Enhanced Automated Essay Grading With Explainability

- **By Team Crypto Coders**

Shubham Kumar Singh, Prajak Sen, Aayush Kumar Singh, Ujjwal Kumar, Soumyo Nath Tripathy

Institute of Engineering & Management, Kolkata

Abstract

In this paper, we delve into the complex realm of essay scoring, where the task necessitates a sophisticated interplay of analysis, summarization, and judgmental expertise. We address the shortcomings of traditional essay scoring methodologies, characterized by their heavy reliance on manual crafting of features, a process known to be both resource-intensive and yielding sparse results. To overcome these limitations, we propose a novel approach leveraging recurrent neural networks (RNNs) to automatically learn syntactic and semantic features. Additionally, we extend our exploration by integrating TensorFlow LSTM features and string kernels, aiming to enrich the feature representation and enhance the model's performance. Through empirical evaluation, we rigorously compare the efficacy of our neural model against state-of-the-art discrete baselines in both in-domain and domain-adaptation essay scoring tasks. Our findings underscore the superiority of our neural model, marking a significant advancement in the field of automated essay scoring.

Keywords: Essay scoring, Analysis, Summarization, Judgmental expertise, Traditional methodologies, Feature crafting, Resource-intensive, Sparse results, Recurrent neural networks (RNNs), Syntactic features, Semantic features, TensorFlow LSTM, String kernels.

Introduction

To advance automated essay scoring (AES) systems, we propose a novel approach that leverages LSTM-based models with a focus on multi-scale essay representation. This approach addresses key limitations observed in previous LSTM-based methodologies, aiming to enhance the accuracy and effectiveness of automated essay grading.

AES systems play a pivotal role in educational assessment, offering significant potential to streamline the grading process and alleviate the burden on educators. With the increasing prevalence of online education, there is a growing demand for more sophisticated AES frameworks capable of accurately evaluating a wide range of essays.

Traditionally, AES frameworks rely on LSTM or CNN architectures for essay representation and scoring. However, existing LSTM-based approaches may fall short in capturing the diverse characteristics of essays across different granularity levels. Our proposed methodology seeks to address this limitation by incorporating multi-scale features extracted from essays, thereby enabling a more comprehensive and nuanced representation of essay content.

In contrast to previous LSTM-based AES systems, our approach explicitly models multi-scale features to capture the nuances of essays at various levels, including token, sentence, and paragraph levels. By leveraging LSTM architectures, we can effectively process sequential data and extract meaningful representations from essays, leading to improved scoring accuracy.

Through experimental evaluation, we demonstrate that our novel LSTM-based approach outperforms state-of-the-art LSTM-based models in automated essay scoring tasks. By focusing on multi-scale essay representation, our approach achieves superior performance by capturing the diverse characteristics of essays more effectively.

In summary, our proposed LSTM-based methodology represents a significant advancement in AES systems, offering enhanced accuracy and robustness in essay grading. By leveraging LSTM architectures and multi-scale features, we provide a comprehensive framework for automated essay scoring that addresses the evolving needs of educators and learners in the digital age.

Motivation

The motivation behind our work stems from the recognition of the multifaceted nature of automated essay scoring (AES) and the diverse challenges it presents. As AES continues to evolve as a valuable tool for educational assessment, it becomes increasingly clear that no single approach can adequately capture the complexity of written expression and provide accurate evaluations. Therefore, our motivation lies in exploring a comprehensive and integrative approach that harnesses the strengths of multiple methodologies to enhance the effectiveness and reliability of AES systems.

LSTM

Traditional LSTM sequential models have proven effective in capturing sequential dependencies and long-range interactions within essays. However, they may overlook important structural, semantic, and coherence-related aspects of writing that are crucial for holistic assessment. To address this limitation, we seek to integrate alternative approaches, including string kernel, word embeddings, and neural coherence, into the AES framework.

String Kernel: The motivation behind incorporating string kernel methods lies in their ability to capture structural properties and syntactic nuances within essays, complementing the sequential understanding provided by LSTM models. By leveraging string kernel techniques, we aim to extract subtle linguistic patterns and enhance the richness of feature representation, thereby improving the accuracy and granularity of essay assessment.

Word embedding approach: Similarly, word embeddings offer a powerful means of capturing semantic relationships and contextual information within essays. By embedding essays into high-dimensional vector spaces, we can gain deeper insights into the underlying meaning and context of the text, facilitating more nuanced and contextually aware evaluations.

Neural Coherence: Furthermore, neural coherence methods provide a mechanism for explicitly modeling the coherence and cohesion of essays, focusing on the logical flow and organization of ideas. By incorporating coherence-based features extracted from neural networks, we aim to assess the structural integrity and argumentative coherence of student writing, providing valuable feedback on essay organization and clarity.

Our motivation for adopting this integrative approach stems from the belief that by combining the strengths of LSTM sequential models with alternative methodologies such as string kernel, word embeddings, and neural coherence, we can develop more robust, comprehensive, and effective AES systems. By embracing diversity in approach and methodology, we strive to enhance the overall quality and utility of automated essay scoring, ultimately benefiting educators, students, and educational institutions alike.

Literature Review

Automated essay grading (AEG) has traversed a fascinating journey, evolving from rudimentary rule-based systems to sophisticated machine learning models empowered by cutting-edge natural language processing (NLP) techniques. This progression has been catalyzed by significant breakthroughs in AI research, exemplified by seminal works such as "On the Use of BERT for Automated Essay Scoring: Joint Learning of Multi-Scale Essay Representation" (Tran et al., 2022) and "Countering the Influence of Essay Length in Neural Essay Scoring" (Considering-Content-XLNet, 2021). These studies showcase the transformative impact of deep learning architectures like BERT and XLNet, which enable AEG systems to discern intricate nuances in essay content, leading to more accurate and nuanced grading outcomes. Furthermore, real-world implementations of AEG in standardized testing by esteemed organizations like Educational Testing Service (ETS) have revolutionized assessment practices, streamlining grading processes and providing timely feedback to test-takers. Similarly, the integration of AEG into online learning platforms such as Coursera has democratized education, enabling learners worldwide to receive personalized feedback on their written assignments. Despite these advancements, AEG faces persistent challenges related to bias, fairness, and interpretability, prompting ongoing research efforts to address these issues and ensure the ethical and equitable deployment of AEG systems. Moreover, resources like Wikipedia serve as invaluable repositories of knowledge, offering comprehensive insights into the historical evolution, methodologies, and applications of AEG. As interdisciplinary collaboration and innovation continue to drive progress in the field, the future of AEG holds immense promise for revolutionizing essay assessment, paving the way for more efficient, accurate, and scalable grading solutions in diverse educational contexts.

