The investigation of Pedró et al. [71], Khaled [46] and Carey et al. [19] states that there is extensive open-ended research about NLP's impact in the education domain. Therefore, the above-motioned research could highlight the vast usage of NLP in the education domain.

Despite researchers adopting NLP for various applications in the education domain, it has been used for grading short answers for over two decades (2001-2022) [16, 56]. However, the work by Burrows et al. [16] states that the first use of the NLP method to evaluate student answer dates back to 1966, which is called project essay grade. According to this research work, the literature survey for short answer grading is divided into three parts: reference answer-based, rubrics-based, and active learning-based.

2.1 Reference Answer-based

An example of short answer grading based on reference answer is presented in Figure 1.1 and Table 3.1. Callear et al. [17] formulated an automated text maker, which grades the short answer based on the number of concepts matching from student answer to reference answer. These concepts are assigned weights based on their corresponding importance. The researchers Mitchell et al. [62] and Bachman et al. [3] used parse tree representation to obtain a template based on which the grades are assigned. Similar to the previous research, these parse representations hold weights according to their significance. Later, one of the well-known methods to grade short answers were C-rater developed by Leacock and Chodorow [51], which used concepts from both student and reference answer to generate model sentences. The score is assigned based on the concept presented in the model sentence after NLP preprocessing (discussed in Chapter 3). C-rater was one of the popular methods to grade essay answers as well. The development of ML paved the way for Pulman and Sukkarieh [74] to present a comparative study between different ML methods on short answer grading using reference answers. In addition, Madnani et al. [57], Hou and Tsao [42] used ML models such as Support Vector Machine (SVM) and logistic regression on simple features such as Part-of-Speech (POS) tags, term frequency, length of response, Bag of Words (BOW) from student answer, and reference answer to detect the score. Gütl [36] made use of NLP preprocessing techniques along with text similarity and statistics measures to grade the answer in the E-learning platform based on the provided reference answer. The researchers Nielsen et al. [65], Bailey and Meurers [4] used corpus to obtain the score for the short answer, where paraphrasing and synonyms correlation drawbacks from the previous research are addressed. The work from Gomaa and Fahmy [33] used corpus-based features to combine or compare with previously used features to provide a different perspective for short answer grading that is to be interpreted as a similarity task. Mohler and Mihalcea [63] contributed to providing a dataset on computer science that is widely used for benchmarking. The research used knowledge-based and corpus-based similarity features between reference and student answers. Klein et al. [49] developed a latent semantic analysis for grading, whereas Basu et al. [6] used a clustering approach to group similar responses and computed similarity with the reference answer. The literature survey from Burrows et al. [16] classified the above previous works on short answer grading into different verticals as concept mapping, information extraction, corpus-based, machine learning, and evaluation. Sultan et al. [89] provided a fast, simple, and high-accuracy approach to grade the short answer by augmenting the textual similarity features such as sentence alignment, semantic vector similarity, and embedding along with the grading constructs such as question demoting, and term weighting. This is the highest-performed SOTA for short answer grading. Research work by Pribadi et al. [72], Pado and Kiefer [68] employed weighted similarity and sorting to grade the short answer, which failed when the student's answer was long. Due to the various research in short answer grading, several competitions have been conducted. Among them, the most famous is SemEval'13 task [16, 20]. These competitions have paved the way to collect more data which successively made use of deep learning to grade the answers. Research works of Sung et al. [90], Reimers and Gurevych [77], Luo [56] and Devlin et al. [27] made use of transformers and transformer-based models such as Bidirectional Embedding Representation from Transformers (BERT) trained on different corpora to grade the short answer, which is also available publicly to fine-tune. In addition, recent short answer grading research employs mostly deep learning methods [78, 24]. Krishnamurthy et al. [50] had compared four deep learning methods, namely, character-based Convolutional Neural Network (charCNN), Bidirectional Long Short Term Memory (Bi-LSTM), word level CNN, and BERT, to present the performance of deep learning on short answer scoring. Zhang et al. [101] had combined the benefit of deep belief networks with feature engineering. Patil and Agrawal [70] has used two reference pairs from the student answers for each grade category and provided reference answers. The features are extracted using LSTM with an attention layer. Researchers Yang et al. [98] and Liu et al. [55] employed attention networks and deep autoencoder models to grade the answer as a classification problem. In addition, Qi et al. [75] and Tan et al. [91] explored the usage of convolutional networks for feature extractions from student responses and reference answers to grade. Further, Gaddipati et al. [31], Camus and Filighera [18] extensively studied transformer models such as BERT, Generative Pre-trained Transformer (GPT), and Embeddings from Language Model (ELMo) on their performance for short answer grading. Also, Cer et al. [21], Gomaa and Fahmy [34] evaluated models such as BERT and skip-thought vector based on transfer learning from the relevant domain corpus. However, the strategy prevailing from template matching to deep learning and, at present, transfer learning is comparing the student's answer to the reference answer and predicting the grade based on certain similarities [45] as expressed in Equation 2.1.

$$score_i = \delta(a_i|q_i, ra_i) = \sum_{s=1}^{N} sim_s(a_i, ra_i) \tag{2.1}$$

Where, i is the number of student answers, a is the student answer in consideration, q is the question corresponding to the student answer, ra is the reference answer corresponding to the answer, s is the number of similarities in computation and sim_s similarity metric between reference and student answer.

2.2 Rubrics-based

The usage of rubrics instead of reference answers has gained limited recognition. The research work by Santos et al. [82] initially attempted to manually grade using rubrics with a web-based Graphical User Interface (GUI). The evaluators provide the questions with rubrics so that when the students deliver the answers, the evaluator selects the rubrics that match and presents the score. Nevertheless, the process is entirely manual; only the system is electronic. Sakaguchi et al. [81] developed an approach combining reference answers and rubrics where the question is provided with one or two reference answers with the key element to be captured. So, the approach used a stacking of two Support Vector Regression (SVR) where the first one looks for a matching reference answer to the student answer; the other SVR obtains the score based on the key elements presented in both the reference and student answer. Marvaniya et al. [58] attempted to create scoring rubrics from the student's answers based on clustering and representative selection from those clusters for each score. Using the selected representative answer and the reference answer increased the performance. Hasanah et al. [40] developed keyword-based rubrics for short answer grading for the Indonesian language, where each keyword contains certain weights. The number of keywords in the student's answer sums up the total score. Also, the paper mentioned specific tools, namely POS tagging and wordnet, which were unavailable in the Indonesian language, to further the research. The research work by Wang et al. [94] used a Bi-LSTM with a pooling layer from Riordan et al. [78] as the base component, along with the word level attention layers (for each rubric) as a rubric component. The concatenated sum of features from the base component and the sigmoid of features from the rubric component predicts the score for the respective answer. The rubrics are used in place of reference answers to predict the scores based on the similarities between the student's answer and rubrics, be it a keyword or key phrase, or whole sentence. Overall, the rubrics are not used to provide feedback but score similar to the reference answer-based methodologies, expressed in Equation 2.3.

$$R = \{r_j\}_{j=1}^{M} \tag{2.2}$$

Where, r is a rubric, R is the set of M number of rubric. Substituting R instead of ra in Equation 2.1 yields,

$$score_i = \delta(a_i|q_i, R_i) = \sum_{s=1}^{N}\sum_{j=1}^{M} sim_s(a_i, r_{ij}) \tag{2.3}$$

2.3 Active Learning-based

Active learning has been used as a wrapper for different models, to mention a few, random forest [64], CNN [8], SVM [13], and deep learning [8] for different tasks such as time-series classification [5], anomaly detection [8], image classification and detection [13]. In addition, active learning is used in the context of NLP for obtaining word sense disambiguation [28] and text classification [29, 59]. Whereas, in short answer grading, active learning has a small contribution. Wang et al. [93] was the first to employ active learning for short answer grading where few answers are labeled with their respective scores with which the scores for other answers are predicted. Goudjil et al. [35] made an overview of an Arabic text classification using SVM with active learning. Also, active learning used batch mode selection for querying instead of the pool based. This is achieved by assigning posterior probability using SVM to the samples to get the most informative instance. However, there are no experimental results to support the idea. Li et al. [54] used active learning for sampling the answers manually by the evaluator instead of defined sampling techniques. Niraula and Rus [66] utilized active learning to get feedback on the one or two words generated for the gap-filling questions. The research work by Horbach and Palmer [41] compared the performance of active learning with different sampling techniques, seed selection, and the number of samples to query (batch size). In addition, Kishaan et al. [48] investigated the performance of active learning for short answer grading based on different query strategies, and ML models. Mieskes and Padó [61] had conducted various experiments to compare the performance of manual grading with automated grading to semi-automated grading. Al-Tamimi et al. [1] had compared active and passive learning for Arabic text classification. The research highlights the effectiveness of using active learning compared to unsupervised learning or manual annotations followed by supervised learning. Thus, previous research emphasized the benefits of using the active learning method to grade the short answer with few labeled data. In addition, specific research has provided optimal parameter setting such as sampling methods, batch size, better-performing ML models, and seed settings.

2.4 Summary

The research work by Al-Tamimi et al. [1], Horbach and Palmer [41] highlighted the significance of using an active learning approach on top of the ML models. However, these active learning methods used reference answers to grade the student's response as a similarity task. Utilizing reference answers restricts the paraphrase or word usage and partial grades/scores. Also, reference answers are expensive since it needs more than one expert opinion. On the other hand, employing rubrics shatters the disadvantage of using reference answers. Nevertheless, more than rubrics to predict scores is required to achieve the objective. The most crucial intention of having an assessment is to improve knowledge. Predicting scores does provide the evaluation, whereas providing feedback [95] rounds off the assessment. Comparing Equation 2.1 and 2.3 presents the scenario of missing feedback. Negative rubrics offer feedback on what was missing in the answer or the justification for the score provided. Therefore, this research aims to provide feedback and a score as expressed in Equation 2.4 using particularly negative rubrics. Table 2.1 provides the limitations of each method.

$$score_i,\ feedback_i = \delta(a_i|q_i, R_i) = \sum_{s=1}^{N}\sum_{j=1}^{M} sim_s(a_i, r_{ij}) \tag{2.4}$$

Method	**Limitation**
Reference answer-based	1. Need more than one expert opinion. 2. No generalization of all answers. 3. Partial grades are not considered.
Rubrics-based	1. Human grader is not included in the evaluation. 2. Does not provide feedback.
Active Learning-based	1. Does not provide feedback. 2. Does not account for grading as regression.

Table 2.1: Limitations of related work.

3 Background

This chapter provides a general overview of pre-processing and popular feature extraction methods used for short answer grading. In addition, the machine learning methods used in this research work are discussed. Readers can skip if the topics are familiar. This chapter helps to provide the concept useful to understand the later chapters.

3.1 Preprocessing Methods in ASAG

The text contains rich information, which is challenging to operate. Preprocessing the text is intended to lower the computational load for the algorithm by reducing its size. Even though preprocessing phase helps lessen the computational burden, it does cause inevitable information loss from the raw text. Therefore, it is essential to understand each preprocessing method and apply it when it is significant. The following preprocessing methods are supported with the example from the Mohler dataset [63] of question I'd 1.5 presented in Table 3.1.

Question:	What is a variable?
Reference Answer:	A location in memory that can store a value.
Student Answer:	A variable is a location in the computer's memory where a value can be stored for use by a program. Each variable has a name, a value, a type, and a size.

Table 3.1: An example short answer of a student for the question with reference answer is provided from the Mohler dataset [63]. The particular example is chosen because of its simplicity to understand. To know more about the dataset, refer to Section 5.1.1.

3.1.1 Spellcheck

Few student responses may contain misspelled words; in that case, spell correction is mandatory to enhance performance. Spell-checking is a statistical approach based on

Peter Norvig's [67] blog post. The spell-checking method uses a word frequency list from a reference corpus and the Levenshtein Distance algorithm. Each word in the text is checked across the list of known words. No correction is applied if the word is present in the list. Otherwise, all permutations of an edit, such as insertions, deletions, replacements, and transpositions of a character from the original word, are generated. It then compares these generated words to known words in a word frequency list. Those words found more often in the frequency list are more likely the correct word. In exceptional cases, both the given word and the generated word are unknown, and that word is not corrected. The correction quality increases based on the corpus used for reference; for better results, use the domain-dependent articles. The sentences from Table 3.1 has no spelling mistakes. Therefore, Table 3.2 presents a different example from the same dataset.

Original: To provide an example or model of how the finished program should ***perfom***. -Provides ***forsight*** of some of the ***challanges*** that would be encountered.

Spellchecked: To provide an example or model of how the finished program should ***perform***. -Provides ***foresight*** of some of the ***challenges*** that would be encountered.

Table 3.2: Spellchecked student responses using Spellchecker based on Peter Norvig's method. The words *perfom* and *forsgiht* from the original sentence are changed to *perform* and *foresight* by insertion of letter *r* and *e* respectively. Also, the word *challanges* is altered to *challenges* by replacement of the letter *a* with *e*.

3.1.2 Case Folding

Transforming the case of the text or making the entire text lower or upper case is called case folding. Majorly, lowercase is favored [39]. Though the method is straightforward, sometimes it gets tricky when “US” is converted to “us”, which alters the meaning completely. Hence, the usage is restricted when the minute details are essential. Case folding the student's answer from Table 3.1 is provided in Table 3.3.

a variable is a location in the computer's memory where a value can be stored for use by a program. each variable has a name, a value, a type and a size.

Table 3.3: Case lowering the entire student answer.

3.1.3 Tokenization

Processing the whole text/sentence might be difficult sometimes. So, the running text is broken down into sentences and sentences into words using punctuations or whitespaces as delimiters. This process is called tokenization, and separated sentences, phrases, or words are called tokens. The complication arises in the text containing phrases like *North Rhine-Westphalia*, *doesn't*, and *Mr. Tony* are single entities, yet tokenization will split them into individual words. Thus, a rule-based or ML approach is preferred to deal with such situations. Tokenizing the student answer after case folding is presented in Table 3.4.

'a', 'variable', 'is', 'a', 'location', 'in', 'the', 'computer', ''s', 'memory', 'where', 'a', 'value', 'can', 'be', 'stored', 'for', 'use', 'by', 'a', 'program', '.', 'each', 'variable', 'has', 'a', 'name', ',', 'a', 'value', ',', 'a', 'type', 'and', 'a', 'size', '.'

Table 3.4: Tokenizing the student answer after case lowering.

3.1.4 Part-of-Speech (POS) Tagging

POS indicates the function of the word in terms of meaning and grammatically within the sentence. POS tagging annotates each word with its respective classes, such as nouns, verbs, adverbs, adjectives, pronouns, and conjunction. These classes and all POS tags used in a corpus are called a tagset. POS tagger is vital in chunking and keyword extraction. Table 3.5 is an example of a student answer with its annotated POS tag after case folding and tokenization. Tagset with its description is provided in the Appendix A.

'a **(DT)**', 'variable **(NN)**', 'is **(VBZ)**', 'a **(DT)**', 'location **(NN)**', 'in **(IN)**', 'the **(DT)**', 'computer **(NN)**', '' **(PUNCT)**', 's **(DT)**', 'memory **(NN)**', 'where **(WRB)**', 'a **(DT)**', 'value **(NN)**', 'can **(MD)**', 'be **(VB)**', 'stored **(VBN)**', 'for **(IN)**', 'use **(NN)**', 'by **(IN)**', 'a **(DT)**', 'program **(NN)**', '. **(PUNCT)**', 'each **(DT)**', 'variable **(NN)**', 'has **(VBZ)**', 'a **(DT)**', 'name **(NN)**', ', **(PUNCT)**', 'a **(DT)**', 'value **(NN)**', ', **(PUNCT)**', 'a **(DT)**', 'type **(NN)**', 'and **(CC)**', 'a **(DT)**', 'size **(NN)**', '. **(PUNCT)**'

Table 3.5: POS tagging the student answer after tokenizing. The word with the corresponding POS tag is given in parentheses.

3.1.5 Dependency Parsing

The linguistic relation between words in a sentence is described by dependency parsing, as shown in Figure 3.1. As the sentence becomes longer, the tree becomes longer and harder to navigate, which is the reason that instead of the student answer, the reference answer is provided as an example.

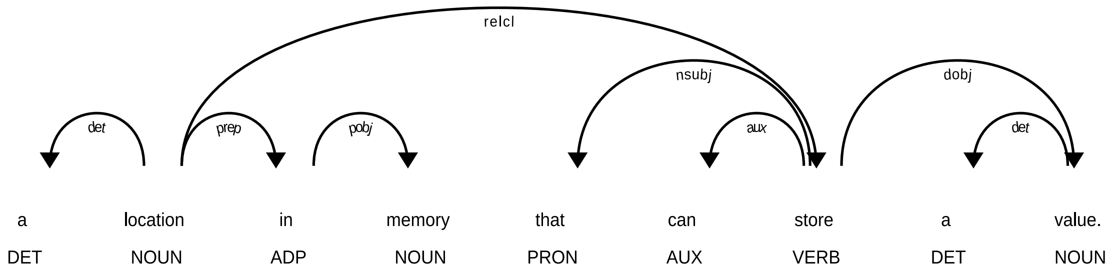


Figure 3.1: Dependency tree for the reference answer from Table 3.1 after POS tagging.

Directed arcs express the relationships between each token in the sentence. The root of the tree is *location* having the children determiner (det) *a*, preposition modifier (prep) *in*, an object of the preposition (pobj) *memory* and relative clause modifier (relcl) *store*. The sentence “a location in memory store” formed from the root node and its children comprises the sentence’s basic meaning. The children of *store* direct object (dobj) *value* make the other part of the sentence which is “store value”.

3.1.6 Punctuation and Stop Words Removal

Prepositions, conjunctions, articles, and pronouns are frequently used to connect different parts of the sentence, called stop words. Punctuation and stop words can be removed from the text as they are insignificant. Table 3.6 depicts the punctuation and stop word removed from student answer after POS tagging.

variable is location computer memory where value can be stored use by program variable has name value type and size

Table 3.6: Punctuation and stop words removed from the student answer after POS tagging. Words such as *a*, *in*, *the* and *for* are removed along with the punctuation . and , from the student answer compared to the sentence in Table 3.5.

3.1.7 Question Demoting

Sometimes students repeat the question in their answers to score. Question demoting removes all the words that appear in the question from answers to restrain repeating the question. An example of a question demoting student answer after punctuation and stop word removal is presented in Table 3.7.

```
location computer memory where value can be stored use by program has name
value type and size
```

Table 3.7: Question demoted from the student answer. Words *variable* and *is* are removed from the student answer compared to the sentence in Table 3.6.

3.1.8 Stemming and Lemmatization

For grammatical reasons, documents contain different word forms, such as formulate, formulated, and formulating but of the same meaning. Stemming and lemmatization convert these words into their base or standard form. Stemming uses heuristics to remove the ending of the term to form a stem. Lemmatization reduces the words to their base form, called a lemma. Table 3.8 presents different words with their stem and lemma.

Word	**Stem**	**Lemma**
Information (noun)	Inform	Information
Informative (adjective)	Inform	Informative
Informal (adjective)	Inform	Informal
Informer (noun)	Inform	Informer
Informers (noun)	Inform	Informer

Table 3.8: Words with their stem and lemma.

From Table 3.8, it can be inferred that stemming maps all four words to the same stem *inform* where the original word's meaning is lost. In addition, from the stem, the actual word can not be deduced. In contrast, lemmatization preserves the type of word. Therefore, lemmatization is preferred to stemming [87]. An example of lemmatization of student answer after question demoting is provided in Table 3.9.

location computer memory where value can be **store** use by program have name value type and size

Table 3.9: Lemmatizing the student answer after question demoting. Word *stored* is transformed into its base form *store* using lemmatization.

3.2 Features for ASAG

The initial requirement for dealing with a text is to identify the optimal number of features. The process of mapping textual data to real-valued numbers or vectors of fixed length is called features. Features are the low-dimensional representation of the raw data without the loss of its information. Also, the machine learning algorithm requires fixed-length vectors as it is not designed to work on the text. The features of ASAG can be classified into two types based on the working, namely response-based and corpus-based.

3.2.1 Response-based

A clear-cut and easy-to-implement method does not require much computation time and memory. Response-based method extracts the features from the provided text/sentences without the aid of external resources. Some of the popularly used response-based methods are discussed below.

Bag-of-Words (BOW)

A vector representation of text that depicts the occurrence of each unique word in a text/sentence is called BOW [38]. It keeps track of word counts and disregards grammatical details and word order. The unique words from the student and reference answer are presented in Table 3.10.

