AI-Explorers

Gurudatta Patil	203050062
Kunal Verma	203050121
Mallela Niteesh Kumar Reddy	203050065
Nilesh Kshirsagar	203059004

Automated Essay Grading

Contents

1 Problem Statement

Grading of essays have been done for many years where the writer writes on certain topic and then a human being grades the essay on the basis of "goodness" of the essay. Humans can be sometimes biased towards grading and most of the times do not have an objective measurement of the grading provided. Automated Essay Scoring (AES) systems can actually help in removing this tedious process of manual grading of essays. At the same time the system grades the essays uniformly without any biases.
Large number of companies also use automatic resume screening programs for selecting the best potential candidates. However the task of grading such diverse essay topics and understanding linguistic nuances of the text using automated systems is not a trivial task.
The objective of our project is to assign grade to the essay from a given range (say 1-10) using natural language processing. We wish to implement different approaches and present an analysis of each of different approaches. The approaches include classical ML based models (Support Vector Machines, Random Forest, Decision trees) as well as Deep Learning based models (RNN and LSTMs). From these models we would like to grade essays and simultaneously provide attributes and traits that were responsible for receiving the score.
We would present the scores and the importance of different features responsible for the grade graphically on web based interface. Our main task would be to explain the the grade which was assigned by the system and which linguistic and grammatical features are considered in grading such essays. Aim and Learning Objectives: Through the project we would be able to understand the computational aspect of linguistics and how algorithms objectively analyse abstract concept of semantics in the language. We would quantitatively present a bar graph/pie chart for top traits which were responsible for the assignment of the score. This would benefit the users to see parts of essays which can be improved as well as which parts were well written.

2 Solution Software Architecture

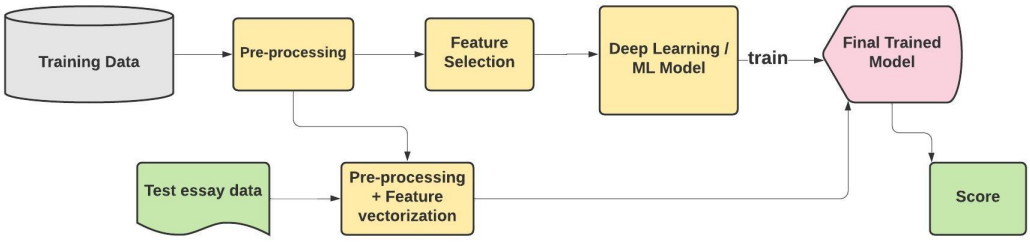

Figure 1: Architecture of our model

2.1 Training Data

We have used kaggle dataset (The Hewlett Foundation: Automated Essay Scoring). We have used eight essay sets. Each of the sets of essays was produced using single prompt. Selected essays range from 150 to 550 words per answer in average length. All essays were written by students ranging from Grade 6 to Grade 10. All of the essays are graded by hand and double-scored. Each of the eight sets of essays has unique features of its own. The variability is meant to measure the limits of the capabilities of scoring engine.There are 28 columns in each of these files. Each student essay has its own unique identifier known as essay-id. It also contains essay-set from 1-8 which is a unique identifier for each set of essays. The first domain is rated by three raters and the score of that domain is calculated accordingly.

2.2 Pre-Processing

We will convert every sentence into lower case as first step of preprocessing and then we remove tagged labels, punctuation, stopwords from each sentence and then we tokenize the sentence.

2.3 Feature selection

We extract features such as word count, sentence count, Part of Speech tags from pre-processed essays. This can be done using NLTK library which has useful functions to tokenize sentences and assign Part of Speech tags. Number

of noun tags, verb tags, adverb tags and adjective tags can be extracted using a tagged sentence. Lexical diversity of the essay can be judged using unique bigram count in the essay. We will use NLTK library for extracting this feature. Apart from these statistical features, we will try to explore some other features such as term-document frequency based and topic relevance based features.