Methodology

Feature Engineering

1. **Word Count Feature**:
   Implement a function to count the number of words in each essay.
   Tokenize each essay and count the number of tokens, excluding stopwords.
   Store the word counts in a new column in the dataset.

2. **Vocabulary Size Feature**:
   Develop a function to calculate the vocabulary size of each essay.
   Tokenize and lemmatize each essay, remove stopwords, and retain unique words.
   Count the number of unique words in each essay and store the results.

3. **Part-of-Speech (POS) Tagging Feature**:
   Create a function to extract POS tag counts for nouns, verbs, adjectives, and adverbs in each essay.
   Tokenize each essay and perform POS tagging using NLTK.
   Count the occurrences of each POS tag and store the results in separate columns.

4. **Sentence Length Feature**:
   Define a function to compute the number of sentences in each essay.
   Tokenize each essay into sentences and count the number of sentences.
   Store the sentence lengths in a new column in the dataset.

5. **Bag of Words (BOW) Feature**:
   Implement a function to create a bag of word representations for each essay.
   Tokenize each essay into words, remove stopwords, and lemmatize the words.
   Create a dictionary where each key corresponds to an essay set
   Extract the most frequent words from each essay set and store them for feature extraction.

6. **Frequent Words Feature**:
   Develop a function to count the occurrences of pre-defined frequent words in each essay.
   Iterate through each essay, and count the occurrences of frequent words.

7. **Spell Check Feature**:
   Create a function to compute the percentage of misspelled words in each essay.
   Tokenize each essay, lemmatize, and remove stopwords
. Check each word against the WordNet dictionary to identify misspelled words.
   Calculate the percentage of misspelled words and store the results.

8. **Essay Perplexity Feature**:
   Define a class to calculate the perplexity of each essay.

Train the perplexity model on a set of training essays to create n-gram counts.
For each essay, compute the perplexity based on the trained model and essay content.
Store the perplexity scores as additional features.

9. **Data Storage**:
Save the preprocessed dataset with extracted features as a pickle file for further analysis.
By following this methodology, we can systematically extract a diverse range of features from the essays, capturing various linguistic, structural, and semantic aspects essential for automated essay scoring. These features will serve as input to the subsequent modeling phase, facilitating the development of robust and accurate AES systems.

essay_set	word_count	vocab_count	n_count	v_count	adj_count	adv_count	sent_len	freek	wrong_spell	perplexity
1	0.397463	0.382239	0.253870	0.38	0.285714	0.234043	0.223881	0.320513	0.102667	188.689232
1	0.479915	0.482625	0.356037	0.60	0.250000	0.319149	0.283582	0.461538	0.124883	199.735366
1	0.320296	0.339768	0.256966	0.32	0.250000	0.127660	0.194030	0.435897	0.043411	168.040531
1	0.635307	0.637066	0.628483	0.62	0.547619	0.297872	0.388060	0.512821	0.175000	247.198548
1	0.535941	0.490347	0.349845	0.49	0.273810	0.446809	0.432836	0.512821	0.174194	168.929019

Fig. : Test Data Statistics

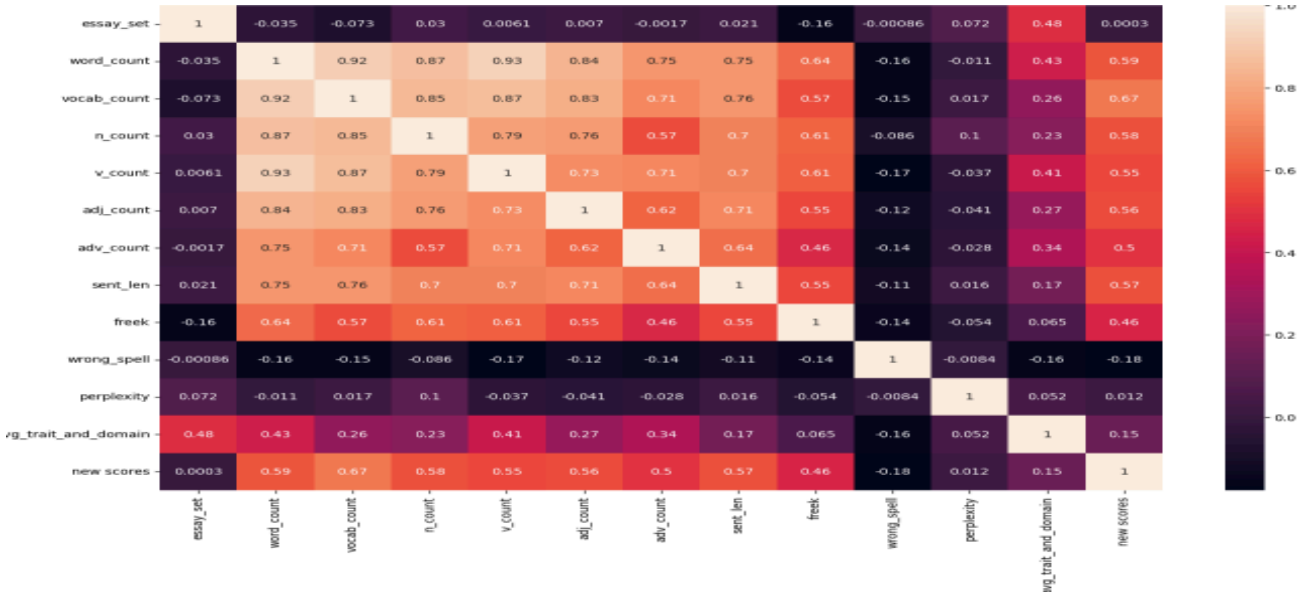


Fig. : Heatmap for generating Data correlation

Data Preprocessing:

In preprocessing the data, we initially calculate the average scores for traits 1, 2, and 3, as well as for the domain attributes, which are then added as new features to the dataset. Subsequently, we construct a new dataset by selecting pertinent attributes from the original data and incorporating the calculated averages.

