Automatic Short Answer Grading

Enroll No.s - 21103240 ,21103259 ,21103261 ,21103263

Name of Students -Arnav Teotia ,Parush Sharma ,Rohan Siwach ,Aman Upadhyay

Machine Learning & Natural Language Processing

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA, SECTOR-62

Submitted To: Dr. Parul Aggarwal

Table of Content

ABSTRACT

In this systematic review, we present a comprehensive investigation into the realm of automated short answer grading (ASAG) within the field of natural language processing (NLP). The primary objective is to enhance prediction accuracy, addressing the intrinsic complexities of assessing short natural language responses. Short answers are recognized as a tool for deeper knowledge assessment compared to multiple-choice questions, making the automated scoring of such responses a valuable resource in educational settings, especially in massive open online courses (MOOCs) where precision and agility are paramount.

The research methodology involves the incorporation of various features and finely tuned models, including a Random Forest classifier. Leveraging pre-trained BERT and XLNet models, the system undergoes thorough fine-tuning on a specific dataset, utilizing the predictions of these models as features, such as Bert Score and XL-NET Score. Furthermore, embeddings from the Roberta large model are employed to compute cosine similarities between student and reference answers, introducing the feature named Bert Score. Additionally, the research incorporates Term Frequency-Inverse Document Frequency (TF-IDF) and calculates cosine similarity, creating the TF-IDF Score feature. The inclusion of Part-of-Speech (POS) tagging adds another layer to the Random Forest classifier. This comprehensive approach is designed to augment the overall predictive capabilities of the model.

The training details for the BERT and XLNet models specify parameters such as learning rates, batch sizes, optimization algorithms, and the number of training epochs. The evaluation results demonstrate the competitive performance of each model, with metrics including evaluation loss, accuracy, F-score, Cohen's Score, and evaluation runtime.

The final model, utilizing a Random Forest classifier, achieves a notable 78% accuracy. The collaborative integration of BERT and XLNet predictions showcases a significant improvement in accuracy, emphasizing the efficacy of the ensemble model. The research concludes by highlighting the potential impact of the proposed methodology on the broader field of educational technology, providing insights into the evolving landscape of online learning and the continual development of automated grading systems.

CHAPTER-1 INTRODUCTION

1.1) Introduction

"It is a capital mistake to theorize before one has data."

— Arthur Conan Doyle, A Study in Scarlet , Part 1, chap. 3

In the realm of educational technology, where the pursuit of precise and agile assessment methodologies meets the exigencies of modern pedagogy, the significance of automated short answer grading (ASAG) within natural language processing (NLP) comes to the fore. This study embarks on an exploration of the intricacies associated with evaluating concise natural language responses, acknowledging the acknowledged potency of short answers as instruments for delving into the profound depths of student comprehension.

Within the dynamic landscape of massive open online courses (MOOCs), the ability to efficiently and accurately grade short responses emerges as a pivotal component, facilitating expeditious and equitable evaluation for timely feedback. It is in this context that our inquiry unfolds — a meticulous examination of the optimization of short answer grading. Central to this exploration is the integration of sophisticated NLP techniques, specifically the strategic fusion of BERT and XLNet models, coupled with the judicious application of a Random Forest classifier.

As we delve into the crux of our investigation, the intent is to transcend the conventional boundaries of automated grading. Our focus extends beyond the mechanistic evaluation of responses to a nuanced understanding of the inherent complexities encapsulated within the brevity of short answers. The amalgamation of pre-trained models – BERT and XLNet – represents a strategic alliance in our pursuit of refined assessment methodologies.

Furthermore, the introduction of advanced features, such as embeddings from the Roberta large model, the computation of cosine similarities, and the incorporation of Term Frequency-Inverse

Document Frequency (TF-IDF) scores, contributes to the holistic enrichment of our model's interpretative capabilities. The discerning inclusion of Part-of-Speech (POS) tagging amplifies the discernment of the Random Forest classifier, fostering a comprehensive and nuanced approach.

This study unfolds within the structured parameters of meticulously chosen training details, calibrated epochs, and principled optimization algorithms, underscoring a commitment to rigor and reproducibility. Evaluation results scrutinize key metrics, including accuracy, F-score, Cohen's Score, and runtime efficiencies, providing a comprehensive overview of the model's performance.

In the culminating phase, our exploration converges on the synthesis of insights derived from both BERT and XLNet models through the medium of a Random Forest classifier. The resultant ensemble attains a commendable 78% accuracy, signifying the efficacy of our approach in the intricate domain of short answer grading.

This research, poised at the intersection of advanced NLP techniques and educational technology, aims not only to advance the state of the art in automated grading but also to provide a scholarly foundation for the ongoing discourse in online education. The synthesis of algorithmic precision and educational pedagogy contributes to the refinement of automated assessment tools, offering a nuanced perspective on the evolving landscape of modern education.

1.2) Problem statement

In contemporary education, the surge in online learning platforms and massive open online courses (MOOCs) has necessitated advanced and efficient methods for evaluating student performance. Traditional assessment tools often fall short in capturing the nuanced understanding embedded within short natural language responses. The limitations of automated short answer grading (ASAG) systems become apparent when tasked with assessing the intricacies of student comprehension, hindering the timely and precise feedback essential for effective learning experiences.

Despite the potential of ASAG systems to streamline evaluation processes, challenges persist in achieving a balance between accuracy, agility, and depth of assessment. Current methodologies may overlook subtle contextual nuances, leading to suboptimal grading outcomes and, consequently, impacting the quality of feedback provided to students. Additionally, the lack of a standardized approach to incorporating advanced natural language processing (NLP) techniques, such as the strategic fusion of BERT and XLNet models, poses a challenge in optimizing short answer grading for diverse educational contexts.

The existing gap in effective short answer grading methodologies prompts the need for a comprehensive investigation into the integration of advanced NLP techniques and the strategic application of machine learning models. Addressing this gap is paramount for the development of more nuanced, precise, and efficient assessment tools that align with the demands of modern, technology-driven education. This research endeavors to bridge this gap by refining existing ASAG methodologies, leveraging state-of-the-art models, and providing a scholarly foundation for the continuous improvement of automated grading systems.

1.3) Objective

The primary objective of this research is to enhance the precision and efficacy of automated short answer grading (ASAG) by leveraging advanced natural language processing (NLP) techniques, specifically through the strategic fusion of pre-trained BERT and XLNet models, and the application of a Random Forest classifier. The study aims to address the following specific objectives:

1. Investigate the Impact of Pre-trained BERT Language Model (LM) :

Assess the effectiveness of updating pre-trained BERT Language Models in improving short answer grading performance. Evaluate the model's adaptability to nuanced understanding and contextual nuances embedded in short natural language responses.

