AUTOMATED ESSAY SCORING

J COMPONENT PROJECT

REPORT

FALL 2020-21

Submitted by

Jain Satin Sunil (17BIT0113)

Sankalp Jain (17BIT0136)

Sidharth Raj Miglani (17BIT0145)

Aradhya Mathur (17BIT0146)

in partial fulfilment for the award of the degree of

B. Tech

in

Information Technology

Vellore-632014, Tamil Nadu, India

School of Information Technology and Engineering

October 2020

ABSTRACT

Writing essays is crucial in the assessment of students' language skills. With the periodic evaluation of the students' performance, manually scoring of essays becomes a stressful and time- consuming process. Automated Essay Scoring (AES) is a tool for evaluating and scoring of essays written in response to specific prompts. Automated essay scoring (AES) can be defined as the process of scoring written essays using computer programs. The process of automating the assessment process could be useful for both educators and learners since it encourages the iterative improvements of students' writings.

AES systems start by analyzing written text to extract useful features that are related to intrinsic writing characteristics. Various features are weighted using a regression procedure to develop the AES statistical model. AES systems consider different features according to the different scoring aspects they focus on. Automated grading if proven effective will not only reduce the time for assessment but comparing it with human scores will also make the score realistic. The project aims to develop an automated essay assessment system by use of machine learning techniques and Neural networks by classifying a corpus of textual entities into a small number of discrete categories, corresponding to possible grades.

The dataset we are using is ‘The Hewlett Foundation: Automated Essay Scoring Dataset’ by ASAP. You can find in the below link.

https://www.kaggle.com/c/asap-aes/data

INTRODUCTION

The impacts that computers have on our writings have been in use for 40 years now. Even the most basics of computers like processing of words is of great help to authors in updating their writing material. The research has revealed that computers have the capacity to function as a more effective cognitive tool. Revision and feedback are an important part of the writing material. Students, in general, require input from their teachers for mastering the art of writing.

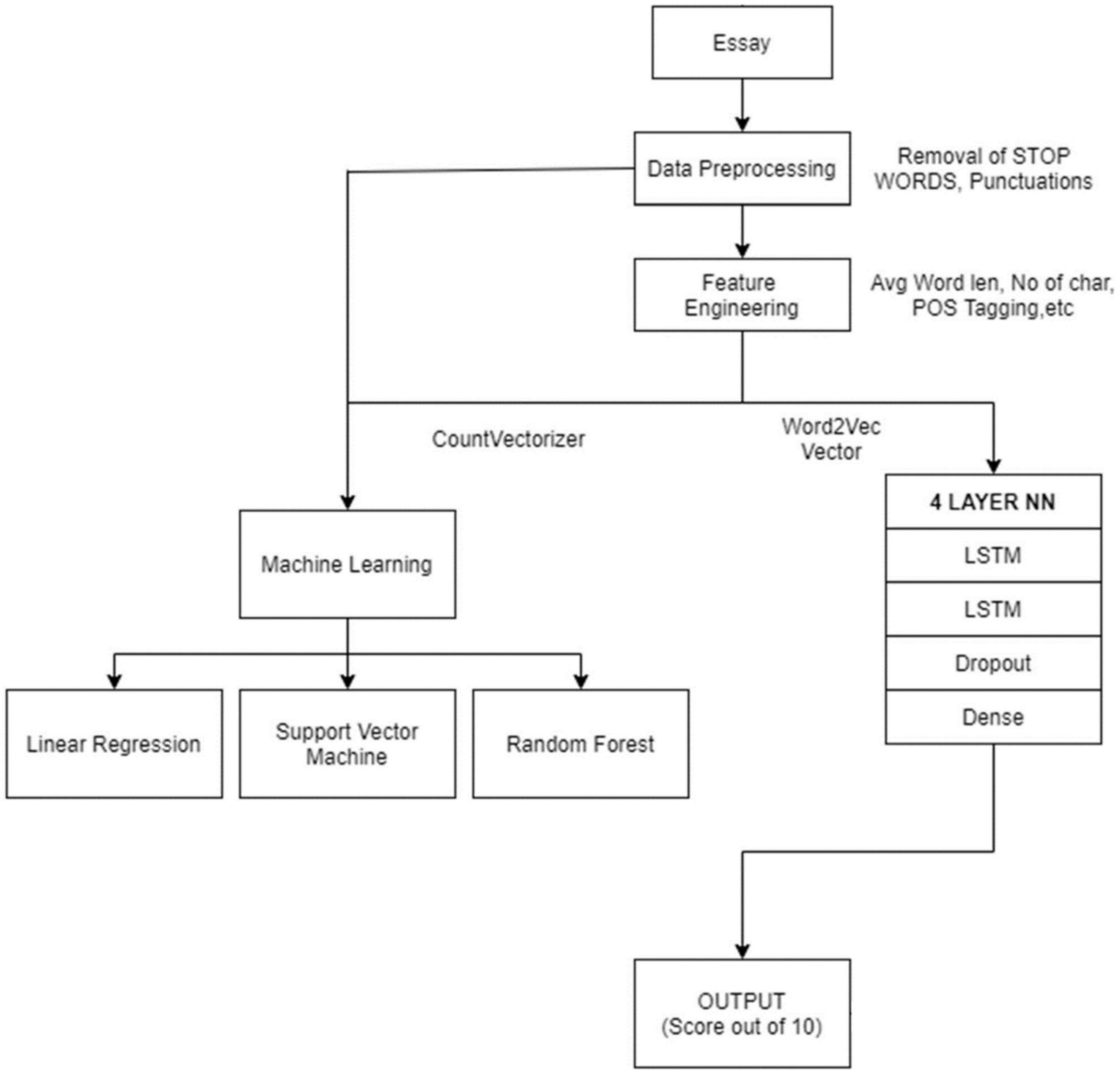
However, responding to student papers can be a burden for teachers. Particularly giving feedback to a large group of students on frequent writing assignments can be pretty hectic from the teacher's point of view. We are creating a system that can be accurate in both providing the feedback as well as grading the performance and can also be time-efficient. In our system, the scores would be much more detail-oriented than the ratings provided by two human raters. Computerized scoring has many weak points as well. The method used by Hamp- Lyons stated the lack of a man to man communication as well as the sense in which the writer rates the essays. Similarly, Page stated that the computers could not assess an essay as human raters do because computer systems are programmed and lack the intensity of human emotions and therefore it will not be able to appreciate the context. Another criticism is the construct objections. That is, a computer can give importance to unimportant factors while rating or providing the score to the user i.e., focusing on traditional aspects rather than orthodox ones.