This amalgamated dataset serves as the foundation for further analysis. Finally, to streamline the dataset and focus on key attributes, we eliminate redundant columns such as the domain scores and individual trait averages. This streamlined dataset is then ready for subsequent analysis and modeling, enabling a more focused and effective approach to grading and assessment.

Let a , b, and c represent the mean values of traits 1, 2, and 3 respectively, calculated across the specified attributes. These mean values are derived using the formula:

$$a = \frac{\sum_{i=1}^{6} X_i}{6}, \quad b = \frac{\sum_{i=7}^{12} X_i}{6}, \quad c = \frac{\sum_{i=13}^{18} X_i}{6}$$

where xi denotes the values of the respective attributes. After obtaining these averages, a new dataset xi is created, incorporating the original attributes along with the calculated mean values. This dataset x is then refined by removing redundant columns such as the domain scores and individual trait averages, resulting in a streamlined dataset ready for further analysis.

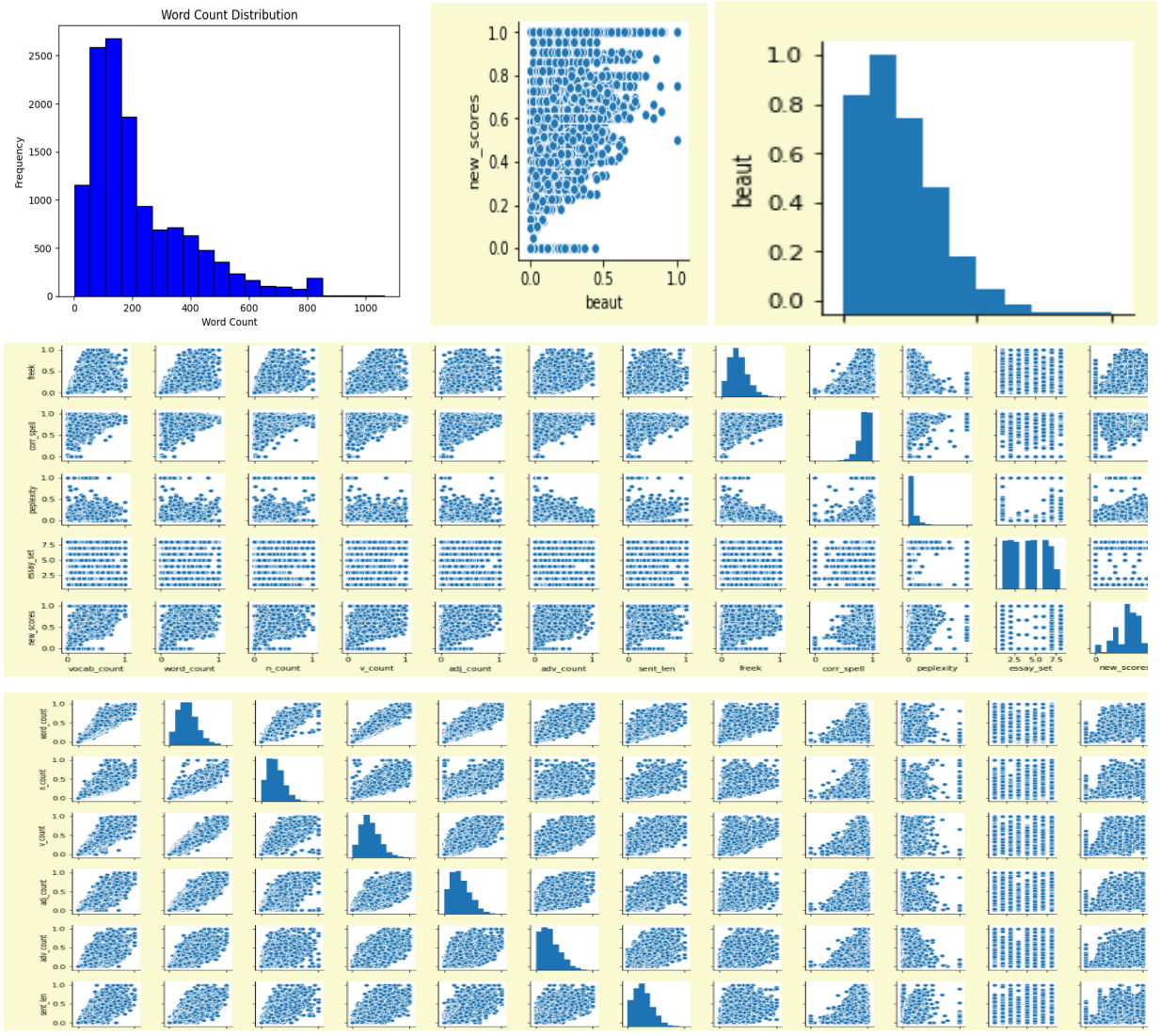


Fig. : Relational plots

The Learning:

1. **MODEL DEFINITION:-**
    - A 2-layer LSTM model is defined using Keras. It consists of two LSTM layers followed by a dropout layer and a dense layer.
    - Relu activation is used in the output layer since the scores are not normalized.
    - Mean squared error is used as the loss function, and RMSprop is used as the optimizer.

LSTM Layer 1:
h_t(1) = LSTM(x_t, h_t-1(1))
h_t(1) represents the hidden state of the first LSTM layer at time t.
x_t denotes the input at time t.
h_t-1(1) is the previous hidden state of the first LSTM layer.