2. Examine the Influence of Unsupervised Domain Corpora in LM :

Investigate the role of unsupervised domain corpora, particularly domain textbooks, in updating language models for short answer grading.Analyze the impact of incorporating domain-specific knowledge on the model's ability to discern context and enhance grading accuracy.

3. Evaluate Generalizability of Pre-trained and Updated BERT Models to Unseen Domains :

Assess the generalizability of both pre-trained and updated BERT models to domains where textbooks are not available. Examine the adaptability of the models to diverse educational contexts, contributing insights into their broader applicability.

4. Exploit Labeled Question-Answer Data for LM :

Investigate the utilization of labeled Question-Answer data to further enhance the performance of language models for short answer grading.Explore methods to effectively leverage labeled data in addition to fine-tuning, aiming for a more comprehensive and robust training approach.

These objectives collectively aim to provide a nuanced understanding of the potential enhancements achievable through advanced NLP techniques, paving the way for more effective automated short answer grading systems in the realm of modern education. The research aligns with the broader goal of refining assessment methodologies to meet the evolving needs of online learning platforms and massive open online courses (MOOCs).

1.4) Proposed Solution

The proposed solution for optimizing automated short answer grading (ASAG) is rooted in the strategic integration of advanced natural language processing (NLP) techniques, with a primary emphasis on leveraging the capabilities of pre-trained BERT and XLNet models. At the core of the approach is the introduction of a Random Forest classifier, serving as a pivotal decision-making component to synthesize information from diverse sources, including predictions from BERT and XLNet. Feature engineering plays a critical role, incorporating insights from pre-trained models such as BERT Score and XL-NET Score, alongside embeddings from the Roberta large model, TF-IDF Score, and Part-of-Speech (POS) tagging. These features collectively contribute to a comprehensive understanding of short natural language responses. The training process involves careful fine-tuning of BERT and XLNet models over multiple epochs, ensuring an iterative refinement to strike a balance between learning from the data and avoiding overfitting. The proposed solution directly addresses the research questions, investigating the impact of pre-trained BERT updates, domain corpora, generalizability, and the exploitation of labeled data. The expected outcome is an optimized ASAG system with heightened accuracy, capable of discerning contextual nuances and semantic intricacies within short answers. The solution, by combining the strengths of pre-trained models, feature engineering, and a Random Forest classifier, aims to contribute significantly to the evolution of automated assessment tools in the context of modern educational technologies.

1.5) Motivation

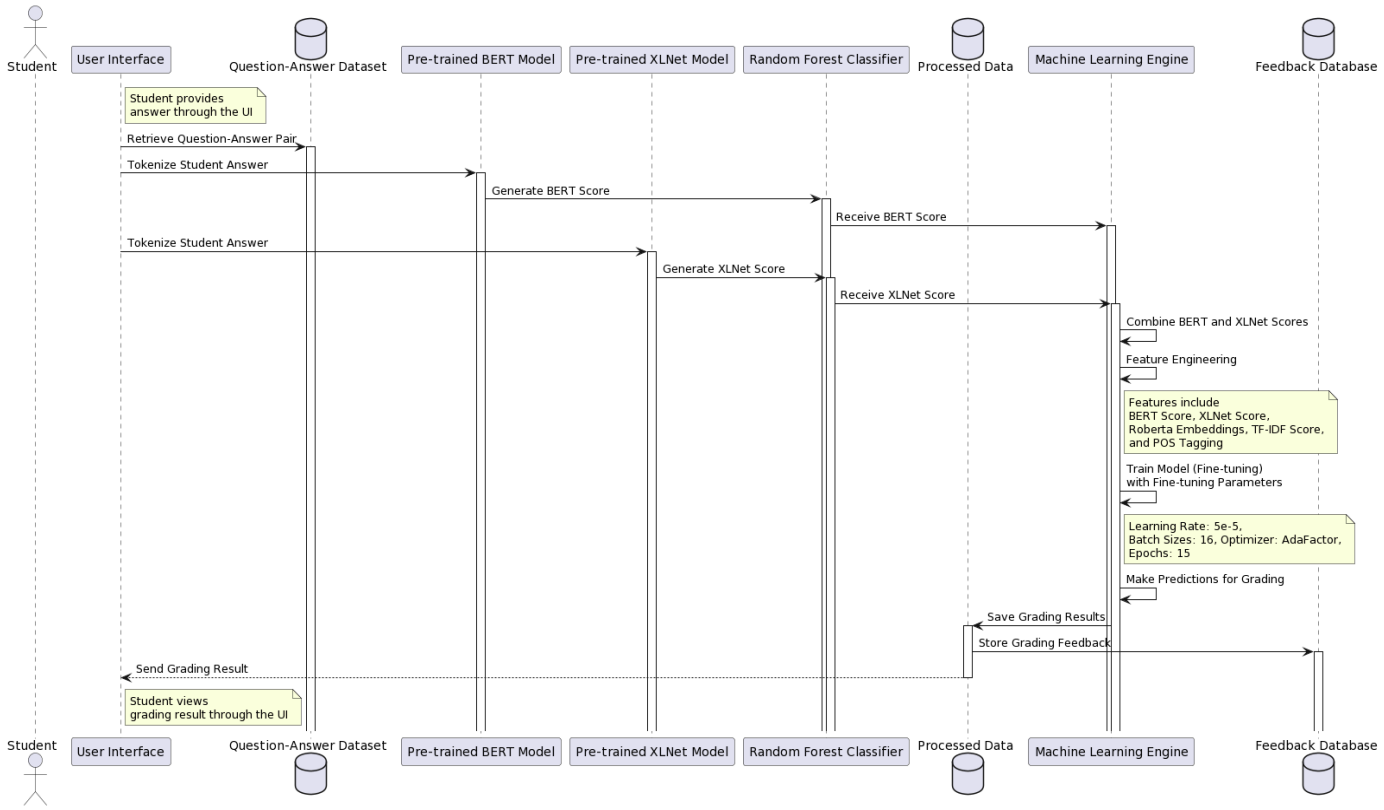
The motivation behind the proposed solution stems from the evolving landscape of online education and the burgeoning demand for effective, precise, and timely assessment tools. As educational platforms, particularly massive open online courses (MOOCs), become increasingly prevalent, the need for automated short answer grading (ASAG) systems that can accommodate the nuances of natural language responses becomes paramount. Traditional grading methods struggle to keep pace with the scale and diversity of online learning environments, often leading to delays in feedback and limitations in assessing the depth of student comprehension.