We analyze natural language, or free-form text used in everyday human-to-human communications, is a vast and complex problem for computer machines irrespective of the medium that has been selected, be it in the form of verbal interaction, written sample, or reading. The murkiness in the given system language and the lack of one definite output to any given communication task make scoring a difficult challenge. In general, in this particular domain implementing machine learning techniques with various features spaces, and a large collection of data having different patterns can give us better outcomes. We also improve our system by implementing an NN approach that outdoes the Machine Learning models.

Our project is divided into 4 modules and each of them are majorly done by one of the team members, they are:

1. Data preprocessing.
2. Application through Machine Learning models.
3. Application through Neural Networks

ARCHITECTURE DIAGRAM



LITERATURE SURVEY

Paper 1: Towards Automated Evaluation of Handwritten Assessments

This paper describes an effective method for automatically evaluating the short descriptive handwritten answers from the digitized images. Authors goal was to evaluate a student's handwritten answer by assigning an evaluation score that is comparable to the human-assigned scores. Existing works in this domain mainly focused on evaluating handwritten essays with handcrafted, non-semantic features. Paper's contribution is two-fold: 1) author models this problem as a self-supervised, feature-based classification problem, which can fine-tune itself for each question without any explicit supervision. 2) The author also introduces the usage of semantic analysis for auto-evaluation in handwritten text space using the combination of Information Retrieval and Extraction (IRE) and, Natural Language Processing (NLP) methods to derive a set of useful features. Author tested the given method on three datasets created from various domains, using the help of students of different age groups. Experiments show that the given method performs comparably to that of human evaluators.

Paper 2: Adapting word2vec to Named Entity Recognition

In this paper, the researchers explored how word vectors built using word2vec can be used to improve the performance of a classifier during Named Entity Recognition. Thereby, they discuss the best integration of word embeddings into the classification problem and consider the effect of the size of the unlabelled dataset on performance, reaching the unexpected result that for this particular task increasing the amount of unlabelled data does not necessarily increase the performance of the classifier.

Paper 3: Automated Essay Grading Using Machine Learning

The essays considered were divided into 8 different sets based on context. They extracted features such as total word count per essay, sentence count, number of long words, part of speech counts etc from the training set essays. We used a linear regression model to learn from these features and generate parameters for testing and validation. They used 5-fold cross-validation to train and test their model rigorously. Further, they used a forward feature selection algorithm to arrive at a combination of features that gives the best score prediction. Quadratic Weighted Kappa, which measures agreement between predicted scores and human scores, was used as an error metric. Their final model was able to achieve a kappa score of 0.73 across all 8 essay sets. They also got a good insight into what kind of features could improve our model, for example, N-Grams and content testing features

Paper 4: Automated Essay Scoring with Ontology based on Text Mining and NLTK tools

One of the common learning activities used in educational levels and disciplines is essay writing. The problems of the essay writing activities are time-consuming, concerns in producing immediate result and/or feedback from teachers to students, and the teachers tend to be subjective in grading the essay activities. The study aims to apply the preliminary approach for automatically generating the domain concept ontology in essays using OntoGen and applied natural language processing algorithms using NLTK (Natural Language Tool Kit) that enhance the teachers essay grading.

Paper 5: An Analysis of Automated Answer Evaluation Systems based on Machine Learning

In this paper, they have summarized the existing mechanism and have analyzed the performance of the system used for automatic grading of the long and descriptive answers as evaluation of the answers continue as one of the most significant factors in the learning and teaching process. As we know, the subjective answers are in either short form or long answers. According to them existing system available for evaluation have shown mediocre result in evaluating and scoring the answers. They identified that there are very inadequate keywords available in short answers and therefore they came to a conclusion that the answers with such inadequate number of keywords needs special care, particularly while calculating the weighting score of the answers. They used corpus based, information extraction, and mapping, methodologies, AI, along with assessment. They found that the example of prior occasions is high for manage based techniques, it audits answers in leaves behind thought mapping systems, or completely with information extraction strategies. They also found authentic procedures, while the features developed with the corpus-based strategies, or Natural Language Processing systems as a significant section of AI structure.