LSTM Layer 2:
h_t(2) = LSTM(h_t(1), h_t-1(2))
h_t(2) represents the hidden state of the second LSTM layer at time t.
h_t(1) is the output of the first LSTM layer at time t.
h_t-1(2) is the previous hidden state of the second LSTM layer.

Dropout Layer:
y = Dropout(h_t(2))
y represents the output after applying dropout to the hidden state of the second LSTM layer.

Output Layer:
y^ = ReLU(W * y + b)
y^ denotes the predicted score.
W and b are the weight matrix and bias vector of the dense layer.
Output Layer with RELU Activation:
y^ = ELU(W * y + b)
y^ denotes the predicted score.
W and b are the weight matrix and bias vector of the dense layer.

Loss Function (Mean Squared Error):
MSE = 1/n * Σ(y - y^)^2
MSE denotes the mean squared error.
n is the number of samples.
y is the true target value.
y^ is the predicted value.

Optimizer (RMSprop):
RMSprop update rule:
v_dw = β * v_dw + (1 - β) * (∂MSE/∂W)^2
v_db = β * v_db + (1 - β) * (∂MSE/∂b)^2

$W = W - (learning\_rate / \sqrt{(v\_dw + \varepsilon)}) * \partial MSE/\partial W$
$b = b - (learning\_rate / \sqrt{(v\_db + \varepsilon)}) * \partial MSE/\partial b$
W and b are the weights and biases.
β is the momentum parameter.
ε is a small constant to avoid division by zero.
$\partial MSE/\partial W$ and $\partial MSE/\partial b$ are the gradients of the loss with respect to weights and biases.

2. **Training with Epochs**

Training with Epochs:

Training with epochs refers to the process of iteratively updating the parameters of a neural network model for a fixed number of times over the entire training dataset. Each iteration through the dataset constitutes one epoch. The purpose of training with epochs is to allow the model to gradually learn from the training data and adjust its parameters to minimize the chosen loss function.

During each epoch, the training process typically involves the following steps:
1. **Forward Pass**: Input data is fed into the model, and predictions are generated.
2. **Loss Calculation**: The model's predictions are compared to the ground truth labels, and the loss is computed using a specified loss function.
3. **Backward Pass (Backpropagation)**: Gradients of the loss with respect to the model parameters are computed using backpropagation.
4. **Parameter Update**: The model parameters (weights and biases) are adjusted using an optimization algorithm (e.g., RMSprop, Adam) to minimize the loss.

Training with multiple epochs allows the model to learn complex patterns from the data and converge to a better solution. However, excessive training epochs may lead to overfitting, where the model performs well on the training data but poorly on unseen data.

Training with k-Fold Cross Validation:

Training with k-Fold Cross Validation is a technique used to assess the performance and generalization ability of a machine learning model. It involves dividing the dataset into k subsets (or folds) of approximately equal size. The model is trained and evaluated k times, each time using a different fold as the validation set and the remaining folds as the training set.

The steps involved in k-Fold Cross Validation are as follows:
1. **Dataset Splitting**: The dataset is divided into k subsets (folds).

2. **Iterative Training and Validation**: For each iteration, one fold is held out as the validation set, and the model is trained on the remaining k-1 folds.
3. **Model Evaluation**: The trained model is evaluated on the validation set, and a performance metric (e.g., accuracy, loss) is computed.

4. **Average Performanc**e: The performance metrics obtained from each iteration are averaged to obtain an overall assessment of the model's performance.

k-Fold Cross Validation helps to provide a more reliable estimate of the model's performance by reducing the variance in the evaluation metric compared to a single train-test split. It also allows for better utilization of the available data, especially in cases where the dataset is limited in size.

By combining training with epochs and k-Fold Cross Validation, we can train robust machine learning models that generalize well to unseen data and are less prone to overfitting.
Sure, here's the information formatted in LaTeX:

Training with Epochs:

1. **Forward Pass:**
   - Input data: X
   - Predictions: $\hat{y} = f_\theta(X), \hat{y}$ is the neural network model .

2. **Loss Calculation:**
   - Mean Squared Error (MSE):

$$\[\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\]$$

3. **Backward Pass (Backpropagation):**
   - Compute gradients:

$$[\nabla_\theta \text{MSE} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)\nabla_\theta \hat{y}_i\]$$

4. **Parameter Update:**
   - Using an optimization algorithm (e.g., RMSprop):

$$\[\theta_{t+1} = \theta_t - \eta \frac{1}{\sqrt{v_t + \epsilon}} \nabla_\theta \text{MSE}\]$$

Training with k-Fold Cross Validation:

1. **Dataset Splitting:**
Divide the dataset into
k subsets (folds):

$$D1, D2, \ldots, DkD1, D2, \ldots, Dk$$

-

2. **Iterative Training and Validation:**

- $\bigcup_{j \neq i} D_j$ (training set). – Validate the model on D_i

3. **Model Evaluation:**
   - Compute performance metric (e.g., MSE) on the validation set.

4. **Average Performance**:
   - Average the performance metrics obtained from each fold to assess the model's overall performance.

Other models we have explored:

Sure, let's discuss the mathematical formulations for each of the mentioned models without diving into the code:

1. **XGBoost (Extreme Gradient Boosting)**:
XGBoost is an ensemble learning method that builds a series of decision trees sequentially. It minimizes a loss function by adding new trees that predict the residuals or errors of the previous trees. The final prediction is the sum of predictions from all the trees.
The objective function to minimize in XGBoost is typically a combination of a loss function

$L(y_i, \hat{y}_i)$ and a regularization term $\Omega(f)$ to control the complexity of the model:
$\text{Objective} = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$

2. **AdaBoost (Adaptive Boosting):**
AdaBoost is also an ensemble learning method, but unlike XGBoost, it assigns weights to each training sample and adjusts these weights at each iteration to focus on the samples that are difficult to classify correctly.
At each iteration, AdaBoost fits a weak learner to the data and assigns a weight \(\alpha_t \) to it based on its performance. The final prediction is a weighted sum of the weak learners' predictions:

$$F(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

3. **RandomForest Regressor:**
RandomForest is an ensemble learning method that constructs multiple decision trees during training and outputs the average prediction of the individual trees.
The prediction of the RandomForest is the average prediction of all the trees:

$$F(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$

4. **SVR (Support Vector Regressor)**:
SVR is a type of Support Vector Machine (SVM) that is used for regression tasks. It finds a hyperplane that best fits the data while minimizing the margin violations.