2.4 Models

After obtaining features, we use them to train our Machine learning and deep-learning models. We have used 4 different models which are:-

1. Support Vector Regression
2. Decision Tree and Random Forests
3. FeedForward Deep Neural Network
4. RNNs: Long Short Term Memory(LSTM)

SciKit Learn library's implementation of Support Vector Regression, Decision Tree and Random Forest was used. DNN and RNN-LSTM implementation from the TensorFlow library was also tested against the feature sets. Each of these models were trained using 5-fold cross validation method and measure the Quadratic Weighted Kappa for each fold. This metric measures the agreement between two ratings. This metric typically varies from 0 to 1, where 0 signifies random agreement between raters and 1 means complete agreement between raters. Every model was hyper-tuned to produce various set of results on the test data.

2.5 Code written

1. Pre-processing steps:- Our code
2. Deep learning/ ML modules:- referenced from git repositories with necessary adaptations. Reference to relevant urls are provided in the references section.
3. User Interface:- Our code

3 Progress report

3.1 Running Modules

3.1.1 Pre-Processing module

Lowercasing, stopword removal, tokenization of strings.
Example
Input : I Don't Like such computers!!?
Output: ['dont', 'like', 'computers']

3.1.2 GUI feature

After completing the local feature importance extractor, we incorporated the graphs which is our final aim in the explanation of feature importance for a particular essay.

3.1.3 Feature Extraction

```
def count_bigrams(essay):
    sentences = nltk.sent_tokenize(essay)
    bigram_count = 0
    for sentence in sentences :
      bigrams = [grams for grams in nltk.ngrams(sentence.split(), 2)]
      bigram_count += len([(item, bigrams.count(item)) for item in sorted(set(bigrams))])
    return bigram_count

def get_wordlist(sentence):
    clean_sentence = re.sub("[^A-Z0-9a-z]"," ", sentence)
    wordlist = nltk.word_tokenize(clean_sentence)
    return wordlist

def tokenize(essay):
    stripped_essay = essay.strip()
    tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
    raw_sentences = tokenizer.tokenize(stripped_essay)
    tokenized_sentences = []
    for raw_sentence in raw_sentences:
        if len(raw_sentence) > 0:
            tokenized_sentences.append(get_wordlist(raw_sentence))
    return tokenized_sentences
```

Figure 2: Feature Extraction Code

```
def sent_count(essay):
    sentences = nltk.sent_tokenize(essay)
    return len(sentences)

def count_pos(essay):
    tokenized_sentences = tokenize(essay)
    noun_count = 0
    adj_count = 0
    verb_count = 0
    adv_count = 0
    for sentence in tokenized_sentences:
        tagged_tokens = nltk.pos_tag(sentence)
        for token_tuple in tagged_tokens:
            pos_tag = token_tuple[1]
            if pos_tag.startswith('N'):
                noun_count += 1
            elif pos_tag.startswith('J'):
                adj_count += 1
            elif pos_tag.startswith('V'):
                verb_count += 1
            elif pos_tag.startswith('R'):
                adv_count += 1
    return noun_count, adj_count, verb_count, adv_count
```

Figure 3: Feature Extraction Code

Following features were extracted from input essays.

- Word count
- Sentence count
- Bigram count
- Count of POS tags - This count includes four separate features - noun count, adjective count, adverb count and verb count

Output Feature Vector : [333, 16, 284, 74, 19, 71, 25]
Each of the above integer in the array corresponds to word count, sentence count, bigram count, nouns, adjectives, adverbs and verbs respectively. Code used to extract these features is given in snippets above.

In case of LSTM and DNN based models the feature vectors are corresponding to the word vectors which are learnt from Word2Vec embeddings. The word embeddings represent words in vector form.

3.1.4 Machine Learning Models

Using the above features, we have trained some machine learning models.

- Support Vector Regressor
- Random Forest Regressor
- Decision Tree Regressor

Each of the above algorithms has a number of hyper-parameters. We trained them using different set of values for these hyper-parameters.

3.1.5 Deep Learning Models

- Feed Forward Neural Network
- Long Short Term Memory(LSTM)

3.1.6 Web Application

We have implemented initial web-based application designed on Django framework. The initial web application takes input an essay and gives an overall score for the given essay. The range of the score is from 0 to 10. This is the Random Forest based model.

