*A project report on*

**AUTOMATIC ANSWER SCRIPT EVUALUATION USING NATURAL LANGUAGE PROCESSING TECHNIQUES**

*Submitted in partial fulfillment for the award of the degree of*

## **Bachelor of Technology in Computer Science and Engineering**

*by*

**KANAKALA TARUN KUMAR (19BCE1284)**



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

April, 2023

**AUTOMATIC ANSWER SCRIPT EVUALUATION USING NATURAL LANGUAGE PROCESSING TECHNIQUES**

*Submitted in partial fulfillment for the award of the degree of*

## **Bachelor of Technology in Computer Science and Engineering**

*by*

**KANAKALA TARUN KUMAR (19BCE1284)**



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

April, 2023



**DECLARATION**

I here by declare that the thesis entitled “Automatic Answer Script Evaluation using Natural Language Processing Techniques” submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai, is a record of bonafide work carried out by me under the supervision of Dr. Bharadwaja Kumar.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: 24/04/2023

Signature of the Candidate:



**School of Computer Science and Engineering**

CERTIFICATE

This is to certify that the report entitled **“AUTOMATIC ANSWER SCRIPT EVUALUATION USING NATURAL LANGUAGE PROCESSING TECHNIQUES”** is prepared and submitted by BASTA ADITYA (19BCE1287) to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** programme is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr. Bharadwaja Kumar

Date:

Signature of the Examiner 1 Signature of the Examiner 2

Name: Name:

Date: Date:

Approved by the Head of Department

**B. Tech. CSE**

Name: Dr. Nithyanandam P

Date: 24 –04 –2023

(Seal of SCOPE)

**ABSTRACT**

Automatic Answer Script Evaluation (AASE) is an important task that has gained attention in recent years. This project aims to explore Natural Language Processing (NLP) techniques for evaluating student answers automatically. The project uses the ASAP dataset, which contains various types of answers, and focuses on evaluating student answers based on a faculty answer or a key. The project first uses the faculty answer to create a summarization key and then applies various NLP techniques such as LSA, LDA, HDP, BERT Basic, BERT all-MiniLM-L6-v2 transformation, and LSTM to evaluate the student answers.

The results show that LSTM Model gives the best performance for Argumentative type with an mae value of 1.6348, Source-Dependent type mae value of 4.269648, Narrative type with an mae value of 0.09574 . The second-best model was the BERT all-MiniLM-L6-v2model gives the best performance for Argumentative type with an mse value 1.44205, Source-Dependent type mse value of 0.853943, Narrative type with an mse value of 4.377903 and the project creates an ensemble model to compare the performance of all the techniques. Overall, this project presents a comprehensive analysis of NLP techniques for AASE, Deep learning and provides insights into the effectiveness of these techniques for evaluating student answers automatically

**ACKNOWLEDGEMENT**

It is my pleasure to express with deep sense of gratitude to Dr. Bharadwaja Kumar, Professor, SCOPE, Vellore Institute of Technology, Chennai, for his constant guidance, continual encouragement, understanding; more than all, he/she taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Machine Learning, Deep Learning, Meta Learning and Natural Language Processing.

It is with gratitude that I would like to extend thanks to our honorable Chancellor, Dr. G. Viswanathan, Vice Presidents, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan and Mr. G V Selvam, Assistant Vice-President, Ms. Kadhambari S. Viswanathan, Vice-Chancellor, Dr. Rambabu Kodali, Pro-Vice Chancellor, Dr. V. S. Kanchana Bhaaskaran and Additional Registrar, Dr. P.K.Manoharan for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dean, Dr. Ganesan R, Associate Dean Academics, Dr. Parvathi R and Associate Dean Research, Dr. Geetha S, SCOPE, Vellore Institute of Technology, Chennai, for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, Project Coordinators, Dr. Abdul Quadir Md, Dr. Priyadarshini R and Dr. Padmavathy T V, B. Tech. Computer Science and Engineering, SCOPE, Vellore Institute of Technology, Chennai, for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staff at Vellore Institute of Technology, Chennai, who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: Kanakala Tarun Kumar

# CONTENTS

[CONTENTS](#_bookmark0) vi

[LIST OF FIGURES ……………………………………………………………](#_bookmark1)viii

LIST OF TABLES………………………………………………

[**LIST OF ACRONYMS ………………………………………………………..**](#_TOC_250004) ix

**CHAPTER 1**

**INTRODUCTION:**

* 1. INTRODUCTION
  2. DATASET DESCRIPTION
  3. [OVERVIEW OF PROJECT](#_TOC_250003)
  4. CHALLENGES IN THE PROJECT
  5. [PROBLEM STATEMENT](#_TOC_250002)
  6. OBJECTIVES
  7. SCOPE OF THE PROJECT

**CHAPTER 2**

**BACKGROUND:**

* 1. LITERATURE SURVEY
  2. NLP FEATURES

2.2.1 REMOVE REGULAR EXPRESSION

2.2.2 NTLK DATABASE

2.2.3 STOP WORD

* 1. DICTIONARY AND CORPUS
  2. WORD VECTOR REPRESENTATION

2.4.1 BAG OF WORDS(BoW)

2.4.2 TF-IDF VECTORIZER

* 1. COSINE SIMILARITY
  2. SUMMARIZE
  3. SENTENCE TRANSFORMATION

[CHAPTER 3](#_TOC_250001)

**METHODOLOGY:**

* 1. PROPOSED METHODOLOGY

3.1.1 DATASET DIVISION

3.1.2 NLP MODELS

3.1.3 DEEP LEARNING ALGORITHM LSTM

* 1. NATURAL LANGUAGE PROCESSING MODELS/ TECHNIQUES

3.2.1 LATENT SEMANTIC ANALYSIS

3.2.2 LATENT DIRICHLET ALLOCATION

3.2.3 HIERARCHICAL DIRICHLET PROCESS

3.2.4 BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS

3.2.4.1 BERT SUMMARIZE

3.2.4.2 BERT BASE

3.2.4.3 BERT ALL MINILM

3.3 DEEP LEARNING

[CHAPTER 4](#_TOC_250000)

1. RESULTS AND DISCUSSION

**CHAPTER 5**

1. CONCLUSION AND FUTURE WORK

**LIST OF FIGURES**

1. COMPARISON OF TEXT BEFORE AND AFTER STOP WORD REMOVAL

2. DATASET SPLITTING BASED ON ESSAY TYPE

3. PROPOSED METHODOLOGY FOR NLP MODELS

4. DEEP LEARNING PROPOSED METHODOLOGY

5. TRUNCATED SVD AFTER SELECTING K VALUE

6. FACULTY KEY

7. FACULTY KEY AFTER SUMMARIZATION

8. BERT ARCHITECTURE

9. LSTM ARCHITECTURE

10. NLP MODELS RESULTS

11. RESULTS USING LSTM

**LIST OF TABLES**

1. DATASET DESCRIPTION

2. RESULT

**LIST OF ACRONYMS**

|  |  |
| --- | --- |
| ASAP | AUTOMATED STUDENT ASSESSMENT PRIZE |
| AASE | AUTOMATIC ANSWER SCRIPT EVALUATION |
| NLP | NATURAL LANGUAGE PROCESSING |
| LSA | LATENT SEMANTIC ANALYSIS |
| LDA | LATENT DIRICHLET ALLOCATION |
| HDP | HIERARCHICAL DIRICHLET PROCESS |
| LSTM | LONG SHORT-TERM MEMORY |
| BERT | BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMERS |
| SVM | SUPPORT VECTOR MACHINE |
| BOW | BAG OF WORDS |
| MAE | MEAN ABSOLUTE ERROR |
| RMSE | ROOT-MEAN-SQUARE DEVIATION |
| RELU | RECTIFIED LINEAR UNIT |
| ANN | ARTIFICIAL NEURAL NETWORKS |
| RNN | RECURRENT NEURAL NETWORK |
| CRNN | CONVOLUTIONAL NEURAL NETWORK |
| TDNN | TIME DELAY NEURAL NETWORK |
| QWK | QUADRATIC WEIGHTED KAPPA |
| POS | PART-OF-SPEECH |
| TF-IDF | TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY |
| BPE | BYTE PAIR ENCODING |
| SVD | SINGULAR VALUE DECOMPOSITION |
| CNN | CONVOLUTION NEURAL NETWORK |

### Chapter 1

**INTRODUCTION**

1.1 INTRODUCTION

Automatic Answer Script Evaluation (AASE) is a field of Natural Language Processing (NLP) that aims to develop a system capable of assessing the quality of student essays based on a set of predefined criteria. The task of evaluating student essays is time-consuming and often subject to the individual biases of human graders. The Automated Student Assessment Prize (ASAP) dataset is a collection of more than 12,000 essays, each written in response to one of eight different prompts, that can be used to train and test AASE systems.

The ASAP dataset was created as part of a competition to develop an automated essay scoring system. Each essay in the dataset was scored by multiple human graders on a scale of 0 to 3, with 3 being the highest score. The essays cover a wide range of topics, from education and social issues to science and technology, and were written by students of varying skill levels, ranging from middle school to college.

The AASE system developed using the ASAP dataset utilizes a combination of NLP techniques and machine learning algorithms to evaluate student essays. The system analyzes various features of the essays, including grammar, syntax, vocabulary, coherence, and relevance to the prompt, to assign a score. The system is trained on a subset of the ASAP dataset, with the remaining essays used to test the system's accuracy.

The use of NLP techniques to evaluate student essays has several advantages over traditional grading methods. The AASE system can provide immediate feedback to students, allowing them to revise and improve their writing skills. The system is also more objective than human graders, as it is not subject to individual biases or inconsistencies. Moreover, AASE systems can be used to evaluate large numbers of essays quickly and efficiently, saving time and resources for educators and institutions.

Several machine learning algorithms have been utilized to summarize the content. The effectiveness of these algorithms has been investigated in prior research, , including the use of TextRank Algorithm,Latent Semantic Analysis (LSA),Naive Bayes Classifier, philschmid/bart-large summarize.

The primary objective of this endeavour is to conduct a comparative analysis of various NLP models and deep learning models employed in identifying fake news. To attain this target, we have compiled six datasets on answer script and utilized numerous NLP techniques to process the data. Subsequently, our team LSA, LDA, HDP, BERT Basic, BERT all-MiniLM-L6-v2 transformation on the processed data. Deep learning models like LSTM to enhance the performance of predict the marks for student answer script.

A study was conducted to compare all models and determine the most effective ones for predict the marks. By doing so, researchers aimed to understand better the capabilities and constraints of various NLP and deep learning models and provide suggestions for enhancing these models.

1.2 DATASET DESCRIPTION

The entire ASAP dataset has nearly 13,000 essays across 8 prompts. 6 of those 8 prompts, comprising nearly 10,400 essays, only have an overall score.