- The objective of SVR is to minimize the following cost function:
  $\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$
- Subject to the constraints:
  $y_i - \langle w, x_i \rangle - b \leq \epsilon + \xi_i$
  $\langle w, x_i \rangle + b - y_i \leq \epsilon + \xi_i^*$
  $\xi_i, \xi_i^* \geq 0$

5. **Bayesian Ridge Regression**:
Bayesian Ridge Regression is a probabilistic regression model based on Bayesian inference. It assumes a Gaussian prior on the regression weights and computes the posterior distribution of the weights given the data.

The objective of Bayesian Ridge Regression is to maximize the log marginal likelihood of the data, which can be formulated as:

$$\max_{w,\alpha} -\frac{1}{2}(\alpha||w||^2 + (y - Xw)^T(y - Xw)) - \frac{n}{2}\log\alpha - \frac{d}{2}\log(2\pi)$$

These mathematical formulations provide insights into how each model optimizes its parameters to make predictions on the given data. Understanding these formulations is crucial for implementing and fine-tuning these models effectively for automatic essay grading.

OUR PROJECT AS END TO END APPLICATION

For Vectorization we have employed TFIDF vectorizer from ScikitLearn to convert our essay text into Model digestible input. We have employed our Language tool API just for handling our input texts and parsing texts through a remote server: https://language-tool-api.mywebsite.net. It's a Python wrapper for language tools and it is also used to spell and grammar check for documents.

The integration of TF-IDF vectorization with the Language Tool API can significantly enhance the effectiveness and usability of a web-based essay grading system. Here's how:

1. **Improved Text Representation**:
TF-IDF vectorization transforms the raw essay text into a numerical representation that captures the importance of each word or term in the document relative to the entire corpus. By using TF-IDF, the model can better understand the content and context of the essays, enabling more accurate grading based on the relevance and frequency of specific words and phrases. This improved text representation enhances the model's ability to identify key features and patterns in the essays, leading to more precise grading outcomes.

2. **Enhanced Language Analysis**:
Integrating the Language Tool API allows for comprehensive spell-checking, grammar-checking, and language analysis of the essays. The API can identify and correct spelling errors, grammatical mistakes, punctuation issues, and other linguistic errors that may affect the quality and readability of the essays.

3. **User-Friendly Interface**:
Incorporating these tools into a web-based platform provides users, such as students and educators, with a seamless and user-friendly experience. Users can simply upload their essays to the platform, which then automatically performs TF-IDF vectorization and language analysis using the Language Tool API. The platform can display the graded essays along with detailed feedback on spelling, grammar, and language usage, allowing users to easily understand their strengths and areas for improvement.

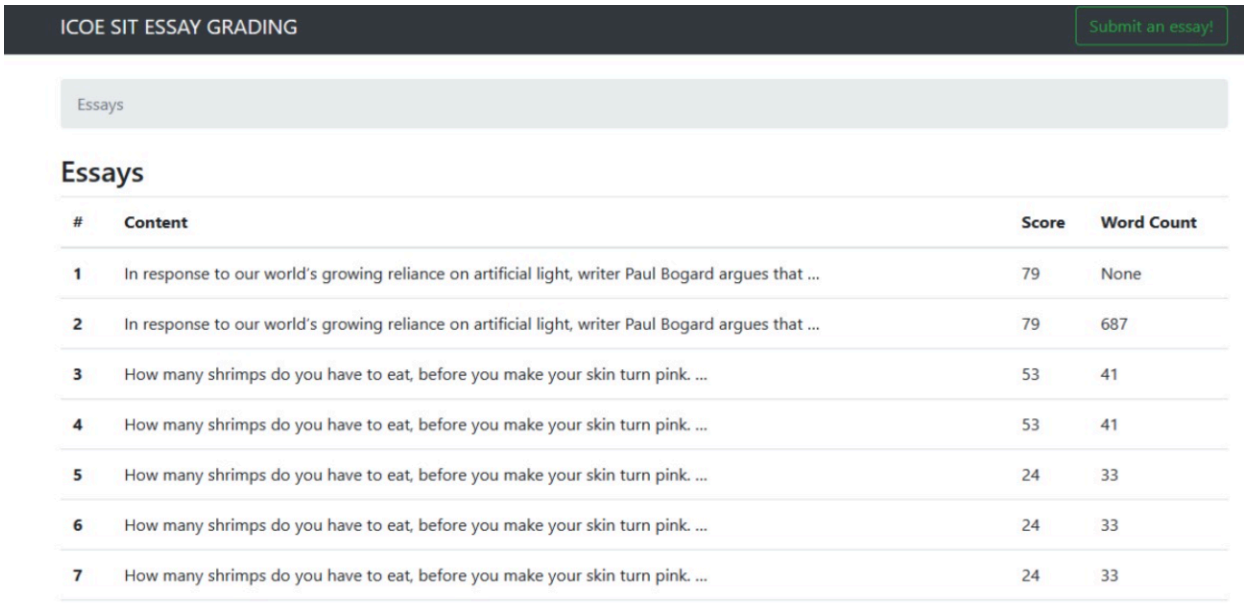


Fig. : Search history

.
In summary, integrating TF-IDF vectorization with the Language Tool API in a web-based essay grading system enhances text representation, language analysis, user experience, and feedback mechanisms. This combination facilitates more accurate and insightful grading while promoting continuous improvement in students' writing skills.