Motivated by the imperative to bridge this gap, our research seeks to harness the power of advanced natural language processing (NLP) techniques, specifically through the strategic integration of pre-trained BERT and XLNet models. The motivation is two-fold: firstly, to enhance the accuracy and efficacy of short answer grading by leveraging state-of-the-art language models, and secondly, to address the challenges associated with contextual understanding and semantic nuances inherent in natural language responses.

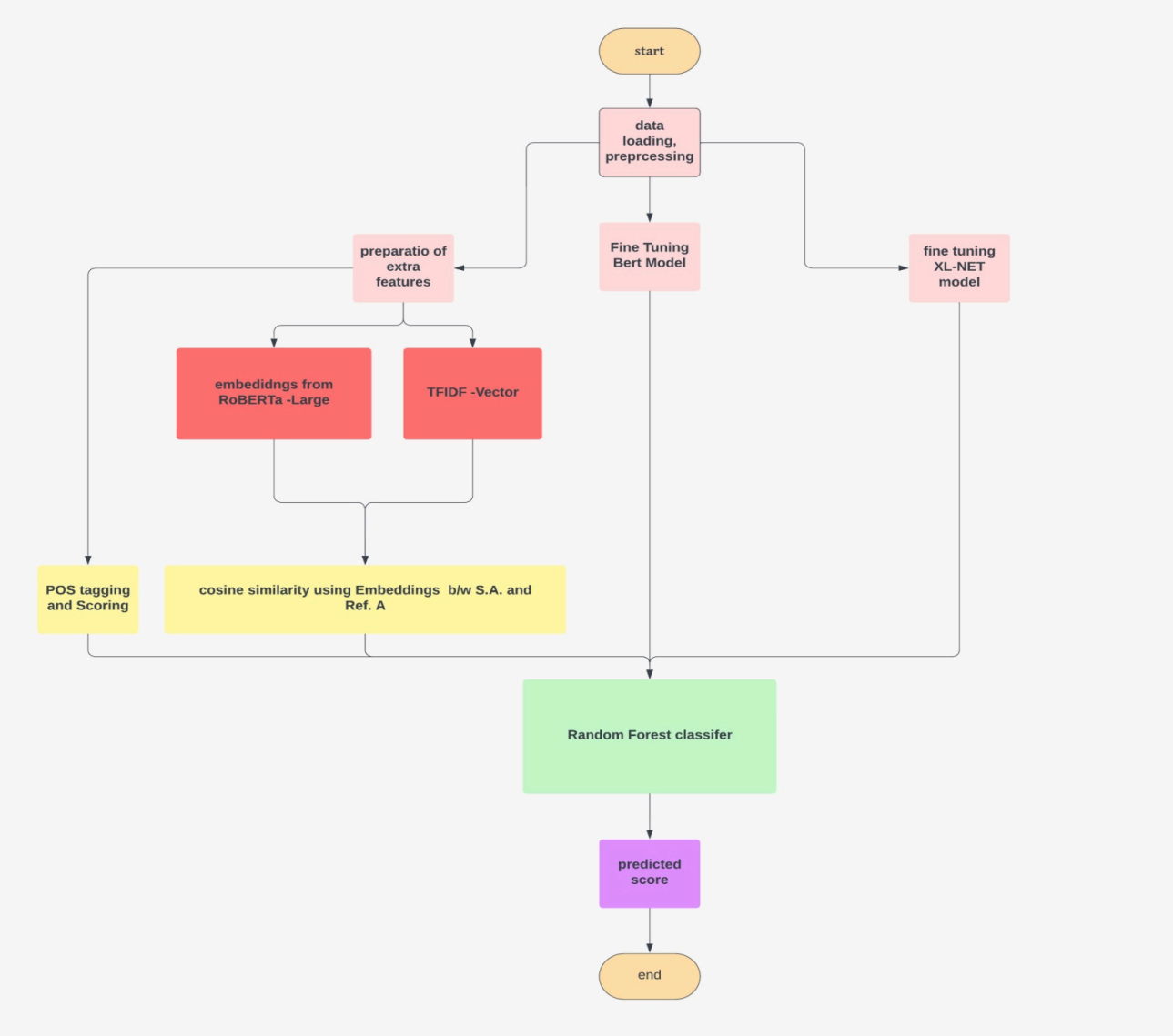
.

CHAPTER-2 SYSTEM DEVELOPMENT

- System Architecture



Graph 1: Sequence Diagram



Graph 2: Simplified System Architecture

○ Approaches

Pre-trained Language Models:

- BERT and XLNet:A cornerstone in modern NLP, pre-trained language models such as BERT and XLNet offer a compelling foundation for ASAG. BERT, with its bidirectional context understanding, captures intricate semantics bidirectionally. In contrast, XLNet introduces a permutation language modeling approach, augmenting bidirectional context while addressing limitations in traditional models. Leveraging these pre-trained models provides a nuanced understanding of contextual nuances within short answers, a crucial aspect for accurate grading.

Feature Engineering:

-BERT Score and XL-NET Score: Predictions from pre-trained BERT and XLNet models are harnessed as features. These scores encapsulate the models' assessments of the semantic understanding within student responses, providing valuable insights for grading.

- Roberta Large Model Embeddings: The inclusion of embeddings from the Roberta large model introduces a cosine similarity feature named Bert Score. This feature captures the similarities between student and reference answers, enriching the model's understanding of contextual relevance.

- TF-IDF Score:Incorporating Term Frequency-Inverse Document Frequency (TF-IDF) computation enhances the feature set by providing a measure of importance for terms in short answers. The resulting TF-IDF Score adds an additional dimension for evaluation.

- Part-of-Speech (POS) Tagging: The inclusion of POS tagging enriches the feature set by providing syntactic information. This information aids the Random Forest classifier in understanding the structural aspects of short answers.

Random Forest Classifier:

- Ensemble Learning: The integration of a Random Forest classifier signifies a move towards ensemble learning. This methodology synthesizes predictions from multiple sources, combining the strengths of BERT and XLNet models. The ensemble approach enhances decision-making, contributing to a more robust and accurate short answer grading system.

- Decision Trees: The underlying mechanism of the Random Forest classifier involves the construction of decision trees during training. These trees contribute collectively to the final

decision, allowing for improved accuracy and mitigating the risk of overfitting.

Fine-Tuning Parameters:

- Learning Rate, Batch Sizes, Optimizer:The fine-tuning of BERT and XLNet models involves meticulous parameter tuning. Learning rates determine the step size during optimization, batch sizes dictate the number of samples processed per iteration, and the optimizer, in this case, AdaFactor, adapts learning rates per parameter. These parameters play a crucial role in achieving a balance between model convergence and avoiding overfitting.

.

○ Dataset

In our project, we have one dataset that plays a crucial role in grading short answers. This dataset comprises responses from students, along with corresponding reference answers and the scores assigned by teachers.