Paper 6: Automatic essay scoring with recurrent neural network

In this paper, they used an ASAP essay dataset, combining feature scoring and a recurrent neural network. The results show that we can compare the result of the quadratic weighted Kappa of each experience to get the best model. GloVe significantly improves the results, and feature extraction can affect the result slightly. In future work, they will apply transfer learning, one-shot learning, and adversarial inputs in our model to get better performance.

Paper 7: A Neural approach to automated essay

Traditional automated essay scoring systems rely on carefully designed features to evaluate and score essays. The performance of such systems is tightly bound to the quality of the underlying features. However, it is laborious to manually design the most informative features for such a system. In this paper, we develop an approach based on recurrent neural networks to learn the relation between an essay and its assigned score, without any feature engineering. We explore several neural network models for the task of automated essay scoring and perform some analysis to get some insights into the models. The results show that our best system, which is based on long short-term memory networks, outperforms a strong baseline by 5.6% in terms of quadratic weighted Kappa, without requiring any feature engineering.

Paper 8: Flexible domain adaptation for automated essay scoring using correlated linear regression

Most of the current automated essay scoring (AES) systems are trained using manually graded essays from a specific prompt. These systems experience a drop in accuracy when used to grade an essay from a different prompt. Obtaining a large number of manually graded essays each time a new prompt is introduced is costly and not viable. We propose domain adaptation as a solution to adapt an AES system from an initial prompt to a new prompt. We also propose a novel domain adaptation technique that uses Bayesian linear ridge regression. We evaluate our domain adaptation technique on the publicly available Automated Student Assessment Prize (ASAP) dataset and show that our proposed technique is a competitive default domain adaptation algorithm for the AES task.

Paper 9: Syntactic, Semantic and Sentiment Analysis

The Joint Effect on Automated Essay Evaluation Manual grading of essays by humans is time-consuming and likely to be susceptible to inconsistencies and inaccuracies. In recent years, an abundance of research has been done to automate essay evaluation processes, yet little has been done to take into consideration the syntax, semantic coherence and sentiments of the essay's text together. The authors proposed a system that incorporates not just the rule-based grammar and surface level coherence check but also includes the semantic similarity of the sentences. They use Graphbased relationships within the essay's content and polarity of opinion expressions. Semantic similarity is determined between each statement of the essay to form these Graph-based spatial relationships and novel features are obtained from it. Their algorithm uses 23 salient features with high predictive power, which is less than the current systems while considering every aspect to cover the dimensions that a human grader focuses on. Fewer features help them get rid of the redundancies of the data so that the predictions are based on more representative features and are robust to noisy data. The resulting agreement between human grader's score and the system's prediction is measured using Quadratic Weighted Kappa (QWK). Their system produced a QWK of 0.793.

Paper 10: Intelligent Auto-grading System

Teachers tend to set the free-text questions for testing the comprehensive ability of students. That leads to the increasing attention to the intelligent auto-grading system for easing the grading load on examiners. In this paper, the authors present a novel
automatic essay scoring system based on Natural Language Processing and Deep Learning technologies. In particular, the proposed system encodes an essay as sequential embeddings and
harnesses a bi-directional LSTM to catch the semantic information. Meanwhile, the system constructs the attention for each essay so that the network can learn to focus on the valid information correctly in an article, which can also provide the reasonable evidence of the predictive result. The dataset for training and testing is the public essay set available in the Automated Student Assessment Prize on Kaggle. The study shows that their system achieves state-of-the-art performance in grade prediction, and more importantly, their intelligent auto-grading system can focus on the critical words and sentences, analyze the logical semantic relationship of the context and predict the interpretable grades.

Paper 11: Automated language essay scoring systems

The authors have reviewed the existing literature using Google Scholar, EBSCO and ERIC to search for the terms “AES”, “Automated Essay Scoring”, “Automated Essay Grading”, or “Automatic Essay” for essays written in English language. Two categories have been identified: handcrafted features and automatically featured AES systems. The systems of the former category are closely bonded to the quality of the designed features. On the other hand, the systems of the latter category are based on the automatic learning of the features and relations between an essay and its score without any handcrafted features. They reviewed the systems of the two categories in terms of system primary focus, technique(s) used in the system, the need for training data, instructional application (feedback system), and the correlation between e-scores and human scores. The paper includes three main sections. First, they present a structured literature review of the available Handcrafted Features AES systems. Second, they present a structured literature review of the available Automatic Featuring AES systems. Finally, they draw a set of discussions and conclusions.