‘and’, ‘be’, ‘by’, ‘can’, ‘computer’, ‘has’, ‘location’, ‘memory’, ‘name’, ‘program’, ‘size’, ‘store’, ‘that’, ‘type’, ‘use’, ‘value’, ‘where’

Table 3.10: BOW from the student and reference answer.

Figure 3.2 presents the vector representation of reference and student answer from Table 3.1. The first and second row represents the vector based on the occurrence of each unique word (x-axis) from the reference and student answer, respectively. According to BOW, the reference answer and student response have 50% similarity, but the student response is more similar to the reference answer. This is due to the matching of words instead of having the sentence semantic in consideration. Therefore, the sentence can be preprocessed by case folding, punctuation and stop words removal, question demoting, and lemmatization before processing using BOW since it does not regard word order and grammar.

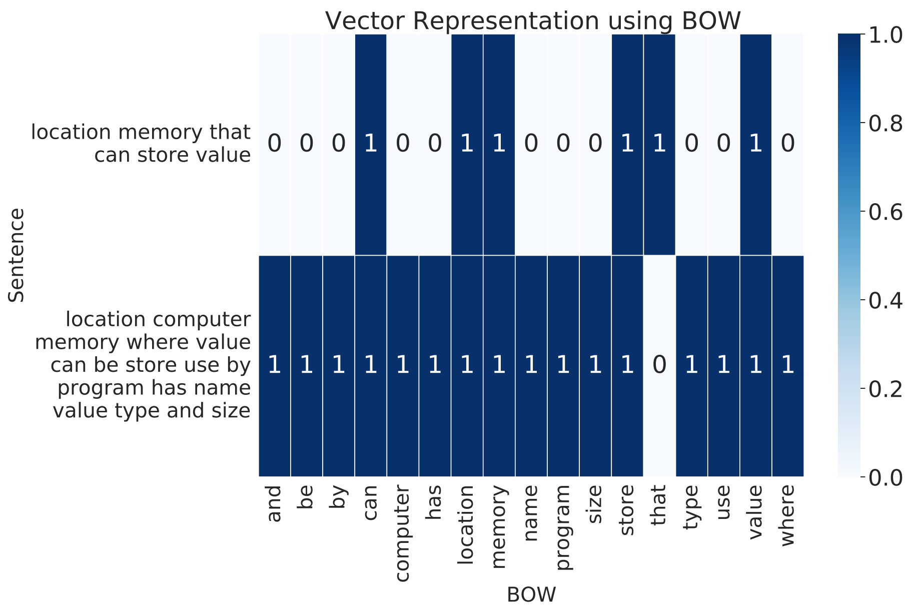


Figure 3.2: Vector representation using BOW in the heatmap for reference and student answer from Table 3.1 after all preprocessing discussed earlier.

N-grams

A vector representation of text that represents the occurrence of each unique N-word sequence in a text/sentence is called an N-gram, an extension of BOW. So, N-grams make the sentence more understandable than BOW [44]. A 2-gram (bigram) is a two-word sequence like *computer memory* and *good morning*. A 3-gram (trigram) is a three-word sequence like *support vector machines* and *full stack developer*. The bigram for the student and reference answer is listed in Table 3.11. Similar to BOW, N-gram disregards the grammar but not the word order. Therefore, the sentence can be preprocessed by case folding, punctuation and stop words removal, question demoting, and lemmatization before processing using N-gram since the filler words can be removed between the notable words, which makes N-gram robust.

(location, memory), (memory, that), (that, can), (can, store), (store, value), (location, computer), (computer, memory), (memory, where), (where, value), (value, can), (can, be), (be, store), (store, use), (use, by), (by, program), (program, has), (has, name), (name, value), (value, type), (type, and), (and, size)

Table 3.11: List of bigram from the student and reference answer

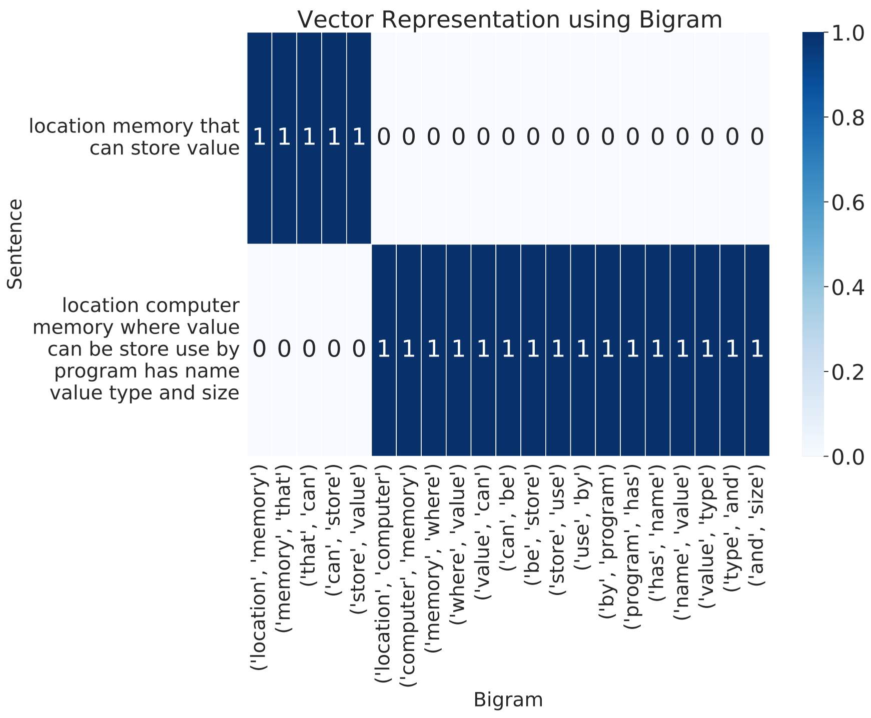


Figure 3.3: Vector representation using Bigram in the heatmap for reference answer and student answer from Table 3.1 after all preprocessing discussed earlier.

Figure 3.3 presents the vector representation of reference and student answer from Table 3.1. The first and second row represents the vector based on the occurrence of the bigram sequence (x-axis) for reference and student answer, respectively. According to bigram, there is no similarity between the reference and student answers as the bigram pairs of reference answers are not presented in the student response or vice-versa.

Chunking

The entity or phrase extraction from unstructured text is called chunking [69]. It is performed upon POS tagging, highlighting the noun group, verb group, and verb [88]. A regex-based grammar is defined initially to extract specific phrases, such as verb or noun phrases. Therefore, when the sentence/text is to be chunked, it is tokenized, followed by POS tagging. These POS tags are parsed with the defined regex-based grammar to produce the required phrase [88]. Figure 3.4 depicts a sentence's noun phrase-based chunking tree.

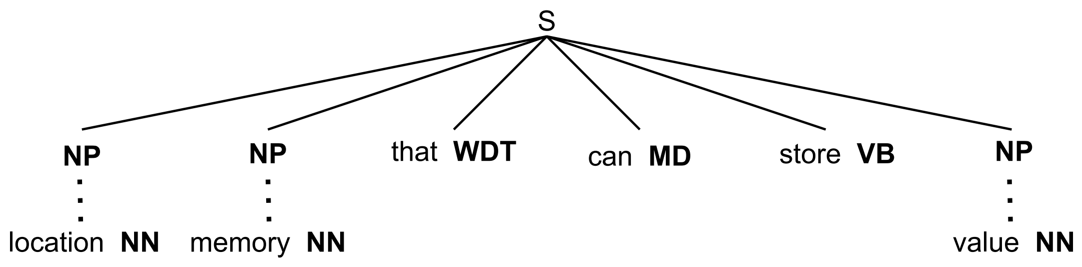


Figure 3.4: Chunking tree based on noun phrase for reference answer from Table 3.1. The image contains noun phrases with only nouns such as *location*, *memory*, and *value* of the reference answer, which are almost the expected keywords in the student answer. The regex grammar used for this noun phrase is "NP: {<DT>?<JJ>*<NN>}" which means one or zero determiners, followed by zero or more adjectives and nouns.

Bilingual Evaluation Understudy (BLEU) Score

BLEU score is a metric used to assess machine translation. It evaluates the quality of machine-generated/translated text by comparing it with a manual translator/evaluator reference text. BLEU score is computed using four steps.

1. The number of N-grams from the predicted translation that appears in the reference translation to the number of N-grams in the expected translation determines the N-gram precision expressed as,

$$p_n = \frac{\sum_{\text{N-gram} \in \text{c}} count_{clip}(\text{N-gram})}{\sum_{\text{N-gram} \in \text{c}} count(\text{N-gram})} \tag{3.1}$$

   where, $count_{clip}$(N-gram) is the number of N-grams from the predicted translation that appears in the reference translation, $count$(N-gram) is the number of N-grams in the predicted translation, and p_n is the N-gram precision.

2. Geometric average precision scores are calculated from the N-gram precision expressed in Equation 3.1.

$$\text{Geometric average precision } = \prod_{n=1}^{N} p_n^{w_n} \tag{3.2}$$

   where, N is N-gram, $N = 4$ and w_n is the weight, $w_n = \frac{N}{4}$.

3. Brevity penalty that penalizes the shorter sentences.

$$\text{Brevity penalty} = \begin{cases} 1, & \text{if } c > r \\ e^{(1-\frac{r}{c})}, & \text{if } c <= r \end{cases} \tag{3.3}$$

   where, c is the number of words in the predicted translation, and r is the number of words in the reference translation.

4. BLEU score is the product of Equation 3.2 and Equation 3.3.

$$\text{BLEU score} = \text{Brevity penalty} * \text{Geometric Average Precision} \tag{3.4}$$

Even though BLEU is used as an evaluation metric for machine translation, it has been used in auto grading to compute similarities [57, 101]. According to Equation 3.4, the BLEU score for the answer based on the reference answer from Table 3.1 is zero. Except for unigram, other N-gram yields zero, which causes N-gram precision and geometric average precision to zero. Hence, the BLEU score is also zero since its a product of brevity penalty and geometric average precision.

Length Ratio

The ratio of the number of tokens in the student answer to the reference answer is called length ratio [89]. This restrains the student's response length and computes its degree of significance.

$$T_{ans} = \{W_i\}_{i=0}^{N} \tag{3.5}$$

where, T_{ans} is a set of tokens for the student answer, W_i is the individual token in the student answer, and N is the number of tokens in the student's answer.

$$T_{ref} = \{W_j\}_{j=0}^{M} \tag{3.6}$$

where, T_{ref} is a set of tokens for the reference answer, W_j is the individual token in the reference answer, and M is the number of tokens in the reference answer.

$$\text{Length ratio} = \frac{N}{M} \tag{3.7}$$

According to Equation 3.7, the length ratio for the student answer and reference answer from Table 3.1 is 2.83.

3.2.2 Corpus-based

The corpus-based method extracts the features from the provided text/sentences with the benefit of external resources. This resource can be other text/sentences in the dataset or domain-related corpus like Wikipedia. The corpus-based method requires much computation time and memory, especially during training on the corpus. Some of the popularly used corpus-based methods are discussed below.

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is a technique for converting words into vectors with moderate semantic information, providing weight to uncommon words. It gives way to associating important words in a document which eliminates the incompetence of BOW. This method is majorly used for keyword extraction and information retrieval [99, 47], yet it has been used for ASAG [63]. TF-IDF comprises two parts they are

1. **Term Frequency (TF)** is the number of word occurrences in a document. TF is the ratio of the recurrence of a word to the total number of words in a document.

$$\text{TF}(t, d) = \frac{\text{number of occurrence of } t \text{ in } d}{\text{total number of all words in } d} \tag{3.8}$$

   where, t is the word under consideration and d is the document.

2. **Inverse Document Frequency (IDF)** weights the uncommon words. Words like *is* occur more often in the documents and obtain a high score in TF. Nevertheless, *is* consists of insignificant task information. Thus, it is desirable to possess a low score for common words like *is* which is accomplished by IDF.

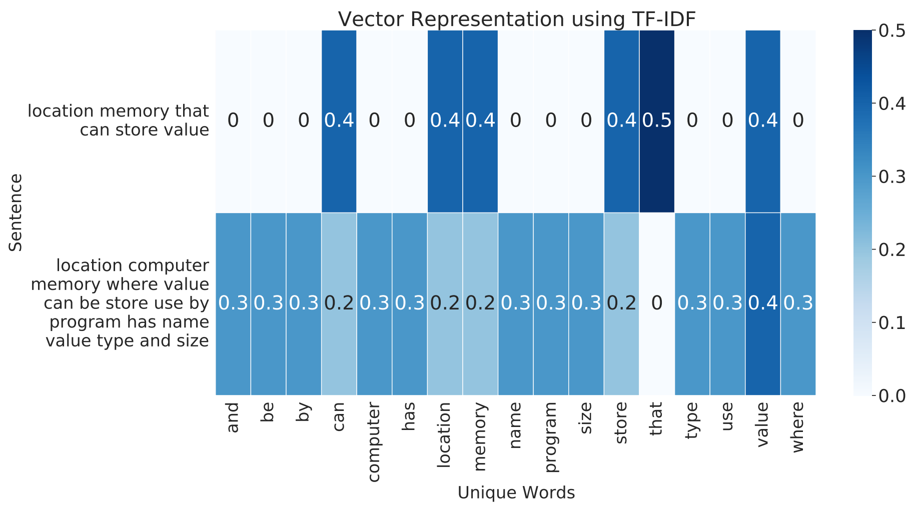
$$\text{IDF}(t) = log(\frac{N}{df_t}) \tag{3.9}$$

where, N is the number of documents and df_t number of the document containing t.

Therefore, TF-IDF is given by the product of Equation 3.8 and 3.9,

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) * \text{IDF}(t) \tag{3.10}$$

Figure 3.5 presents the vector representation of reference and student answer from Table 3.1 using TF-IDF. The first and second row represents the vector based on the TF-IDF of each unique word (x-axis, considering given sentence as documents) for reference and student answer, respectively. According to TF-IDF, the reference and student answer has 40% similarity, but the response matches the reference answer precisely. Hence, when the TF-IDF is trained on relevant domain corpus like Wikipedia (e.g., related to computer science) instead of considering the sentence from the dataset itself, the score will increase [89].



Sentence	and	be	by	can	computer	has	location	memory	name	program	size	store	that	type	use	value	where
location memory that can store value	0	0	0	0.4	0	0	0.4	0.4	0	0	0	0.4	0.5	0	0	0.4	0
location computer memory where value can be store use by program has name value type and size	0.3	0.3	0.3	0.2	0.3	0.3	0.2	0.2	0.3	0.3	0.3	0.2	0	0.3	0.3	0.4	0.3

Figure 3.5: Vector representation using TF-IDF in the heatmap for reference and student answer from Table 3.1 after all preprocessing discussed earlier.

Word Embedding

Word embedding learns a vector representation for each word of fixed length from a corpus. It is competent to capture the context of a phrase with syntactic and semantic correlation with other terms in a document. This is achieved by training a neural network on several corpora. Training on domain-related corpus yields better results comparatively [60].

The BOW, N-gram, and TF-IDF capture the word frequency, whereas word embedding performs word prediction, which is significant while comparing two sentences. The student and reference answer from Table 3.1 hardly have different meanings, yet the BOW and TF-IDF provided a mediocre performance. However, the word embedding learns to cluster similar words in context and semantics, as shown in Figure 3.6.

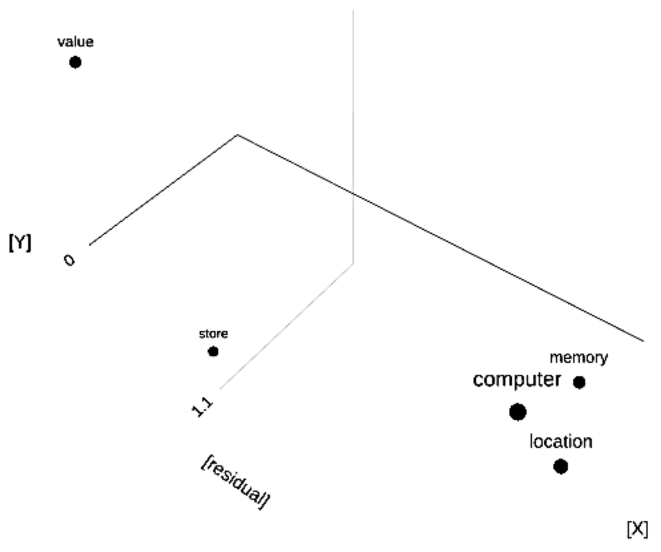


Figure 3.6: Word embedding of major keywords of reference answer from Table 3.1 is plotted as a 3D graph using a word embedding demo from Carnegie Mellon University-computer science department [23]. The words *computer*, *memory*, and *location* are clustered together since it is related to a certain degree. Whereas *store* and *value* are in different clusters indicating different entities or not related.

Widely used efficient word embedding models are Word2Vec, Glove, and FastText [9, 76, 10]. The research work by [60] has compared the performance of the embedding models mentioned earlier on the ASAG dataset and found FastText to be a better model. Therefore, this research work also uses the FastText word embedding model. FastText is the extension of word2vec where instead of learning vector for each word, it learns vector for n-gram of characters. Thus, it grasps the semantics of shorter words, prefixes, and suffixes, which even process out-of-vocabulary words compared to the other two. According to FastText, reference and student answers have 69% similarity.

Sentence Embedding using S-BERT

Sentence embedding learns a vector representation for each sentence of fixed length from a corpus. Word embedding performs better in correlating the words but does not infer from the words. Sentence embedding is competent in capturing the context of a sentence with syntactic and semantic correlation with the sentence in a document. Therefore, it can grasp the entire document's intention, context, and other nuances. Among different sentence embedding models available, this research work intends to use S-BERT as it seems to perform well comparatively [56, 27, 7]. S-BERT stands for Sentence-Bidirectional Encoder Representations from Transformers.

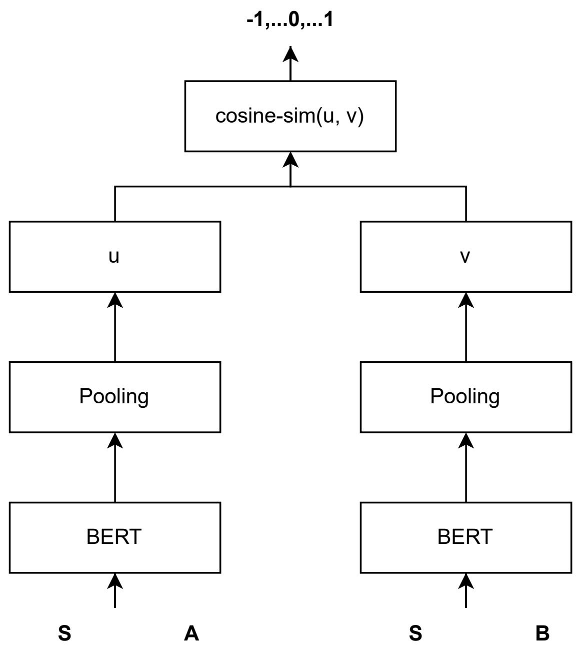


Figure 3.7: S-BERT architecture at inference computing cosine similarity between two sentences. Reproduced from [p. 5][77].

Figure 3.7 depicts the S-BERT architecture that processes two sentences simultaneously. So, it is also called a twin network. A pooling layer has been stacked to the sentence, creating a fixed-size vector representation for the sentence of varying length. The utilization of CLS tokens is a popularly used pooling strategy. To fine-tune the weights that produce significant embedding, siamese and triplet networks were adopted [83].