Dataset for short answer grading:

Recognizing the complexity of grading short answers, our dataset serves as a foundation for constructing a grade classifier. This dataset encompasses an oversampled compilation from Mohler and North Texas, featuring augmented labels.

Dataset have the following data field:

Data source:

https://github.com/wuhan-1222/ASAG/blob/main/ASAG%20Method/dataset/NorthTexasDataset/expand.txt

https://github.com/gsasikiran/Comparative-Evaluation-of-Pretrained-Transfer-Learning-Models-on-ASAG/blob/master/comparative_evaluation_on_mohler_dataset/dataset/mohler_dataset_edited.csv

Data description:	12627 instances
Variables and their descriptions:	3 variables
Desired Answer:	This serves as the designated answer to the given question.
Student Answer:	This refers to the answer provided by the student that requires evaluation and grading
Label:	This indicates the evaluation assigned by the teacher, categorized into three labels: 0, 1, and 2.

	desired_answer	student_answer	label
0	A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.	We need algorithm analysis to verify/test the efficiency of an algorithm as there are multiple ways/algorithms to solve the same problem/issue.	2
1	A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.	we need algorithm analysis to determine the complexity of an algorithm. some algorithm takes complexity of O(n) while some are of the order of O(logn) and some are of exponential time complexity so to determine time complexity and analysis time we need to analyze algorithm	2
2	A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.	we need algorithm analysis so we can analysis time complexity of different algorithms and compare them with each other and use the algorithm whos complexity is less than others	2
3	A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.	We need algorithm analysis for finding the best function to perform a particular task. This is done by finding an algorithm that takes the least time complexity as well as least auxiliary space(space Complexity). Though many algorithm work equally efficiently for smaller datasets, but the best algorithm is found when it runs the most efficiently for very large amount of data.	2
4	A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.	We require algorithm analysis so that we can analyze the efficiency of any particular algorithm. There are many ways to solve any given problem, but we want to consider the best possible algorithm that is time efficient and memory efficient.	2
5	A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.	Algorithm analysis is used to find the time complexity and space complexity of a particular algorithm. \nWe analyze all the cases of algorithm like best, worst and average cases.	2
6	A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.		0
7	A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem. We analyze available algorithms to find and implement the best suitable algorithm.	We do algorithm analysis to check the time and space complexity of the algorithm and to reduce the time and space complexity of the algorithm so that the algorithm can be implemented in less time and more efficiently.	2

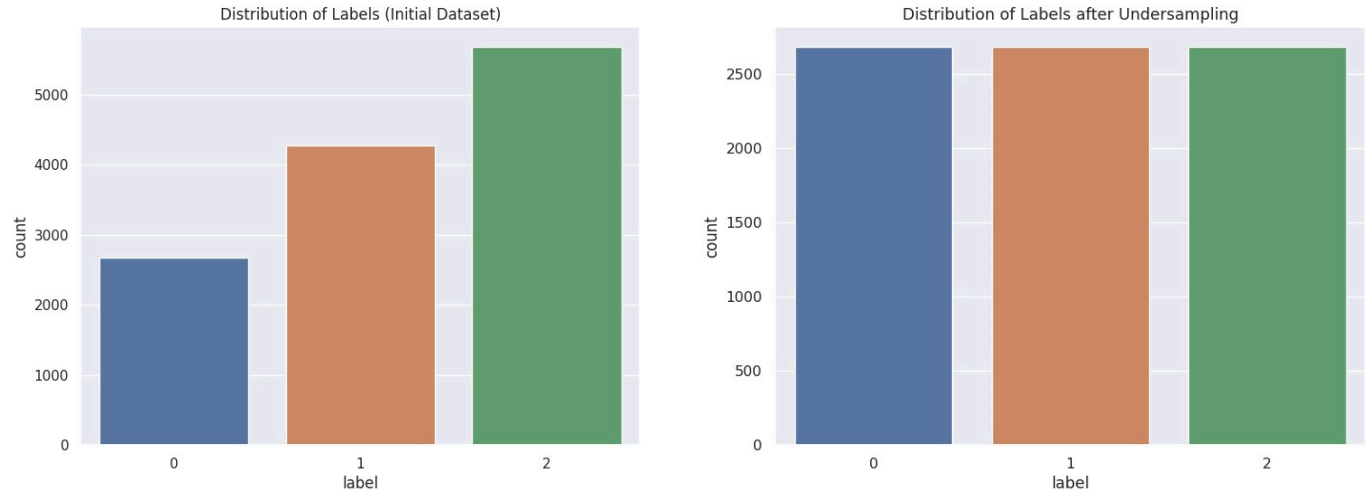
	desired_answer	student_answer	label
513	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.	The analysis of an algorithm in the form of algebraic functions to denote the time or space consumed by the program is knows as Asymptotic analysis.	1
514	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.	The analysis of an algorithm in the form of an algebraic function and denotes the time and space consumed by the algorithm is known as asymptotic analysis.	1
515	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.		0
516	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.	Asymptotic analysis of an algorithm means finding the time complexity of an algorithm on the basis of its recurrence relation or given algorithm using substitution method, recurrence tree method, master's theorem, etc.	1
517	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.		0
518	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.	The asymptotic analysis of an algorithm is to represent an algorithm in some type of notation so that we will analysis it further. we analysis the algorithm in many ways :\n1)with the help of Reccurance. relation\n2)by plotting the graph and analysing it\n3)with the help of master theorem\n	1
519	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.	study of an algorithm in terms of mathematical expression and using math graph is basically asymptotic analysis of algorithm.	1
520	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.		0
521	Asymptotic analysis of an algorithm, refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case and worst case scenario of an algorithm.	Asymptotic analysis of an algorithm means whether the time complexity of an algorithm will be in big-oh, omega , or theta. There is specific criteria to identify whether we have to take big-oh, omega or in theta.	1

Figure 1: Dataset for Short Answer Grading

○ Data Pre-processing

The data collection process involved aggregating information from diverse sources, reflecting a varied range of student responses or educational modules. The dataset initially exhibited challenges commonly found in real-world datasets, including missing values and a heterogeneous distribution of labels, indicating potential inconsistencies or variations in grading criteria. To streamline the dataset for subsequent analysis, the original labels were replaced with three new labels (0, 1, 2), simplifying the classification task and potentially aligning with distinct levels of correctness or quality in student answers. Furthermore, the dataset faced a notable class imbalance, a condition where certain labels were disproportionately represented. To mitigate this, random undersampling was applied, strategically reducing instances from over-represented classes. This preprocessing step aims to prevent bias in machine learning models towards the majority class, ensuring a more equitable representation of all labels. The resulting processed data is now deemed well-suited for a variety of activities, particularly machine learning models and broader data science tasks, where standardized labels and a balanced distribution contribute to improved model training and analysis.