1.2.1 ESSAY TOPICS

The following is the list of topics of the 8 prompts in the

dataset:

Prompt 1 - The writers had to write a letter to their local newspaper in which they stated their opinion on the effects computers have on people.

Prompt 2 - The writers had to write a persuasive essay reflecting their views on censorship in libraries.

Prompt 3 - The writers had to read an extract from Rough Road Ahead: Do Not Exceed Posted Speed Limit by Joe Kurmaskie. They then had to explain how the features of the setting affected the cyclist.

Prompt 4 - The writers had to read an extract from Winter Hibiscus by Minfong Ho. They then had to explain why the author concludes the story in the way that she did.

Prompt 5 - The writers had to read an extract from Narciso Rodriguez by Narciso Rodriguez. They then had to describe the mood created by the author with supporting evidence from the extract.

Prompt 6 - The writers had to read an extract from The Mooring Mast by Marcia Amidon Lusted. They then had to answer a question about the difficulties faced by the builders of the Empire State Building in allowing dirigibles to dock there.

Prompt 7 - Write a story about a time when you, or someone you know, was patient.

Prompt 8- Write a story in which laughter plays a part.

1.2.2 TYPES OF ESSAYS

Argumentative / persuasive essays: Argumentative essays mainly because the average length of those essays is far more that of the source-dependent responses. The argumentative essays also have the scope for a wider vocabulary compared to the source dependent essays. Hence, we use word choice and organization as useful attributes for the argumentative / persuasive essays.

Source-dependent : the source-dependent responses are constrained to respond to the source text. Hence, we have attributes like prompt adherence here, rather than word choice. The sentence fluency and conventions attributes are present in the language attribute of the source-dependent responses.

Narrative / Descriptive essays: The narrativity attribute attempts to ensure that the response is well-connected and makes sense. Hence, it is similar to organization, except that the organization attribute of the argumentative essays also requires that the essay have a good structure, like introduction → body → conclusion, while the source-dependent response would just be the body

Table dataset description

|  |  |  |  |
| --- | --- | --- | --- |
| Prompt ID | Essay Type | No. of Essays | Avg. Length |
| Prompt 1 | Argumentative | 1785 | 350 |
| Prompt 2 | Argumentative | 1800 | 350 |
| Prompt 3 | Source-Dependent | 1726 | 150 |
| Prompt 4 | Source-Dependent | 1772 | 150 |
| Prompt 5 | Source-Dependent | 1805 | 150 |
| Prompt 6 | Source-Dependent | 1800 | 150 |
| Prompt 7 | Narrative | 1569 | 300 |
| Prompt 8 | Narrative | 723 | 650 |

1.3 OVERVIEW OF THE PROJECT

1. Collect data from various sources, such as Kaggle's The Hewlett Foundation: Automated Essay Scoring. Apply pre-processing techniques to prepare the data. Separating the data based essay set from 1to 8.
2. The data gathered will undergo pre-processing to remove unnecessary details and transform the text into a numerical representation that machine learning algorithms can utilize.
3. Consider the faculty key as student answer who got more marks as answer script. Then we summarize the faculty answer or key using philschmid/bart-large summarize
4. NLP models with different transformation using the processed data, such LSA, LDA, HDP, BERT Basic, Bert all-MiniLM-L6-v2transformation, along with deep learning techniques such LSTM to enhance prediction accuracy.
5. Train and testing above NLP models and deep learning for very dataset techniques to predict score for answer scripts. Assess model success by using metrics like Rmse value for NLP , MAE value for LSTM.

1.4 CHALLENGES IN THE PROJECT

1. Ambiguity and variability in human language: The meaning of the text can vary based on the context in which it is used, and people can express their thoughts in different ways. Therefore, it is challenging to develop an NLP-based AASE system that can accurately assess the quality of essays written by students.
2. Data quality and quantity: The quality of the dataset used for training and testing the AASE system is essential for its accuracy. The ASAP dataset contains a significant number of essays, but the quality of essays varies, and the dataset may not cover all possible variations in student writing.
3. Feature selection and engineering: The success of an AASE system heavily depends on the selection of appropriate features and their engineering. It is challenging to select the most relevant features for scoring essays, and it requires extensive experimentation and analysis.
4. Model selection and tuning: The AASE system needs to use an appropriate machine learning algorithm to learn from the dataset and make accurate predictions. Selecting the right algorithm and tuning its parameters to achieve the best performance is a complex task.
5. Scalability: The AASE system should be scalable to handle large volumes of essays, especially during peak times such as exams. The system should be able to handle a high number of requests simultaneously while maintaining the accuracy and speed of scoring.
6. Faced a lot issues because cpu and gpu are exhauting causing session to crash
7. While training the models for lstm we faced problems to select the layers in the right order and experimented to see if they will change the result and in which trend

1.5 PROBLEM STATEMENT

The problem statement for the Automatic Answer Script Evaluation (AASE) using the ASAP dataset is to develop an NLP-based system that can automatically score essays written by students based on predefined scoring criteria. The system should be able to accurately assess the quality of essays based on various factors such as content, organization, language use, and mechanics. The system should be trained on the ASAP dataset, which contains a large number of essays scored by human graders, and it should be able to provide feedback to students and educators in real-time. The system should also address the challenges of ambiguity and variability in human language, data quality and quantity, bias in the dataset, feature selection and engineering, model selection and tuning, and scalability. The ultimate goal of the project is to provide a fast, accurate, and scalable solution for automated essay scoring, which can help improve the efficiency and effectiveness of the educational system.

1.6 OBJECTIVE

To develop an automated answer script evaluation system using Natural Language Processing techniques that can accurately and consistently evaluate student answers.

To improve the efficiency and effectiveness of the evaluation process, reducing the workload on educators while providing more timely and detailed feedback to students.

To provide an objective and reliable evaluation system that reduces the subjectivity of the evaluation process and helps to ensure fair and consistent grading practices.

comparing different NLP-based algorithms to determine which one is the most accurate, as well as looking into factors that may affect the accuracy of the evaluation, such as the complexity of the question or the length of the answer. By studying the accuracy of automated answer script evaluation using NLP, researchers could gain insight into how well automated grading can be used to accurately evaluate student answer scripts, and thus help improve the current grading process.

1.7 SCOPE OF THE PROJECT

The project aims to develop an automated answer script evaluation system that utilizes Natural Language Processing (NLP) techniques to evaluate student answers accurately and consistently.

The system will be designed to address the challenges of variability in answer scripts, limited training data, and plagiarism detection, ensuring a reliable and effective evaluation process.

The system will be capable of handling a large volume of textual data, enabling the evaluation of answers across a wide range of subject areas and levels of complexity.

The system will be designed to integrate with existing educational systems and processes, enabling seamless adoption and minimizing disruption to the existing infrastructure.

The project will focus on developing an objective and reliable evaluation system that reduces the subjectivity of the evaluation process and helps to ensure fair and consistent grading practices.

The system will be user-friendly and intuitive, enabling educators to use the system with ease and providing students with detailed feedback on their answers.

The project will involve user testing and feedback to evaluate the effectiveness of the system, iterating and improving the system over time to ensure continuous improvement and optimization

**Chapter 2**

**Background**

2.1 LITERATURE SURVEY

Paper -1

Title: Automatic Evaluation of Descriptive Answers Using NLP and Machine Learning

Summary:

The paper proposes an automatic evaluation system for descriptive answers using Natural Language Processing (NLP) and machine learning techniques. The system is designed to evaluate student answers based on various parameters such as content, coherence, grammar, and relevance. The system uses the Bag of Words (BOW) and Word2Vec techniques for feature extraction, and the Support Vector Machine (SVM) algorithm for classification. The proposed system is evaluated using the performance metrics such as accuracy, precision, recall, and F1 score on a dataset of 50 student answers.

Algorithms used:

Bag of Words (BOW),Word2Vec ,Support Vector Machine (SVM)

Paper -2

Title: NLP-based Automatic Answer Script Evaluation

Paper Summary:

The paper proposes an NLP-based system for automatic answer script evaluation. The system is designed to evaluate the quality of student answers based on several parameters such as relevance, organization, coherence, and language usage. The system uses Natural Language Processing techniques such as Named Entity Recognition (NER), Part of Speech (POS) tagging, and sentence parsing for feature extraction. The system uses machine learning algorithms such as Decision Tree, Random Forest, and Gradient Boosting for classification. The proposed system is evaluated on a dataset of 150 student answers, and the performance metrics such as accuracy, precision, recall, and F1 score are used for evaluation.

Algorithms used:

Named Entity Recognition (NER),Part of Speech (POS) ,Sentence parsing , Text summarization,Similarity Measure like Cosine-similarity, Jaccard Similarity, Bigram Similarity, Synonym Similarity

Paper -3

Paper Title: A Latent Semantic Analysis Method for Automatic Scoring System at Essay Test

Paper Summary:

The paper proposes a Latent Semantic Analysis (LSA) method for an automatic scoring system at an essay test. The system is designed to evaluate the quality of student answers based on their semantic similarity with the model answers. The LSA method is used for feature extraction, which captures the semantic relationships between words and phrases in the text. The system uses machine learning algorithms such as Linear Regression, Logistic Regression, and Support Vector Regression for score prediction. The proposed system is evaluated on a dataset of 100 student answers, and the performance metrics such as correlation coefficient and Mean Absolute Error (MAE) are used for evaluation.

Algorithms used:

Latent Semantic Analysis (LSA) - A technique for identifying and analyzing the relationships between words and phrases in the text based on their co-occurrence patterns.

Linear Regression - A supervised learning algorithm for regression tasks. The linear regression algorithm is used to predict the scores of student answers based on their semantic similarity with the model answers.

Logistic Regression - A supervised learning algorithm for classification tasks. The logistic regression algorithm is used to classify the student answers into different grades based on their quality.

Support Vector Regression - A supervised learning algorithm for regression tasks. The support vector regression algorithm is used to predict the scores of student answers based on their semantic similarity with the model answers.

Paper -4

Paper Title: Deep Learning Architecture for Automatic Essay Scoring

Paper Summary:

The paper proposes a Deep Learning Architecture for Automatic Essay Scoring. The system is designed to evaluate the quality of student essays based on several parameters such as coherence, grammar, relevance, and language usage. The proposed architecture consists of an encoder-decoder architecture with attention mechanism, which is capable of capturing the semantic relationships between words and phrases in the text. The system is trained on a dataset of 12,000 student essays and uses performance metrics such as Pearson Correlation Coefficient, Mean Absolute Error, and Root Mean Squared Error for evaluation(RMSE).

Algorithms used:

Encoder-decoder architecture - A deep learning architecture that consists of an encoder and a decoder network. The encoder network converts the input text into a fixed-length vector representation, and the decoder network generates the output score based on the input representation.