3.2 Work in Progress

3.2.1 Trait/Feature importance extractor

This module refers to the extraction or identification of the features which played the main role in the assignment of the grade by a particular model. We have designed feature extractor for the Machine Learning model according to the feature importance. We are using LIME library which helps in the explainable aspect of machine learning and deep learning models.

3.3 Individual Contribution

Name	Contribution
Gurudatta Patil	Machine Learning modules :- SVR, Decision trees , GUI
Kunal Verma	Deep learning Modules:- LSTM model, GUI
M Niteesh	Deep Learning Modules:- Feedforward model, GUI
Nilesh K	Machine Learning Modules:- SVR, Random forests, GUI

Table 1: Contributions

4 Critical Evaluation

We present the obtained results from various models and experimentation in the Table 2. We see that the **LSTM model** produces the best results as it is helpful in processing of **longer sentences and essays**.

Model	Feature set	Parameters	Quadratic Weighted Kappa
LSTM	Word2Vec	Hidden Layers=2	**0.906**
FFNN	Word2Vec	Hidden Layers=2	0.8297
		Hidden Layers=3	0.9002
SVR	Sentences, words, bigram count, nouns, etc.	Kernel=rbf	0.65555
		Kernel=polynomial	0.6026
Random Forest Regressor	Sentences, words, bigram count, nouns, etc.	n_estimators = 50	0.7245
		n_estimators = 100	0.7195
Decision Trees	Sentences, words, bigram counts, nouns, etc.	Max Depth = 50	0.5621
		Max Depth = 100	0.6066

Table 2: Results for different models. Quadratic Weighted Kappa measure is used (value closer to 1 indicates good performance)

We tried using both machine learning and deep learning models for essay grading. We observed following differences when training these models :-

- Machine learning models take as input features extracted from essays whereas deep learning models do not require such features.
- Machine learning models are trained quicker compared to deep learning models, especially LSTM, which requires sequential input.
- Comparatively, deep learning models give better value of qwk.
- Due to feature importance values, it is easier to interpret machine learning model predictions, compared to deep learning models.

4.1 Feature Importance

4.1.1 Global feature importance

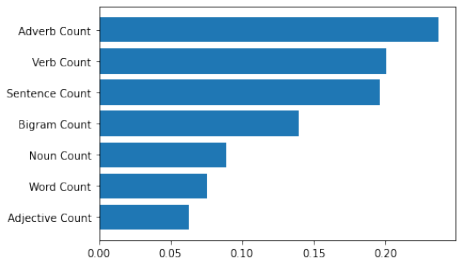

Figure 4: Feature Importance graph for RFR-100

Feature importance graph generated by training Random Forest Regressor with 100 maximum trees is shown in the figure 4. As per the graph, adverb

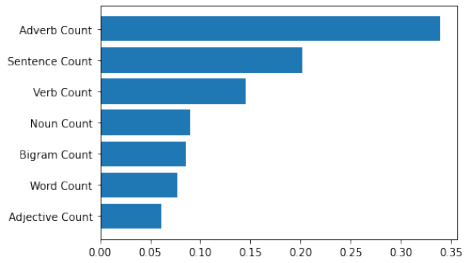

Figure 5: Feature Importance graph for DTR-100

count and verb count are most important features while adjective count has the least effect on predicted essay grade.

Feature importance graph generated by training Decision Tree Regressor with max depth value 100 is shown in the figure 12. As per the graph, adverb count is the most important feature, but with this algorithm second important feature is number of sentences in the essay. Similar to random forest, adjective count is the least important feature.

4.1.2 Local Feature importance

Although it is important to see which features are important overall but at the same time we want to focus on features that were important in the grading of an essay during test time. For example, Number of grammatical mistakes may be an important global feature for deciding grade but for a particular essay during test time, the number of diverse words were more important in assigning the grade. We are using a tool called Lime to extract important features for a particular essay example. The figure shows output generated by lime for an essay. It shows value of each feature for essay under consideration and range of predictions model can make. It shows which features contribute negatively and which features contribute positively to the result.