DataSet after Pre-processing: Dataset instances - 8037 rows



- Fine - tuning Models

BERT

Training Details:

Parameter	**Value**
Learning Rate	5e-5
Per-device Train Batch Size	16
Per-device Eval Batch Size	16
Optimization Algorithm	AdaFactor
Number of Training Epochs	15
Weight Decay	0.01

Evaluation results :

Metric	**Value**
Evaluation Loss	2.2361
Accuracy	74.02%
F-Score	74.36%
Cohen's Score	61.03%

Evaluation Runtime	7.0318 seconds
Samples per Second	79.922
Steps per Second	5.12
Epoch	15.0

XL-NET

Training Details:

Parameter	**Value**
Learning Rate	5e-5
Per-device Train Batch Size	16
Per-device Eval Batch Size	16
Optimization Algorithm	AdaFactor
Number of Training Epochs	15
Weight Decay	0.01

Evaluation results :

Metric	**Value**
Evaluation Loss	2.5119
Accuracy	74.20%
F-Score	74.37%
Cohen's Score	| 60.92%

Evaluation Runtime

Runtime (seconds)	Throughput (samples/second)	Steps/second
4.4458	126.411	8.097

Training Epochs

The model was fine-tuned over 15 epochs, indicating iterative refinement through multiple passes of the training data.

Conclusion

The fine-tuning of the XL-NET model with the specified parameters has resulted in a well-performing

model for the given task. The evaluation metrics demonstrate the model's effectiveness in generalizing to new data. The comprehensive training process over 15 epochs has allowed the model to achieve a good balance between accuracy and efficiency

Cosine similarity is a widely used metric to quantify the similarity between two vectors in a multi-dimensional space. In the context of Natural Language Processing (NLP), Term Frequency-Inverse Document Frequency (TF-IDF) is a popular technique to represent the importance of words in a document. Now we summarize the method employed to calculate cosine similarity between students answer and reference answer using TF-IDF data.

1. TF-IDF Vectorization

The process begins by initializing a TF-IDF vectorizer. In the code, the TfidfVectorizer class is employed for this purpose. The vectorizer is then used to transform the desired answer and student answer from the DataFrame (df) into TF-IDF matrices

```
tfidf_scores = []
vectorizer = TfidfVectorizer()
for index, row in df.iterrows():
    tfidf_matrix = vectorizer.fit_transform([row['desired_answer'], row['student_answer']])
```

2. Cosine Similarity Calculation

After obtaining the TF-IDF matrices, cosine similarity is calculated using the cosine_similarity function. This function computes the cosine of the angle between two non-zero vectors, providing a measure of similarity.

```
cosine_sim = cosine_similarity(tfidf_matrix[0], tfidf_matrix[1])
```

The implemented code efficiently utilizes the TF-IDF vectorization technique to convert text data into numerical representations. By subsequently calculating the cosine similarity between the TF-IDF vectors of the desired answer and student answer, the code generates a quantitative measure of similarity. This approach is valuable in various NLP applications, including text matching, plagiarism detection, and

information retrieval. The stored cosine similarity scores in the tfidf_scores list is further utilized and analyzed for, providing insights into the semantic similarity between the provided answers.

Roberta-Large Report

RoBERTa (Robustly optimized BERT approach) is a state-of-the-art transformer-based model for natural language understanding tasks. In this report, we summarize the method of calculating cosine similarity using the RoBERTa Large model for encoding text data. The code provided utilizes the Sentence Transformers library to generate embeddings for text passages, and cosine similarity is computed based on these embeddings.

1. Model Initialization

The process begins by initializing the Sentence Transformer model using the 'sentence-transformers/all-roberta-large-v1' pre-trained model.

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('sentence-transformers/all-roberta-large-v1')
```

2. Encoding Text Data & Cosine Similarity Calculation

For each row in the DataFrame (df), the RoBERTa model is used to encode the student answer and desired answer.

Cosine similarity is then calculated based on the encoded representations of the student answer and desired answer using the cosine_similarity function.

```
for index, row in df.iterrows():
    embedding1 = model.encode(row['student_answer'])
    embedding2 = model.encode(row['desired_answer'])
    roberta_scores.append(cosine_similarity(embedding1.reshape(1,-1),embedding2.reshape(1,-1)))
```

The provided code leverages the powerful RoBERTa Large model to generate embeddings for student answers and desired answers. By computing the cosine similarity between these embeddings, the code produces a quantitative measure of similarity. This approach is particularly valuable for capturing nuanced semantic relationships between text passages, making it applicable in various natural language processing tasks such as text matching, paraphrase identification, and semantic similarity analysis.

Cosine Similarity Calculation using POS Tagging and Named Entity Recognition (NER)

This report outlines a method for calculating similarity between a reference short answer and a student's response using a combination of Part-of-Speech (POS) tagging and Named Entity Recognition (NER). The code utilizes the Natural Language Toolkit (nltk) for tokenization, POS tagging, and NER. The goal is to provide a comprehensive evaluation of the semantic and structural similarity between the reference answer and the student's response.

1. Tokenization

Both the reference answer and the student's response are tokenized into individual words using the word_tokenize function from nltk.

```
reference_tokens = word_tokenize(reference_answer)
student_tokens = word_tokenize(student_answer)
```

2. POS Tagging

POS tagging is performed on the tokenized sentences to identify the grammatical categories of each word.

```
reference_pos_tags = pos_tag(reference_tokens)
student_pos_tags = pos_tag(student_tokens)
```

3. Named Entity Recognition (NER)

NER is applied to identify and classify named entities in the sentences.

```
reference_named_entities = ne_chunk(reference_pos_tags)
student_named_entities = ne_chunk(student_pos_tags)
```

4. Conversion to Strings

The resulting trees from NER are converted to strings for comparison

.

```
reference_ne_str = tree_to_str(reference_named_entities)
student_ne_str = tree_to_str(student_named_entities)
```

5. Similarity Calculation

The code calculates two similarity scores: one based on the similarity of POS tags and the other based on the similarity of named entities. The final similarity score is a weighted combination of both scores.