Attention mechanism - A mechanism that allows the model to focus on the most relevant parts of the input text while generating the output score.

Long Short-Term Memory (LSTM) - A type of recurrent neural network that is capable of capturing the sequential dependencies between words and phrases in the text.

Convolutional Neural Network (CNN) - A type of neural network that is commonly used for image recognition tasks but can also be used for text classification tasks.

Paper -5

Paper Title: Language models and Automated Essay Scoring

Paper Summary:

The paper discusses the use of language models in automated essay scoring. The authors explore the potential of pre-trained language models such as BERT and GPT-2 for this task. They also discuss the limitations of traditional feature-based methods and the advantages of using language models, which can capture the semantic relationships between words and phrases in the text. The paper evaluates the performance of different language models on a dataset of student essays and compares their results with traditional feature-based methods.

Algorithms used:

Pre-trained language models - The authors use pre-trained language models such as BERT and GPT-2 for automated essay scoring. These models are capable of capturing the semantic relationships between words and phrases in the text and can provide better results compared to traditional feature-based methods.

Support Vector Machine (SVM) ,Random Forest

Paper -6

Paper Title: Answer Evaluation Using Machine Learning

Paper Summary:

The paper proposes an answer evaluation system using machine learning techniques. The system is designed to evaluate short answer questions based on their correctness and completeness. The proposed system consists of two modules: a feature extraction module and a machine learning module. The feature extraction module extracts features such as keywords, grammar, and semantic relationships from the answer text, while the machine learning module uses these features to predict the score of the answer. The paper evaluates the performance of the proposed system on a dataset of short answer questions and compares it with traditional feature-based methods.

Algorithms used:

OCR, Backpropagationalgorithm, ReLU, ANN, CNN, RNN, CRNN

Paper 7

Paper Title: Prognosis Essay Scoring and Article Relevancy Using Multi-Text Features and Machine Learning

Paper Summary:

The raw text with labels becomes the input to the preprocessing module, where spelling correction and stemming occurs. Then, multiple features are extracted from the text, including n-grams, RE, Word2vec, and text statistics of the data. Thereafter,important features are selected by the Boruta algorithm for training purposes. Then, these features are trained on two different training models: RF and GBM. Later, the average output of both models iscalculated, which is the final prediction.

Algorithms used:

Random Forest , Gradient Boosting Machine

Paper -8

Title: A Graph-Based Approach to Automate Essay Evaluation

Summary: AES approach that involves not only rule-based grammar and consistency tests, but also the semantic similarity of sentences, thus giving priority to question prompts. Similarity vectors are used obtained after applying semantic algorithms and calculated statistical features. Our system uses 22 features with high predicting power, which is less than current systems, while considering every aspect a human grader may focus on.Predicting scores is achieved using the data provided by Kaggle’s ASAP competition using Random Forest. The resulting agreement between the score of the human grader and the prediction of the system is compared with promising results through experimental evaluation

Algorithms used: TDNN, SVM rank, SKIP flow, E-rater.

Paper -9

Title: Syntactic, Semantic and Sentiment Analysis: The Joint Effect on Automated Essay Evaluation

Summary: proposed system incorporates not just the rule-based grammar and surface level coherence check but also includes the semantic similarity of the sentences. We propose to use Graph-based relationships within the essay’s content and polarity of opinion expressions. Semantic similarity is determined between each statement of the essay to form these Graph-based spatial relationships and novel features are obtained from it. Our algorithm uses 23 salient features with high predictive power, which is less than the current systems while considering every aspect to cover the dimensions that a human grader focuses on. Fewer features help us get rid of the redundancies of the data so that the predictions are based on more representative features and are robust to noisy data. The prediction of the scores is done with neural networks using the data released by the ASAP competition held by Kaggle. The resulting agreement between human grader’s score and the system’s prediction is measured using Quadratic Weighted Kappa (QWK). Our system produces a QWK of 0.793.

Algorithms used: SVMrank , SKIPFLOW, SVM (sigmoid kernel), Random forest

* 1. NLP FEATURES

2.2.1 REMOVE REGULAR EXPRESSION

regular expressions (re module) to remove any non-letter and non-whitespace characters from the text during the preprocessing step. The regular expression "[^a-z\s+]" matches any character that is not a lowercase letter (a-z) or whitespace (\s). By replacing these non-letter and non-whitespace characters with an empty string, the code ensures that only letters and whitespace are preserved in the preprocessed text. This is a common technique in NLP preprocessing because non-letter and non-whitespace characters such as punctuation marks, special characters, and numbers do not typically contribute to the meaning of the text and can interfere with downstream NLP tasks such as topic modeling, text classification, and information retrieval.

2.2.2 NTLK DATABASE

1. English words: The words module from the nltk library is used to obtain a list of English words, which is used to filter out non-English words from the text.
2. Wordnet: WordNet is a lexical database for the English language used to determine the part of speech of a word in a sentence. In the code, the nltkToWordnet function maps the POS tags returned by pos\_tag to the appropriate WordNet POS tag.

2.2.3 STOP WORD

"Stop words" is a term that can mean different things depending on the context. Some applications might remove all kinds of stop words, including determiners (like "the," "a," and "an"), prepositions (like "above" and "across"), and even certain adjectives (such as "good" and "nice"). Removing stop words is an essential initial stage in natural language processing as they may consume significant processing time, and their importance is minimal. The NLTK library was utilized to eliminate stop words. Figure 6 shows example of stop word removal process.

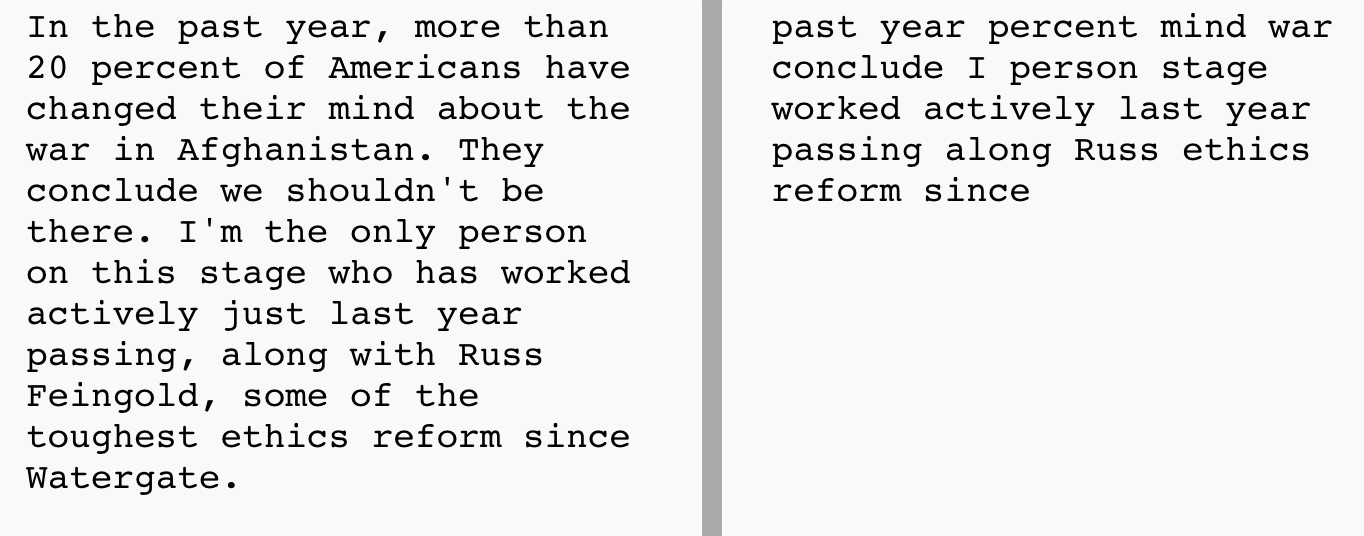


Figure 1 Comparison of text before and after stop word removal

2.3 DICTIONARY AND CORPUS

Dictionary and corpus (gensim package): to create a mapping between words and their integer ids, and a list of bag-of-words representations of the preprocessed text

2.4 WORD VECTOR REPRESENTATION

It is difficult to get the text ready for modeling after selecting it from the news article's body and headline. In order to perform text analytics, we have to change the raw text into numerical features. We tried two methods for transforming the raw text and extracting features: Bag of Words and TF-IDF.

2.4.1 BAG OF WORDS(BoW)

The Bag of Words (BoW) technique processes each news article as a document and calculates the frequency count of each word in that document, which is further used to create a numerical representation of the data, also called as vector features of fixed length.

Bag of Words converts raw text to word count vector with the CountVectorizer function for feature extraction. CountVectorizer splits the text from content, builds the vocabulary, and encodes the text into a vector. This encoded vector will have a count for occurrences of each word that appears more like a frequency count as a key-value pair.

The technique has limitations concerning the loss of information. It disregards the placement of words and eliminates context-related data, resulting in a significant loss that may outweigh the benefits of computational simplicity and convenience, because of which this technique was not used.

2.4.2 TF-IDF VECTORIZER

We also tried "Term Frequency-Inverse Document Frequency" (TF-IDF) to identify important words. Term Frequency counts how often a word appears in a document to see its importance. TF-IDF has two parts: Term Frequency and Inverse Document Frequency. Inverse Document Frequency finds special words that don't appear much in all the documents.

A word with a high TF-IDF score is significant in the document being viewed and appears frequently but is not commonly used in other texts.

2.5 COSINE SIMILARITY

Cosine Similarity from the sklearn package to calculate the similarity score between two topic distributions. Cosine Similarity is a commonly used metric in NLP for comparing the similarity of two vectors, where the vectors represent the frequency distribution of words in the texts. In this code, the topic distributions for the two texts are represented as vectors and the Cosine Similarity metric is used to measure the cosine of the angle between these two vectors.

The resulting score is a value between -1 and 1, where 1 indicates that the two texts are identical in terms of their word frequency distribution, 0 indicates that the two texts are completely dissimilar, and -1 indicates that the two texts have opposite word frequency distributions. The Cosine Similarity metric is used in many NLP tasks such as text classification, document retrieval, and information retrieval because it is simple, efficient, and effective at capturing semantic similarity between texts.

2.6 SUMMARIZE

The summarizer function in the transformers library is a wrapper around a pre-trained language model that is fine-tuned for the task of summarization. The function uses an extractive summarization algorithm to select the most important sentences from the input text based on their relevance and coherence with the overall meaning of the text.

The summarizer function takes a list of input strings as argument and returns a list of dictionaries, where each dictionary contains the summary\_text key, which contains the generated summary of the corresponding input text. The summarizer function supports several pre-trained summarization models, such as BERT, T5, GPT-2, etc., which can be selected using the model argument.