Paper 12: Neural Networks for Automated Essay Grading

The biggest obstacle to choosing constructed-response assessments over traditional multiple-choice assessments is the large cost and effort required for scoring. This project was to use different neural network architectures to build an accurate automated essay grading system to solve this problem. All models so far have been built using predefined features without deep learning features. They built an automated essay grader using neural networks. It aimed at seeing how well neural networks perform compared to machine learning with predefined features.Their aim is to build a model that could take in an essay and automatically output the grade of that essay. Within the scope of this project, they only worked with essays written students in grade 7 to grade 10. The grading scale of the essay can vary. Their models were capable of acknowledging the difference in scale and outputting the corresponding grade

Paper 13: Weighted word2vec based on the distance of words

Word2vec is a novel technique for the study and application of natural language processing(NLP). It trains a word embedding neural network model with a large training corpus. After the model is trained, each word is represented by a vector in the specified vector space. The vectors obtained possess many interesting and useful characteristics that are implicitly embedded with the original words. The idea of word2vec is that there are relations between the words if they appear in the neighborhood. These relations are employed by considering various context windows in training the network model. However, word2vec doesn't consider the influence of distance between the words. It only considers whether or not the words appear in the same context window. We consider that word distances in the context bear certain semantic sense which can be exploited to better train the network model. To formalize the influence of different distances in the context, the fuzzy concept is adopted. Various experiments show that our proposed improvement can result in better language models than Word2Vec.

Paper 14: Coherence-Based Automatic Essay Assessment

In this paper, a discourse-based method that merges semantic and syntactic models for automated essay assessment is proposed. The approach combines shallow linguistic features and discourse patterns in order to predict an essay's score by using decision trees regression techniques. Unlike current approaches, our method directly measures an essay coherence by using corpus-based
semantics and text centering techniques so as to determine discourse patterns underlying high-quality essays when compared with human assessed essays. Experiments using standard datasets showed that the proposed discourse-based approach outperformed traditional shallow features-based methods.

Paper 15: An Analysis of Automated Answer Evaluation Systems based on Machine Learning

Evaluation of the answers remain as one of the most important factors in the learning and teaching process. Automatic evaluation of the answers is very necessary thus, many systems has been developed in this digital era. Usually, the subjective answers are in either short form or long answers. The existing system available for evaluation has shown mediocre result in evaluating and scoring the answers. In such frameworks, the data recovery technique to gauge likeness between understudies answer and references answer is utilized, yet such scoring framework doesn't give the best outcome yet. There are very few keywords available in short answers. The answers with such limited number of keywords needs special care, especially while calculating the weighting score of the answers. In the presented study, we try to summarize the existing mechanism and analyses the performance of the system used for automatic grading of the long and descriptive answers.

Paper 16: An Automated Grader for Chinese Essay Combining Shallow and Deep Semantic Attributes

Writing is a pivotal part of the language exam, which is considered as a useful tool to accurately reflect students' language competence. As Chinese language tests become popular, manual grading becomes a heavy and expensive task for language test organizers. In the past years, there is a large volume of research about the automated English evaluation systems. Nevertheless, since the Chinese text has more complex grammar and structure, much fewer studies have been investigated on automated Chinese evaluation systems. In this paper, we propose an automated Chinese essay evaluation system called AGCE (Automated Grader for Chinese Essay), which combines shallow and deep semantic attributes of essays. We implement and train our AGCE system on a Chinese essay dataset, which is created by ourselves based on more than 1000 student essays from a Chinese primary school. Experimental results indicate that our AGCE system achieves the quadratic weighted Kappa of 0.7590 on a small dataset, which is of higher grading accuracy compared with other four popular neural network methods trained on large-scale datasets. In addition, our AGCE system can provide constructive feedback about Chinese writing, such as misspelling feedback and grammatical feedback about writers' essays, which is helpful to improve their writing capability.

METHODOLOGY

Our Project is divided into 3 modules as follows:

1) Data Preprocessing:

We began by doing some standard preprocessing steps like filling in null values and selecting valid features from the entire dataset after a thorough study.Next we plotted a graph to get a measure of the skewness of our data and applied normalisation techniques to reduce this skewness. The next step involved cleaning the essays to make our training process easier for getting a better accuracy.To achieve this we removed all the unnecessary symbols ,stop words and punctuations from our essays. To increase our accuracy even more we even planned to add some extra features like the number of sentences , number of words, number of characters, average word length etc. Moreover , we even worked on techniques like getting the noun ,verb ,adjective and adverb counts using parts of speech tagging as well as getting the total misspellings in an essay by comparison with a corpus.We applied various machine learning algorithms on this data as explained in thenext section.

2) Machine Learning

For making our data ready to apply algorithms, we require one more step. Machine learning algorithms can not be applied on sentences or words, they can only be used upon numeric data. Our

dataset has a field which has essays that need to be converted into a numeric form first in order to train it.To do this we use something known as a CountVectorizer.Now the CountVectorizer works by tokenizing a collection of text documents and returning an encoded vector with a length of the entire vocabulary along with an integer count for the number of times each word appeared in the document. After this step our data is finally ready for predictive modelling.

Initially we applied machine learning algorithms like linear regression, SVR and Random Forest on the dataset without addition of features that were mentioned in the preprocessing section before. Our results were not really satisfactory as our mean squared error was quite high for all the above algorithms.After this initial evaluation, we added the extra features,applied CountVectorizer again on this modified dataset and applied the same three algorithms. There was a great improvement in the performance of all three algorithms especially Random forest for which the mean squared error reduced drastically.

3) Applying Neural Networks:

Preprocessing steps for neural networks are different from preprocessing steps for machine learning algorithms. Our training data is fed into the Embedding Layer which is Word2Vec.