Figure 3.8 presents an example of pre-trained S-BERT learning the reference answer. The thickness of the flow/edges (purple-colored thread) indicates the degree of association

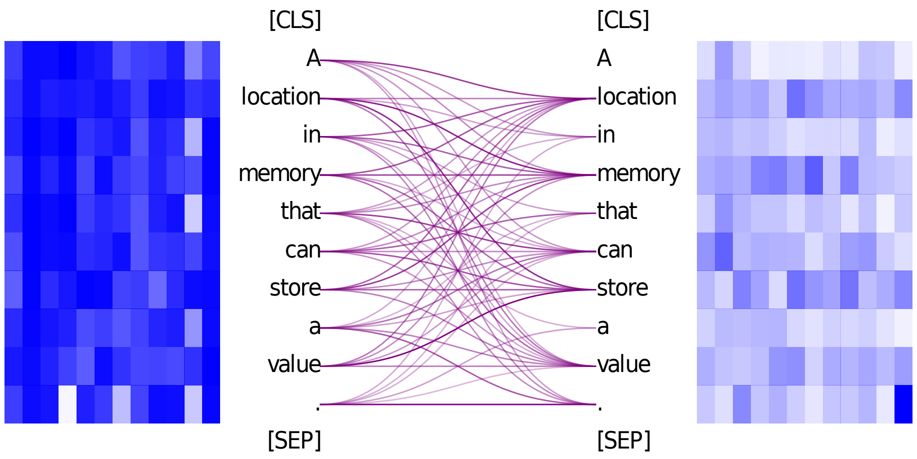


Figure 3.8: S-BERT learning reference answer from Table 3.1 without any preprocessing. Either side has heatmap columns of the layers/heads learned. Reproduced from [43].

between the words. The word *location* is highly associated with *memory* and *store*. Also, *value* is highly associated with *store*. The following words are the keywords necessary to consider in student answers. According to S-BERT, the reference and student answer has 75% similarity.

3.2.3 Cosine Similarity

Features such as BOW, N-gram, TF-IDF, word, and sentence embedding produce vectors. To compute the similarity between those vectors, the cosine angle between those vectors is calculated, called cosine similarity, expressed in Equation 3.11.

$$s(u_i, v_i) = cos(\theta) = \frac{u * v}{\|u\| \, \|v\|} = \frac{\sum_{i=1}^{n} u_i * v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}} \tag{3.11}$$

where, u_i is the vector-1, v_i is the vector-2, n is the length of the vectors and $s(u_i, v_i)$ is the cosine similarity between vector u_i and v_i. The similarity value varies from -1 to 1, with three possible outcomes.

- **Similar Vectors :** The value is 1, and the angle is 0.
- **Unrelated Vectors :** The value is 0, and the angle is 90.
- **Opposite Vectors :** The value is -1, and the angle is 180.

3.2.4 Linear Sum Assignment

The Linear Sum Assignment Problem (LSAP) is designed to achieve optimal work-task pair [12]. This is one of the major problems in combinatorial optimization and linear programming since it employs finding minimal workers to finish the task or limited time to finish the tasks by allocating optimal workers.

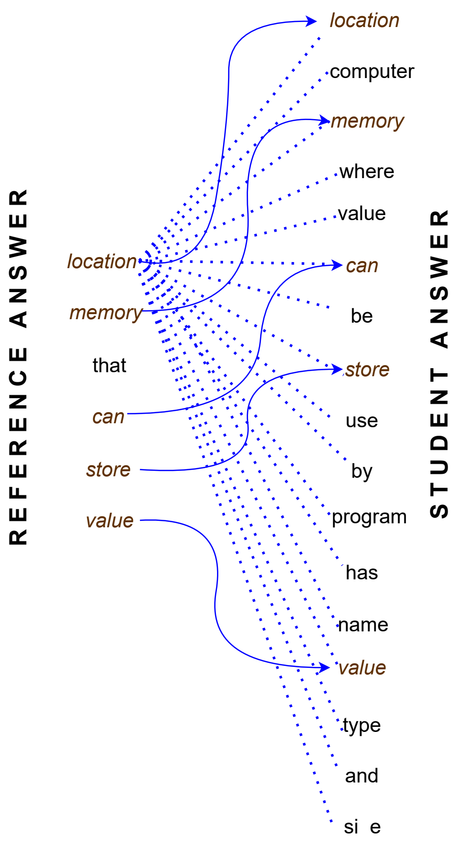


Figure 3.9: Working of linear sum assignment on ASAG task. Each token of the reference answer is associated with the tokens of the student answer (indicated using dotted lines) to find similar token pairs in brown color (indicated using arrow-headed lines).

Consider a cost matrix C[i,j] of size n x m, which needs to obtain the sum of each row to each column pair as minimum [15]. The LSAP is expressed as,

$$\text{LSAP}(i,j) = min(\sum_i \sum_j C_{i,j} X_{i,j}) \tag{3.12}$$

where, X is a boolean matrix $X[i,j] = 1$ iff the row i matches the column j.

The word level working of LSAP given the tokens of reference and student answer is provided in Figure 3.9. For ASAG, the LSAP is calculated as follows,

1. Obtain word embedding u for the list of words in the student answer.
2. Obtain word embedding v for the list of words in reference answer.
3. Build a cost matrix $C_{i,j}$ as a cosine similarity between u_i and v_j

$$C_{i,j} = s(u_i, v_j) \tag{3.13}$$

$$\text{LSAP}(i,j) = max(\sum_i \sum_j C_{i,j} X_{i,j}) \tag{3.14}$$

4. Obtain the optimal row and column vector pairs corresponding to the words using LSAP from Equation 3.14. Maximizing LSAP captures the highest similarity value corresponding to similar words.
5. Finally, these values are summed up and normalized using the length of the reference answer tokens T_{ref} to get the final value. Also, these values can be filtered using a threshold based on performance.

3.3 Machine Learning (ML) Models

The short answer scoring is considered chiefly a regression task [89, 63, 60]. In contrast, few have regarded it as a classification task [35, 94]. This research work prefers ML models that work for both tasks. The subsequent ML models used in this research work for training on the features extracted from the short answer for grading the same are explained.

3.3.1 Ridge Regression

Ridge regression is a form of linear regression method in ML which reduces overfitting. Ridge regression is favored when there are highly correlated multiple variables. Linear regression is expressed as

$$Y = WX + B \tag{3.15}$$

where, Y is the predicted value, X is the input value or feature vector, W is the weight matrix and B is the bias. The loss is computed by,

$$\text{Loss} = \sum_{i=1}^{n} (\hat{y}_i - (w_i x_i + b))^2 \tag{3.16}$$

where, $\hat{y}$ is the actual value. When there is a penalty for the loss in the Equation 3.16,

$$\text{RSS}_{\text{ridge}}(w, b) = \sum_{i=1}^{n} (\hat{y}_i - (w_i x_i + b))^2 + \alpha \sum_{j=1}^{p} w_j^2 \tag{3.17}$$

Ridge regression penalizes the coefficients of the variables, which prevents overfitting. The penalty added is called the L2 penalty or regularisation term, which shrinks the coefficient size. Also, this helps reduce overfitting, notably with smaller datasets or many variables. Therefore, ridge regression is a model-tuning method employed when the data suffer from multicollinearity. Also, ridge regression is used in this research to compare the performance with the work [89, 60] that uses the same.

3.3.2 Support Vector Machines (SVM)

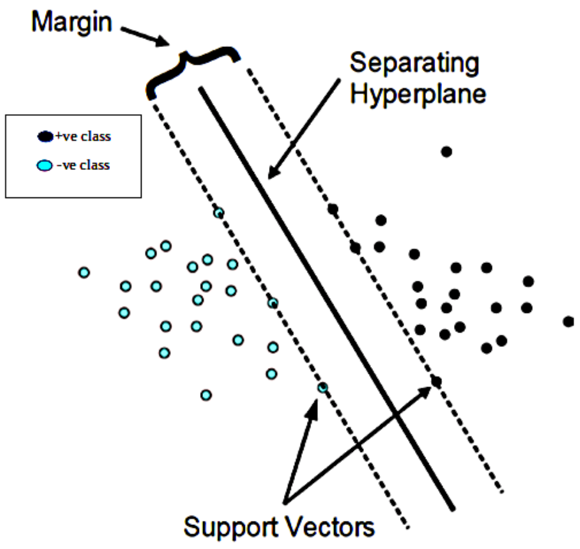


Figure 3.10: SVM as a classifier. Reproduced from [26].

SVM can be used as a classifier as well as a regressor, yet the working principle of SVM for both remains the same [14]. Figure 3.10 illustrates SVM as a classifier that separates different classes by fitting an optimal hyperplane with maximum margin [52]. The regressor fits the hyperplane with the maximum number of points within the margin/decision boundary [53]. The margin/decision boundary is the shortest distance between the hyperplane and observations. The target is to maximize the space between the margins to fit new observations with higher confidence. The observations through which the margins are computed on both sides of the hyperplane are called support vectors. Figure 3.10 depicts the working on linear data. Nevertheless, SVM works on non-linear data by converting it into linear data while projecting from a lower to a higher dimension. Working with higher dimensional data increases the computational burden. SVM avoids the computation overhead using a kernel trick. Instead of projecting the data to a higher dimension, the kernel trick is computing a dot product. Some popular kernel functions in SVM are linear, polynomial, non-linear, Radial Basis Functions (RBF), and sigmoid. These kernel functions are associated with SVM and can be used for other ML models, such as logistic regression.

3.3.3 Random Forest

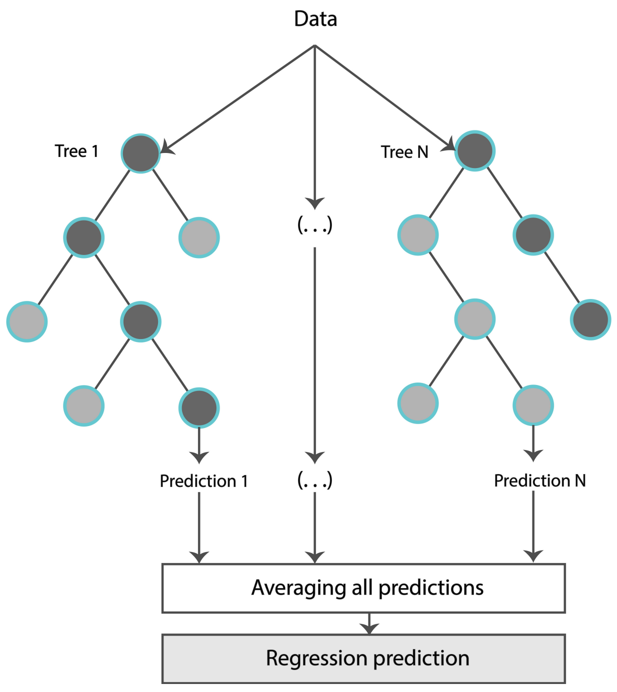


Figure 3.11: Random forest as regressor. Reproduced from [11, p. 5].

Random forest is an ensemble method that follows a bagging approach applicable to classification and regression tasks. The random forest contains an “N” number of decision trees as learning models as presented in Figure 3.11. Random forest trains these decision trees on different subsets of data based on sampling (including replacements). This is called bootstrapping. Each decision tree holds high variance, but when combined parallel, the variance is low as every decision tree is perfectly trained on that particular subset of data. Hence the output is not based on a single decision tree but on "N" decision trees. The leaf nodes of these decision trees consist of their prediction. In the case of the classification task, the final output is achieved by selecting the class of significant votes/predictions. In the regression task, the outcome is the mean of all the leaf node predictions. This is called Aggregation. The aggregation process in random forest improves its accuracy and reduces overfitting. Therefore, the random forest can work with small to large datasets with an increasing number of trees. Also, the random forest can work with missing data by computing values for them.

3.4 Active Learning

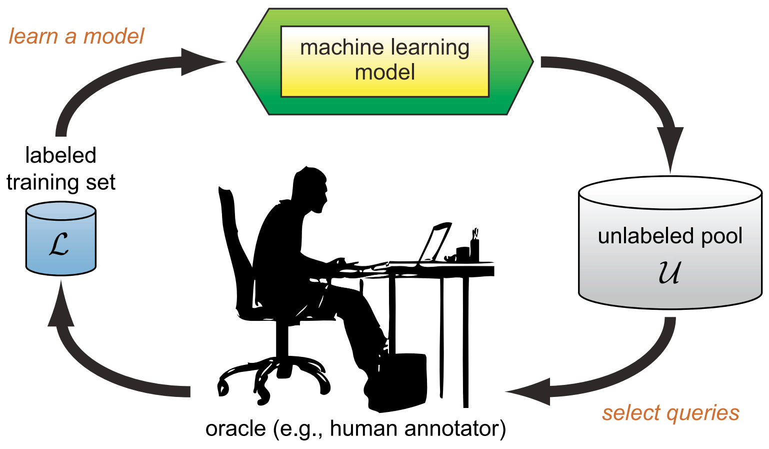


Figure 3.12: Active learning cycle. Reproduced from [84, p. 5].

Active learning seeks to reduce the labeling congestion by querying the instances from unlabeled data to annotate by a human annotator/oracle [84]. In this research work, the oracle is the grader/evaluator. The active learner aims to achieve high accuracy using as much labeled data as possible by minimizing the cost and time of labeling the data. Active learning can be employed where the data are abundant, but labels are expensive, time-consuming, and sparse [93]. Figure 3.12 illustrates the idea of an active learning cycle in which the machine learning model is initially trained on the available labeled data. To achieve more performance in prediction, the model queries oracle and learns from the annotation along with the previously annotated data. The querying continues till desired/requested performance is obtained. The instances presented in the query to the oracle are sampled from the pool of unlabeled data. The main concern for the sampling is to obtain an adequate strategy to select the better informative instance from the pool. There are different query strategies: uncertainty sampling, query-by-committee, expected model change, error reduction, variance reduction, and density-weighted methods. Among them, uncertainty sampling is popularly used due to its performance [84, 48]. However, these strategies work for the classification task, not for regression. Therefore, this research work has devised a query strategy that works both for classification and regression.

Uncertainty Sampling

A powerful querying strategy for active learning is uncertainty sampling [48]. Uncertainty sampling queries oracle with the most uncertain instance from the unlabeled data. Uncertainty sampling is expressed as,

$$U(x) = argmin_{x \in X_u} P(\hat{y}|x;\theta) \tag{3.18}$$

where, X_u is the pool of unlabeled data, x is the instance from the pool of unlabeled data, $\hat{y}$ is the class label of high posterior probability for model θ and $U(x)$ is the sampled instance based on uncertainty based on model's belief that x will be mislabeled. Therefore, this strategy cannot work with continuous values (regression).

Intensity Sampling

While working with uncertainty sampling (or a similar strategy), it cannot obtain the data's structure, which can lead to suboptimal queries. Hence, this research work has devised a custom query method based on information density expressed as,

$$I(x) = argmin_{x \in X_u} \frac{1}{|X_u|} \sum_{x' \in X_u} sim(x, x') \tag{3.19}$$

where, x' is the other instances from the pool of unlabeled data, $sim(x, x')$ is a cosine similarity function, and $I(x)$ is the sampled instance based on information density. Information density increases with the similarity between the given instance and the rest of the data. Thus, this strategy can work with both discrete (classification) and continuous values (regression).

4
Methodology

This chapter focus on the methodology implemented to grade short answer using rubrics. Also, the evaluation metrics used to assess the technique are discussed. The methodology is generic that can be applied to any short answer grading. The methodology is fragmented into the concept, technical approach, and architecture.

4.1 Concept

The main idea of this methodology is to provide both summative (score) and formative (feedback) assessments of short answers as expressed in Equation 2.4. Having reference answers does not suffice to provide feedback during grading. Hence, the question should hold rubrics instead of reference answers. The similarity is computed between each rubric and student answer per question. Words in the student answer and rubrics are represented as vectors using different feature extraction methods and are compared using cosine similarities. This allows for a higher coarse approach, where the answers can be graded based on the rubrics they comprise. In addition, the rubrics-based similarity could be an appropriate justification for the grade provided to a student along with the feedback that was missing to be included. Before extracting features, each student's answer and rubrics are preprocessed using different methods such as case folding, spell check, and stop word removal, as mentioned in Chapter 3, Section 3.1.

4.2 Technical Approach

This methodology is a two-step approach, first is to determine the potential features and model for short answer grading using reference answers with the benchmark dataset. The second is to use these features and model for grading the short answer using rubrics. The benchmark dataset is provided by Mohler and Mihalcea [63].

The first step is to experiment on the Mohler dataset to determine the significant features and model. The idea is to have the least correlated features, which are determined by p-value (feature selection). The features indented to be used are BOW, N-gram, chunking, length ratio, word, and sentence embedding. These features are selected based on their performance in previous research works. The selected features are fed into the model wrapped by active learning, which selects the instance among the provided features to be labeled using an intensity sampling query strategy. The model tested in this research work are random forest, SVM, and ridge regression.

The determined model and features are used for grading short answers from Autonomous Mobile Robot (AMR) course using rubrics. In the previous step, the features are labeled using scores, whereas in this step, it uses rubrics. The scores are calculated based on the number of rubric presented in the answer, and a missing rubric is provided as feedback.

4.3 Architecture

As the approach is a two-step process, so does the architecture, reference answer-based, and rubrics-based.

4.3.1 Reference Answer-based

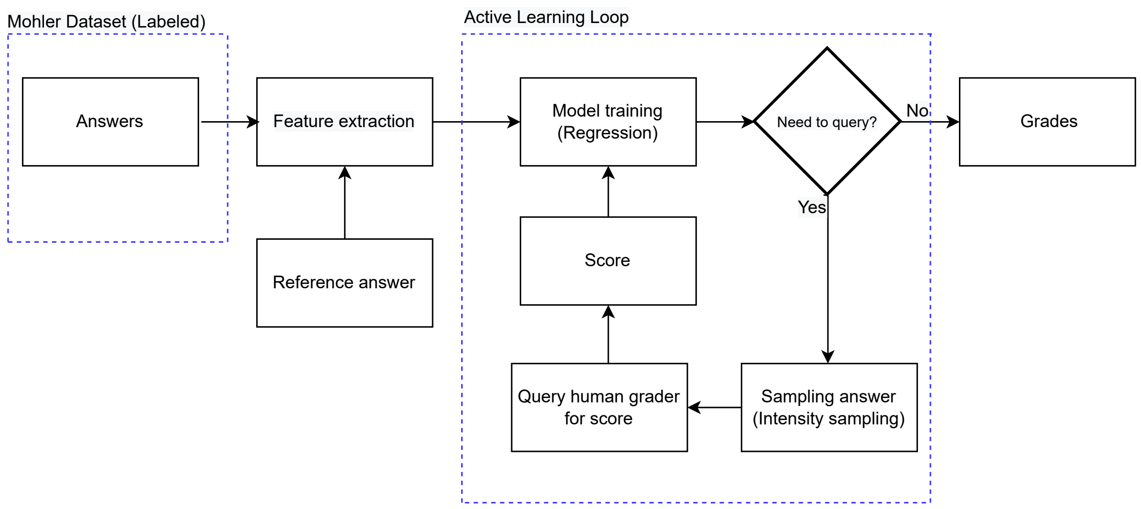


Figure 4.1: Pipeline for reference answer-based short answer grading. The dataset used in this is from Mohler and Mihalcea [63].

The complete pipeline of the approach proposed for the reference answer-based short answer grading is depicted in Figure 4.1. The answers from the Mohler dataset are extracted features after certain preprocessing suitable to that particular method. Among these features, potential features are selected for training the model. The model is regression-based, wrapped using active learning that queries a single instance to the grader to label the score from the provided feature set using intensity sampling. The model continues to train until it reaches 25% of the total data being queried, after which it predicts the score based on the training. The pseudo algorithm for the method, along with the query strategy, is presented below.