Data Acquisition:

The first step in developing the enhanced AEG system is to acquire a suitable dataset for training and evaluation. One of the commonly used datasets in the AEG research community is the ASAP-AES dataset, available on Kaggle. This dataset contains essays written in response to prompts from the SAT Reasoning Test, along with human-assigned scores and essay features such as word count and syntactic complexity. In addition to the Kaggle dataset, we will explore other potential sources of data, including

datasets from educational institutions or research labs, to augment the training data and improve the robustness of the model. We take the data from https://www.kaggle.com/c/asap-sas/

Evaluation:

Evaluation based on Kappa score for automated essay grading system
The Kappa score, also known as Cohen's Kappa coefficient, is a statistical measure commonly used to assess the inter-rater agreement between two raters. In the context of automated essay grading systems, where the system's scores are compared against human scores, the Kappa score can provide valuable insights into the system's performance.

Here's how the Kappa score evaluation process typically works for automated essay grading systems:

1. **Data Preparation**:
   Collect a dataset of essays along with their corresponding human-assigned scores and system-generated scores.
   Ensure that the dataset covers a diverse range of topics, writing styles, and difficulty levels to provide a comprehensive evaluation.

2. **Calculation of Agreement**:
   Calculate the Kappa score to quantify the agreement between the human-assigned scores and the system-generated scores.
   The Kappa score is calculated using the formula:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

3. **Interpretation of Kappa Score**:
   Interpret the Kappa score based on commonly accepted guidelines to assess the level of agreement between the human-assigned scores and the system-generated scores.

4. **Analysis and Improvement**:
   Analyze the Kappa score and identify areas where the automated essay grading system performs well areas where improvement is needed.
   Use insights from the evaluation to refine the system's algorithms, features, or training data to enhance its performance and agreement with human raters.

5. **Validation and Iteration**:
   Validate the improvements made to the system using additional datasets and repeat the evaluation process.Iterate on the system's design and implementation based on feedback from multiple rounds of evaluation, aiming to achieve higher levels of agreement with human ratings.This comprehensive evaluation process, guided by the Kappa score, ensures rigorous assessment and continuous improvement

of automated essay grading systems, ultimately enhancing their reliability and accuracy in evaluating written content.This integration provides both the theoretical understanding of Kappa score evaluation and its mathematical formulation in the context of automated essay grading systems.

Performance Metrics & Explainability Metrics:

In our quest to refine and optimize our automated essay grading system, we've embarked on a journey to develop a comprehensive evaluation framework that delves deep into the intricacies of essay assessment. Building upon traditional metrics like the Kappa score, we've meticulously curated a suite of evaluation metrics that reflect both the performance and explainability of our system. I have taken the reference from Feedback Prize - English Language Learning | Kaggle

Performance Metrics:

1. **Unveiling Content Depth**:
   Word Counts: word counts provides insights into the depth and breadth of ideas
   POS (Part-of-Speech) Count: the distribution of different parts of speech assess

2. **Structural Sophistication**:
Sentence Length: Examining sentence length aids in understanding the complexity and readability of essay structure.

3. **Language Mastery**:
Beauty: Assessing the beauty of language encompasses elements such as eloquence, imagery, and stylistic finesse.
Grammar and Spelling: Identifying and correcting grammatical and spelling errors ensures linguistic accuracy and clarity.
Cohesion: Analyzing the coherence and logical flow of ideas enhances the essay's overall effectiveness.

4. **Semantic Precision**:
Phraseology: Evaluating the use of phrases and idiomatic expressions contributes to the essay's depth and sophistication.
Semantic Similarity: Assessing semantic similarity between essays helps gauge the originality and uniqueness of content.

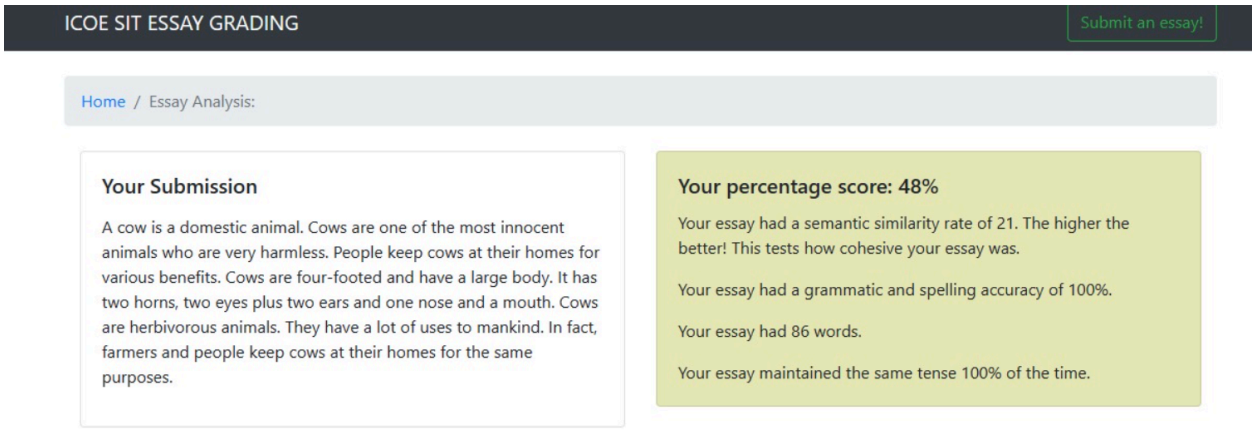


Fig. : Application output

Explainability Metrics:

1. **Transparency**:

Feature Importance: Determining the importance of different features in the grading process enhances transparency and understanding.

Model Outputs: Providing clear explanations of the model's outputs and decision-making process improves interpretability.