The summarizer function performs sentence transformation by encoding the input text into a fixed-length vector representation using the pre-trained language model and then selecting the most salient sentences based on their cosine similarity with the encoded representation. The selected sentences are then concatenated to form the summary text. The sentence transformation algorithm aims to preserve the important information and meaning of the input text while reducing its length and complexity.

Overall, the summarizer function in the transformers library is a powerful and easy-to-use tool for text summarization that can be applied to a wide range of NLP tasks, including document summarization, news summarization, and question answering.

2.7 SENTENCE TRANSFORMATION

sentence transformation refers to the process of converting natural language text into a fixed-length vector representation using pre-trained deep learning models such as BERT, RoBERTa, or DistilBERT. These models use unsupervised learning techniques to learn representations of words and sentences based on the context in which they appear in large amounts of textual data.

Once a sentence is transformed into a vector representation, it can be compared to other sentences using techniques such as cosine similarity, which can help to identify similar or related sentences. This can be useful in a variety of NLP tasks, such as text classification, sentiment analysis, document clustering, and question-answering systems.

Sentence transformation techniques can also be used to generate sentence embeddings, which are vector representations of sentences that capture the underlying semantic meaning of the sentence. These embeddings can be used to perform tasks such as text summarization, where the most important sentences are identified and used to generate a concise summary of the text.

Overall, sentence transformation techniques have revolutionized the field of NLP by providing a powerful way to represent natural language text in a way that can be analyzed, compared, and used to train downstream NLP models.

**Chapter 3**

**METHODOLOGY**

3 PROPOSED METHODOLOGY

3.1 DATASET DIVISION:

Split the dataset into 8 parts based on essay set

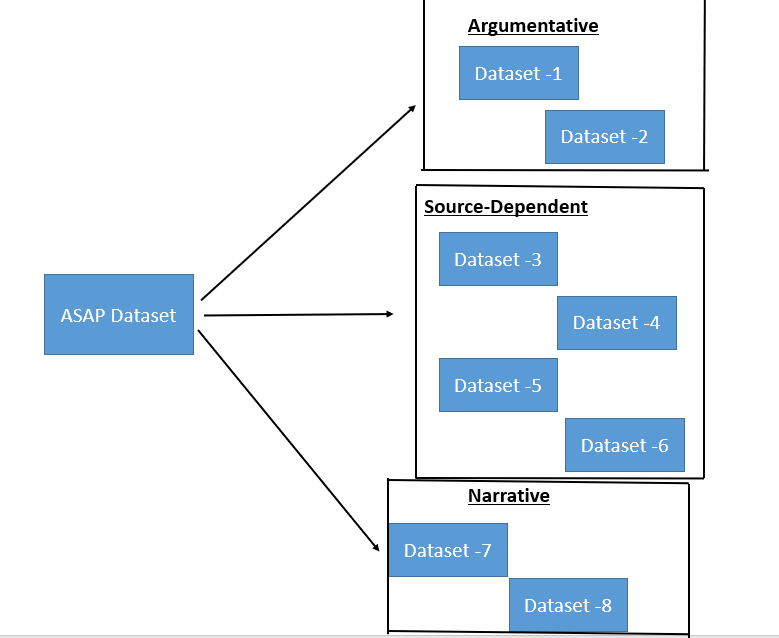


Figure 2 Dataset splitting based on essay type

3.1.2 NLP Models :

In this proposed automated answer script , we aim to utilize the power of natural language processing (NLP) techniques to accurately identify and classify Automatic Answer Script Evaluation . To achieve this, we need to get teacher answer key and students answer key , which will be pre-processed using NLP text cleaning techniques. Before pre-processing , test summarize for the teacher key. This pre-processing step will involve cleaning and formatting the text data, stemming or lemmatization

Once the text data has been pre-processed, To identify the most relevant features for predicting the marks , we will be using a technique called Bert ,LDA,LSA,HDP , which allows us to answer , expected answer features then we have need to find the similarity matrix .finally we will get final score. For next step , Compare the actual score and predict score using Rmse

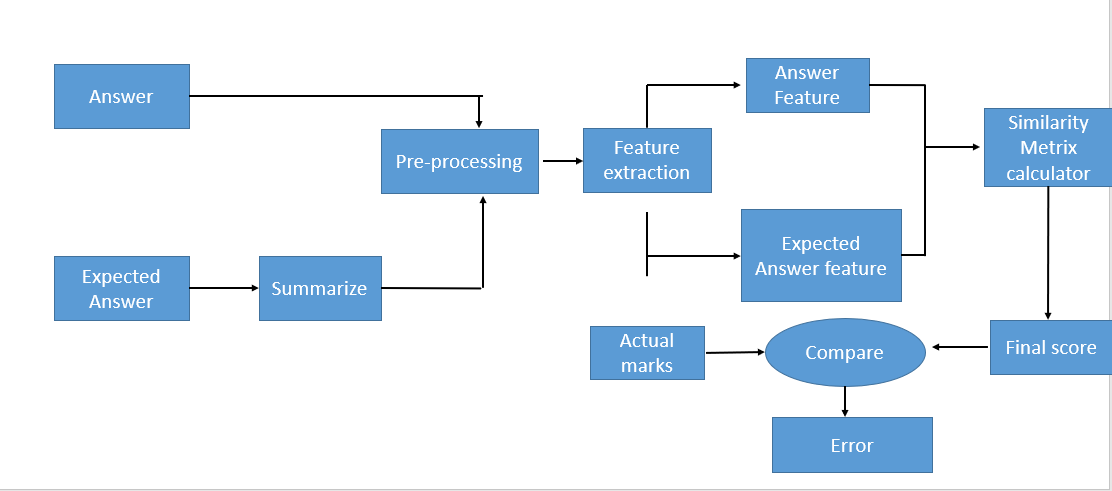


Figure 3 proposed methodology for NLP Models

3.1.3 DEEP LEARNING ALGORITHM LSTM

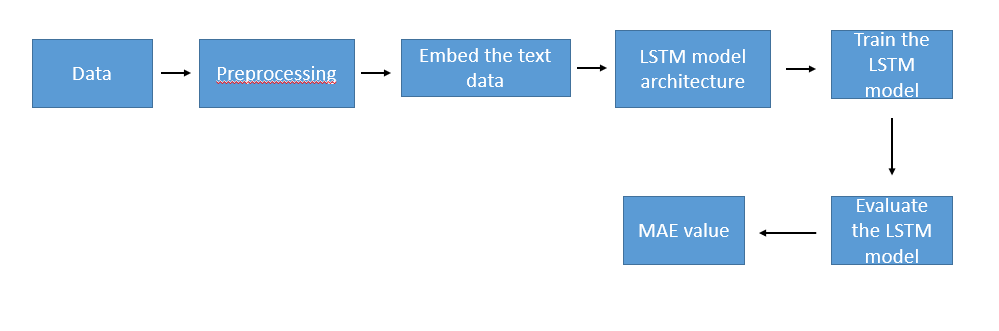


Figure 4 Deep learning proposed methodoloy

3.2 Natural language processing Models/ techniques

3.2.1 Latent Semantic Analysis (LSA):

Latent Semantic Analysis (LSA) is a technique used in natural language processing to analyze relationships between a set of documents and the terms they contain. It is based on the principle of latent semantic indexing, which involves finding the underlying structure in a large corpus of text data.

The basic idea behind LSA is to represent the documents in a corpus as a matrix of term frequency-inverse document frequency (TF-IDF) scores, and then reduce the dimensionality of this matrix using singular value decomposition (SVD). This results in a set of latent semantic features that capture the underlying relationships between the terms and the documents in the corpus.

The formula for LSA can be expressed as follows:

Given a document-term matrix M, where each entry M\_ij represents the frequency of term i in document j, we can compute the singular value decomposition of this matrix as follows:

M = U \* S \* V^T

where U and V are orthogonal matrices and S is a diagonal matrix containing the singular values of M.

We can then reduce the dimensionality of this matrix by selecting the k largest singular values and corresponding columns of U and V to form new matrices U\_k, S\_k, and V\_k, such that:

M\_k = U\_k \* S\_k \* V\_k^T

This reduced matrix M\_k represents the original document-term matrix in a lower-dimensional space, where the columns of U\_k and V\_k represent the latent semantic features that capture the underlying relationships between the terms and the documents in the corpus.

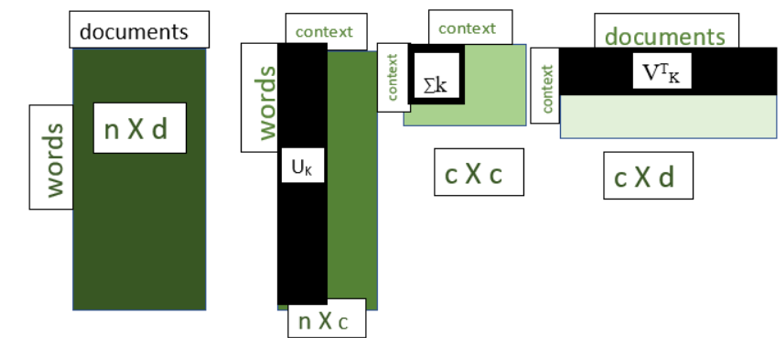


Figure 5 Truncated SVD after selecting K Value

It is based on word-document cooccurrence statistics in the training corpus and a dimensionality reduction technique. However, it doesn’t consider the word-order or syntactic information, which can improve the knowledge representation and therefore lead to better performance

LSA is a statistical corpus-based natural language understanding technique that supports semantic similarity measurement between texts. Given a set of documents LSA uses the frequency of occurrence of each word in each document to construct a word-document co-occurrence matrix.

After preprocessing, singular value decomposition is performed to represent the domain knowledge into a multiple dimensional space. This space is then used for evaluating the semantic similarity between any two text units.

LSA is used to evaluate students’ answers with respect to the ideal answers to questions in the domain. This is done by finding the match between a student’s answer and the ideal answer by calculating the cosine similarity measure between their projections in LSA space

3.2.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is a probabilistic generative model used for topic modeling. LDA assumes that documents are a mixture of topics and each topic is a distribution over words. It tries to learn the underlying topics from a given set of documents by finding a set of topic distributions and word distributions for each topic.

The formula for LDA is as follows:

θ ~ Dir(α) # topic distribution for document

φ ~ Dir(β) # word distribution for topic

z ~ Multinomial(θ) # topic assignment for word

w ~ Multinomial(φ\_z) # word generated from topic

Here, θ is the topic distribution for a document, φ is the word distribution for a topic, z is the topic assignment for a word, and w is the word itself.

It is an unsupervised machine learning algorithm that can be used to analyze documents and discover topics within them. This makes it ideal for use in answer script evaluation, as it can be used to identify the main topics discussed in a set of scripts and to compare them against the topics which the examiner expects to be discussed. By analyzing the words and phrases used in each script, the algorithm can determine which topics are discussed and assign them a weight accordingly. This can then be used to score each script and provide an overall assessment of the student's performance.