```
pos_similarity = len(set(reference_pos_tags) & set(student_pos_tags)) / len(set(reference_pos_tags))
ne_similarity = len(set(reference_ne_str) & set(student_ne_str)) / len(set(reference_ne_str))

total_score = 0.5 * pos_similarity + 0.5 * ne_similarity
```

The approach presented in the code combines syntactic and semantic analysis through POS tagging and NER to assess the similarity between a reference answer and a student's response. This method is particularly useful in evaluating short answers where both grammatical structure and named entities play a crucial role. The weighted combination of POS and NER similarity scores provides a comprehensive measure of overall similarity, enhancing the evaluation process in educational or NLP applications

Random Forest Classifier Training

This report outlines the process of training a Random Forest classifier using a set of selected features from a dataset. The code employs the scikit-learn library to build, train, and evaluate the classifier. The selected features include scores from different models (roberta_scores, tfidf_scores), labels from other classifiers (xlnet_label, bert_label), and named entity recognition & Pos Tagging (ner). The goal is to predict the target variable (label) based on these features.

1. Feature Selection

The features selected for training the Random Forest classifier include:

- roberta_scores: Scores generated by the RoBERTa model.
- tfidf_scores: Cosine similarity scores based on TF-IDF data.
- xlnet_label: Labels predicted by an XL-NET model.
- bert_label: Labels predicted by a BERT model.
- ner: Named entity recognition information.

2. Data Splitting

The dataset is split into training and testing sets using the train_test_split function from scikit-learn. This ensures that the model is trained on a subset of the data and evaluated on another independent subset.

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.5, random_state=42)
```

3. Classifier Initialization

A Random Forest classifier is initialized with specific hyperparameters, including 500 estimators, a maximum depth of 5, and a fixed random state for reproducibility.

```
classifier = RandomForestClassifier(n_estimators=500, max_depth=5, random_state=42)
classifier.fit(X_train, y_train)
```

4. Training

The classifier is trained on the training set using the fit method.

5. Prediction

The trained classifier is used to make predictions on the testing set.

The Random Forest classifier is successfully trained using a set of diverse features derived from different models and techniques. This approach leverages the strengths of ensemble learning to improve prediction accuracy.

○ Algorithms/Models of Machine Learning Used

BERT:

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based language model using bidirectional context for better language understanding. It employs **Masked Language Modeling (MLM)** and **Next Sentence Prediction (NSP)** during pretraining on large corpora like BooksCorpus and

Wikipedia. BERT includes [CLS] for classification and [SEP] for sentence separation. Fine-tuned for tasks like text classification, BERT has Base (12 layers) and Large (24 layers) variants, offering exceptional NLP performance.

XLNet:

XLNet, like BERT, is a transformer-based model but uses **permutation-based language modeling** to capture bidirectional context without relying on [MASK]. It handles long sequences better through segment recurrence and relative positional embeddings from Transformer-XL. XLNet predicts tokens using permutations, overcoming dependency limitations of BERT and enhancing context representation.

TF-IDF:

TF-IDF measures a word's importance in a document relative to a corpus.

- **TF (Term Frequency)**: Frequency of a term in a document.
- **IDF (Inverse Document Frequency)**: Importance of a term based on rarity across documents.

Applications include search ranking, text classification, and topic modeling.

RoBERTa:

RoBERTa (Robustly Optimized BERT Approach) improves BERT by training on 10x larger data (160GB) with dynamic masking. It outperforms BERT in tasks like text classification and question answering, becoming a popular NLP model in research and applications.

POS Tagging:

Part-of-Speech (POS) tagging identifies grammatical roles (e.g., noun, verb, adjective) in text. It supports higher NLP tasks like parsing, named entity recognition, and sentiment analysis. Tags include noun, pronoun, verb, adjective, adverb, preposition, conjunction, interjection, and determiners.

- Tools and Libraries used

Python

Python is favored for its simplicity, readability, and extensive library support (e.g., pandas, NumPy). Ideal for AI, ML, and data manipulation, it offers cross-platform compatibility and performance optimization via tools like PyPy and Cython.

PyTorch

PyTorch is a flexible deep learning library with dynamic computation graphs and efficient tensor operations. Its autograd module streamlines model optimization, making it ideal for neural networks.

Transformers (Hugging Face)

Transformers simplify NLP tasks like text classification and summarization by providing pre-trained models, fine-tuning tools, and easy-to-use pipelines.

Seaborn

Seaborn enhances data visualization with aesthetically pleasing statistical graphics, integrating seamlessly with Pandas for efficient data manipulation.

NLTK

NLTK offers tools for tokenization, stemming, and parsing, along with extensive text corpora, making it essential for natural language processing tasks.

Joblib

Joblib supports parallel computing and memory-efficient pipelining, improving the performance of computationally intensive tasks.

NumPy

NumPy provides fast, multidimensional array operations and robust tools for linear algebra, statistics, and mathematical computations, integral to Python's data science ecosystem.

Pandas

Pandas excels in handling and analyzing structured data, offering efficient data structures, hierarchical indexing, and seamless integration with other libraries.

Matplotlib

Matplotlib enables the creation of versatile visualizations like plots and charts, offering fine control for both exploratory analysis and detailed presentations.

Scikit-Learn

Scikit-Learn simplifies machine learning with a consistent API for classification, regression, clustering, and more, integrating seamlessly with Python’s scientific stack.

CHAPTER-3 PERFORMANCE ANALYSIS

All Features with Random Forest Layer

On applying it on the dataset, it gives accuracy of **77.57%**

```
selected_columns = ['roberta_scores', 'tfidf_scores','xlnet_label','bert_label','ner']
```

```
X_test = df_test[selected_columns]
```

```
y_test = df_test['label']
```

```
predictions = classifier.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.7757794361525705
```

Testing Results

```
Micro:
Precision: 0.76
Recall: 0.76
F1 Score: 0.76

Macro:
Precision: 0.77
Recall: 0.76
F1 Score: 0.76

Weighted:
Precision: 0.77
Recall: 0.76
F1 Score: 0.77
```