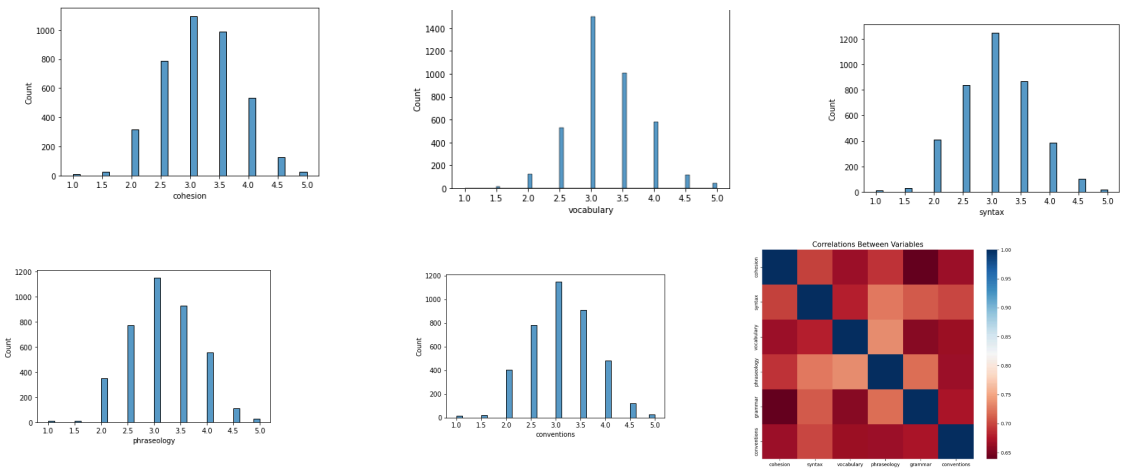
2. **Consistency**:

Scoring Consistency: Ensuring consistency in scoring across different essays and graders enhances trust and reliability.

Our evaluation methodology is not just about performance; it's about understanding and transparency. Drawing inspiration from reputable sources and employing advanced techniques, we've crafted a robust evaluation framework that reflects the complexity and nuance of writing quality.

	word_count	vocab_count	n_count	v_count	adj_count	adv_count	sent_len	freek	wrong_spell	beaut	perplexity
0	275	77	67	29	21	15	13	26	8.771930	26	163.218737
1	353	95	75	35	15	21	20	33	1.526718	25	154.934685
2	471	137	91	51	29	26	15	35	13.170732	38	112.750824
3	410	103	103	49	21	17	24	31	4.733728	38	156.539091
4	496	132	122	39	38	22	34	47	4.824561	54	141.841194

Fig. : Our new attributes to judge the essays



	cohesion	syntax	vocabulary	phraseology	grammar	conventions
0	3.5	3.5	3.0	3.0	4.0	3.0
1	2.5	2.5	3.0	2.0	2.0	2.5
2	3.0	3.5	3.0	3.0	3.0	2.5
3	4.5	4.5	4.5	4.5	4.0	5.0
4	2.5	3.0	3.0	3.0	2.5	2.5
...	...	...	...	...	...	...
3906	2.5	3.0	3.0	3.5	2.5	2.5
3907	4.0	4.0	4.0	4.0	3.5	3.0
3908	2.5	3.0	3.0	3.0	3.5	3.0
3909	4.0	4.5	4.5	4.0	4.5	4.5
3910	3.5	2.5	3.5	3.0	3.0	3.5

Fig. : 5 notable features for judgements

Results and Analysis

Performance Metrics:

In assessing the performance of our automated essay scoring system, we adopt a multifaceted approach that encompasses semantic correlation, cohesiveness, grammar, and phraseology as key performance metrics. Semantic correlation metrics are utilized to gauge the alignment between essay content and given prompts, leveraging techniques like semantic similarity scores for an accurate assessment. Cohesiveness analysis evaluates the logical flow and coherence of ideas within essays, examining transitional elements and overall structural integrity. Grammar metrics focus on detecting and rectifying grammatical errors, ensuring linguistic accuracy. Additionally, phraseology assessment scrutinizes the richness of vocabulary, diversity of sentence structures, and appropriateness of phrase usage to gauge the essay's linguistic sophistication. By integrating these metrics, we aim to provide a comprehensive evaluation of our system's ability to accurately assess essay quality, ultimately striving to deliver reliable and informative scores that reflect the proficiency and merit of the written content.

	essay_id	essay_set	essay	avg_trait_and_domain	new_scores
0	1	1	Dear local newspaper, I think effects computer...	8.0	0.6
1	2	1	Dear @CAPS1 @CAPS2, I believe that using compu...	9.0	0.7
2	3	1	Dear, @CAPS1 @CAPS2 @CAPS3 More and more peopl...	7.0	0.5
3	4	1	Dear Local Newspaper, @CAPS1 I have found that...	10.0	0.8
4	5	1	Dear @LOCATION1, I know having computers has a...	8.0	0.6

Fig. : Scored essays

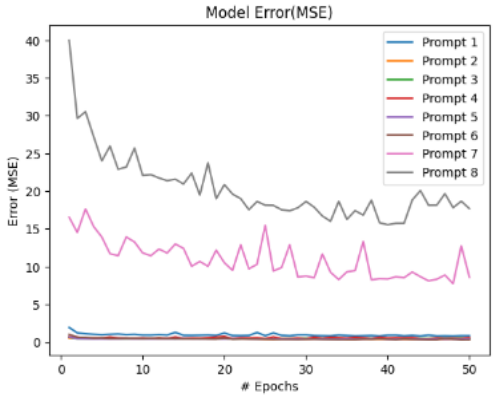
```
Kappa Score: 0.6891631136212879
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your mode
  saving_api.save_model(

61] print("Average Kappa score after a 5-fold cross validation: ", np.round(np.array(results).mean(),decimals=4))

Average Kappa score after a 5-fold cross validation:  0.6661
```

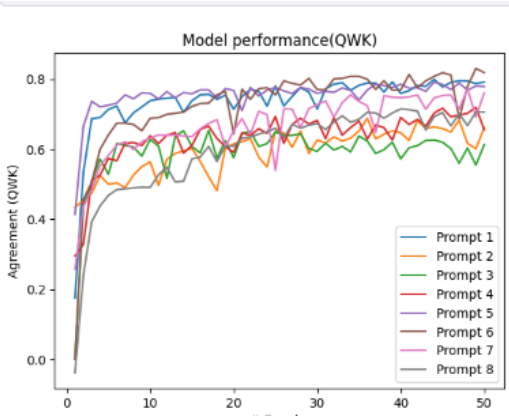
Fig. : OUR KAPPA SCORE

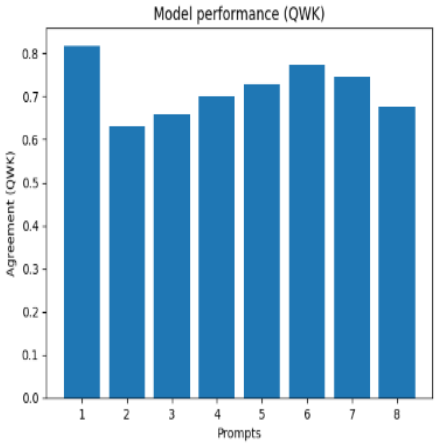
Explainability Analysis:

In assessing the explainability of our automated essay grading system, we first examine the model architecture and parameters, such as embedding dimension, hidden dimension, and number of epochs, which define its structure and complexity. Additionally, the loss function, represented by the Mean Squared Error (MSE) in our case, provides insight into how the model optimizes its parameters during training. By understanding these components, stakeholders can gain clarity on how the model processes

and evaluates essay data. Furthermore, visualization techniques, such as graphs displaying agreement (Quantified Weighted Kappa, QWK) and error (MSE) over epochs, offer transparency into the model's performance and the validation process. Through these explainability metrics, we aim to enhance transparency and understanding of our automated essay grading system's inner workings.








```
corresponding loss 763.16156
corresponding loss 501.65042
corresponding loss 258.86816
corresponding loss 118.326324
corresponding loss 59.73414
corresponding loss 38.996483
corresponding loss 30.283293
corresponding loss 25.192894
corresponding loss 21.473267
corresponding loss 18.467579
corresponding loss 16.181269
corresponding loss 14.329625
corresponding loss 12.78288
corresponding loss 11.400213
corresponding loss 10.238663
corresponding loss 9.18619
corresponding loss 8.308787
corresponding loss 7.628619
corresponding loss 7.052447
corresponding loss 6.557523
corresponding loss 6.1354785
corresponding loss 5.7908707
corresponding loss 5.456056
corresponding loss 5.1838856
corresponding loss 4.957857
```

Fig. : Loss functions and comparative model analysis

Future Implementation Plan

Roadmap:

In our exploration of automated essay scoring, we have delved into various cutting-edge approaches and methodologies as evidenced by the exploration of recent advancements in the field. This includes investigations into models like Tran-BERT-MS-ML-R, which leverages BERT for joint learning of multi-scale essay representation, achieving a top-ranking Quadratic Weighted Kappa (QWK) score of 0.791 in 2022. Additionally, studies such as Considering-Content-XLNet have addressed challenges like essay length bias, achieving a competitive QWK score of 0.786 in 2021. Other notable contributions include research on models like HISK+BOSWE, SkipFlow, and MHMLW, each showcasing innovative techniques such as string kernels, neural coherence features, and ensemble methods for scoring essays automatically. Through our exploration of these diverse approaches, we aim to stay at the forefront of advancements in automated essay scoring, continually refining our methods to enhance accuracy and reliability in assessing written content.

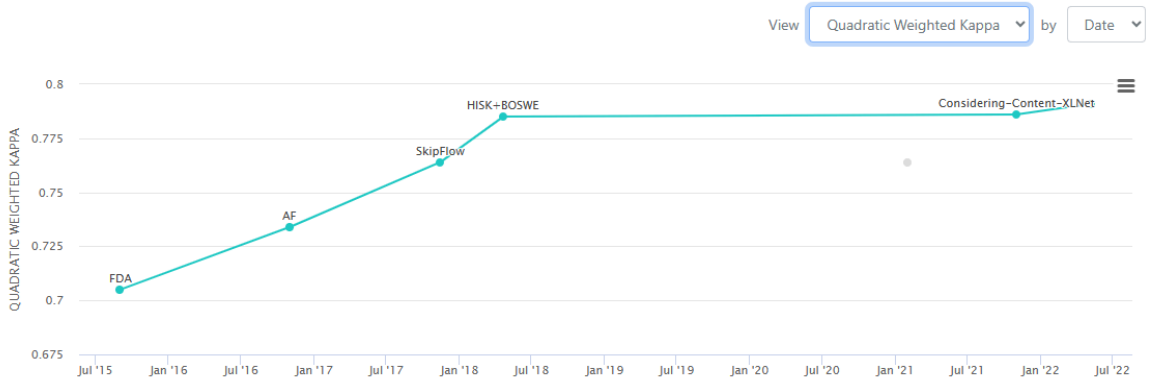


Fig. : All proposed models

Conclusion

This project presents a novel approach to Automated Essay Grading (AEG) that prioritizes transparency and interpretability through the integration of explainable AI techniques. By providing clear explanations for essay scores, the proposed system aims to empower both educators and students in the learning process. Educators gain insights into the strengths and weaknesses of student essays, enabling them to provide targeted feedback and support. Students benefit from a deeper understanding of the grading criteria and areas for improvement in their writing skills. Future developments will focus on refining the model, deploying it in educational settings, and conducting empirical studies to evaluate its effectiveness in improving student learning outcomes.

References

1. Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v.2. The Journal of Technology, Learning, and Assessment, 4(3).
2. Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. In arXiv: Computation and Language.
3. Yue Cao, Hanqi Jin, Xiaojun Wan, and Zhiwei Yu. 2020. Domain-adaptive neural automated essay scoring. In SIGIR ’20: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information, pages 1011–1020.
4. Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1741–1752.
5. J Cohen. 1968. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. Psychological Bulletin, 70(4):213–220.
6. Peter Phandi, Kian Ming A Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression.
7. Lawrence M Rudner and Tahung Liang. 2002. Automated essay scoring using Bayes’ theorem. The Journal of Technology, Learning and Assessment, 1(2).
8. Alex Smola and Vladimir Vapnik. 1997. Support vector regression machines. Advances in neural information processing systems, 9:155–161.
9. Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. Journal of Machine Learning Research, 9(2579-2605):85.
10. Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. arXiv preprint arXiv:1606.04289.
11. Yigal Attali and Jill Burstein. 2004. Automated essay scoring with e-rater R v. 2.0. ETS Research Report Series, 2004(2):i–21.