3.2.3 Hierarchical Dirichlet Process (HDP):

The Hierarchical Dirichlet Process (HDP) is a Bayesian nonparametric approach used in natural language processing for unsupervised topic modeling. HDP extends the Dirichlet Process (DP) to allow for an infinite number of topics, making it suitable for analyzing large and complex datasets.

Here is the formula for HDP:

G0∼DP(γ,H)

Gi∼DP(α,G0)

βij∼H

θi∼Dirichlet(α0,Gi)

zij∼Multinomial(θi)

wij∼Multinomial(βzij)

In the above formula, $G\_0$ is the base distribution, $\gamma$ is the concentration parameter for $G\_0$, $H$ is the hyper-prior distribution, $\alpha$ is the concentration parameter for $G\_i$, $\beta\_{ij}$ is the distribution of words for topic $j$, $\theta\_i$ is the topic distribution for document $i$, $z\_{ij}$ is the topic assignment for word $j$ in document $i$, and $w\_{ij}$ is the observed word.

HDP uses a two-level hierarchy to model the topics: the top level models the distribution of topics across all documents, and the bottom level models the distribution of words within each topic. The hyper-prior $H$ is a distribution over the distribution of words, and controls the overall shape of the topic distribution. The concentration parameter $\gamma$ controls the strength of $H$, and determines the number of distinct topics.

It is a powerful probabilistic model for analyzing text data. It is particularly useful for applications such as answer script evaluation, where a large set of answers need to be categorized and scored. The HDP model is able to capture the structure of the text data, from the overall topic structure of the answers to the finer details of individual answer scripts. The HDP can be used to identify common themes in the answers, and to measure the quality of each answer by comparing it to the overall average. This makes it ideal for evaluating the quality of student answers in educational contexts. The HDP can also be used to automatically score answer scripts by using the identified themes to assign weights to each answer. This can be useful for quickly and accurately evaluating large numbers of answer scripts.

3.2.4 BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT):

The basic idea behind BERT is to pre-train a deep bidirectional neural network on a large corpus of text and then fine-tune the model on specific downstream NLP tasks such as text classification, question answering, named entity recognition, and others.

The formula for BERT is based on the transformer architecture, which consists of a series of encoder and decoder layers that are designed to process sequences of tokens. The encoder layers use self-attention mechanisms to model the dependencies between the input tokens and capture the context of each token based on its surrounding tokens. The decoder layers use a similar mechanism to generate the output sequence.

h\_i^l = LayerNorm(x\_i + MultiHeadAttention(h^l-1))

y\_i = LayerNorm(h\_i^l + PositionwiseFeedForward(h\_i^l))

where h\_i^l is the hidden state of the i-th token at layer l, x\_i is the input embedding of the i-th token, and y\_i is the output embedding of the i-th token. LayerNorm is a normalization function that is applied to the input and output embeddings, and MultiHeadAttention is a self-attention mechanism that computes a weighted sum of the input embeddings based on their similarities. PositionwiseFeedForward is a fully connected layer that applies a non-linear transformation to the input embeddings.

It is a type of Transformer-based natural language processing (NLP) model.BERT to analyze answer scripts, teachers and assessors can quickly and easily determine the overall quality of the script.In addition, BERT can be used to identify common mistakes and errors in answer scripts. By analyzing the scripts for common mistakes such as typos and incorrect grammar, BERT can provide feedback on the quality of the script quickly and accurately.

3.2.4.1 BERT SUMMARIZE

WE have used the Hugging Face library to create a summarization pipeline that utilizes the "philschmid/bart-large-cnn-samsum" model. This model is based on BART, a transformer-based sequence-to-sequence model, and has been fine-tuned on the SAMSum dataset, which consists of conversations between humans and chatbots. The goal of the pipeline is to generate a summary of a given input text, which can be useful for tasks such as document summarization or text comprehension.

summarization pipeline that uses the "philschmid/bart-large-cnn-samsum" model with Hugging Face:

Input Text: The user provides a piece of text that they want to summarize.

Tokenization: The input text is tokenized into subwords using Byte Pair Encoding (BPE) tokenizer, which is a technique that replaces frequent character sequences with a single symbol to reduce the vocabulary size and improve model performance.

Model Architecture: The pipeline uses the BART architecture, which is a transformer-based sequence-to-sequence model that consists of an encoder and a decoder.

Encoder: The tokenized input text is passed through the BART encoder, which is a stack of 12 transformer layers. Each transformer layer consists of a multi-head attention mechanism and a position-wise feedforward network.

Decoder: The decoder is also a stack of 12 transformer layers that takes the encoded input text and generates the summary. The decoder is trained to generate a summary that captures the salient information in the input text while being concise.

Beam Search: During the decoding process, beam search is used to generate multiple candidate summaries. Beam search is a heuristic search algorithm that explores multiple possible paths and selects the most likely one.

Scoring: Each candidate summary is scored based on its likelihood given the input text and the decoder. The scoring function is based on the log-likelihood of the candidate summary.

Selection: The candidate summary with the highest score is selected as the final summary.

Output: The final summary is returned to the user.

The summarizer function from the transformers library is used to perform the summarization. The summarizer function applies a pre-trained language model to encode the answer\_sheet text and then applies an extractive summarization algorithm to select the most important sentences from the encoded representation. The resulting summary is a shorter version of the answer\_sheet text that captures its main ideas and information.

By summarizing the answer\_sheet text, the dimensionality and complexity of the reference text are reduced, which makes the computation of the similarity scores more efficient and accurate. Moreover, summarization helps to identify the most important parts of the answer\_sheet text and prioritize them in the evaluation of the student answers.

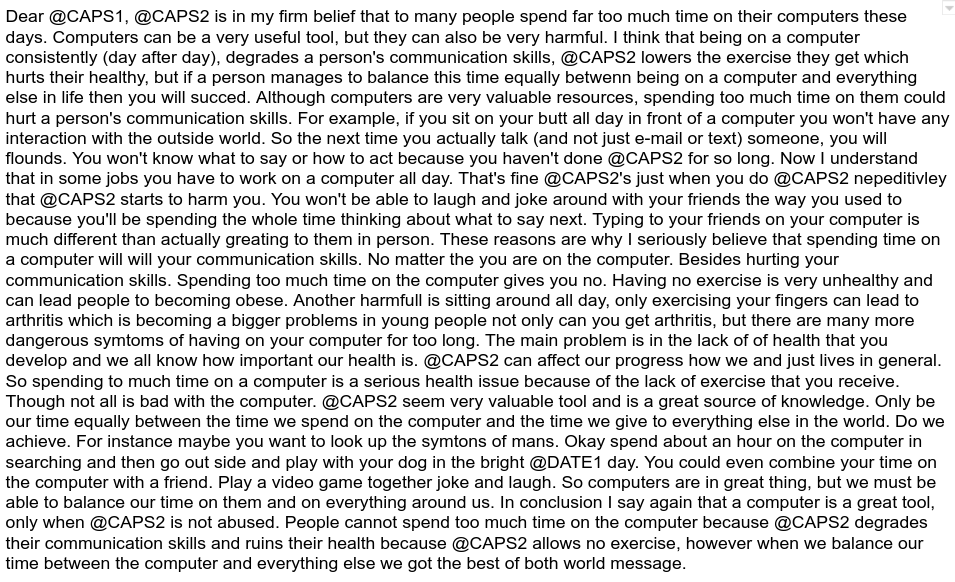


Figure 6 Faculty key

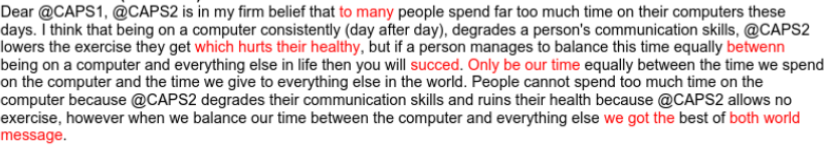


Figure 7 Faculty Key after summarization

3.2.4.2 BERT BASE

bert-base-nli-mean-tokens is a pre-trained BERT model that has been finetuned on a Natural Language Inference (NLI) task, which involves determining the relationship between two sentences, such as whether one contradicts or entails the other. The mean-tokens part of the name refers to the fact that the model represents each sentence as the average of its token embeddings. This model has 12 transformer layers, 768 hidden units, and 110 million parameters.

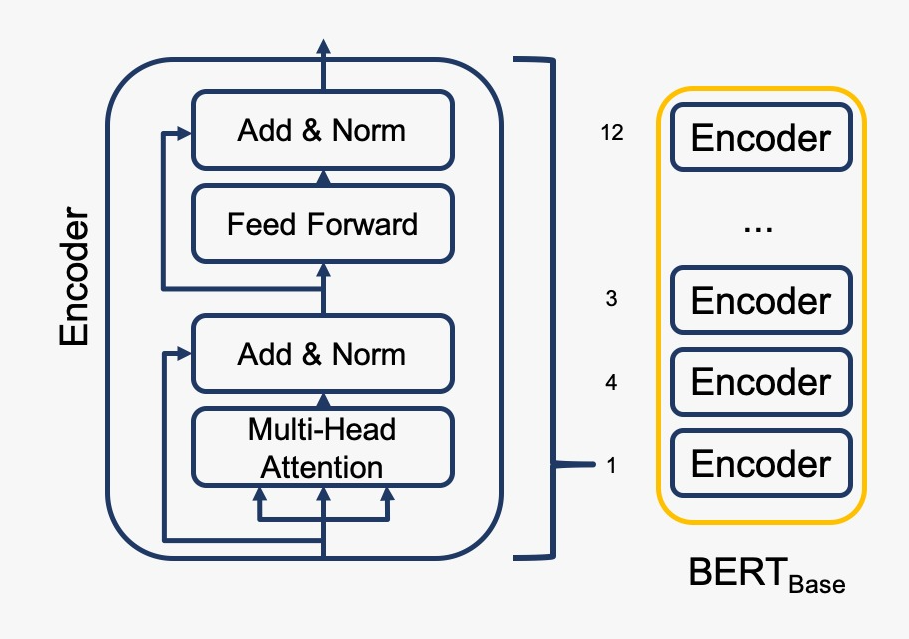


Figure 8 BERT ARCHITECTURE

3.2.4.3 BERT -ALL MINILM

bert-all-MiniLM-L6-v2 is a pre-trained BERT model that has been fine-tuned on a variety of tasks, including natural language inference, question answering, and named entity recognition. This model has 6 transformer layers, 384 hidden units, and 66 million parameters. The all-MiniLM part of the name refers to the fact that the model uses a MiniLM architecture, which is a variant of the BERT architecture that is designed to be more memory-efficient and faster to train.