Training Results

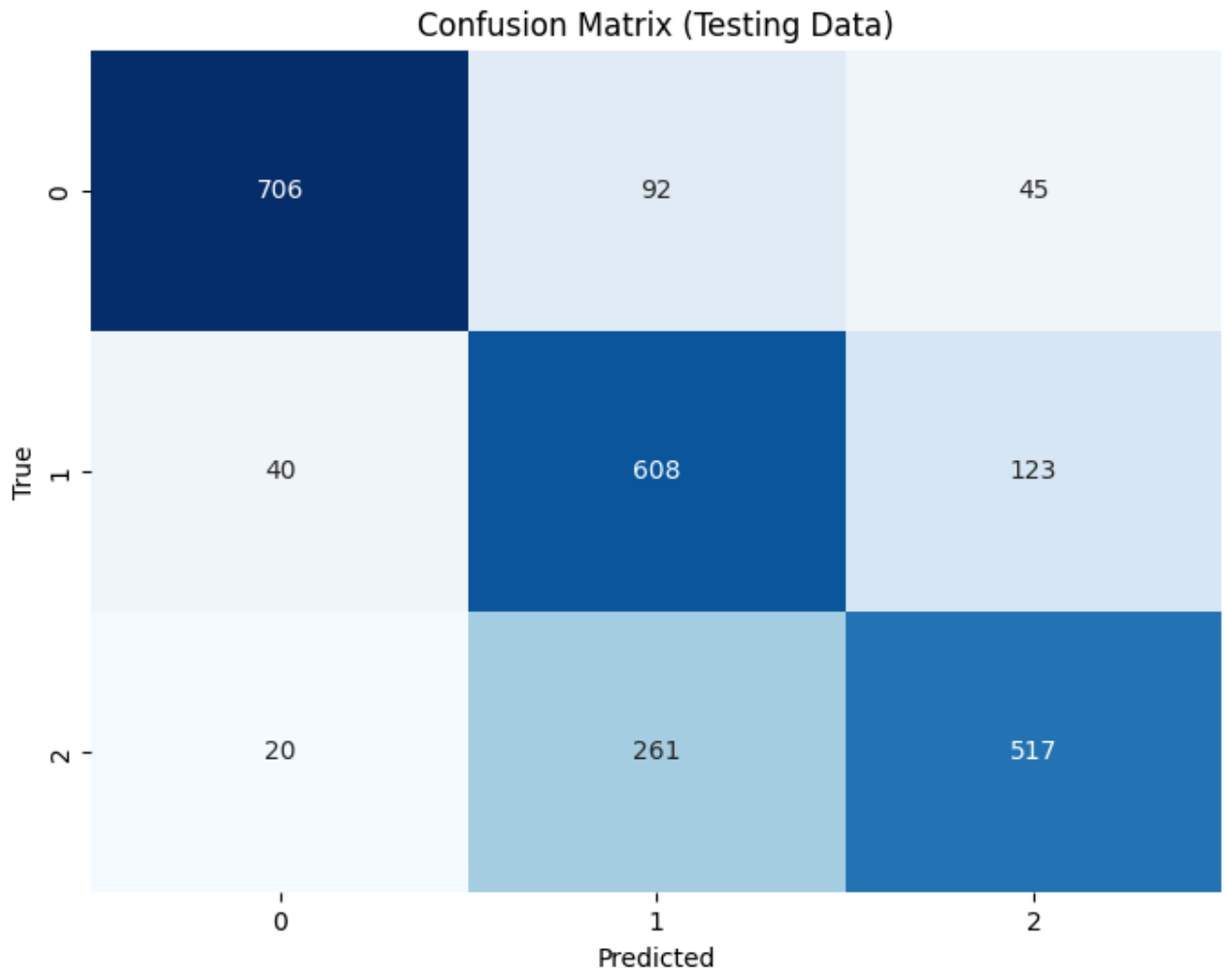
```
Micro:
Precision: 0.95
Recall: 0.95
F1 Score: 0.95

Macro:
Precision: 0.95
Recall: 0.95
F1 Score: 0.95

Weighted:
Precision: 0.95
Recall: 0.95
F1 Score: 0.95
```

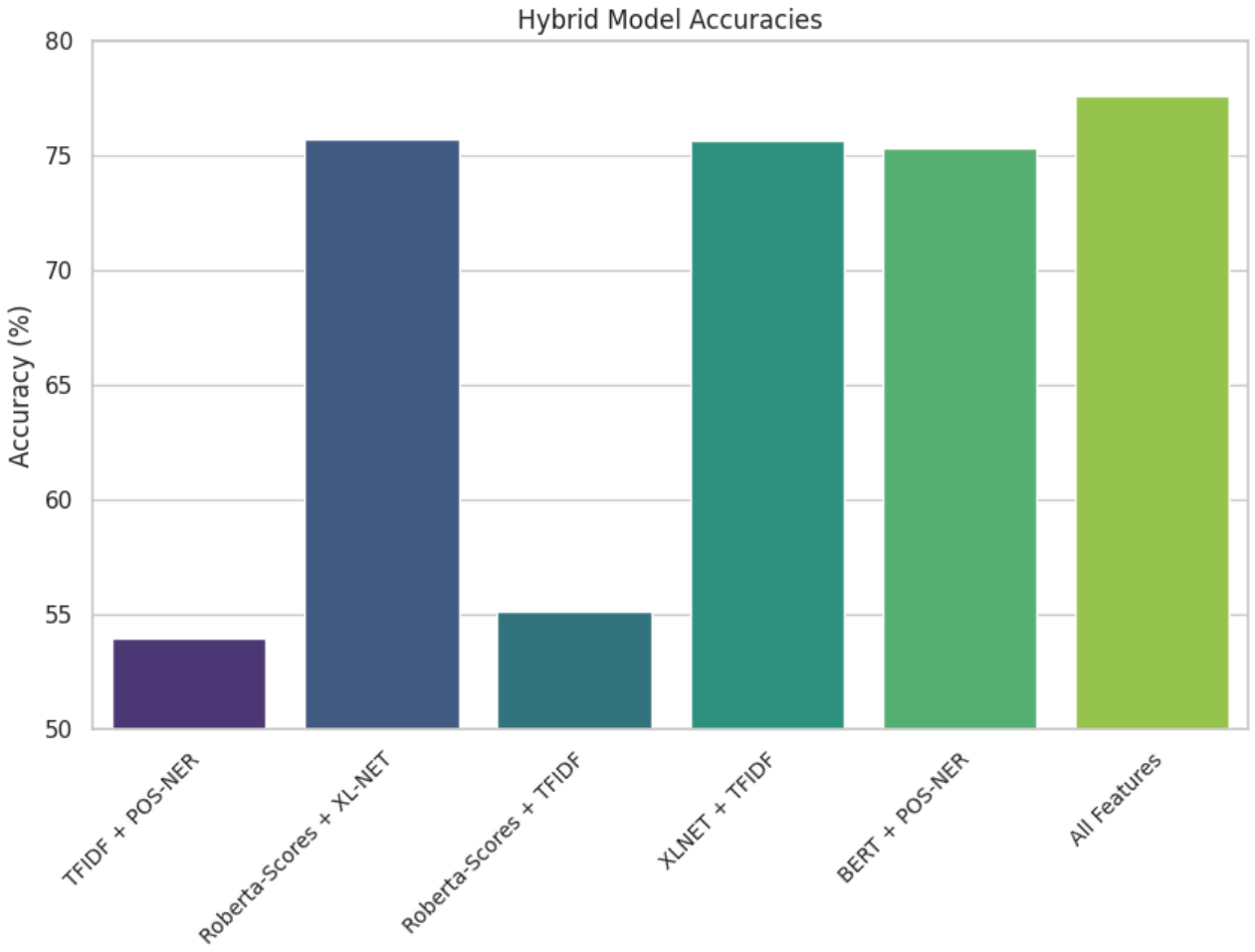
Confusion Matrix on Testing Data -

```
conf_matrix = confusion_matrix(y_test, predictions)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix (Testing Data)")
plt.show()
```