Algorithm 1: Reference Answer-based Grading

```
Input: Question, Students answer, Reference answer
Output: Grades

Function intensityquery($learner$, $pool$):
    intensity $\leftarrow information\_density(pool, cosine)$
    query_data $\leftarrow pool[argmax(intensity)]$
    return query_data

Function Main:
    /* features for the data are obtained as array */
    features $\leftarrow potential\_features(Qn, Studans, Refans)$
    /* initially get a single random index to initiate the model */
    initial $\leftarrow Features(random\_index)$
    pool $\leftarrow Features.remove(initial)$
    /* initiating active learner */
    learner $\leftarrow Activelearner(model, intensityquery, initial)$
    counter $\leftarrow 0$
    /* Query for 25% of total data */
    for $counter < 0.25 * len(features)$ do
        query_data $\leftarrow learner.query(pool)$
        score $\leftarrow learner.teach(query\_data)$
        pool $\leftarrow pool.remove(query\_data)$
        counter $+= 1$
    /* predict remaining answer's grade and save the model for later */
    grades $\leftarrow learner.predict(pool)$
    model $\leftarrow learner.save()$
```

The features are essential as they impact the efficiency of the approach. The features extracted from the short answer should depict the raw data in the low dimensional space. Therefore, extracting the features and selecting the significant among them is crucial. Figure 4.2 presents the feature extraction and selection process to yield the feature array

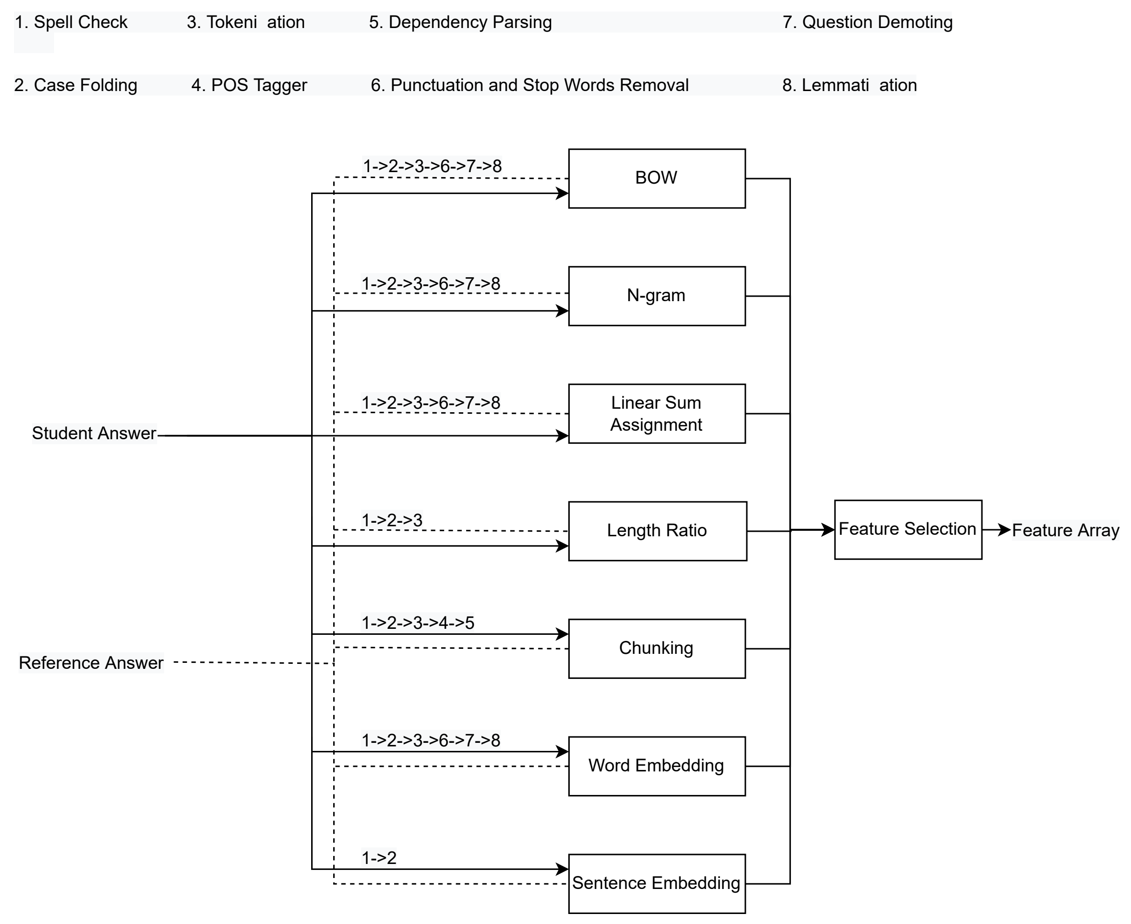


Figure 4.2: Feature extraction and selection pipeline for reference answer-based short answer grading.

for reference answer-based short answer grading. Features such as BOW, N-gram, linear sum assignment, length ratio, chunking, word, and sentence embedding are extracted from student and reference answers. Each feature requires a different preprocessing set, denoted by the number according to the order above the flow line in Figure 4.2. Among the extracted features, potential features are selected for the model training. The potential features are obtained as least correlated by fitting a p-value lesser than 0.5. Each preprocessing and feature extraction method is discussed in detail in Chapter 3. The word embedding used in this research work is FastText trained on domain-related corpus by Metzler [60]. Sentence embedding from Huggingface with the model *all-mpnet-base-v2*, which is trained on a large and varied dataset of over 1 billion sentence training pairs, is employed. The pseudo-code for chunking and linear sum assignment is given below. The remaining features, BOW, N-gram, and length ratio, are popularly used methods [89], which are available publicly.

```
Algorithm 2: Feature Extraction
    Input: Question, Students answer, Reference answer
    Output: Array

    Function Chunking(sentence):
        /* regex expression for noun phrase */
        chunks ← parser(NP: <DT>? <JJ> * <NN>)
        token ← tokenization(sentence)
        tags ← pos_tag(token)
        chunk_tree ← chunks.parse(tags)
        for token in tree.pos() do
            if token is "NP" then
                phrase ← token
        return phrase

    Function Linearsumassignment(refans, studans, threshold):
        studans_we ← word_embedding(studans)
        refans_we ← word_embedding(refans)
        for word1 in studans_we do
            for word2 in refans_we do
                cost_matrix ← cosinesimilarity(word1, word2)
        row_idx, col_idx ← linearsumassignment(costmatrix, maximize)
        for elem in cost_matrix[row_idx][col_idx] do
            if cost_matrix[row_idx][col_idx] ≥ threshold then
                /* normalized using the length of reference answer */
                sim ← cost_matrix[row_idx][col_idx]/len(refans_we)
        return sum(sim)
```

4.3.2 Rubrics-based

The complete pipeline of the approach proposed for the rubrics-based short answer grading is provided in Figure 4.3. The answers from the AMR dataset are extracted features after certain preprocessing suitable to that particular features. These features are the ones benchmarked using the Mohler dataset. The features are extracted between answers and rubrics instead of reference answers. The model is a classification-based wrapped using active learning that queries a single instance to the grader to label the

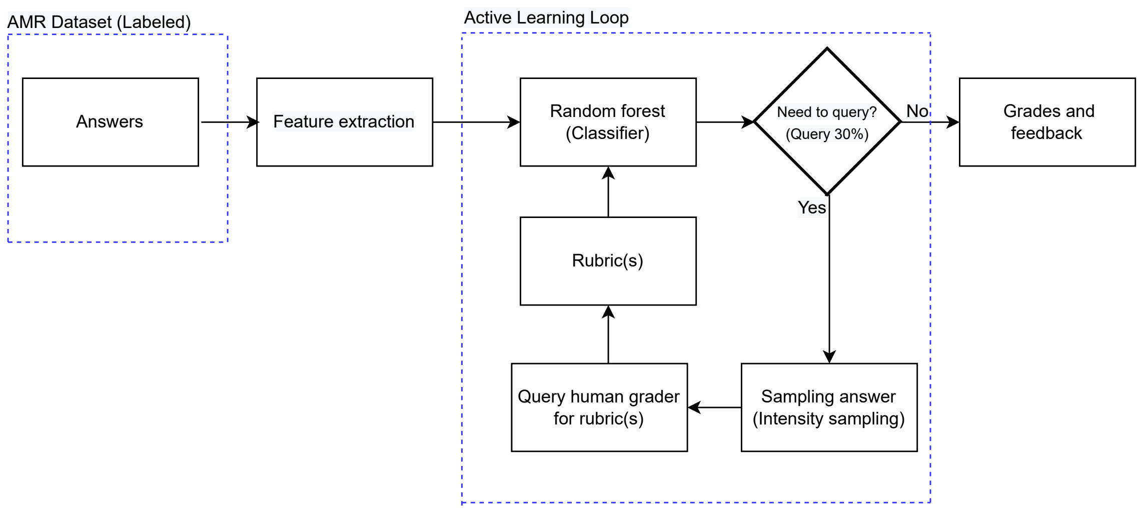

AMR Dataset (Labeled)
Answers
Feature extraction
Active Learning Loop
Random forest (Classifier)
Need to query? (Query 30%)
No
Grades and feedback
Yes
Rubric(s)
Query human grader for rubric(s)
Sampling answer (Intensity sampling)

Figure 4.3: Pipeline for rubrics-based short answer grading. The dataset used in this is the AMR dataset which is collected internally.

rubric (using rubrics table) from the provided feature set using intensity sampling. The model continues to train until it reaches 25% of the total data being queried, after which it predicts rubrics based on the training. The rubrics table consists of the score for all the linear combinations of the provided rubrics. The pseudo algorithm for the method and the generation of a rubrics table from the rubrics are presented below.

Algorithm 3: Generating Rubrics Table

```
Input: Rubrics
Output: Array of rubrics with score

Function Main:
    for count in len(rubrics) do
        for sent in iter(set(rubrics), count) do
            if $sent \mathrel{!}= [\,]$ then
                Rubricstable(rubrics) $\leftarrow sent$
                /* Since it is negative rubrics */
                Rubricstable(score) $\leftarrow -len(sent)$
    return Rubricstable
```

Algorithm 4: Rubrics-based Grading

```
Input: Question, Students answer, Rubrics
Output: Grades, Feedback

Function Main:
    /* features for the data are obtained as array */
    features ← $potential\_features(Qn, Studans, Rubrics)$

    /* initially get a single random index to initiate the model */
    initial ← $Features(random\_index)$
    pool ← $Features.remove(initial)$

    /* initiating active learner */
    learner ← $Activelearner(model, intensityquery, initial)$
    counter ←0

    /* Query for 25% of total data */
    for $counter < 0.25 * len(features)$ do
        query_data ← $learner.query(pool)$
        rubrics ← $learner.teach(query\_data)$
        pool ← $pool.remove(query\_data)$
        counter +=1

    /* predict remaining answer's grade and save the model for later */
    feedback ← $Rubricstable[rubrics][learner.predict(pool)]$
    grades ← $len(rubrics)$+$Rubricstable[score][feedback]$
    model ← $learner.save()$
```

4.4 Evaluation Metrics

Evaluation of ASAG is done by comparing the grades predicted by the model and the grades given by the human evaluator [40]. The success of ASAG is determined by how much the model generates similar grades to the human grader. Following are the two evaluation metrics used in this research work.

4.4.1 Pearson Correlation

Correlation is the degree of the linear relation between two variables. It can be computed between two numerical entities (e.g., height and weight) or between two categorical entities (e.g., product type and reviews). Pearson correlation is a majorly used evaluation metric for ASAG [16]. Hence, this research work prefers to use the same as expressed in Equation 4.1.

$$\rho_{xy} = \frac{COV(x,y)}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n}(x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \hat{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \hat{y})^2}} \tag{4.1}$$

where, x and y are the variables to find correlation, ρ_{xy} is Pearson correlation coefficient, $Cov(x,y)$ is the covariance of variables x and y, σ_x is the standard deviation of x, σ_y is the standard deviation of y, $\hat{x}$ is the mean value for the variable x and $\hat{y}$ is the mean value for the variable y. Pearson correlation measures the association between two variables, x, and y. Its value ranges from -1 to 1. A value of -1 is a negative correlation, 0 is no correlation, and +1 is a positive correlation.

4.4.2 Root Mean Square Error (RMSE)

RMSE is a popular evaluation metric to measure the prediction quality of supervised learning using actual value. It uses euclidean distance to compute the difference between predicted and actual values. RMSE is determined by obtaining the standard deviation of the residuals (difference between truth and prediction), which can be expressed as,

$$\text{RMSE}_{xy} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{4.2}$$

where, n is the number of measurements/true values, y_i is the predicted value and $\hat{y}_i$ is the true value.

4.4.3 Why not Accuracy?

Though accuracy can evaluate the model, there are better performance indicators for the unbalanced dataset [56]. For example, let x be a binary class (for simplicity) consisting of labels [1, 1, 0, 1, 1] such as 90% positive and 10% negative samples. The model had predicted label as [1, 1, 1, 1, 1], which has learned the dominating class but not the other as the data might need to be more sufficient to train the model. However, it obtains a 90% accuracy rate, whereas the correlation is undefined since there is no variance and the standard deviation becomes zero, as in Equation 4.1. The undefined correlation indicates that the model's performance is unsatisfactory, whereas accuracy does not indicate the model's learning. As ASAG datasets are mostly unbalanced, correlation metric is preferred compared to accuracy [16]. Also, correlation depicts how much the model performs similarly to the evaluator in grading the short answer.

5 Evaluation

This chapter describes the different datasets used in this research work along with the performance and analysis of the proposed methodology on those datasets.

5.1 Experimental Setup

The methodology proposed in Chapter 4 should be experimented to validate it. Thus, the experimental setup describes the transformation of process and methods presented in Chapter 3 and 4 to the working model.

5.1.1 Dataset Description

This research work experimented with two different datasets, namely Mohler [63] and AMR. These datasets differ in the grading approach where the former is reference-based, latter is rubrics-based. A comprehensive description of each dataset is provided below.

Mohler Dataset

The Mohler dataset from the University of North Texas comprises a short answer assignment from the Data Structure course. This assignment consists of questions from introductory computer science with the answers given by undergraduate students as presented in Table 5.1. The student responses are stored with the WebCT online service for learning [63]. This dataset is a collection of two exams with ten questions and ten assignments with four to seven questions each. Thus, the dataset holds eighty-one questions with 2,273 student responses. Two human evaluators graded these answers from zero (incorrect) to five (perfect). Hence the average of the grades is considered the gold standard to investigate the performance of ASAG [33]. The dataset is publicly available at the author's blog[1] at the Department of Electrical Engineering and Computer Science from the University of Michigan. The dataset's attributes are provided in Table 5.2.

[1]https://web.eecs.umich.edu/~mihalcea/downloads.html

Question:	What is the role of a prototype program in problem solving?
Reference Answer:	To simulate the behaviour of portions of the desired software product.
Student Answer 1:	To simulate portions of the desired final product with a quick and easy program that does a small specific job. It is a way to help see what the problem is and how you may solve it in the final project (5).
Student Answer 2:	the prototype program gives a general idea of what the end product will do, without the time and effort to write out the entire program (3).

Table 5.1: Question and answer from the Mohler dataset with gold standard score in parentheses.

Attributes	**Value/type**
Number of participants	28-29
Number of questions	81
Number of answers	2273
Number of answers/questions	28-29
Grade range	0-5
Domain	Computer science
Grading method	Reference answer-based
Dataset file extension	.csv

Table 5.2: Attributes of the Mohler dataset.

AMR Dataset

The AMR dataset from the Bonn-Rhein-Sieg University of Applied Science includes short answers from the Autonomous Systems course. This assignment consists of questions from Localization using maps with the answers given by graduate students as provided in Table 5.3. The student responses are gathered with the Jupyter Notebook published via an internal server for exams. This dataset is a collection of five questions from the mid-term and final exams with 190 student responses. These answers were graded by a human evaluator from zero (incorrect) to three (perfect) and provided feedback. This feedback is used as the key element for rubrics for this research work. Since the dataset is limited, internally developed, and still part of the examination, it is not publicly available. The dataset's attributes are provided in Table 5.4.

Question:	How can a continuous map representation be modified to reduce the computational cost?
Feedback:	closed world assumption, different map decomposition techniques.
Student Answer 1:	By using abstraction and capture relevant features, and neglect not so important information from the map, we can reduce the computational cost of the continuous map (1).
Student Answer 2:	Continous map representation can be reduced taking the closed world assumption into consideration where only regions with objects/obstacles are stored and all other non-object/free areas are left sparse. This result in a sparse map that is computationally and memory more efficient. Another way is to add an extra layer of abstraction where features like straight lines are extracted from continous points and these sets of lines approximate the exact map to a reasinable degree. This way, instead of saving many points only line(groupimg multiple points) prameters are saved (3).

Table 5.3: Question and answer from the AMR dataset with evaluator score in parentheses. The feedback provided is applicable to the student answer 1, since the student answer 2 secures full score.

Attributes	**Value / type**
Number of participants	38
Number of questions	5
Number of answers	190
Number of answers/questions	38
Grade range	0-3
Domain	Robotics
Grading method	Rubrics-based
Dataset file extension	.csv

Table 5.4: Attributes of the AMR dataset.

5.1.2 Machine Configuration

Both the Mohler and AMR datasets are small. So, it only requires a little computation power, which can be executed in a Personal Computer (PC) of the specification provided in Table 5.5. However, experimenting in various settings requires good memory and computation power. The experiment is conducted on the platform for Scientific Computing at Bonn-Rhein-Sieg University [37]. This platform's subspace (cluster) is granted for student's research work. The details are below in Table 5.5. Assure to check for error-free before executing the script on the University cluster. A detailed software prerequisite and frameworks used are explained in Appendix B.

Components	**Cluster specs**	**PC specs**
Operating system	Nitrogen, Linux 7.8	Ubuntu, Linux 20.04
Memory	50 GB for computation and 100 GB for storage	50 GB SSD
Central Processing Unit (CPU)	50 nodes with 2x Xeon processors having 16 cores	16GB RAM, 9th Generation Intel® Core™ i7-9750H
Graphical Processing Unit (GPU)	Nvidia Tesla V100	4GB, NVIDIA GeForce® GTX 1660 Ti

Table 5.5: Machine configuration utilized for this research [37]. Configuration of both cluster and PC used is mentioned where the PC is of model *Lenovo Legion Y540*.

5.1.3 Pre-processing

Before extracting features, the student and reference answers are preprocessed. The proposed approach includes the following preprocessing steps:

- **Spell check** is performed using pyspellchecker based on Peter Norvig's blog post that holds an English dictionary (not particular to the domain).
- **Case folding** is performed by lowering text cases using a default string library in python.
- **Tokenization, POS tagging, dependency parsing and lemmatization** are accomplished using SpaCy library.
- **Stop words removal** is performed using the stop words list associated with the SpaCy library.
- **Punctuation removal** is done by filtering out the "PUNCT" POS tag.
- **Question demoting** is performed by filtering out the words in the question from the student's answer.

All the preprocessing methods are not applied to the student and reference answers. Depending on the features to be extracted specific set of preprocessing is performed as indicated in Figure 4.2.

5.1.4 Features

Extracting features is crucial as it represents the raw data in a lower dimension. The proposed approach includes the following features extracted from the student and reference answer:

- **BOW** is a list of unique words in the given text computed by the countvector function from the Scikit-learn library.
- **N-gram** is performed using NLTK library that concatenates N successive tokens. This research work experimented with bigram and trigram. Among them, trigram produced a better result. Therefore, the research work used trigram but generally denoted as N-gram.
- **Linear sum assignment** is an optimization function from the Scipy library.
- **Length ratio** is performed by dividing the number of student answer tokens by the number of reference answer tokens.
- **Chunking** is performed by the NLTK library that parses the regex grammar along with tokens consisting of their respective POS tags.
- **Word embedding** is performed using Gensim library. The Gensim library helps to load and execute the pre-trained model. The word embedding used in this research work is FastText trained on domain corpora such as computer science and machine learning from Metzler [60].
- **Sentence embedding** is performed using sentence transformer library from Huggingface[2]. The pre-trained model used is *all-mpnet-base-v2*[3] which is trained on higher than one billion sentence pairs.

Among the above seven features, BOW, N-gram, chunking, word, and sentence embedding produce two vectors for student and reference answers, which are converted to a single value using cosine similarity. These values for each student answer constitute an array called a feature array.

[2]`https://huggingface.co/sentence-transformers/all-mpnet-base-v2`
[3]`https://www.sbert.net/docs/pretrained_models.html`

5.1.5 ML Models

Active learning wraps the ML models and converts the features to scores and feedback.

- **Active learning** is used from the ModAL library, which is comparatively effortless to use upon ML models from other frameworks.
- **Random forest and SVM** is used from Scikit-learn library.
- **Ridge regression** is used from Scikit-learn and mord library. Mord library is considered an extension of the Scikit-learn library for ordinal regression models [60]. Ridge regression from the mord library is denoted as M-Ridge, whereas the Scikit-learn library is denoted as Ridge.

This research does not focus on finding the best model with the best parameters. Hence, the default parameters of the model have been used.

5.1.6 Performance Metrics

The following metrics evaluate the performance of the approach.

- **Pearson correlation** is computed from the statistics function in the Scipy library.
- **RMSE** is computed from the metrics function in the Scikit-learn library.

The framework mentioned above and its usage are detailed in Appendix B. The code is available publicly[4] for further research and usage.

5.2 Results

The results are discussed individually based on the methods of grading. Reference answer-based grading is used to acquire the potential feature set and the model. Since the rubrics-based method is the first distinct approach to grade the short answer, which has yet to have a publicly or popularly available dataset to benchmark the method. There is no available dataset with negative rubrics along with the respective score. Therefore, the Mohler dataset is used to benchmark the approach to be used in the AMR dataset.