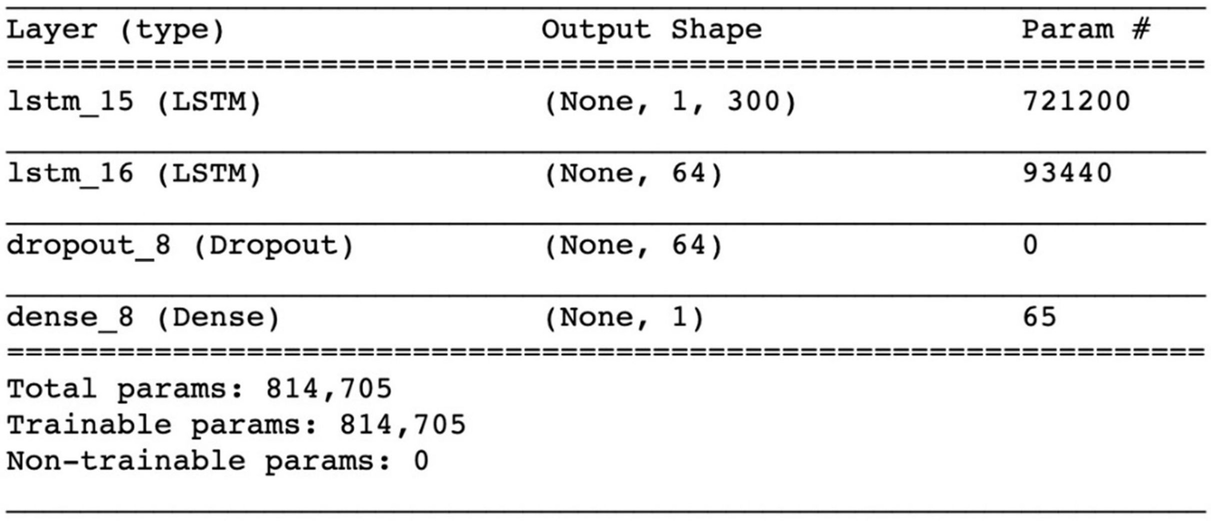
Word2Vec is a shallow, two-layer neural network which is trained to reconstruct linguistic contexts of words. It takes as its input a large corpus of words and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Word2Vec is a particularly computationally-efficient predictive model for learning word embeddings from raw text. Features from Word2Vec are fed into LSTM.

LSTM can learn which data in a sequence is important to keep or throw away. This largely helps in calculating scores from essays. Finally the Dense layer with output 1 predicts the score of each essay.

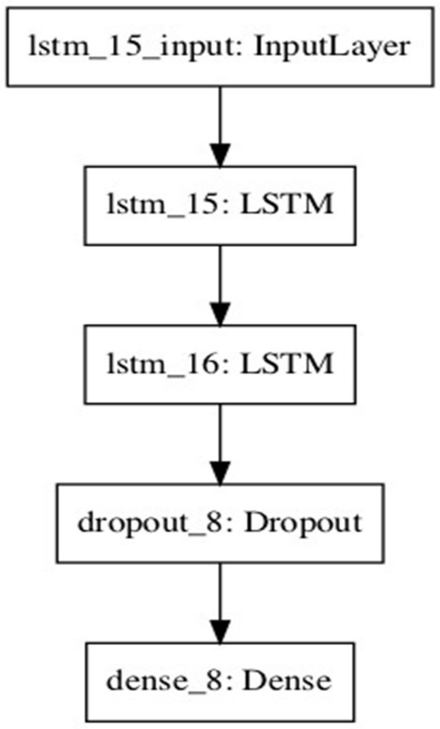
PROPOSED MODEL

We create a list of words from each sentence and from each essay. This list is fed into the Word2Vec model. This model makes sense of the available words by assigning numerical vector values to each word. Features are generated by passing the essays through Word2Vec model. The Word2Vec model acts as an Embedding Layer in a neural network. Features from this model are passed through our LSTM layers. We implement 2 LSTM layers. The first layer accepts all features from the Embedding Layer (Word2Vec) as input and passes 300 features as output to the second LSTM layer. The second layer accepts 300 features as input and 64 features as output. Next we add a Dropout layer with value 0.5. Finally a fully connected Dense Layer with output 1 which represents the score of Essay. The model was compiled with loss function Mean Squared Error and Optimizer Root Mean Square. The model was trained for 150 epochs with batch size of 64.