4.1.3 Fooling AES system

As we can from the above feature importance graphs that model has learned to give high importance to features like adverb count,verb count and sentence count etc.Based on this we can observe that model gives high scores to essays which have more sentences with adverbs and verbs even though written essay is not related to the question.

In figures 5, 6 and 7 above we can see that when given input containing only adverbs and end of sentence marker, the system assigned it full score based on importance of these features.

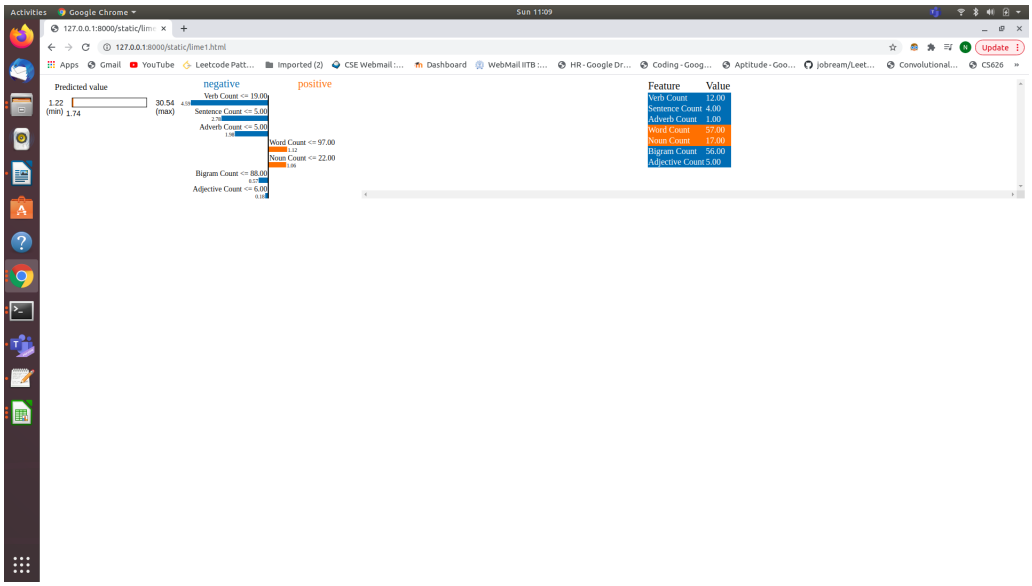

Figure 6: LIME Output

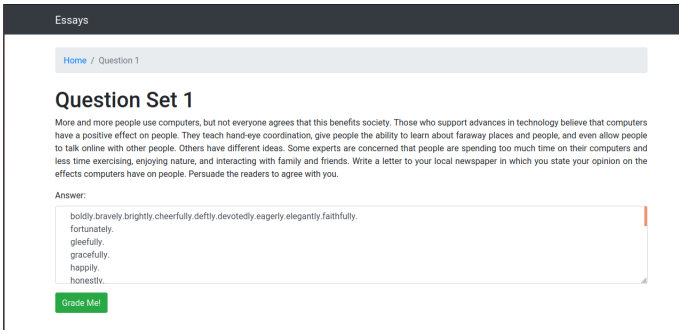

Figure 7: Fooling the system

5 Final Deliverables

5.1 Essay Scoring Model

First, we preprocess training essays and then extract features from them. Based on these features, we train some machine learning and deep learning models. The training model generating best predictions for essay score was selected. Any test essay will be first preprocessed and features extracted from it. The scoring model obtained from training will then be used to estimate the score of the new essay.

Figure 8: Grade assigned

Figure 9: Feature Importance

5.2 Web-based application

We are using **python3** programming language for the implementation of the architecture on the back-end. For developing the user interface we are making a web-based application using the **Django framework** so that we can host this on a server.

5.2.1 User Interface

The user interface contain a set of topics for which the user can submit an essay related to anyone of those topics. After submitting the essay, the application grades the essay according to our trained model.

5.2.2 Results

On the results page, we will display a histogram chart which displays the information regarding the important features which were considered during grading such as fluency in language, grammar, spelling mistakes etc. We have two types of feature importance values which tell us why model assigned a particular grade to the essay - Local feature importance which gives importance of features considered while grading a particular essay instance and another is Global feature importance to determine which features model considers as overall important features.