[4]`https://github.com/Ganesamanian/Computer-Assisted-Short-Answer-Grading-with-Rubrics-using-Active-Learning`

5.2.1 Reference Answer-based

The Mohler dataset is used to benchmark the methodology and detect the optimal parameters for the approach, such as features, percentage of querying, strategy for querying, and model.

Feature Selection

Seven features are extracted: BOW, N-gram (trigram), chunking, linear sum assignment, length ratio, word, and sentence embedding after specific preprocessing of student answers and their corresponding reference answers from the Mohler dataset. These features are pivotal because they are diverse and uncorrelated since analogous features do not improve performance. Therefore, to eliminate the redundant features, a hypothesis is framed that the features are not correlated by fixing the p-value to be less than or equal to 0.5.

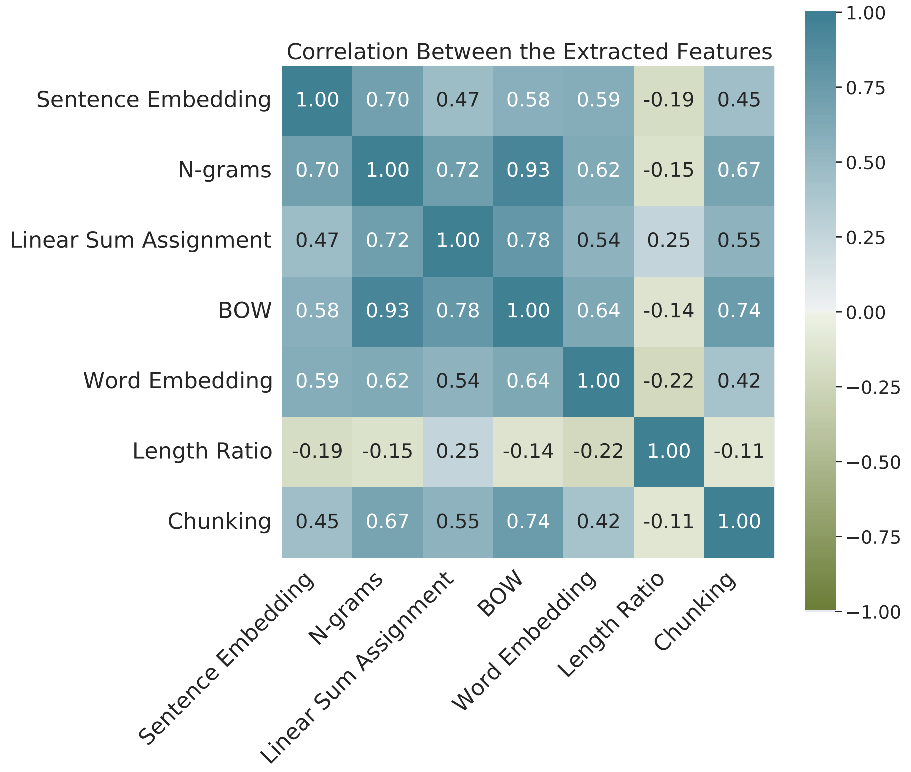
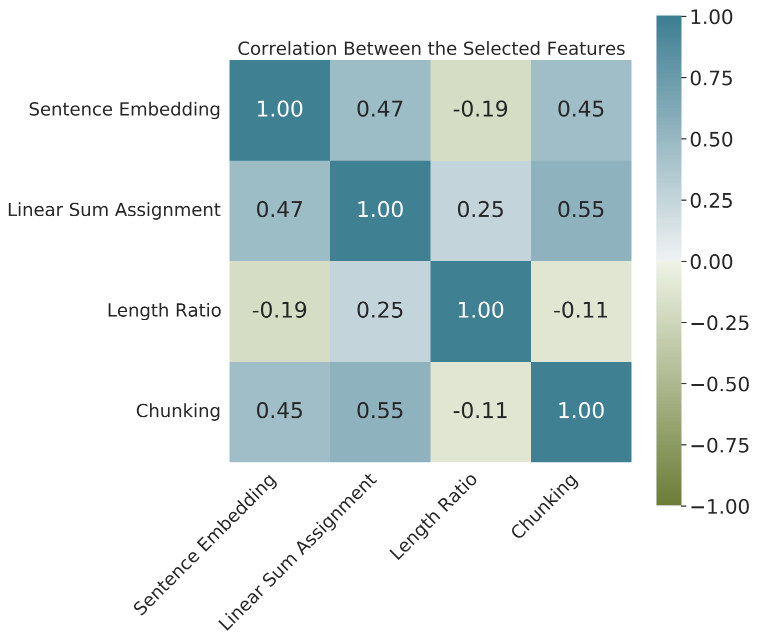


Figure 5.1: Correlation between the extracted features is represented using heatmap. The values in each cell correspond to the correlation value between the features forming the respective cell.

The correlation heatmap between the extracted features is presented in Figure 5.1. The diagonal of the matrix is one since it is the correlation with itself. At the same time, other cell indicates the correlation between the following features forming the individual cells. Employing the hypothesis on this correlation heatmap, potential features are selected. Thus, according to the hypothesis, the features that correlate above 0.5 are removed. The feature sentence embedding has a high correlation with N-grams (trigram) of 0.70, followed by a correlation with BOW and word embedding of 0.58 and 0.59, respectively. However, the sentence embedding feature can not be removed because the features are to be removed systematically based on the number of highest correlation values. Consider BOW that correlates with an N-gram (trigram) of 0.93, which is very close to one. Hence, BOW or N-gram (trigram) must be removed since the other is redundant. Recognizing the previous research work, the BOW has performed better and is popularly used comparatively. Thus the N-gram feature was removed initially. However, BOW highly correlates with other features, namely, linear sum assignment, chunking, word and sentence embedding of values 0.78, 0.74, 0.64, and 0.58, respectively. The BOW feature was removed next to N-gram, followed by word embedding. Since word embedding holds a correlation value of 0.59 with sentence embedding. Also, word embedding is internally used in linear sum assignment (refer Section 3.2.4).



	Sentence Embedding	Linear Sum Assignment	Length Ratio	Chunking
Sentence Embedding	1.00	0.47	-0.19	0.45
Linear Sum Assignment	0.47	1.00	0.25	0.55
Length Ratio	-0.19	0.25	1.00	-0.11
Chunking	0.45	0.55	-0.11	1.00

Figure 5.2: Correlation between the selected features is represented using heatmap.

The correlation heatmap after feature selection is provided in Figure 5.2. These features are less correlated because each one distinctly represents the raw text, such as 1. length ratio measures the length of the answer, 2. sentence embedding measures the semantics of the sentence as a whole, 3. chunking provides importance to the noun phrase in the sentence which consists of keywords or keyphrases and 4. linear sum assignment uses word embedding to determine the similar word pairs. The features chunking and linear sum assignment have a correlation value of 0.55, which is higher than the determined p-value of the hypothesis. However, removing either of the features degrades the performance, so both are retained. Chunking and linear sum assignment can be similar sometimes, as the word pair selected by linear sum assignment could be in the chunked noun phrase. Hence, the selected features are length ratio, chunking, linear sum assignment, and sentence embedding.

Determining Querying Percentage

When the approach was formulated, the querying percentage was assumed to be 25%, with which the model trains to predict the new instances. This percentage value was chosen to have a minimal contribution from the evaluator and not burden the evaluator. However, assuming the querying percentage without examining might mislead from the optimal value. The objective is to find the minimum querying percentage that yields the best performance. So the first question from the Mohler dataset, along with its student response and reference answer, is selected. The first question has an id 1.1 consisting of 28 student responses and a reference answer. The features selected from the previous step are extracted from the student answers and a reference answer. The model employed for this analysis is Random Forest Regressor (RFR). Initially, the model is trained with one instance and tested on 27 student responses. In the next step, the first instance is queried based on intensity sampling and tested on the remaining 26 responses. This process continues until no instances are left to query. Figure 5.3 represents the Pearson correlation and RMSE values corresponding to the increasing order of query percentage for 100 iterations. The Pearson correlation has a random negative value, and RMSE holds a random positive value (between 1 to 2) at the start. As the querying percentage increases, the Pearson correlation tends to saturate on the high value of 1, whereas RMSE tends to lower towards 0.2. Therefore, an optimal range is between 25-35% (7-10 number of queries), where the saturation begins. Hence the assumed value tends to be one of the optimal querying percentages.

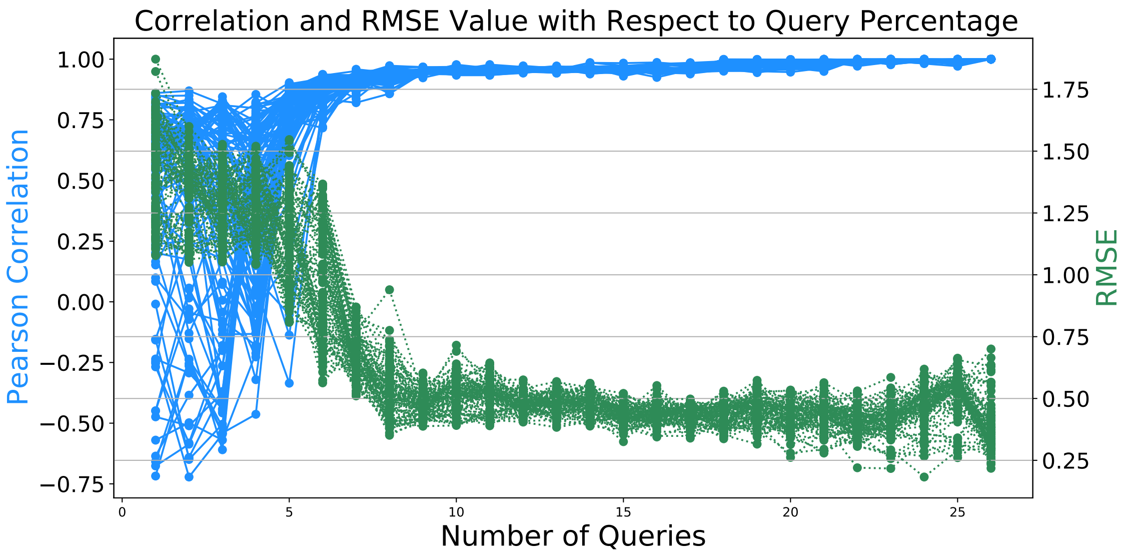


Figure 5.3: Pearson correlation and RMSE value with respect to query percentage. The graph presents the performance of RFR on question 1.1 of the Mohler dataset for the selected features. The evaluation is done for 100 iterations. Solid blue lines correspond to Pearson correlation and dashed green lines denote RMSE.

Though Figure 5.3 supports highlighting the optimal querying range, it is challenging to examine values for each iteration. So, Figures 5.4 and 5.5 provides a candle plot of Pearson correlation and RMSE of 100 iterations for 100% querying. The candle plot consists of four parameters, namely 1. **open** indicates the performance value at the start (for first query), 2. **close** indicates the performance value at the end (for end query), 3. **high** indicates highest value between the open and close and 4. **low** indicates lowest value between the open and close. In Figure 5.4, consider the first block (first iteration) the top, having value 1, is *close*, and at the bottom, having a value near to 0.5, is *open*. The thin line after the *open* is *low*, and the line after the *close* is *high* which is not present in this case since the highest value is one. In addition, the green color block represents increasing values, and the red color represents decreasing values; hence Pearson correlation is green, and RMSE is red. Also, the red color block has *close* at the bottom and *open* at the top or vice-versa to the green block, which was explained earlier. The average Person correlation and RMSE are 1.00 and 0.36 for 100 iterations with a 100% querying percentage. However, the candle plot does not provide information about the optimal percentage since it depicts the performance at each iteration. Therefore, Figure 5.3 is presented before, followed by the candle plot Figures 5.4 and 5.5.

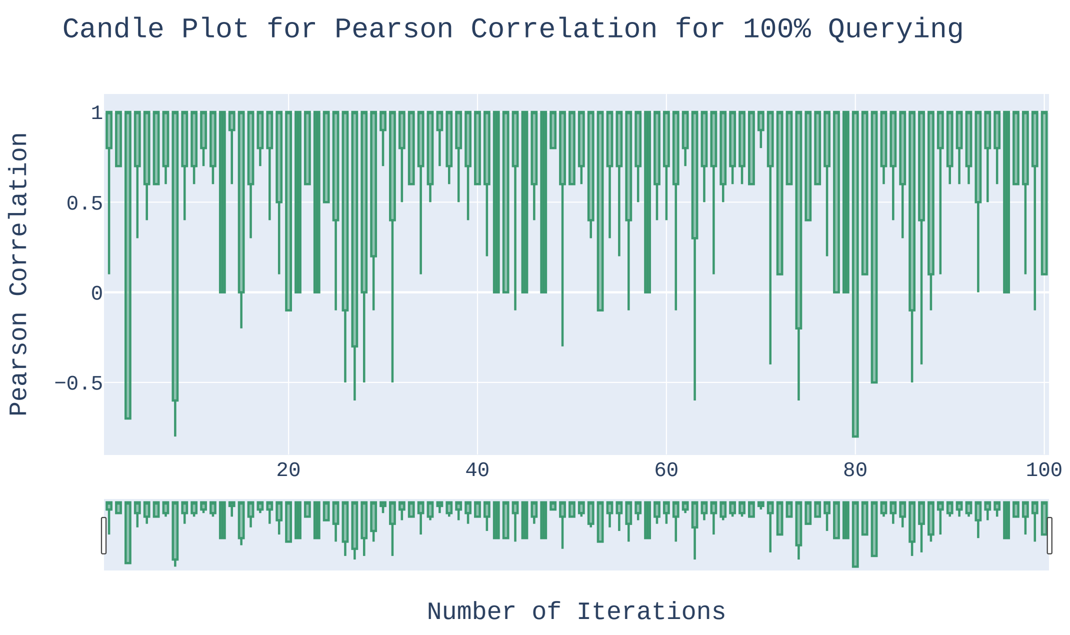


Figure 5.4: Candle plot for Pearson correlation of 100 iterations for 100% querying.

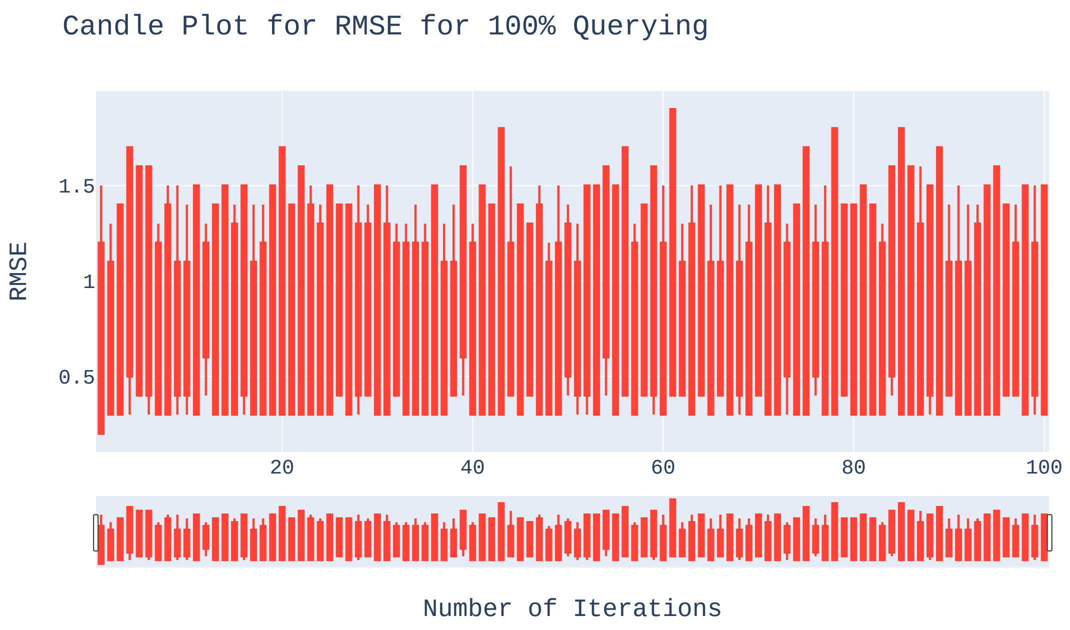


Figure 5.5: Candle plot for RMSE of 100 iterations for 100% querying.

The performance of the RFR for three settings for 100 iterations is presented in Table 5.6. The optimal range is 25-35%; thus, the model's performance with the same is to be examined instead of 100% querying. Three settings for querying percentage are investigated, which are 25% that queries seven instances to label, 30% corresponds to eight instances, and 35% corresponds to ten instances. Among them, 30% querying is optimal as it provides a better result with fewer training instances. Also, 30% querying is one question more than 25%, but the performance nearly matches the 35% querying, which is two more questions for querying. Pearson correlation and RMSE of 100 iterations for 30% querying are presented in Figures 5.6 and 5.7 respectively. Comparing the Figures 5.4 and 5.6, it is evident that 30% querying almost achieves the same performance as 100% querying. Hence, instead of assumed 25%, 30% querying is chosen throughout this research.

Metrics	**25%**	**30%**	**35%**
Pearson's ρ	0.91	**0.93**	0.95
RMSE	0.65	**0.53**	0.51

Table 5.6: Performance based on the optimal querying percentage. The optimal value is highlighted.

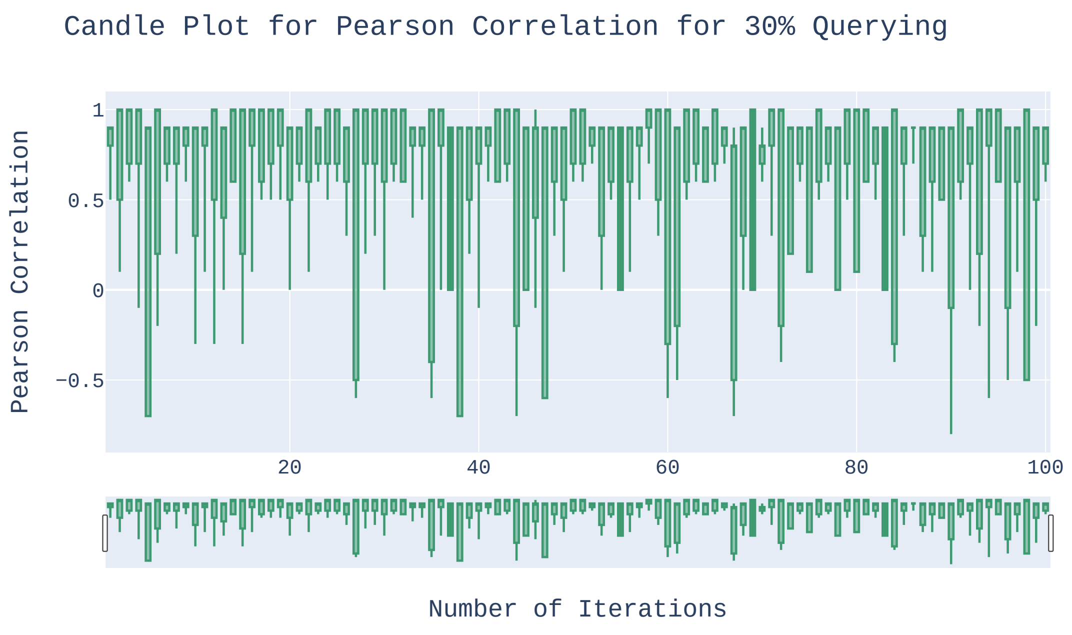


Figure 5.6: Candle plot for Pearson correlation of 100 iterations for 30% querying.