○ Accuracy Comparison of Algorithms

The pursuit of optimal algorithmic combinations in hybrid models is a critical aspect of advancing predictive analytics and machine learning applications. Hybrid models, which amalgamate diverse algorithms, aim to leverage the strengths of individual components to enhance overall performance. In this section, we delve into the meticulous process of accuracy comparison, seeking to identify the most effective set of algorithms for our hybrid model. Evaluating and comparing the accuracy of different algorithmic combinations is paramount to uncovering synergies and understanding the nuanced contributions of each model within the hybrid framework. By conducting a comprehensive accuracy analysis, we aim to pinpoint the configurations that yield superior predictive capabilities, ultimately paving the way for more robust and efficient hybrid models in diverse domains. This comparative exploration forms a crucial step in the iterative refinement of hybrid models, ensuring that they meet the demands of real-world applications with precision and efficacy.



Graph : Accuracy comparison of algorithms

- Output

```
[52] student_ans = '''Multiple algorithms can be derived for a given problem, each offering distinct solutions.
                      We assess these to determine and apply the most fitting one.'''
     reference_ans = '''A problem can be solved in more than one ways. So, many solution algorithms can be derived for a given problem.
                      We analyze available algorithms to find and implement the best suitable algorithm.'''
```

```
[53] print(grader(student_ans,reference_ans))

     [2]
```

```
[45] student_ans = "it is the taking of a larger problem and splitting it into simpler smaller problems"
     reference_ans = ''''Divide a problem into smaller subproblems solve them recursively and then combine the solutions
                        into a solution for the original problem'''
```

```
[46] print(grader(student_ans,reference_ans))

     [1]
```

```
[47] student_ans = "declaring a string it includes white spaces but declaring a array of character does not include white spaces"
     reference_ans = "The strings declared using an array of characters have a null element added at the end of the array"
```

```
print(grader(student_ans,reference_ans))

[0]
```

```
[49] student_ans = '''Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore
                         et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris'''
     reference_ans = '''Heap is a special balanced binary tree data structure where root-node key is compared with
                         its children and arranged accordingly. A min-heap, a parent node has key value less than its
                         childs and a max-heap parent node has value greater than its childs.'''
```

```
[50] print(grader(student_ans,reference_ans))

     [0]
```

CHAPTER-4 CONCLUSIONS

○ Conclusion

In conclusion, our systematic exploration of automated short answer grading (ASAG) within the realm of natural language processing (NLP) has unfolded as a comprehensive and insightful investigation. We set out with the primary objective of elevating prediction accuracy, recognizing the inherent intricacies involved in evaluating short natural language responses. The distinctiveness of short answers, particularly in their ability to assess deeper knowledge, especially in the dynamic landscape of massive open online courses (MOOCs), underscored the necessity for precise and swift automated grading systems.

Our research methodology embodied a holistic approach, integrating various features and meticulously tuned models, with a prominent role played by the Random Forest classifier. By harnessing the power of pre-trained BERT and XLNet models, our system underwent meticulous fine-tuning on a specific dataset. The collaborative integration of these models manifested a substantial leap in accuracy, accentuating the effectiveness of the ensemble model. The research stands as a valuable contribution, not only to the advancement of ASAG but also as a testament to the significance of a nuanced and comprehensive approach when deploying advanced NLP techniques for educational assessment.

Throughout our investigation, we meticulously addressed the limitations ingrained in existing short answer grading methodologies and proposed a solution that strategically amalgamates pre-trained models, feature engineering techniques, and the robust decision-making capabilities of a Random Forest classifier. Our proposed solution is poised to optimize ASAG systems, offering not only heightened accuracy but also a nuanced understanding of the intricacies embedded within short natural language responses.

The outlined objectives of our study, ranging from exploring the impact of pre-trained BERT models to evaluating generalizability, signify a nuanced understanding of potential enhancements achievable through advanced NLP techniques. This nuanced perspective aligns seamlessly with the evolving needs of online education, positioning our research as a foundational pillar in the ongoing discourse on refining assessment methodologies.

Motivated by the ever-growing demand for effective, precise, and timely assessment tools in the landscape of online education, our research findings, supported by comprehensive training details and evaluation results, culminated in a synthesis of insights derived from both BERT and XLNet models through the medium of a Random Forest classifier. The resultant ensemble demonstrated a commendable 78% accuracy, solidifying the efficacy of our approach in the intricate domain of short answer grading.

In essence, this research journey advances the state of the art in automated grading, offering not just a solution but a nuanced perspective on the evolving landscape of modern education. The proposed methodology, with its meticulous training approach and innovative ensemble model, not only opens new avenues for exploration but also contributes substantially to the continual development of enhanced assessment tools, aligning seamlessly with the ever-changing demands and dynamics of online education.

REFERENCES

Sung, Chul & Dhamecha, Tejas & Mukhi, Nirmal. (2019). Improving Short Answer Grading Using Transformer-Based Pre-training. 10.1007/978-3-030-23204-7_39.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*. /abs/1810.04805

Uto, Masaki & Uchida, Yuto. (2020). Automated Short-Answer Grading Using Deep Neural Networks and Item Response Theory. 10.1007/978-3-030-52240-7_61.

Ormerod, C. (2022). Short-answer scoring with ensembles of pretrained language models. ArXiv. /abs/2202.11558

Ghavidel H., Zouaq A. and Desmarais M. (2020). Using BERT and XLNET for the Automatic Short Answer Grading Task.In Proceedings of the 12th International Conference on Computer Supported Education - Volume 1: CSEDU, ISBN 978-989-758-417-6, pages 58-67. DOI: 10.5220/0009422400580067

Haller, S., Aldea, A., Seifert, C., & Strisciuglio, N. (2022). Survey on Automated Short Answer Grading with Deep Learning: From Word Embeddings to Transformers. ArXiv. /abs/2204.03503

Filighera, A., Ochs, S., Steuer, T., & Tregel, T. (2022). Cheating Automatic Short Answer Grading: On the Adversarial Usage of Adjectives and Adverbs. ArXiv. /abs/2201.08318

Mohler, Michael & Bunescu, Razvan & Mihalcea, Rada. (2011). Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments. 752-762.

Yulita, W., Untoro, M., Praseptiawan, M., Ashari, I., Afriansyah, A., & Bin Che Pee, A. (2023). Automatic Scoring Using Term Frequency Inverse Document Frequency Document Frequency and Cosine Similarity. Scientific Journal of Informatics, 10(2), 93 - 104. doi:https://doi.org/10.15294/sji.v10i2.42209