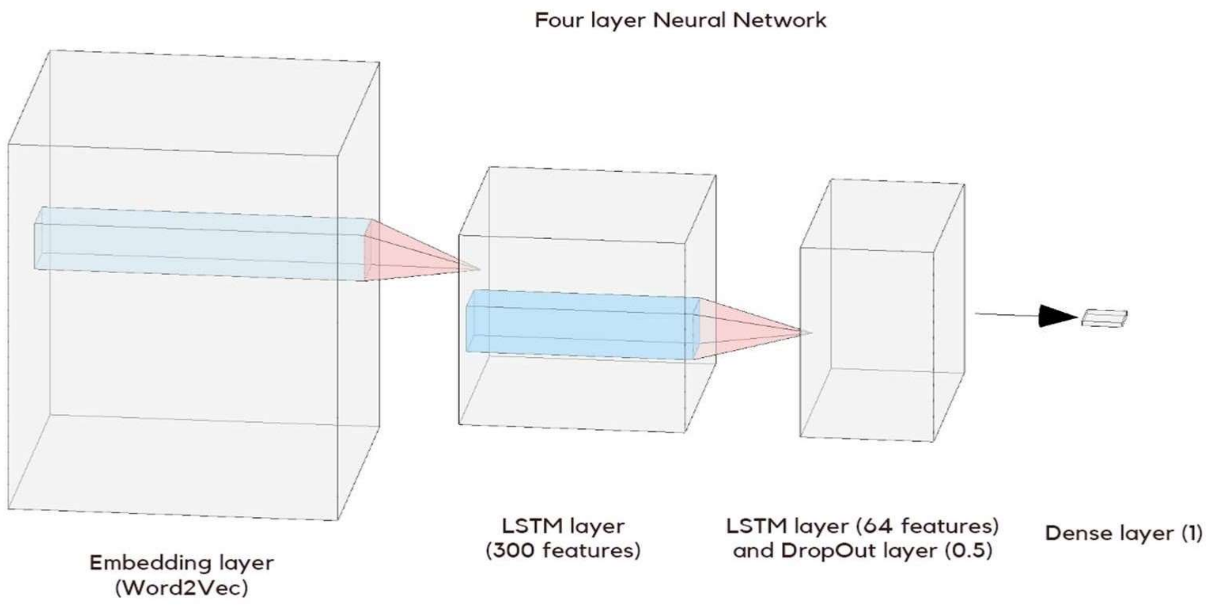
Model Summary



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_15 (LSTM)               (None, 1, 300)            721200
_________________________________________________________________
lstm_16 (LSTM)               (None, 64)                93440
_________________________________________________________________
dropout_8 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 65
=================================================================
Total params: 814,705
Trainable params: 814,705
Non-trainable params: 0
_________________________________________________________________
```



NN Architecture


Four layer Neural Network
Embedding layer (Word2Vec)
LSTM layer (300 features)
LSTM layer (64 features) and DropOut layer (0.5)
Dense layer (1)

RESULTS AND DISCUSSION

SAMPLE CODE

Module – 1

```
def clean_essay(essay): x=[]
    for i in essay.split():
        if
            i.startswith("@
            "): continue
        else:
            x.appen
    d(i) return '
    '.join(x)
df['essay'] = df['essay'].apply(lambda x:clean_essay(x))
stop_words = set(stopwords.words('english'))

def remove_stop_words(essay):
    word_tokens = word_tokenize(essay)
    filtered_sentence = []
    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)
    return ' '.join(filtered_sentence)
df['clean_essay'] = df['essay'].apply(lambda x:remove_stop_words(x))

def remove_puncs(essay):
    essay = re.sub("[^A-Za-z ]","",essay)
    return essay

df['clean_essay'] = df['clean_essay'].apply(lambda x:remove_puncs(x))
```

```
def sent2word(x):
 x=re.sub("[^A-Za-z0-9]"," ",x) words=nltk.word_tokenize(x)
return words

def essay2word(essay):
   essay = essay.strip()
   tokenizer = nltk.data.load('tokenizers/punkt/english.pickle') raw =
   tokenizer.tokenize(essay)
   final_words=[] for i in raw:
      if(len(i)>0): final_words.append(sent2word(i))
   return final_words

def noOfWords(essay): count=0
   for i in essay2word(essay): count=count+len(i)
   return count

def noOfChar(essay): count=0
   for i in essay2word(essay): for j in i:
         count=count+len(j) return count

def avg_word_len(essay):
   return noOfChar(essay)/noOfWords(essay)

def noOfSent(essay):
   return len(essay2word(essay))

def count_pos(essay):
   sentences = essay2word(essay) noun_count=0
   adj_count=0 verb_count=0 adverb_count=0
```

```
for i in sentences:
     pos_sentence =
     nltk.pos_tag(i) for j in
     pos_sentence:
       pos_tag = j[1]
       if(pos_tag[0]=='N'):
         noun_count+=1
       elif(pos_tag[0]=='V'):
         verb_count+=1
       elif(pos_tag[0]=='J'):
         adj_count+=1
       elif(pos_tag[0]=='R'):
         adverb_count+=1
   return noun_count,verb_count,adj_count,adverb_count

 data = open('big.txt').read()
 words = re.findall('[a-z]+', data.lower())

 def check_spell_error(essay):
   essay=essay.lower()
   new_essay = re.sub("[^A-Za-z0-9]"," ",essay)
   new_essay = re.sub("[0-
   9]","",new_essay) count=0
   all_words =
   new_essay.split() for i in
   all_words:
     if i not in words:
       count+=1
   return count
```

Module – 2

```
vectorizer = CountVectorizer(max_features = 10000, ngram_range=(1, 3),
stop_words='english') count_vectors = vectorizer.fit_transform(prep_df['clean_essay'])
feature_names =
vectorizer.get_feature_names() X =
count_vectors.toarray()
X_full = np.concatenate((prep_df.iloc[:, 5:].as_matrix(), X), axis =
1) y_full = prep_df['final_score'].as_matrix()
X_train, X_test, y_train, y_test = train_test_split(X_full, y_full, test_size = 0.3)