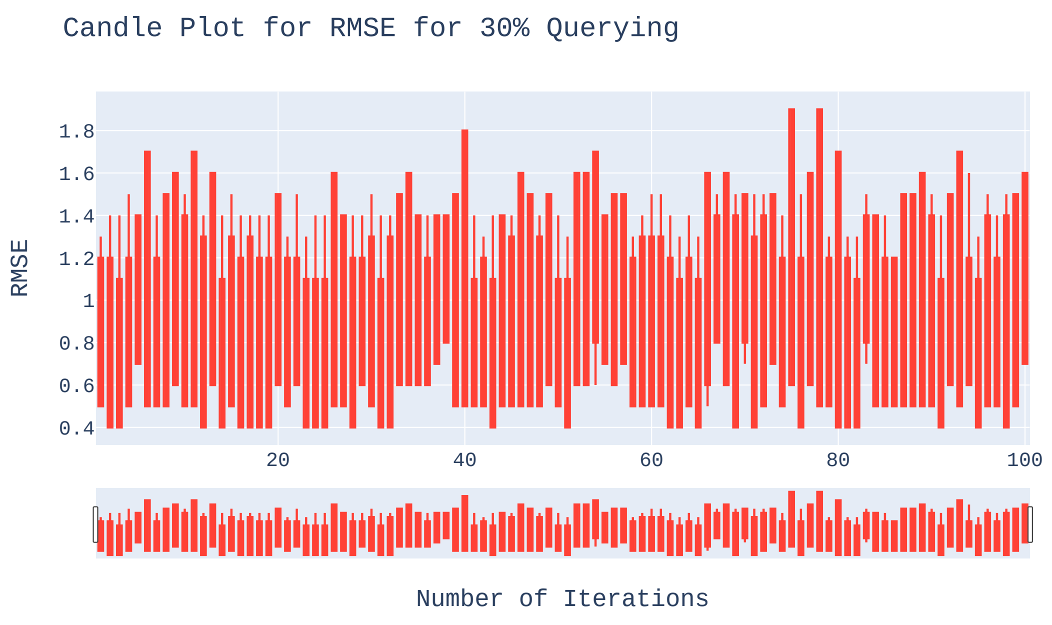


Figure 5.7: Candle plot for RMSE of 100 iterations for 30% querying.

The intensity sampling query strategy is evaluated by benchmarking its performance compared to the performance of uncertainty sampling. Since uncertainty sampling works only with classification models, intensity sampling is also performed with classification and regression models. The random forest model is used for both classification and regression tasks. Table 5.7 highlights performance based on query strategy for 30% of querying. Regarding the classification task, the intensity sampling has performed 4% higher than the uncertainty sampling. Querying informative instances is more effective than uncertain instances. In comparison, the regression task performs better than the classification task because the continuous values are much similar to the grading pattern of the evaluator. Therefore, this research work proceeds with intensity sampling as a querying strategy irrespective of the task.

Task	**Querying Strategy**	**Pearson's ρ**	**RMSE**
Classification	Uncertainty Sampling	0.85	0.78
Classification	**Intensity Sampling**	0.89	0.72
Regression	Intensity Sampling	**0.93**	**0.53**

Table 5.7: Performance based on task and querying strategy. Random forest is used for both the classification and regression tasks. Intensity sampling is better performing irrespective of the task.

Model Selection

A potential model to work with active learning is to be selected. Regarding the previous work, ridge regression [89, 60] and Support Vector Regression (SVR) which is SVM as regressor [63], are popularly used. This work uses RFR and the earlier-mentioned methods to examine the better-performing model for the selected features and querying percentage. Table 5.8 presents each ML model's performance on the Mohler dataset for the selected features. These ML models are wrapped with active learning that queries 30% of the total data using intensity sampling. Therefore, the performance is on the remaining 70% evaluated for 100 iterations.

Metric	**RFR**	**SVR**	**Ridge**	**M-Ridge**
Pearson's ρ	**0.637**	0.593	0.571	0.512
RMSE	**0.855**	0.882	0.915	0.958

Table 5.8: Performance of the ML models on the Mohler dataset. M-Ridge represents ridge regression from the mord library. Whereas the other three models are from the Scikit-learn library. The performance is arranged in their descending order from left to right.

	RFR	**SVR**	**M-Ridge**	**Ridge**
Execution Time (s)	50	29.6	1.79	1.62

Table 5.9: Execution time of the ML models on 70% of the Mohler dataset in seconds based on PC configuration. The time of execution is arranged in their descending order from left to right.

Among the ML models mentioned in Table 5.8, the RFR performed high, with a Pearson correlation value of 0.637 and an RMSE of 0.855. This is due to concatenating hundred decision trees in parallel, which has the drawback of a high processing time of 50s for 1,592 answers depicted in Table 5.9, comparatively. SVR shows the second-highest performance with almost 40% less execution time than RFR. Whereas Ridge and M-Ridge execute almost 94% less in time than SVR but the performance is low. Therefore, there should be a trade-off between performance and execution time. Based on trade-off, SVR is the significant model; based on performance, RFR, and execution time, it is Ridge. The values tabulated above are the average for a hundred iterations of random data sampling for training and testing. Figures 5.8 and 5.9 depicts the Pearson correlation and RMSE value of each ML model per iteration with their average value for hundred iterations.

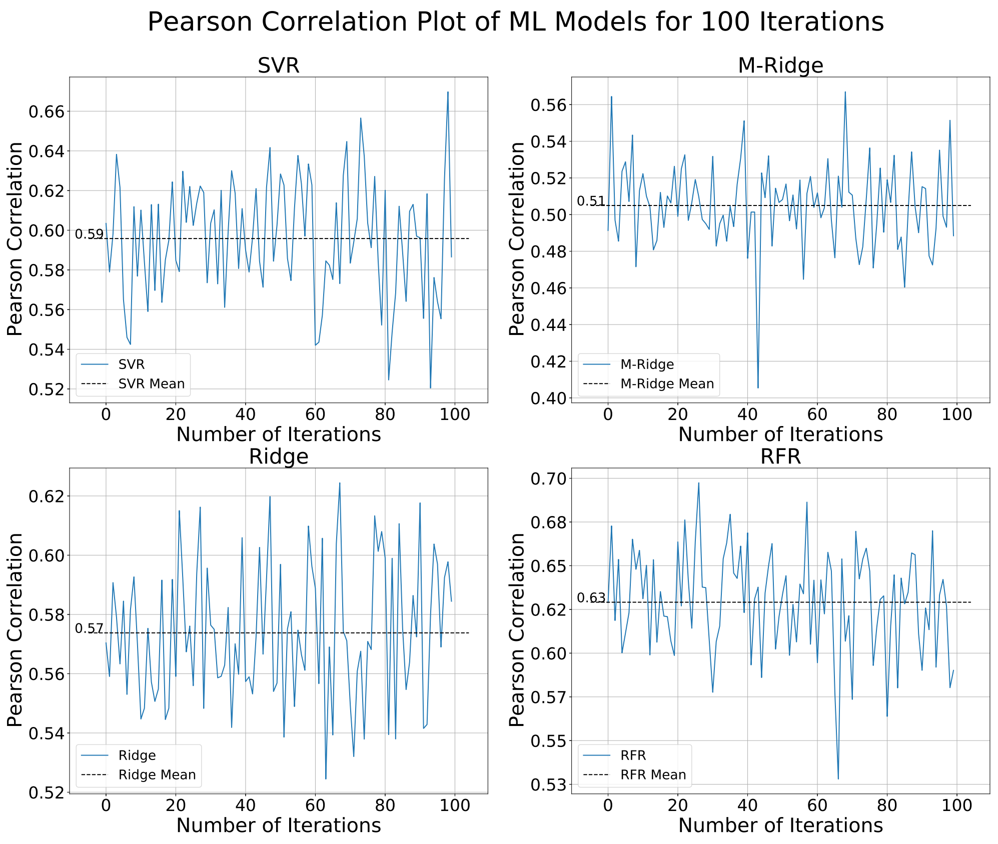


Figure 5.8: Pearson correlation graph for each model for 100 iterations. The dashed line represents the average value which is mentioned above the line.

A hundred iterations for each model are performed in the same condition, such as machine configuration, feature set, and environment. The only variability is the sampling of the test set for each iteration. The graphs in Figure 5.8 show that all the models oscillate in their performance across 100 iterations because of the randomly sampled data for training. Even though the performance oscillates, it remains within a range, indicating no abnormalities. When the training set consists of well-balanced all diverse classes, the model learns effectively to perform better predictions on the training data. Otherwise, the performance is comparatively low, based on the diversity; performance varies proportionally with the diversity of classes. This observed pattern could not be adequately expressed in an equation or formulation. Additionally, the answers that has equations or expressions, are difficult to process.

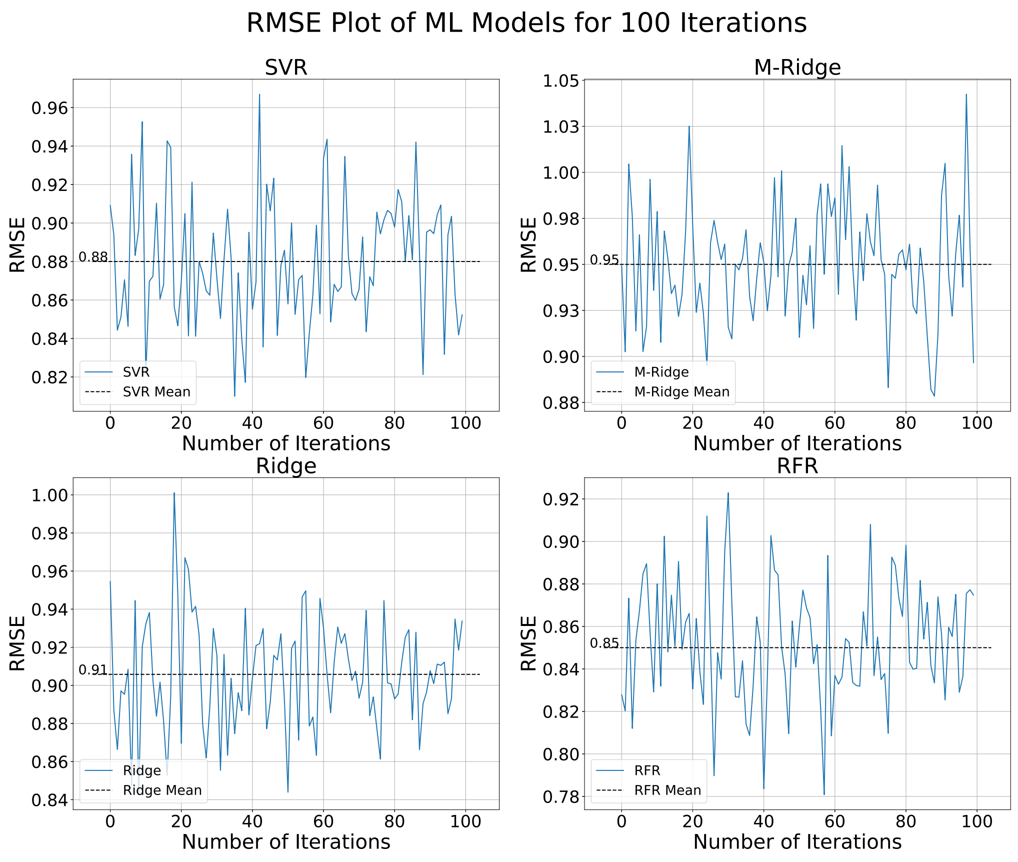


Figure 5.9: RMSE graph for each model for 100 iterations. The dashed line represents the average value which is mentioned above the line.

The model RFR and SVR have surpassed the SOTA results as depicted in Table 5.10. The results of this research outran the previous results since the significant ideas from each previous work were combined. Sultan et al. [89] found a better performance by combining the features that form this research's base. The BERT model for sentence embedding was inspired by Gaddipati et al. [31]. In addition, fastText for word embedding is employed from Metzler [60] due to its performance, and also it is trained in the domain articles. In addition, Metzler [60] has extracted verb phrase that increases the performance, and this research experimented with a noun phrase as most of the keywords do contain nouns [88]. Linear sum assignment was included as a new technique regarding the previous work to find similar word pairs between two sentences. Therefore, this research selects random forest (RFR) for rubrics-based grading since the model outperforms other models despite taking longer execution time. Since better hardware can reduce the execution time.

Model	**Approach**	**Pearson's ρ**	**RMSE**
SVR [Mohler and Mihalcea [63]]	BOW	0.431	0.999
SVR [Mohler and Mihalcea [63]]	TF-IDF	0.327	1.022
Ridge [Sultan et al. [89]]	TF-IDF + SIM	0.592	0.887
Ridge [Metzler [60]]	SIM (Word2Vec) + VP	0.545	0.945
Ridge [Metzler [60]]	SIM (Glove) + VP	0.509	1.002
Ridge [Metzler [60]]	SIM (FastText) + VP	0.537	0.956
Ridge [Gaddipati et al. [31]]	SIM (ELMo)	0.485	0.978
Ridge [Gaddipati et al. [31]]	SIM (BERT)	0.318	1.057
RFR	SIM (BERT & FastText) + NP	**0.637**	**0.855**
SVR	SIM (BERT & FastText) + NP	**0.593**	**0.882**
Ridge	SIM (BERT & FastText) + NP	0.571	0.915

Table 5.10: Result comparison on Mohler dataset with former approaches. In the table, *SIM* denotes similarity, *VP* is Verb Phrase and *NP* is Noun Phrase.

5.2.2 Rubrics-based

This approach uses the parameter benchmarked with the Mohler dataset for short answer grading using rubrics. The method queries first the rubrics associated with the question and short answer to grade instead of reference answer, which was used in the earlier approach. The rubrics related to short answer presented in Table 5.3 is provided in Table 5.11. Once the rubrics are provided, the approach generates a negative rubrics table based on the linear combination of provided rubrics as depicted in Table 5.12. In the previous approach, features such as chunking, linear sum assignment, length ratio, and sentence embedding are computed between each student and a reference answer per question. In this approach, the features are computed between each rubric and short answer. Then the active learning upon random forest (as a classifier) queries the evaluator to annotate with a rubric. After the model is trained based on querying, it predicts the rubrics for the remaining student responses. Based on the predicted rubrics, the corresponding rubrics score is added to the total score (since negative rubrics summing is preferred). The approach uses a random forest as a classifier instead of a regressor since the rubrics are a discrete quantity. Presently, the rubrics aid in achieving a partial score of one or two or three; it does not support attaining an intermediate score between the partial score, such as 0.5 or 1.5 or 2.5.

Rubrics - 1	Employing closed world assumption greatly reduces the computational cost.
Rubrics - 2	Different map decomposition techniques reduces the computational cost of its application.
Rubrics - 3	Selecting and representing maps with only features necessary for localization. Abstraction of these features as polygons or straight lines will reduce the computational cost.

Table 5.11: Rubrics for short_answer_A in AMR dataset.

Rubrics	**Score**	**Class**
correct	0	0
Rubrics - 1	-1	1
Rubrics - 2	-1	2
Rubrics - 3	-1	3
Rubrics - 1 & 2	-2	4
Rubrics - 1 & 3	-2	5
Rubrics - 2 & 3	-2	6
Rubrics - 1 & 2 & 3	-3	7

Table 5.12: Negative rubrics table for short_answer_A in AMR dataset.

To illustrate the working of the approach, consider the 1^{st} student response from Table 5.3 was evaluated to have a score of one since the answer does not contain 1^{st} and 2^{nd} rubrics. As per Table 5.12, it is of class *4*, and its corresponding score is *-2* when added with the total score *3* it yields a score of *1 (=3+(-2))*. Thus, the score and the missing rubrics (1&2) are provided as feedback. The performance for 100 iterations for one short answer (Short_Answer_A) is presented in Figures 5.10 and 5.11. Figure 5.10 is not the same as Figure 5.7 as it contains a negative impact presented in iteration *36* of decreasing correlation. This is due to the effect of the high imbalanced distribution of the assigned score as depicted in Figure 5.12. The correlation is higher whenever the answer corresponds to all the classes shown for querying. The correlation is comparatively less when the answer of either grade class of *3* or *0* is not presented for querying. In case of an answer from only one class, be it either *1* or *2*, the performance is decreasing as indicated by the red bar in Figure 5.10. However, such a case holds a probability range from 2-5%, yet the actions are to be taken so that at least one answer from each class is presented for querying. So that the negative impact can be avoided. Figure 5.11 consists of the positive impact of increasing RMSE, which tends to occur when the model learns a new class at the end of querying or does not show any variance until specific iterations.

An overview of different grade classes for Short_Answer_A from the AMR dataset is

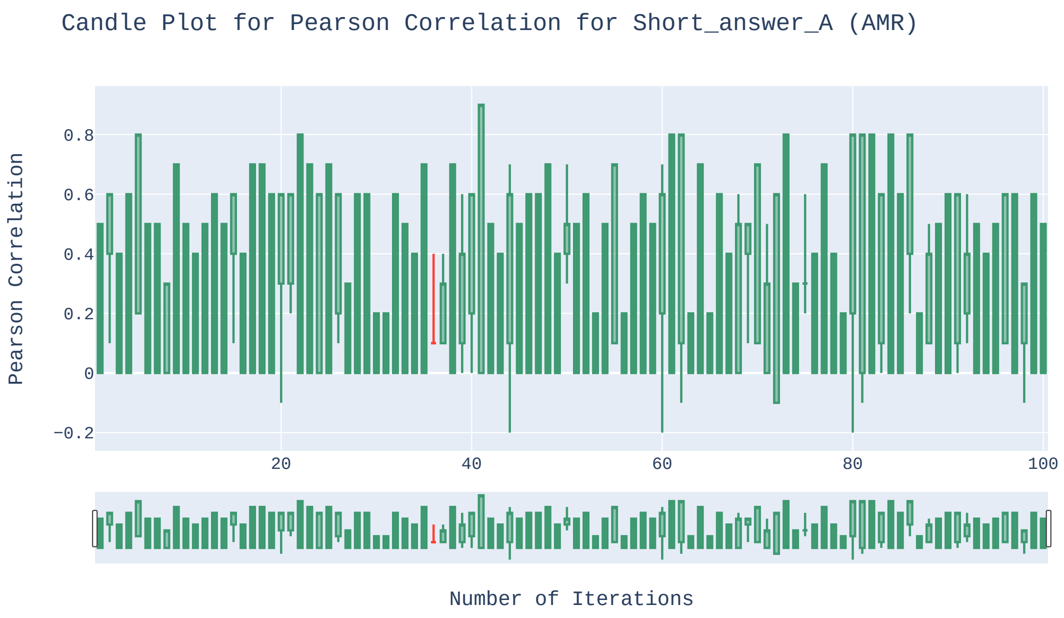


Figure 5.10: Candle plot for Pearson correlation for Short_Answer_A from AMR dataset.

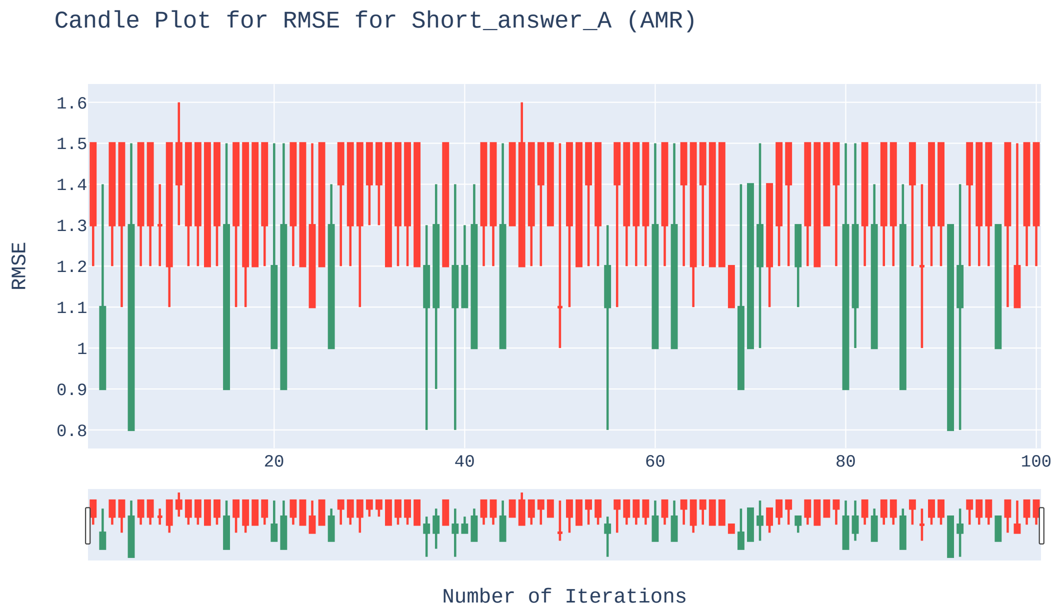


Figure 5.11: Candle plot for RMSE for Short_Answer_A from AMR dataset.