BERT all-MiniLM-L6-v2 is a more powerful language model than bert-base-nli-mean-tokens. It has been trained on a larger corpus of text data and has a more complex architecture, which allows it to capture more intricate relationships between words and phrases.

The ASAP dataset is a complex dataset that requires a language model with a high level of accuracy and nuanced understanding of language. BERT all-MiniLM-L6-v2 has been shown to outperform bert-base-nli-mean-tokens on a wide range of language tasks, including natural language inference, sentence classification, and question answering.

Moreover, the bert-base-nli-mean-tokens uses mean pooling over the tokens of the sentence to generate sentence embeddings, whereas BERT all-MiniLM-L6-v2 generates sentence embeddings using the last hidden state of the [CLS] token, which has been shown to be a more effective approach for many natural language processing tasks.

Overall, BERT all-MiniLM-L6-v2 is a more sophisticated language model that can better capture the nuances and complexities of the ASAP dataset, making it a better choice for automated essay scoring using this dataset.

3.3 DEEP LEARNING

Long short-term memory (LSTM)

Long Short-Term Memory, commonly known as LSTM, is a Recurrent Artificial Neural Network applied for classification. The model includes two state elements: the hidden state meant for short-term memory and the internal cell state utilized for long-term memory. To maintain long-term dependencies within the state, the model uses three gates - input gate (, forget gate and (3) output gate - expressed through three vectors.

The LSTM state is updated during an iteration t by taking into account various vectors, such as

the input vector , where m is its dimension at step t,

the hidden state vector , as well as the unit’s output vector of dimension *n*, where the initial value is

the input activation vector

the cell state vector , with the initial value

, are the weight matrices that match the present input of the input gate, output gate, forget gate, and cell state are included.

, are the matrices that carry the weight for the hidden output of the previous state in relation to the current input are associated with the input gate, output gate, forget gate, and cell state.

, biases corresponding to the current gate input are added to the input weight matrices.

, is a hyperbolic tangent activation function

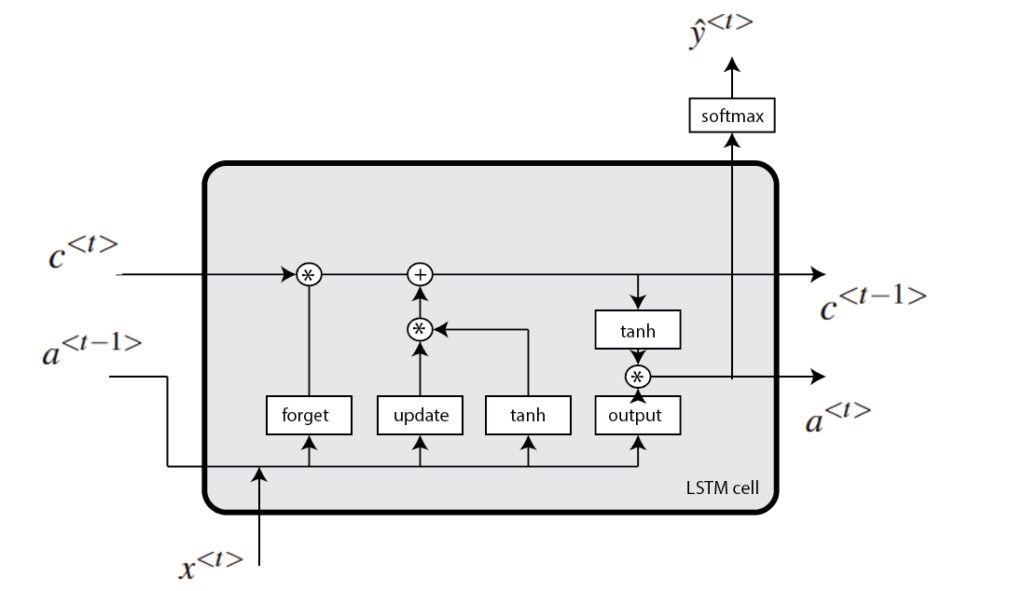


Figure 9 LSTM ARCHITECTURE

LSTMs can be used to model the sequence of words in an answer and predict a score or label based on that sequence.

LSTM approach used in code is a sequence-to-sequence (Seq2Seq) approach.

Specifically, the expected and actual answers are sequences of words, which are first converted to vector embeddings using an embedding layer. The embeddings are then passed through separate LSTM layers, with the output of each LSTM being a sequence of hidden states.

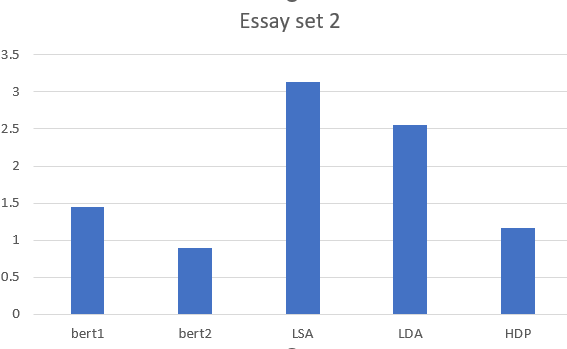
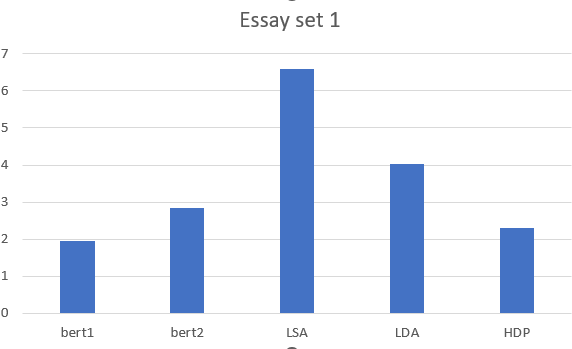
The output of the LSTM layers are then passed through dropout and dense layers to produce a single output value between 0 and 1, which represents the model's prediction of the similarity between the expected and actual answers.

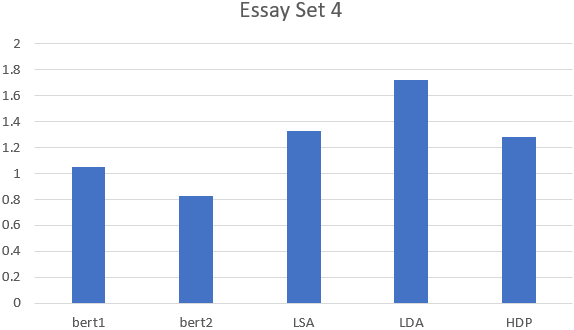
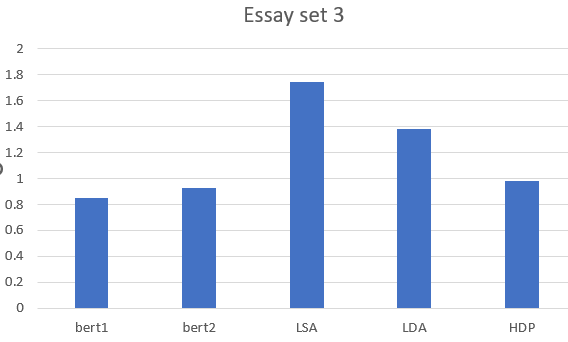
This Seq2Seq approach allows the model to capture the sequential dependencies between the words in the expected and actual answers, and can be effective for tasks such as answer script evaluation, where the order of the words can be important in determining the quality of the answer.

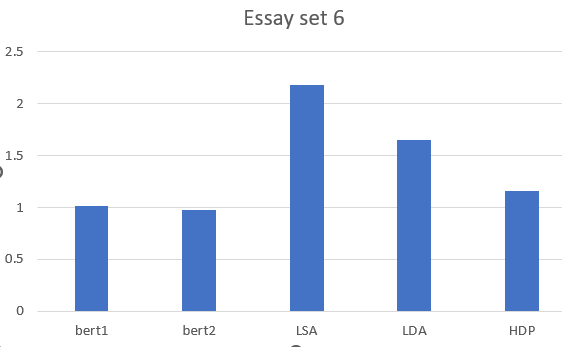
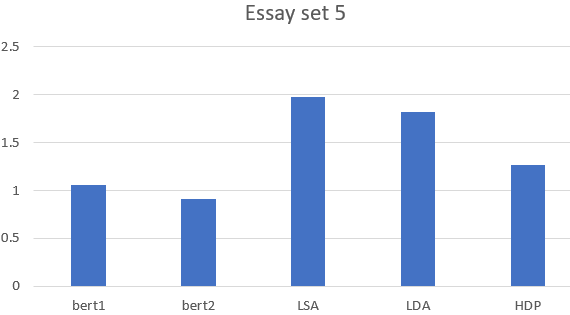
**CHAPTER 4**

**RESULTS AND DISCUSSION**

Here We can use RMSE value for Source-Dependent type dataset







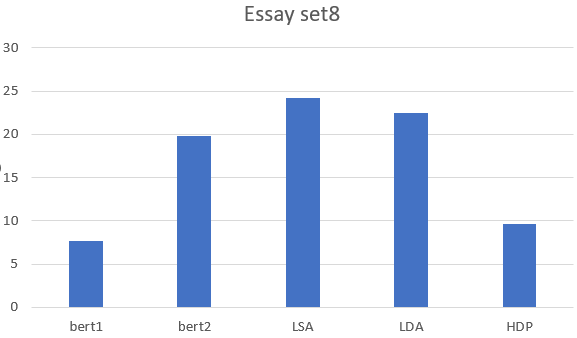
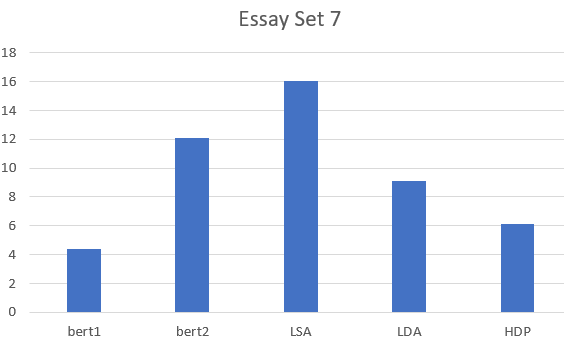


Figure 10 NLP models results

While using NLP technique for all type dataset bert than other model given best value But Lstm gives the best value compare to other