#Save Trained Model
# rf = RandomForestRegressor(n_estimators = 1000, random_state
= 42) # rf.fit(X_train, y_train)
# pickle.dump(rf, open('Saved_Models/RF_with_PP', 'wb'))

#Use Saved Model
rf = pickle.load(open('Saved_Models/RF_wth_pp', 'rb'))
y_pred = rf.predict(X_test)
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
```

Module – 3

```
def get_model():
   model = Sequential()
   model.add(LSTM(300,    dropout=0.4,    recurrent_dropout=0.4,
                          input_shape=[1,300], return_sequences=True))
   model.add(LSTM(64,recurrent_dropout=0.4))
   model.add(Dropout(0.5))
   model.add(Dense(1,  activation='relu'))
   model.compile(loss='mean_squared_error', optimizer='rmsprop',
    metrics=['mae']) model.summary()
    return model
#Training Word2Vec model
num_features = 300
min_word_count = 40
num_workers = 4
context = 10 downsampling = 1e-3
model = Word2Vec(train_sents, workers=num_workers, size=num_features,
                min_count =
                min_word_count, window =
                context,
                sample = downsampling)

model.init_sims(replace=True) model.wv.save_word2vec_format('word2vecmodel.bin',
binary=True)

def makeVec(words, model, num_features):
    vec = np.zeros((num_features,),dtype="float32")
    noOfWords = 0.
    index2word_set =
    set(model.wv.index2word) for i in words:
        if i in index2word_set:
            noOfWords += 1
            vec =
    np.add(vec,model[i]) vec =
    np.divide(vec,noOfWords)
    return vec

def getVecs(essays, model,
    num_features): c=0
    essay_vecs = np.zeros((len(essays),num_features),dtype="float32") for i in essays:
        essay_vecs[c] = makeVec(i, model, num_features) c+=1
    return essay_vecs
```

```
clean_train=[] for i in train_e:
    clean_train.append(sent2word(i))
training_vectors = getVecs(clean_train, model, num_features)
clean_test=[]
for i in test_e:
    clean_test.append(sent2word(i))
testing_vectors = getVecs(clean_test, model, num_features)
training_vectors = np.array(training_vectors) testing_vectors = np.array(testing_vectors)

# Reshaping train and test vectors to 3 dimensions. (1 represnts one timestep)
training_vectors=np.reshape(training_vectors,(training_vectors.shape[0],1,training_vectors.
shape[1]))
testing_vectors=np.reshape(testing_vectors,(testing_vectors.shape[0],1,testing_vectors.sha
pe[1]))
lstm_model = get_model()
lstm_model.fit(training_vectors, y_train, batch_size=64, epochs=150)
```

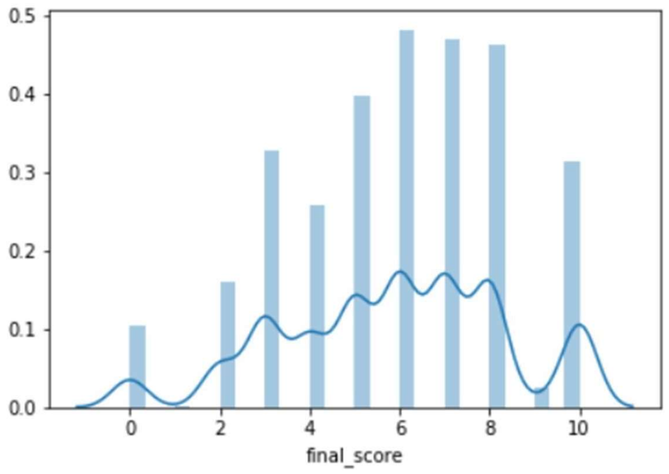
Code to Plot Graph