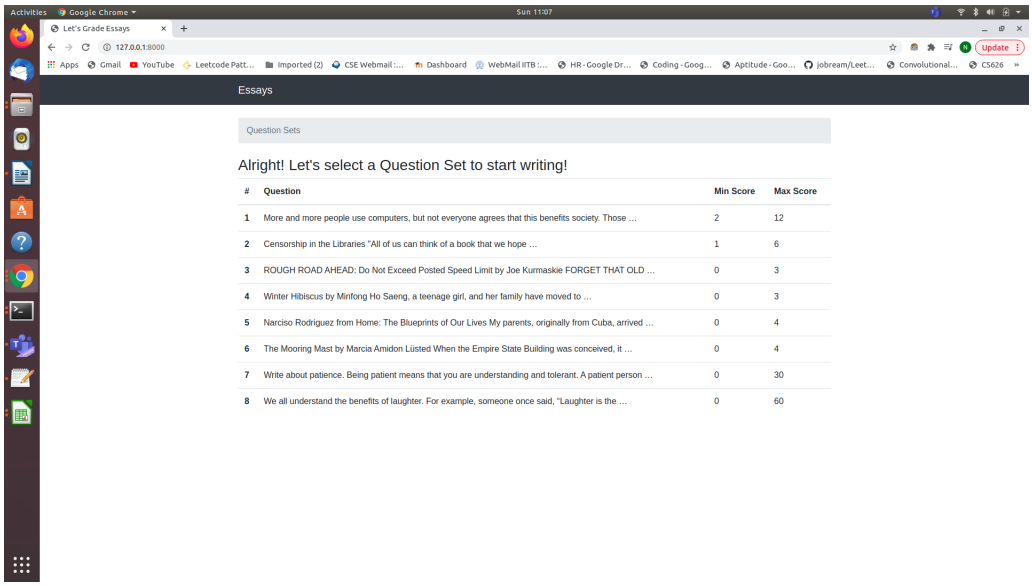

Figure 10: User Interface

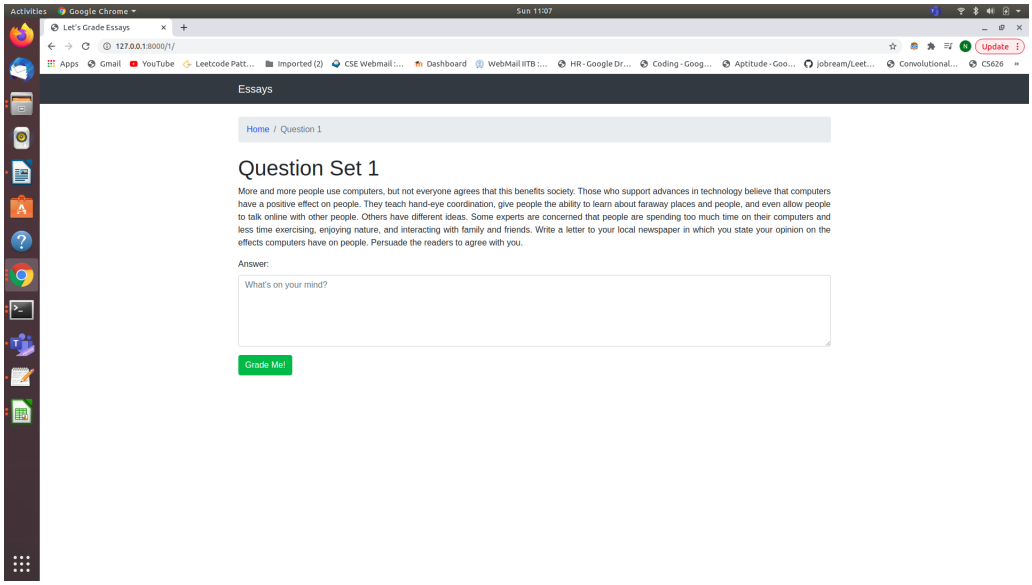

Figure 11: User Interface

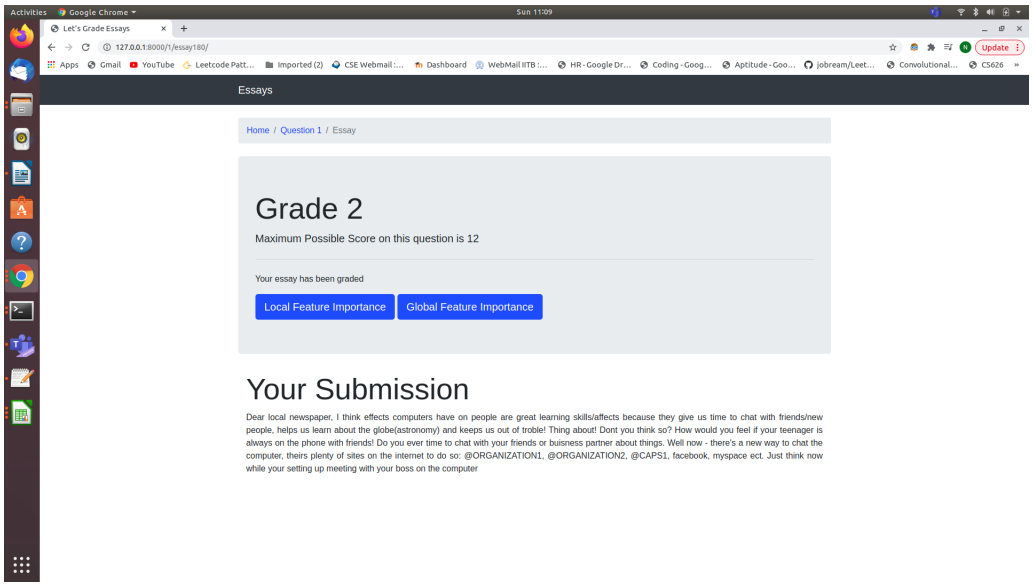

Figure 12: User Interface

5.3 Limitations with Automatic Essay Grading for Indian languages

The approach we are using for essay grading requires a dataset of essays graded by human raters. We searched extensively for such a dataset in any Indian language, but we were unable to find it. We could find only one research paper that talks about essay grading for Indian languages, but it mainly talked about inclusion of some Indian language phrases in English essays and how to deal with them. Hence we do not have enough resources to train a model for grading Indian language essays.

6 Future Projects

6.1 Short Answer Grading system

This project is similar to essay grading project in some aspects. A model answer will be given for a question and based on similarity of the student's answer to the model answer, student will be graded. This can also include a feedback mechanism, where student will get to know what points he missed in the answer. The similarity between model answer and a student's answer can be derived using bag of words approach.

6.2 Adding support for Indian Languages