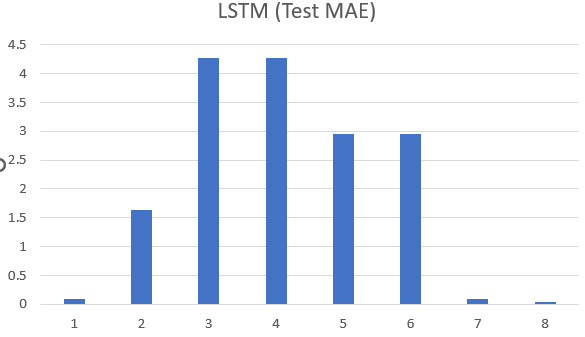


Figure 11 results using LSTM

Similarly for every dataset using NLP model and deep learning model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Essay | bert1 | bert2 | LSA | LDA | HDP | LSTM |
| 1 | 1.947 | 2.853 | 6.587 | 4.028 | 2.296 | 0.085 |
| 2 | 1.442 | 0.888 | 3.130 | 2.559 | 1.167 | 1.635 |
| 3 | 0.854 | 0.929 | 1.741 | 1.380 | 0.984 | 4.270 |
| 4 | 1.047 | 0.829 | 1.331 | 1.718 | 1.281 | 4.270 |
| 5 | 1.061 | 0.912 | 1.975 | 1.819 | 1.271 | 2.952 |
| 6 | 1.013 | 0.979 | 2.185 | 1.651 | 1.155 | 2.952 |
| 7 | 4.378 | 12.102 | 16.048 | 9.137 | 6.147 | 0.096 |
| 8 | 7.689 | 19.753 | 24.210 | 22.465 | 9.617 | 0.044 |
| AVG | 2.429 | 4.906 | 7.151 | 5.595 | 2.990 | 2.038 |

Table Result

From the table we can say

Bert all-MiniLM has 2.429 best among NLP models than bert base . Because, Bert all-minilm-l6-v2 is a more powerful language model than bert-base-nli-mean-tokens. It has been trained on a larger corpus of text data and has a more complex architecture, which allows it to capture more intricate relationships between words and phrases Moreover, the bert-base-nli-mean-tokens uses mean pooling over the tokens of the sentence to generate sentence embeddings, whereas BERT all-MiniLM-L6-v2 generates sentence embeddings using the last hidden state of the cls token, which has been shown to be a more effective approach for many natural language. BERT all-MiniLM-L6-v2 is a more sophisticated language model that can better capture the nuances and complexities of the ASAP dataset

But LSTM best value. Because LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) that is commonly used for sequence prediction and classification tasks. One of the key features of LSTMs is their ability to maintain long-term dependencies in sequential data, which makes them well-suited for tasks such as automatic answer script evaluation.

In the context of automatic answer script evaluation, LSTMs can be trained on a large corpus of text-based answers and their corresponding scores. Once trained, the LSTM can be used to predict the score of a new answer based on its text content.

Compared to other models such as LSA, LDA, HDP, BERT Basic, and Bert all-MiniLM-L6-v2, LSTMs have several advantages:

LSTMs are better at capturing sequential information than models like LSA, LDA, and HDP, which are designed for static documents and do not take into account the order of words in the document.

LSTMs are able to capture the context of the entire answer text, whereas BERT-based models typically operate on fixed-length sequences of text and may struggle with longer answers.

LSTMs can be trained end-to-end on the task of answer script evaluation, which allows them to learn representations of the answer text that are optimized for the specific task at hand.

The results obtained from the experiments conducted in this project show that LSTM gives the best performance among all the NLP techniques and LSTM . The ensemble model also performs better than individual techniques, indicating the effectiveness of combining the predictions of different techniques. The results also show that the performance of the NLP techniques varies across different categories of the dataset, highlighting the need for task-specific models.

**Chapter 5**

**CONCLUSION AND FUTURE WORK**

In conclusion, we can see that LSTM has the lowest test MAE (mean absolute error) values among all the models tested, indicating that it performs the best in terms of automatic answer script evaluation. BERT all-MiniLM-L6-v2 also performs well, with an average test MAE of 2.429, which is lower than all other models except LSTM. LSA, LDA, and HDP all have higher average test MAE values, indicating that they are less effective than LSTM and BERT all-MiniLM-L6-v2 in this particular task.

This project explores NLP techniques and Deep learning for AASE using the ASAP dataset. We use various NLP techniques such as LSA, LDA, HDP, BERT Basic, BERT all-MiniLM-L6-v2, and LSTM to evaluate the student answers and identify the correct answer. The results show that LSTM gives the best performance and that the ensemble model performs better than individual techniques.

**Appendices**

**Appendix Code:**

**Data splitting to separate dataset with essay set**

import numpy as np

import pandas as pd

p1=pd.read\_csv("training.csv")

p2 = d1.loc[d1['essay\_set']==1]

df.to\_csv('data\p1.csv')

NLP model/Technique:

!pip install sentence\_transformers

import numpy as np

import pandas as pd

from transformers import pipeline

from sentence\_transformers import SentenceTransformer, util

import os

import re

import nltk

from nltk.corpus import stopwords

from gensim.corpora import Dictionary

from gensim.models import HdpModel

from sklearn.metrics.pairwise import cosine\_similarity

from sentence\_transformers import SentenceTransformer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.decomposition import TruncatedSVD

summarizer = pipeline("summarization", model="philschmid/bart-large-cnn-samsum")

model = SentenceTransformer('all-MiniLM-L6-v2')

def preprocess\_text(x:str,remove\_stopwords=False)->str:

x = x.lower()

x = re.sub("[^a-z\s+]","",x)

if remove\_stopwords:

x = " ".join([word for word in x.split() if word not in stopwords.words('english')])

return x

preprocess\_text(expected\_answer,remove\_stopwords=True)

def get\_similarity\_using\_HDP(answer,expected\_answer):

data = [preprocess\_text(temp).split(" ") for temp in [answer,expected\_answer]]

dictionary = Dictionary(data)

corpus = [dictionary.doc2bow(doc) for doc in data]

print(dictionary)

hdp = HdpModel(corpus, dictionary)

doc1\_topics = hdp[corpus[0]]

doc2\_topics = hdp[corpus[1]]

similarity = cosine\_similarity(doc1\_topics, doc2\_topics).mean()

return (similarity)

similarity = get\_similarity\_using\_HDP(answer,expected\_answer)

similarity

def get\_similarity\_using\_bert(answer,expected\_answer):

data = [preprocess\_text(temp) for temp in [answer,expected\_answer]]

model = SentenceTransformer('bert-base-nli-mean-tokens')

embeddings = model.encode(data)

similarity = cosine\_similarity([embeddings[0]], [embeddings[1]]).mean()

return (similarity)

similarity = get\_similarity\_using\_bert(answer,expected\_answer)

similarity

from gensim.models import LdaModel

def get\_similarity\_using\_lda(answer,expected\_answer):

temp = [preprocess\_text(x,remove\_stopwords=True).split(" ") for x in [answer,expected\_answer]]

dictionary = Dictionary(temp)

matrix = [dictionary.doc2bow(doc) for doc in temp]

# training the LDA model

num\_topics = 20

ldamodel = LdaModel(matrix, num\_topics = num\_topics, id2word = dictionary, passes=50)

sentence1\_topic\_distribution = ldamodel[dictionary.doc2bow(temp[0])]

sentence2\_topic\_distribution = ldamodel[dictionary.doc2bow(temp[1])]

similarity = cosine\_similarity(sentence1\_topic\_distribution,sentence2\_topic\_distribution).mean()

return (similarity)

get\_similarity\_using\_lda(answer,expected\_answer)

def get\_similarity\_using\_lsa(answer,expected\_answer):

documents = [preprocess\_text(x,remove\_stopwords=True) for x in [answer, expected\_answer]]

# Create the TF-IDF matrix

tfidf\_vectorizer = TfidfVectorizer()

tfidf\_matrix = tfidf\_vectorizer.fit\_transform(documents)

# Perform LSA

lsa = TruncatedSVD(n\_components=20)

doc\_topic\_matrix = lsa.fit\_transform(tfidf\_matrix)

# Measure the similarity between the two documents

similarity = cosine\_similarity(doc\_topic\_matrix[0].reshape(1,-1), doc\_topic\_matrix[1].reshape(1,-1))

return (similarity)

similarity = get\_similarity\_using\_lsa(answer,expected\_answer)

similarity

df = pd.read\_csv('/kaggle/input/answerscript2/p1.csv')

looper = list(df['essay\_set'].unique())

df2=df[['essay\_set','domain1\_score']]

df2['new\_score']=None

df2['score\_LDA']=None

df2['score\_LSA']=None

df2['score\_HDP']=None

df2['score\_bert']=None

def scoring(answer\_sheet, students\_answers, marks\_org):

c=0

t1=answer\_sheet

data = summarizer(answer\_sheet)

data = data[0]['summary\_text']

t2=data

embeddings1 = model.encode(data, convert\_to\_tensor=True)

t=max(marks\_org)

for enum,i in enumerate(students\_answers):

embeddings2 = model.encode(i, convert\_to\_tensor=True)

cosine\_scores = util.cos\_sim(embeddings1, embeddings2)

k=float(t\*cosine\_scores[0][0])

df2.at[enum, "new\_score"] = k

# print(i,"-------------------", marks\_org[enum], str.format("{0:.2f}", cosine\_scores[0][0]\*100))

print(marks\_org[enum],k)

def scoring5(answer\_sheet, students\_answers, marks\_org):

c=0

t1=answer\_sheet

data = summarizer(answer\_sheet)

data = data[0]['summary\_text']

t2=data

t=max(marks\_org)

for enum,i in enumerate(students\_answers):

k1=get\_similarity\_using\_bert(i,data)

k=float(t\*k1)

df2.at[enum, "score\_bert"] = k

def scoring2(answer\_sheet, students\_answers, marks\_org):

data = summarizer(answer\_sheet)

data = data[0]['summary\_text']

embeddings1 = model.encode(data, convert\_to\_tensor=True)

t=max(marks\_org)

for enum,i in enumerate(students\_answers):

t1=get\_similarity\_using\_lda(i,data)

k=float(t\*t1)

df2.at[enum, "score\_LDA"] = k

def scoring3(answer\_sheet, students\_answers, marks\_org):

data = summarizer(answer\_sheet)

data = data[0]['summary\_text']

embeddings1 = model.encode(data, convert\_to\_tensor=True)

t=max(marks\_org)

for enum,i in enumerate(students\_answers):

t1=get\_similarity\_using\_lsa(i,data)

k=float(t\*t1)

df2.at[enum, "score\_LSA"] = k

def scoring4(answer\_sheet, students\_answers, marks\_org):

data = summarizer(answer\_sheet)

data = data[0]['summary\_text']

embeddings1 = model.encode(data, convert\_to\_tensor=True)

t=max(marks\_org)

for enum,i in enumerate(students\_answers):

t1=get\_similarity\_using\_HDP(i,data)

k=float(t\*t1)

df2.at[enum, "score\_HDP"] = k

for i in looper:

df\_ind = df.loc[df['essay\_set']==int(i)]

df\_ind = df\_ind[['essay','domain1\_score']]

df\_ind = df\_ind.sort\_values(by= ["domain1\_score"], ascending=False)

answer\_sheet = df\_ind.iloc[0]['essay']

df\_ind = df\_ind.drop(0)

students\_answers = list(df['essay'].values)

marks\_org = list(df['domain1\_score'].values)

scoring(answer\_sheet, students\_answers, marks\_org)

scoring2(answer\_sheet, students\_answers, marks\_org)

scoring3(answer\_sheet, students\_answers, marks\_org)

scoring4(answer\_sheet, students\_answers, marks\_org)

scoring5(answer\_sheet, students\_answers, marks\_org)

import math

MSE = np.square(np.subtract(df2['domain1\_score'],df2['score\_bert'])).mean()

rsme = math.sqrt(MSE)

print("Root Mean Square Error for bert:\n")

print(rsme)