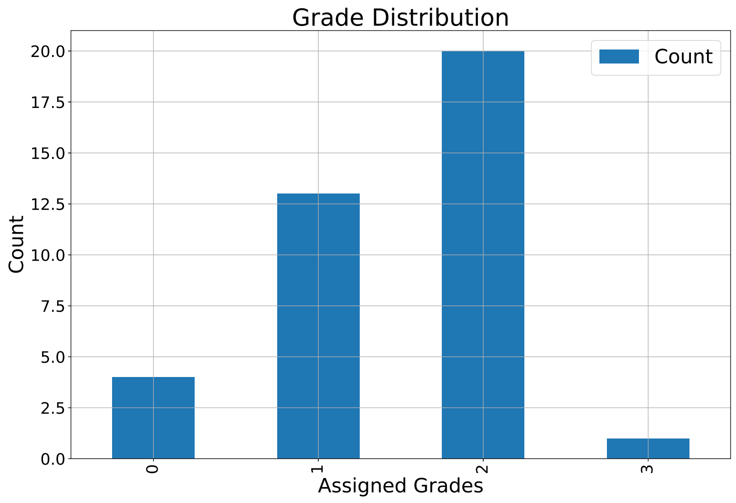


Figure 5.12: Grade distribution for Short_Answer_A (AMR).

provided in Figure 5.12. As mentioned earlier AMR dataset consists of five questions, so the experiment is performed on all five questions for 100 iterations, and their average performance is provided in Table 5.13. Similar to the Mohler dataset, the evaluation can not be performed on the whole dataset as the rubric is not a single element like reference answer. So, it has to be evaluated separately for each question. The overall average of Pearson correlation and RMSE accounts for 0.61 and 0.83, respectively. Thus, the methodology achieved in providing a formative assessment. Also, the performance can be further increased by having the embedding models trained on the domain-related articles as performed by Metzler [60] for the Mohler and Neural Network (internal dataset from the Bonn-Rhein-Sieg University of Applied Science) dataset. Therefore, further research on enhancing the features, rubrics, and models has to be performed to improve the current results. An example of score and feedback is provided in Table 5.14 for each category from AMR dataset for Short_Answer_A. A glimpse of actual model result is presented in Appendix C.

	Pearson's ρ	**RMSE**
Short_Answer_A	**0.67**	0.87
Short_Answer_B	0.64	0.83
Short_Answer_C	0.56	0.80
Short_Answer_D	0.51	**0.76**
Short_Answer_E	**0.67**	0.88
Average	0.61	0.83

Table 5.13: Performance of the approach on AMR dataset.

Student Answer	**Score**	**Feedback**
Continuous Map is a very simple algorithm, and it requires fewer parameters. It can be reduced by capturing only data points which are immediately relevant to localization.	0	Employing closed world assumption greatly reduces the computational cost, Different map decomposition techniques reduces the computational cost of its application, and Selecting and representing maps with only features necessary for localization. Abstraction of these features as polygons or straight lines will reduce the computational cost.
By using abstraction and capture relevant features, and neglect not so important information from the map, we can reduce the computational cost of the continuous map	1	Employing closed world assumption greatly reduces the computational cost and Different map decomposition techniques reduces the computational cost of its application.
The computational cost of a continuous map representation can be reduced by abstraction. Objects are reduced to 2D shapes, such as polygons. This omits futher details like color, texture, etc, but reduces the cost of the map. Furthermore, the closed-world assumption allows all irrelevant objects to be ignored from the map, only including those that can affect the robot.	2	Different map decomposition techniques reduces the computational cost of its application.
Continuous map representation can be reduced taking the closed world assumption into consideration where only regions with objects/obstacles are stored and all other non-object/free areas are left sparse. This result in a sparse map that is computationally and memory more efficient. Another way is to add an extra layer of abstraction where features like straight lines are extracted from continous points and these sets of lines approximate the exact map to a reasinable degree.	3	

Table 5.14: Score and feedback provided using rubrics-based approach for short_answer_A of each classes from AMR dataset.

6 Conclusion

In this research work, an active learning-based computer-assisted short answer grading was developed and benchmarked with the Mohler dataset from the computer science domain. Also, negative rubrics were generated and used for grading, which was evaluated using the AMR dataset from the Robotics domain.

The first phase of this research is about benchmarking the approach with the Mohler dataset. This phase is used to determine the potential features such as sentence embedding, chunking, length ratio, and linear sum assignment, to detect the optimal querying range to be 25-35%, and to decide the best model as random forest and SVM that surpasses the performance of existing SOTA methods. These finalized parameters for the proposed methodology achieve almost four percent higher results than the existing methods.

The next phase of this research is about using the proposed methodology with the decided settings to grade short answers using negative rubrics instead of reference answers. So, for the AMR dataset, the rubrics are generated from the evaluator's feedback. When the rubrics related to the question are fed, the method generates a rubric table with which it queries the grader to teach. Based on the teaching, the model trains to predict the rubrics for the provided short answer, which is used to compute the score. Finally, the score and the rubrics missing in the student response are provided as feedback. The proposed approach achieves an average performance of 0.61 for Pearson correlation and 0.83 for RMSE. The sentence and word embeddings trained on robotics-related articles will improve the performance. Thus, this research work sheds light on using rubrics for short answer grading and the performance enhancement of having collective features.

Further, this section enumerates the contributions, lessons learned, limitations, challenges, and future directions of this research work.

6.1 Contributions

The contributions of this research work are:

1. **Literature review:** A comprehensive literature review is performed from the initial phase of short answer grading to the current scenario. Also, the popularly used features to grade are presented in Chapter 3. A consolidated performance of ASAG is provided in Table 5.10 in Chapter 5.

2. **Querying strategy:** Sampling method for active learning that works on regression as well as classification was formulated (refer Section 3.4). Also, the formulated method performs better than the popularly used existing technique, as depicted in Table 5.7.

3. **Negative rubrics:** This research is the first to use negative rubrics for grading short answers to provide formative assessment. Negative rubrics help students in learning from their mistakes and also make them prepare for their following exams.

4. **Active learning and rubrics:** This research is the first to include an evaluator in the scoring process along with the ML method using active learning for short answer grading using rubrics. Figure 1.1 provides an evident advantage of using rubrics for grading instead of the prevailing traditional approach of using reference answers.

6.2 Lessons Learned

The major part of this research work is a development based on learning. The lessons learned are summarised as follows:

1. **Text preprocessing:** This is essential and extremely sensitive as it tends to eliminate important information at times. As stated earlier, text containing many acronyms should be handled carefully for case folding. For example, case lowering "US" to "us", where the former refers to a country, the latter refers to a first-person pronoun that changes the whole meaning. Therefore, this research work uses a specific set of preprocessing for a particular feature depicted in Figure 4.2.

2. **Feature selection:** Feature selection is crucial as it influences performance at a higher rate. Sometimes, having redundant features degrades performance. Therefore, by trial and error, this research work has a fixed p-value as a selection criterion of a correlation value of 0.5.

3. **Appending features:** Instead of having single features to grade, collection of a certain number of features does provide better results [89]. In recent times, this

has been the least considered that eliminates the perspective of representing the raw text as different representation works together to produce better results.

4. **Short answer scoring methods:** There are many methods to achieve short answer scoring based on the environment, domain, short answer, and necessity. However, the formulation of a method to grade short answers is to incorporate essential details from the previous research works and enhance the same with the latest techniques.

6.3 Limitations

This research work has the following drawbacks:

1. **Benchmark dataset:** As this is the first research work to have negative rubrics to grade short answers, there is no dataset to benchmark the approach. Therefore, this method used the Mohler dataset to benchmark the approach, yet working with rubrics differs from working with reference answers.
2. **Training on domain articles:** Research has proved that training the models for embedding [60] or other feature extraction [89] on domain-related articles had improved the performance comparatively. This research uses sentence embedding trained on standard articles and word embedding trained on machine learning and computer science articles. Hence, this could restrict the features from performing better on the AMR dataset.
3. **Multi-rubrics:** The approach generates a rubrics table of eight classes for three rubrics which generates 2^n computational complexity. Therefore, having an approach for multi-rubrics prediction could reduce this computational complexity.
4. **Regressor:** The grading model should be a regressor instead of a classifier to perform much similarly to the grader. The rubrics-based approach is currently a classifier which could also restrict the model's performance.

6.4 Challenges

The followings are the challenges faced during this research work:

1. **Dataset with rubrics:** This research work required a dataset consisting of negative rubrics for short answer grading. Only one publicly available for rubrics (positive rubrics): the ASAP-SAS dataset from Kaggle [94]. However, not all the questions have rubrics, and only partial rubrics are available. At the same time,

most datasets are based on reference answers [63, 20]. Therefore, this research work has created rubrics from the feedback provided by the evaluator for the AMR course.

2. **Shortcoming of embedding models:** There are no embedding models available related to the domains. Therefore, the researcher has to train independently to improve the performance [60]. Also, the models do not perform well on the rephrased sentence and text with more antecedents, expressions, and acronyms.

6.5 Future Works

Following are the future work related to this research:

1. **Dataset creation:** This research work has used a small dataset to pitch the approach. However, a dataset with more questions similar to the Mohler dataset with corresponding negative rubrics should be created to evaluate the approach extensively. So, a further step could be creating a dataset and corresponding negative rubrics.

2. **Training the model:** Due to time constraints, the sentence and word embedding used in this research work could not be trained on domain-related articles. Training on domain-related articles will improve the current performance of the approach.

3. **Query strategy:** The query strategy is to ensure that at least one response from each grade class is presented from the sampling. Also, should have intermediate querying instead of continuous querying at the start.

4. **Multi-rubrics prediction approach:** As depicted in Figure 6.1, multi-rubrics prediction can be achieved by stacking the N classifier for N rubrics. Each classifier gives a probability score for the rubrics presented in the student response as S1, S2, till SN. These scores are thresholded, when the scores exceed the threshold, indicating that the following rubrics are in the answer and the scores are summed. These scores are finally subtracted from the total scores, and feedback is provided as the rubric that is not in the student answer as presented in the below pseudo code. The current approach should incorporate the following modification inside the active learning loop. This modification makes the approach rubrics configurable and indicates how confident the method grades the answer. Also, this could restrain the 2^n computational complexity. However, active learning does not accept more than one model. So research is to be done on stacking the models inside active learning and developing the same.

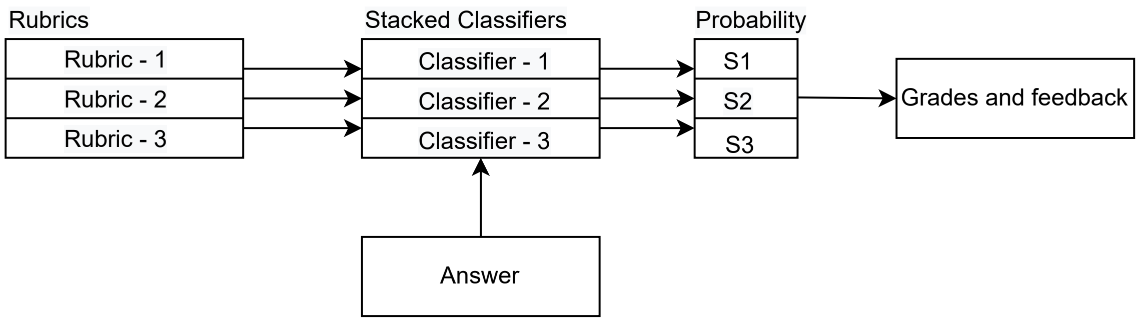


Figure 6.1: Concept sketch for multi-rubrics prediction. Assuming it for three rubrics.

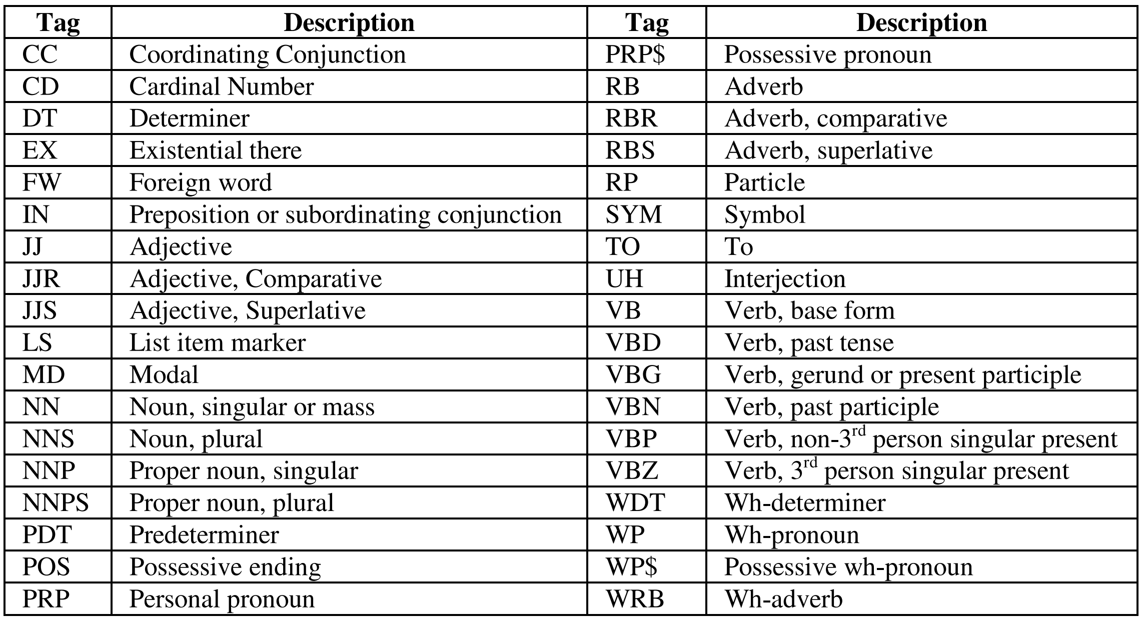
Algorithm 5: Stacking of Models for Rubrics

```
Input: Question, Students answer, Rubrics
Output: Grades, Feedback

Function Main:
    /* get the rubrics                                                  */
    rubrics ← $listofrubricfromuser$
    rubric_classifier ← {}
    for value, rubric in enumerate(rubrics) do
        /* features for the data are obtained as array                  */
        features ← $potential\_features(Qn, Studans, rubric)$
        /* form N classifer for N rubrics                               */
        rubric_classifier["value"] ← $model.train(features)$
    score ← 0
    matchrubrics ← [ ]
    matchgrade ← 0
    for key, classifier in rubric_classifier.items() do
        score ← $classifier.predict(answer)$
        if score >= threshold then
            matchrubrics.append(rubrics[key])
            matchgrade += score
    /* Since negative rubrics                                           */
    grade ← $total\_grade + matchgrade$
    feedback ← $[for\ rubric\ in\ rubrics\ if\ rubric\ not\ in\ matchrubrics]$
```

A
POS Tagger

Below Figure A.1 depicts the list of POS tags included in the Penn Treebank tagset.



Tag	Description	Tag	Description
CC	Coordinating Conjunction	PRP$	Possessive pronoun
CD	Cardinal Number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential there	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	To
JJR	Adjective, Comparative	UH	Interjection
JJS	Adjective, Superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VBN	Verb, past participle
NNS	Noun, plural	VBP	Verb, non-3rd person singular present
NNP	Proper noun, singular	VBZ	Verb, 3rd person singular present
NNPS	Proper noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

Figure A.1: Penn Treebank POS tags. Image from [86, p. 5].

B

Frameworks

To replicate this research work following frameworks are necessary

1. Python3 == 3.6.13
2. Numpy == 1.19.5
3. Pandas == 1.1.5
4. Seaborn == 0.11.2
5. Plotly == 5.10.0
6. Gensim == 3.8.3
7. SpaCy == 3.4.1
8. NLTK == 3.6.7
9. Sklearn/ Scikit-learn == 0.24.2
10. Scipy == 1.5.4
11. ModAL == 0.4.1
12. Pyspellchecker == 0.6.3
13. Matplotlib == 3.3.4
14. Sentence_transformers == 2.2.2
15. Jupyter Notebook == 5.4.1
16. Anaconda == 3.0.0 (optional)

It is advisable to use Anaconda, which makes a constrained environment. The script for this research work is written in Python following the Python3 format. Gensim framework is utilized for working with word embeddings where the word embeddings FastText is used from Metzler [60] work. The spellchecker library is used to check the spelling. Apart from spell check, other preprocessing is done using the SpaCy toolkit. Sentence_transformer framework is used from Huggingface, which has trained on a large and diverse dataset of over 1 billion training pairs. NLTK toolkit is used for chunking. Libraries like Numpy and Scipy are used for computation, like concatenating and storing the features and finding Pearson correlation. Plotly and Seaborn libraries provide better visualization along with Matplotlib. Scikit-learn provides machine learning models like SVM, random forest, and calculate RMSE. modAL is used specifically for the active learning component. The code and trained model can be downloaded using the following Github command

$ git clone https://github.com/Ganesamanian/Computer-Assisted-Short-Answer-Grading-with-Rubrics-using-Active-Learning.git

The repository includes

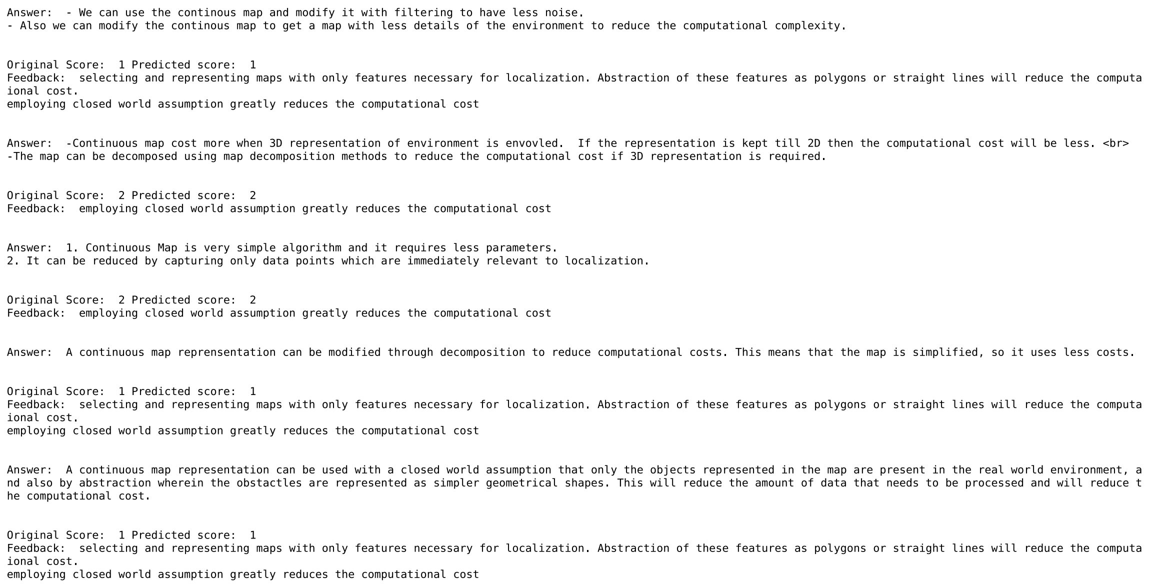
1. Dataset directory contains only the Mohler dataset as the AMR dataset is confidential.
2. Script directory contains the script files to execute the code and load the file to run the saved model.
3. Model directory contains the saved models.
4. Result directory contains experiment results.

Check the machine configuration before executing the code; further instructions are provided in the GitHub repository.

C

Rubrics-based Grading

Below Figure C.1 depicts the results of rubrics-based grading of Short_Answer_A from the AMR dataset. The complete results can not be produced due to size issues; therefore, a glimpse is presented below.



```
Answer:  - We can use the continous map and modify it with filtering to have less noise.
- Also we can modify the continous map to get a map with less details of the environment to reduce the computational complexity.


Original Score:  1 Predicted score:  1
Feedback:  selecting and representing maps with only features necessary for localization. Abstraction of these features as polygons or straight lines will reduce the computa
ional cost.
employing closed world assumption greatly reduces the computational cost


Answer:  -Continuous map cost more when 3D representation of environment is envovled.  If the representation is kept till 2D then the computational cost will be less. <br>
-The map can be decomposed using map decomposition methods to reduce the computational cost if 3D representation is required.


Original Score:  2 Predicted score:  2
Feedback:  employing closed world assumption greatly reduces the computational cost


Answer:  1. Continuous Map is very simple algorithm and it requires less parameters.
2. It can be reduced by capturing only data points which are immediately relevant to localization.


Original Score:  2 Predicted score:  2
Feedback:  employing closed world assumption greatly reduces the computational cost


Answer:  A continuous map reprensentation can be modified through decomposition to reduce computational costs. This means that the map is simplified, so it uses less costs.


Original Score:  1 Predicted score:  1
Feedback:  selecting and representing maps with only features necessary for localization. Abstraction of these features as polygons or straight lines will reduce the computa
ional cost.
employing closed world assumption greatly reduces the computational cost


Answer:  A continuous map representation can be used with a closed world assumption that only the objects represented in the map are present in the real world environment, a
nd also by abstraction wherein the obstactles are represented as simpler geometrical shapes. This will reduce the amount of data that needs to be processed and will reduce t
he computational cost.


Original Score:  1 Predicted score:  1
Feedback:  selecting and representing maps with only features necessary for localization. Abstraction of these features as polygons or straight lines will reduce the computa
ional cost.
employing closed world assumption greatly reduces the computational cost
```

Figure C.1: Rubrics-based grading for Short_Answer_A from AMR dataset.