This task will require manual creation of a dataset. Alternatively we can add a translation module in the proposed architecture and pipeline. We can convert the Indian language to English and then apply the same procedure. However this method generally doesn't perform well as the linguistic and semantics of that particular Indian language may get lost during translation.

6.3 Miscellaneous

1. Although the current work only generates a ranking, it could be updated to provide feedback as well.
2. The essays can be classified based on the content types (e.g. Expository, Descriptive, Narrative, Compare--contrast, Persuasive/argumentative).

7 References

1. `https://github.com/mankadronit/Automated-Essay--Scoring` – Used for making LSTM and FFNN model
2. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html` `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html` `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html` - Referred Scikit learn documentation for hyperparameters for SVR, Random Forest Regressor and Decision Tree Regressor
3. `https://www.kaggle.com/c/asap-aes/overview` - Kaggle Competition Link for dataset
4. `https://www.ijcaonline.org/journal/number11/pxc387391.pdf` - Research about Indian Language
5. `https://thesai.org/Downloads/Volume11No5/Paper_38-A_Trait_based_Deep_Learning.pdf` - Shows use of LSTM Model for giving feedback to essay writer
6. `https://towardsdatascience.com/nlp-preprocessing-with-nltk-3c04ee00edc0` - Used for preprocessing functions from NLTK library
7. GUI reference – used for User interface development