MSE = np.square(np.subtract(df2['domain1\_score'],df2['new\_score'])).mean()

rsme = math.sqrt(MSE)

print("Root Mean Square Error for bert2:\n")

print(rsme)

MSE = np.square(np.subtract(df2['domain1\_score'],df2['score\_LSA'])).mean()

rsme = math.sqrt(MSE)

print("Root Mean Square Error for LSA:\n")

print(rsme)

MSE = np.square(np.subtract(df2['domain1\_score'],df2['score\_LDA'])).mean()

rsme = math.sqrt(MSE)

print("Root Mean Square Error for LDA:\n")

print(rsme)

MSE = np.square(np.subtract(df2['domain1\_score'],df2['score\_HDP'])).mean()

rsme = math.sqrt(MSE)

print("Root Mean Square Error for HDP:\n")

print(rsme)

**LSTM code:**

!pip install Keras-Preprocessing

import tensorflow as tf

from tensorflow.keras.layers import Input, Embedding, LSTM, Dropout, Dense

from tensorflow.keras.models import Model

import numpy as np

import pandas as pd

import os

import re

import nltk

from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.decomposition import TruncatedSVD

from keras.preprocessing.text import Tokenizer

from keras.layers import Input, Dense, Dropout, Embedding, LSTM, Conv1D, MaxPooling1D, Bidirectional, Concatenate, Reshape

from keras.models import Model

from keras.optimizers import Adam

from keras.preprocessing.text import Tokenizer

from keras\_preprocessing.sequence import pad\_sequences

from sklearn.model\_selection import train\_test\_split

from keras.layers import Input, Embedding, LSTM, Dropout, Dense

from keras.callbacks import EarlyStopping

df = pd.read\_csv('/kaggle/input/answerscript4/p3.csv')

import nltk

from nltk.corpus import stopwords

nltk.download('stopwords')

df\_ind = df.loc[df['essay\_set']==3]

df\_ind = df\_ind[['essay','domain1\_score']]

df\_ind = df\_ind.sort\_values(by= ["domain1\_score"], ascending=False)

answer\_sheet = df\_ind.iloc[0]['essay']

df\_ind = df\_ind.drop(0)

students\_answers = list(df['essay'].values)

marks\_org = list(df['domain1\_score'].values)

t=max(marks\_org)

# Set the random seed for reproducibility

np.random.seed(42)

# Set the maximum sequence length and embedding dimension

MAX\_SEQUENCE\_LENGTH = 1000

EMBEDDING\_DIM = 100

# Set the number of LSTM units and dropout rate

NUM\_LSTM\_UNITS = 128

DROPOUT\_RATE = 0.2

# Set the batch size and number of epochs

BATCH\_SIZE = 64

EPOCHS = 10

# Define the function to preprocess the text

def preprocess\_text(x, remove\_stopwords=False):

x = x.lower()

x = re.sub("[^a-z\s+]","",x)

if remove\_stopwords:

x = " ".join([word for word in x.split() if word not in stopwords.words('english')])

return x

# Load the essays dataset

essays\_df = pd.read\_csv('/kaggle/input/answerscript4/p7.csv', encoding='latin-1')

# Remove essays that have a domain1\_score of NaN

essays\_df = essays\_df[~essays\_df['domain1\_score'].isna()]

# Remove stopwords from the essays

essays\_df['essay'] = essays\_df['essay'].apply(preprocess\_text, remove\_stopwords=True)

essays\_df['expected']=answer\_sheet

# Split the dataset into training, validation, and test sets

train\_df, test\_df = train\_test\_split(essays\_df, test\_size=0.2, random\_state=42)

train\_df, val\_df = train\_test\_split(train\_df, test\_size=0.2, random\_state=42)

# Tokenize the texts

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(train\_df['essay'])

# Convert the texts to sequences and pad them to the specified maximum length

X\_expected\_train = pad\_sequences(tokenizer.texts\_to\_sequences(train\_df['expected']), maxlen=MAX\_SEQUENCE\_LENGTH)

X\_actual\_train = pad\_sequences(tokenizer.texts\_to\_sequences(train\_df['essay']), maxlen=MAX\_SEQUENCE\_LENGTH)

y\_train = train\_df['domain1\_score'].values /t

X\_expected\_val = pad\_sequences(tokenizer.texts\_to\_sequences(val\_df['expected']), maxlen=MAX\_SEQUENCE\_LENGTH)

X\_actual\_val = pad\_sequences(tokenizer.texts\_to\_sequences(val\_df['essay']), maxlen=MAX\_SEQUENCE\_LENGTH)

y\_val = val\_df['domain1\_score'].values /t

X\_expected\_test = pad\_sequences(tokenizer.texts\_to\_sequences(test\_df['expected']), maxlen=MAX\_SEQUENCE\_LENGTH)

X\_actual\_test = pad\_sequences(tokenizer.texts\_to\_sequences(test\_df['essay']), maxlen=MAX\_SEQUENCE\_LENGTH)

y\_test = test\_df['domain1\_score'].values /t

# Define the inputs and embedding layer

expected\_input = Input(shape=(MAX\_SEQUENCE\_LENGTH,), dtype='int32', name='expected\_input')

actual\_input = Input(shape=(MAX\_SEQUENCE\_LENGTH,), dtype='int32', name='actual\_input')

embedding\_layer = Embedding(input\_dim=len(tokenizer.word\_index) + 1, output\_dim=EMBEDDING\_DIM, input\_length=MAX\_SEQUENCE\_LENGTH)

# Encode the inputs with the embedding layer

expected\_encoded = embedding\_layer(expected\_input)

actual\_encoded = embedding\_layer(actual\_input)

# Define the LSTM layer

lstm\_layer = LSTM(NUM\_LSTM\_UNITS)

# Define the dropout layer

dropout\_layer = Dropout(DROPOUT\_RATE)

# Define the output layer

output\_layer = Dense(1, activation='sigmoid')

# Encode the expected and actual inputs

expected\_encoded = embedding\_layer(expected\_input)

actual\_encoded = embedding\_layer(actual\_input)

# Pass the expected and actual inputs through the LSTM layer

expected\_output = lstm\_layer(expected\_encoded)

actual\_output = lstm\_layer(actual\_encoded)

# Apply dropout to the LSTM outputs

expected\_output = dropout\_layer(expected\_output)

actual\_output = dropout\_layer(actual\_output)

# Pass the LSTM outputs through the output layer

expected\_output = output\_layer(expected\_output)

actual\_output = output\_layer(actual\_output)

model\_inputs = [expected\_input, actual\_input]

model\_outputs = [actual\_output]

# Create the model

model = Model(inputs=model\_inputs, outputs=model\_outputs)

# Compile the model with MAE as the loss function

model.compile(loss='mean\_absolute\_error', optimizer='adam', metrics=['mae'])

# Train the model

model.fit(x=[X\_expected\_train, X\_actual\_train], y=y\_train, batch\_size=BATCH\_SIZE, epochs=EPOCHS, validation\_data=([X\_expected\_val, X\_actual\_val], y\_val))

# Evaluate the model

loss, mae = model.evaluate([X\_expected\_test, X\_actual\_test], y\_test, batch\_size=BATCH\_SIZE)

print('Test Loss:', loss)

print('Test MAE:', mae)

**REFERENCES**

[1] Prof. Sumedha P Raut1 , Siddhesh D Chaudhari2, VarunBWaghole3, Pruthviraj U Jadhav4, Abhishek B Saste5(2022).Automatic Evaluation of Descriptive Answers Using NLP and Machine Learning.

[2] Rahman, M., & Siddiqui, F. H. (2020). *NLP-based automatic answer script evaluation* (Doctoral dissertation, DUET Journal).

[3] Handayani, L., Alika, W. O., & Negara, B. S. (2020, June). A Latent Semantic Analysis Method for Automatic Scoring System at Essay Test. In *Journal of Physics: Conference Series* (Vol. 1566, No. 1, p. 012119). IOP Publishing.

[4] Tashu, T. M., Maurya, C. K., & Horvath, T. (2022). Deep Learning Architecture for Automatic Essay Scoring. *arXiv preprint arXiv:2206.08232*.

[5] Rodriguez, P. U., Jafari, A., & Ormerod, C. M. (2019). Language models and automated essay scoring. *arXiv preprint arXiv:1909.09482*.

[6] Bharadia, S., Sinha, P., & Kaul, A. (2018). Answer Evaluation Using Machine Learning.

[7] Mehmood, A., On, B. W., Lee, I., & Choi, G. S. (2017). Prognosis essay scoring and article relevancy using multi-text features and machine learning. *Symmetry*, *9*(1), 11.

[8] Bhatt, R., Patel, M., Srivastava, G., & Mago, V. (2020, October). A graph based approach to automate essay evaluation. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 4379-4385). IEEE.

[9] Janda, H. K., Pawar, A., Du, S., & Mago, V. (2019). Syntactic, semantic and sentiment analysis: The joint effect on automated essay evaluation. *IEEE Access*, *7*, 108486-108503.

[10] Lazer, S., & Student and Teacher Assessments, E. T. S. (2010). Automated Scoring for the Assessment of Common Core Standards.

[11] Alikaniotis, D., Yannakoudakis, H., & Rei, M. (2016). Automatic text scoring using neural networks. *arXiv preprint arXiv:1606.04289*.

[12] McNamara, D. S., Crossley, S. A., Roscoe, R. D., Allen, L. K., & Dai, J. (2015). A hierarchical classification approach to automated essay scoring. *Assessing Writing*, *23*, 35-59.

[13] Fauzi, M. A., Utomo, D. C., Setiawan, B. D., & Pramukantoro, E. S. (2017, August). Automatic essay scoring system using N-gram and cosine similarity for gamification based E-learning. In Proceedings of the International Conference on Advances in Image Processing (pp. 151-155).

[14] Nedungadi, P., & Raj, H. (2014). Unsupervised word sense disambiguation for automatic essay scoring. In Advanced Computing, Networking and Informatics Volume 1 (pp. 437-443). Springer, Cham.

[15] <https://www.kaggle.com/c/asap-aes>

[16] <https://medium.com/swlh/feature-transform-of-text-data-nlp-c6ccedbeb3cc>