References

[1] Abdel-Karim Al-Tamimi, Esraa Bani-Isaa, and Ahmed Al-Alami. Active learning for arabic text classification. In *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, pages 123–126, 2021. doi: 10.1109/ICCIKE51210.2021.9410758.

[2] Noah Arthurs and AJ Alvero. Whose truth is the "ground truth"? college admissions essays and bias in word vector evaluation methods. 07 2020.

[3] Lyle F Bachman, Nathan Carr, Greg Kamei, Mikyung Kim, Michael J Pan, Chris Salvador, and Yasuyo Sawaki. A reliable approach to automatic assessment of short answer free responses. Association for Computational Linguistics, 2002.

[4] Stacey Bailey and Detmar Meurers. Diagnosing meaning errors in short answers to reading comprehension questions. pages 107–115, 07 2008. doi: 10.3115/1631836.1631849.

[5] Sergio Barraza, William Lindskog, Davide Badalotti, Oskar Liew, and Arash Toyser. Active learning framework for time-series classification of vibration and industrial process data. *Annual Conference of the PHM Society*, 13, 11 2021. doi: 10.36001/phmconf.2021.v13i1.3059.

[6] Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading. *TACL*, 1:391–402, 12 2013. doi: 10.1162/tacl_a_00236.

[7] Hamid Bekamiri, Daniel Stefan Hain, and Roman Jurowetzki. A survey on sentence embedding models performance for patent analysis. *ArXiv*, abs/2206.02690, 2022.

[8] Marco Bellini, Georges Pantalos, Peter Kaspar, Lars Knoll, and Luca De-Michielis. An active deep learning method for the detection of defects in power semiconductors. In *2021 32nd Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, pages 1–5, 2021. doi: 10.1109/ASMC51741.2021.9435657.

[9] Snehal Bhoir, Tushar Ghorpade, and Vanita Mane. Comparative analysis of different word embedding models. In *2017 International Conference on Advances in Computing, Communication and Control (ICAC3)*, pages 1–4, 2017. doi: 10.1109/ICAC3.2017.8318770.

[10] Geetanjali Bihani and Julia Taylor Rayz. Model choices influence attributive word associations: A semi-supervised analysis of static word embeddings. In *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 568–573, 2020. doi: 10.1109/WIIAT50758.2020.00085.

[11] Vasiliki Bikia, Georgios Rovas, Stamatia Pagoulatou, and Nikolaos Stergiopulos. Determination of aortic characteristic impedance and total arterial compliance from regional pulse wave velocities using machine learning: An in-silico study. *Frontiers in Bioengineering and Biotechnology*, 9, 05 2021. doi: 10.3389/fbioe.2021.649866.

[12] Sébastien Bougleux and Luc Brun. Linear sum assignment with edition. *ArXiv*, abs/1603.04380, 2016.

[13] Lorenzo Bruzzone and Claudio Persello. Active learning for classification of remote sensing images. In *2009 IEEE International Geoscience and Remote Sensing Symposium*, volume 3, pages III–693–III–696, 2009. doi: 10.1109/IGARSS.2009.5417857.

[14] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[15] Rainer E. Burkard and Ulrich Derigs. *The Linear Sum Assignment Problem*, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980.

[16] Steven Burrows, Iryna Gurevych, and Benno Stein. The eras and trends of automatic short answer grading. *International Journal of Artificial Intelligence in Education*, 25, 03 2015. doi: 10.1007/s40593-014-0026-8.

[17] David Callear, Jenny Jerrams-Smith, and Victor Soh. Caa of short non-mcq answers. 01 2001.

[18] Leon Camus and Anna Filighera. Investigating transformers for automatic short answer grading. *Artificial Intelligence in Education*, 12164:43 – 48, 2020.

[19] John Carey, Richard Churches, Geraldine Hutchinson, Jeff Jones, and Paul Tosey. Neuro-linguistic programming and learning: teacher case studies on the impact of nlp in education. 2010.

[20] Daniel Matthew Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In **SEMEVAL*, 2017.

[21] Daniel Matthew Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *ArXiv*, abs/1803.11175, 2018.

[22] Guanliang Chen, David Lang, Rafael Ferreira, and Dragan Gaević. Predictors of student satisfaction: A large-scale study of human-human online tutorial dialogues. In *EDM*, 2019.

[23] Carnegie Mellon University computer science department. Word embedding demo, 2022. URL https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/. Accessed on: 2022-10-24. [Online].

[24] Aubrey Condor, Max Litster, and Zachary A. Pardos. Automatic short answer grading with sbert on out-of-sample questions. In *EDM*, 2021.

[25] Mathieu d'Aquin and Remi Venant. Towards the prediction of semantic complexity based on concept graphs. 07 2019.

[26] Towards data science. Demystifying Maths of SVM — Part 1, 2020. URL https://towardsdatascience.com/demystifying-maths-of-svm-13ccfe00091e. Accessed on: 2020-12-12. [Online].

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[28] Dmitriy Dligach and Martha Palmer. Good seed makes a good crop: Accelerating active learning using language modeling. pages 6–10, 01 2011.

[29] Rosa Figueroa, Qing Zeng-Treitler, Long Ngo, Sergey Goryachev, and Eduardo Wiechmann. Active learning for clinical text classification: Is it better than random sampling? *Journal of the American Medical Informatics Association : JAMIA*, 19: 809–16, 06 2012. doi: 10.1136/amiajnl-2011-000648.

[30] Samuel Fonseca, Filipe Pereira, Elaine Teixeira de Oliveira, David de Oliveira, Leandro Carvalho, and Alexandra Cristea. Automatic subject-based contextualisation of programming assignment lists. 06 2020.

[31] Sasi Kiran Gaddipati, Deebul Nair, and Paul-Gerhard Plöger. Comparative evaluation of pretrained transfer learning models on automatic short answer grading. *ArXiv*, abs/2009.01303, 2020.

[32] Lucas Galhardi and Jacques Brancher. *Machine Learning Approach for Automatic Short Answer Grading: A Systematic Review*, pages 380–391. 11 2018.

[33] Wael Gomaa and Aly Fahmy. Short answer grading using string similarity and corpus-based similarity. *International Journal of advanced Computer Science and Applications (IJACSA)*, 3, 12 2012. doi: 10.14569/IJACSA.2012.031119.

[34] Wael Gomaa and Aly Fahmy. *Ans2vec: A Scoring System for Short Answers*, pages 586–595. 01 2020.

[35] Mohamed Goudjil, Mouloud Koudil, Nacereddine Hammami, Mouldi Bedda, and Meshrif Alruily. Arabic text categorization using svm active learning technique: An overview. In *2013 World Congress on Computer and Information Technology (WCCIT)*, pages 1–2, 2013. doi: 10.1109/WCCIT.2013.6618666.

[36] Christian Gütl. e-examiner: Towards a fully-automatic knowledge assessment tool applicable in adaptive e-learning systems. 2007.

[37] Hochschule Bonn-Rhein-Sieg (H-BRS). Platform for Scientific Computing at Bonn-Rhein-Sieg University, 2020. URL https://wr0.wr.inf.h-brs.de/wr/index.html. Accessed on: 2020-12-21. [Online].

[38] Md. Rakibul Hasan, Maisha Maliha, and M. Arifuzzaman. Sentiment analysis with nlp on twitter data. In *2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)*, pages 1–4, 2019. doi: 10.1109/IC4ME247184.2019.9036670.

[39] Uswatun Hasanah, Tri Astuti, Rizki Wahyudi, Zanuar Rifai, and Rilas Agung Pambudi. An experimental study of text preprocessing techniques for automatic short answer grading in indonesian. In *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)*, pages 230–234, 2018. doi: 10.1109/ICITISEE.2018.8720957.

[40] Uswatun Hasanah, Adhistya Permanasari, Sri Kusumawardani, and Feddy Pribadi. A scoring rubric for automatic short answer grading system. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 17:763, 04 2019. doi: 10.12928/telkomnika.v17i2.11785.

[41] Andrea Horbach and Alexis Palmer. Investigating active learning for short-answer scoring. pages 301–311, 01 2016. doi: 10.18653/v1/W16-0535.

[42] Wen-Juan Hou and Jia-Hao Tsao. Automatic assessment of students' free-text answers with different levels. *International Journal on Artificial Intelligence Tools*, 20:327–347, 04 2011. doi: 10.1142/S0218213011000188.

[43] Huggingface. Explorable transformers, 2022. URL `https://huggingface.co/exbert/?model=bert-base-cased`. Accessed on: 2022-10-24. [Online].

[44] Sabir Ismail and M. Shahidur Rahman. Bangla word clustering based on n-gram language model. In *2014 International Conference on Electrical Engineering and Information Communication Technology*, pages 1–5, 2014. doi: 10.1109/ICEEICT.2014.6919083.

[45] Navjeet Kaur and Kiran Jyoti. Automated assessment of short one-line free-text responses in computer science. 04 2013. doi: 10.47893/IJCSI.2013.1103.

[46] Dr Khaled. Natural language processing and its use in education. *International Journal of Advanced Computer Science and Applications*, 5, 12 2014. doi: 10.14569/IJACSA.2014.051210.

[47] Arfiani Nur Khusna and Indri Agustina. Implementation of information retrieval using tf-idf weighting method on detik.com's website. In *2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pages 1–4, 2018. doi: 10.1109/TSSA.2018.8708744.

[48] Jeeveswaran Kishaan, Mohandass Muthuraja, Deebul Nair, and Paul-Gerhard Plöger. Using active learning for assisted short answer grading. 2020.

[49] Richard Klein, Angelo Kyrilov, and Mayya Tokman. Automated assessment of short free-text responses in computer science using latent semantic analysis. 2011.

[50] Surya Krishnamurthy, Ekansh Gayakwad, and Nallakaruppan Kailasanathan. Deep learning for short answer scoring. *International Journal of Recent Technology and Engineering*, 7:1712–1715, 03 2019.

[51] Claudia Leacock and Martin Chodorow. Crater: Automated scoring of short-answer questions. *Language Resources and Evaluation - LRE*, 37:389–405, 11 2003. doi: 10.1023/A:1025779619903.

[52] L. Li, Y. Wu, Y. Ou, Q. Li, Y. Zhou, and D. Chen. Research on machine learning algorithms and feature extraction for time series. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–5, 2017. doi: 10.1109/PIMRC.2017.8292668.

[53] X. Li, X. Long, G. Sun, G. Yang, and H. Li. Overdue Prediction of Bank Loans Based on LSTM-SVM. In *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1859–1863, 2018. doi: 10.1109/SmartWorld.2018.00312.

[54] Yuexiang Li, Xinpeng Xie, Linlin Shen, and Shaoxiong Liu. Reversed active learning based atrous densenet for pathological image classification, 07 2018.

[55] Tianqiao Liu, Wenbiao Ding, Z. Wang, Jiliang Tang, Gale Yan Huang, and Zitao Liu. Automatic short answer grading via multiway attention networks. In *AIED*, 2019.

[56] Jinzhu Luo. Automatic short answer grading using deep learning. Master's thesis, Illinois State University, USA, October 2021.

[57] Nitin Madnani, Jill Burstein, John Sabatini, and Tenaha O'Reilly. Automated scoring of a summary-writing task designed to measure reading comprehension. In *BEA@NAACL-HLT*, 2013.

[58] Smit Marvaniya, Swarnadeep Saha, Tejas I. Dhamecha, Peter Foltz, Renuka Sindhgatta, and Bikram Sengupta. Creating scoring rubric from representative student answers for improved short answer grading. 2018.

[59] Andrew McCallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *ICML*, 1998.

[60] Tim Daniel Metzler. Computer-assisted grading of short answers using word embeddings and keyphrase extraction. Master's thesis, Hochschule Bonn-Rhein-Sieg, Germany, 2019.

[61] Margot Mieskes and Ulrike Padó. Work smart - reducing effort in short-answer grading. 2018.

[62] Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. Towards robust computerised marking of free-text responses. 01 2002.

[63] Michael Mohler and Rada Mihalcea. Text-to-text semantic similarity for automatic short answer grading. pages 567–575, 01 2009. doi: 10.3115/1609067.1609130.

[64] Hieu T. Nguyen, Joseph Yadegar, Bailey Kong, and Hai Wei. Efficient batch-mode active learning of random forest. In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pages 596–599, 2012. doi: 10.1109/SSP.2012.6319769.

[65] Rodney Nielsen, Wayne Ward, and James Martin. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15:479–501, 10 2009. doi: 10.1017/S135132490999012X.

[66] Nobal Niraula and Vasile Rus. Judging the quality of automatically generated gap-fill question using active learning. pages 196–206, 01 2015. doi: 10.3115/v1/W15-0623.

[67] Peter Norvig. How to Write a Spelling Corrector, 2016. URL `https://norvig.com/spell-correct.html`. Accessed on: 2022-10-23. [Online].

[68] Ulrike Pado and Cornelia Kiefer. Short answer grading: When sorting helps and when it doesn't. 12 2015.

[69] Bhogadi Godha Pallavi, E Ravi Kumar, Ramesh Karnati, and Ravula Arun Kumar. Lstm based named entity chunking and entity extraction. In *2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR)*, pages 1–4, 2022. doi: 10.1109/ICAITPR51569.2022.9844180.

[70] Pranjal Patil and Ashwin Agrawal. Auto grader for short answer questions. 2018.

[71] Francesc Pedró, Miguel Subosa, Axel Rivas, and Paula Valverde. Artificial intelligence in education : challenges and opportunities for sustainable development. 2019.

[72] Feddy Setio Pribadi, Teguh Bharata Adji, and Adhistya Erna Permanasari. Automated short answer scoring using weighted cosine coefficient. *2016 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, pages 70–74, 2016.

[73] Sergei Prokhorov and Victor Safronov. Ai for ai: What nlp techniques help researchers find the right articles on nlp. pages 76–765, 09 2019. doi: 10.1109/IC-AIAI48757.2019.00023.

[74] Stephen G. Pulman and Jana Z. Sukkarieh. Automatic short answer marking. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[75] Hui Qi, Yue Wang, Jinyu Dai, Jinqing Li, and Xiaoqiang Di. *Attention-Based Hybrid Model for Automatic Short Answer Scoring*, pages 385–394. 10 2019.

[76] Ashik Ahamed Aman Rafat, Mushfiqus Salehin, Fazle Rabby Khan, Syed Akhter Hossain, and Sheikh Abujar. Vector representation of bengali word using various word embedding model. In *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, pages 27–30, 2019. doi: 10.1109/SMART46866.2019.9117386.

[77] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *ArXiv*, abs/1908.10084, 2019.

[78] Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. Investigating neural architectures for short answer scoring. pages 159–168, 01 2017. doi: 10.18653/v1/W17-5017.

[79] Jorge Rodriguez-Ruiz, Alvaro Alvarez-Delgado, and Patricia Caratozzolo. Use of natural language processing (nlp) tools to assess digital literacy skills. In *2021 Machine Learning-Driven Digital Technologies for Educational Innovation Workshop*, pages 1–8, 2021. doi: 10.1109/IEEECONF53024.2021.9733779.

[80] Annalisa Wahyu Romadon, Kemas M Lhaksmana, Isman Kurniawan, and Donni Richasdy. Analyzing tf-idf and word embedding for implementing automation in job interview grading. In *2020 8th International Conference on Information and Communication Technology (ICoICT)*, pages 1–4, 2020. doi: 10.1109/ICoICT49345.2020.9166364.

[81] Keisuke Sakaguchi, Michael Heilman, and Nitin Madnani. Effective feature integration for automated short answer scoring. pages 1049–1054, 01 2015. doi: 10.3115/v1/N15-1111.

[82] Patricia Santos, Xavier Colina, Hernández-Leo Davinia, Javier Melero, and Josep Blat. Enhancing computer assisted assessment using rubrics in a qti editor. pages 303–305, 07 2009. doi: 10.1109/ICALT.2009.92.

[83] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[84] Burr Settles. Active learning literature survey. 2009.

[85] Thanveer Shaik, Xiaohui Tao, Yan Li, Christopher Dann, Jacquie McDonald, Petrea Redmond, and Linda Galligan. A review of the trends and challenges in adopting natural language processing methods for education feedback analysis. *IEEE Access*, 10:56720–56739, 2022. doi: 10.1109/ACCESS.2022.3177752.

[86] Vivek Singh, Mousumi Mukherjee, and Ghanshyam Mehta. Sentiment and mood analysis of weblogs using pos tagging based approach. volume 168, pages 313–324, 08 2011. doi: 10.1007/978-3-642-22606-9_33.

[87] Kirill Smelyakov, Danil Karachevtsev, Denis Kulemza, Yehor Samoilenko, Oleh Patlan, and Anastasiya Chupryna. Effectiveness of preprocessing algorithms for natural language processing applications. In *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC ST)*, pages 187–191, 2020. doi: 10.1109/PICST51311.2020.9467919.

[88] R. Subhashini and V. Jawahar Senthil Kumar. Shallow nlp techniques for noun phrase extraction. In *Trendz in Information Sciences Computing(TISC2010)*, pages 73–77, 2010. doi: 10.1109/TISC.2010.5714612.

[89] Md Arafat Sultan, Cristóbal Salazar, and Tamara Sumner. Fast and easy short answer grading with high accuracy. pages 1070–1075, 01 2016. doi: 10.18653/v1/N16-1123.

[90] Chul Sung, Tejas Dhamecha, and Nirmal Mukhi. *Improving Short Answer Grading Using Transformer-Based Pre-training*, pages 469–481. 06 2019.

[91] Hongye Tan, Chong Wang, Qinglong Duan, Yu Lu, Hu Zhang, and Ru Li. Automatic short answer grading by encoding student responses via a graph convolutional network. *Interactive Learning Environments*, pages 1–15, 2020.

[92] Khushboo Thaker, Lei Zhang, Daqing He, and Peter Brusilovsky. Recommending remedial readings using student knowledge state. 07 2020.

[93] Hao-Chuan Wang, Chun-Yen Chang, and Tsai-Yen Li. Assessing creative problem-solving with automated text grading. *Computers Education*, 51:1450–1466, 12 2008. doi: 10.1016/j.compedu.2008.01.006.

[94] Tianqi Wang, Naoya Inoue, Hiroki Ouchi, Tomoya Mizumoto, and Kentaro Inui. Inject rubrics into short answer grading system. pages 175–182, 01 2019. doi: 10.18653/v1/D19-6119.

[95] Mike Wu, Noah Goodman, Chris Piech, and Chelsea Finn. Prototransformer: A meta-learning approach to providing student feedback, 07 2021.

[96] Yunkai Xiao, Gabriel Zingle, Qinjin Jia, Shoaib Akbar, Yang Song, Muyao Dong, Li Qi, and Dr Gehringer. Problem detection in peer assessments between subjects by effective transfer learning and active learning. 01 2020.

[97] Zhiqiang Xing, Letong Li, and Weibin Wang. Evaluation for education of the sustainable development of engineering college students base on nlp method. In *2021 2nd International Conference on Information Science and Education (ICISE-IE)*, pages 705–709, 2021. doi: 10.1109/ICISE-IE53922.2021.00165.

[98] Xi Yang, Yuwei Huang, Fuzhen Zhuang, Lishan Zhang, and Shengquan Yu. *Automatic Chinese Short Answer Grading with Deep Autoencoder*, pages 399–404. 06 2018.

[99] Lu Yao, Zhang Pengzhou, and Zhang Chi. Research on news keyword extraction technology based on tf-idf and textrank. In *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*, pages 452–455, 2019. doi: 10.1109/ICIS46139.2019.8940293.

[100] Muhammad Yusuf and Kemas M Lhaksmana. An automated interview grading system in talent recruitment using svm. In *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*, pages 34–38, 2020. doi: 10.1109/ICOIACT50329.2020.9332109.

[101] Yuan Zhang, Rajat Shah, and Min Chi. Deep learning + student modeling + clustering: a recipe for effective automatic short answer grading. In *EDM*, 2016.