```
import matplotlib.pyplot as plt
%matplotlib inline

data = {'SVR without PreP':3.68, 'Random Forest without PreP':3.68, 'SVR with
PreP':2.83,  'Random Forest with PreP':0.88, 'LSTM': 3.25}
algos = list(data.keys())
values = list(data.values())
fig = plt.figure(figsize = (10, 5))
plt.bar(algos, values, width = 0.4)
plt.xlabel("Algorithms applied")
plt.ylabel("MSE")
plt.title("Comparative Analysis of Algorithms")
plt.show()
```

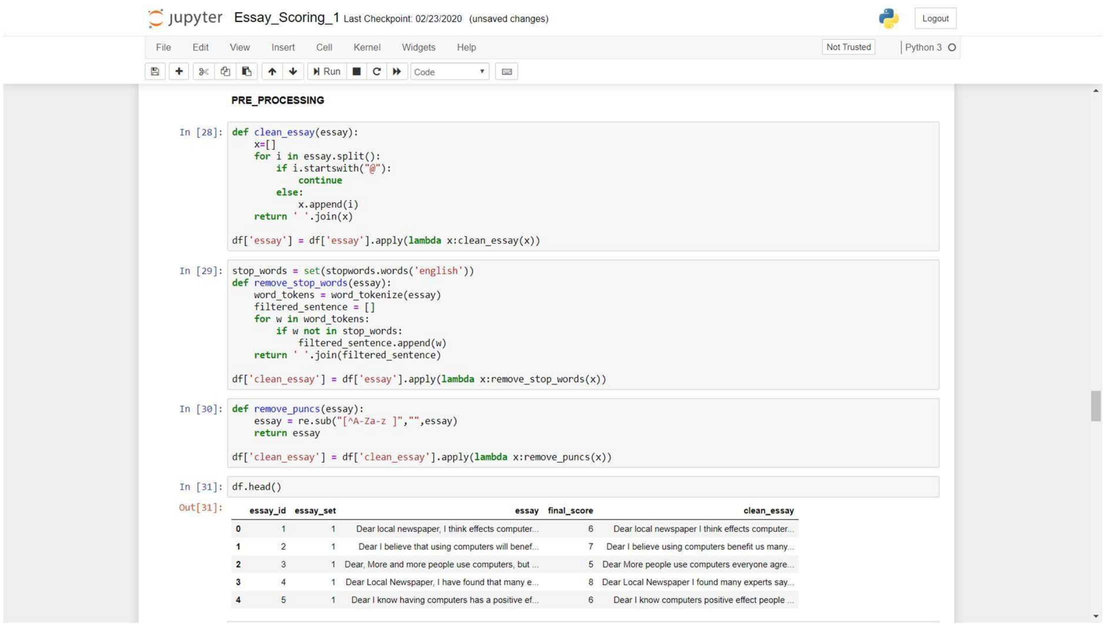
SCREENSHOTS

Module – 1

Removing Skewness from data



Preprocessing on Dataset


jupyter Essay_Scoring_1 Last Checkpoint: 02/23/2020 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

PRE_PROCESSING

```
In [28]: def clean_essay(essay):
             x=[]
             for i in essay.split():
                 if i.startswith("@"):
                     continue
                 else:
                     x.append(i)
             return ' '.join(x)

         df['essay'] = df['essay'].apply(lambda x:clean_essay(x))
```

```
In [29]: stop_words = set(stopwords.words('english'))
         def remove_stop_words(essay):
             word_tokens = word_tokenize(essay)
             filtered_sentence = []
             for w in word_tokens:
                 if w not in stop_words:
                     filtered_sentence.append(w)
             return ' '.join(filtered_sentence)

         df['clean_essay'] = df['essay'].apply(lambda x:remove_stop_words(x))
```

```
In [30]: def remove_puncs(essay):
             essay = re.sub("[^A-Za-z ]","",essay)
             return essay

         df['clean_essay'] = df['clean_essay'].apply(lambda x:remove_puncs(x))
```

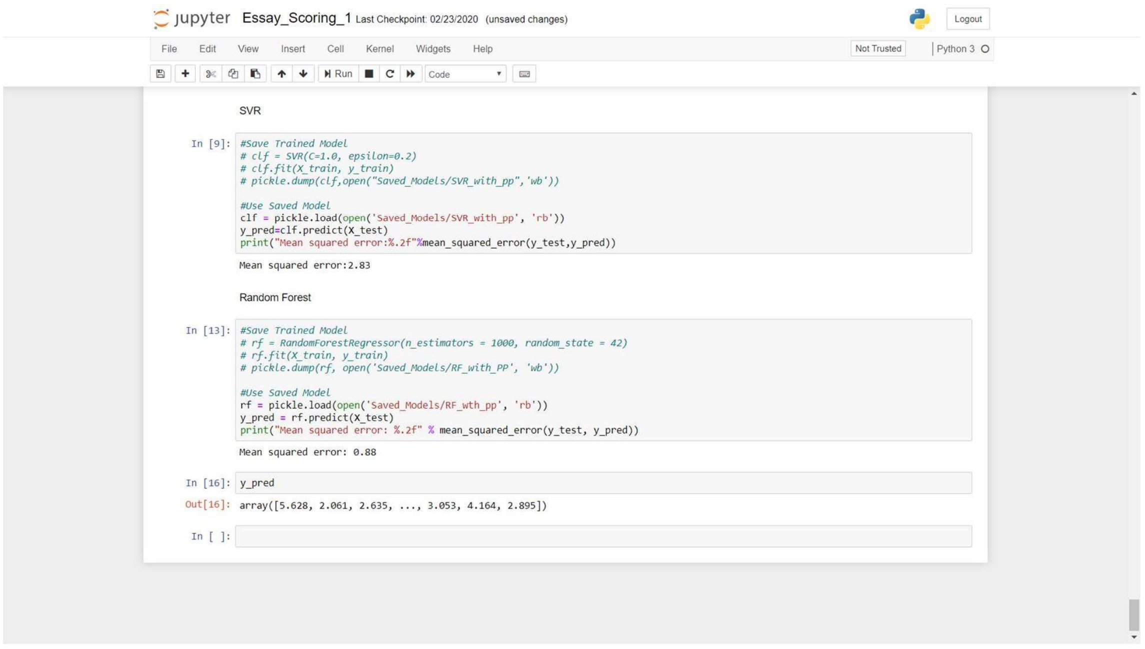
```
In [31]: df.head()
```

Out[31]:

	essay_id	essay_set	essay	final_score	clean_essay
0	1	1	Dear local newspaper, I think effects computer...	6	Dear local newspaper I think effects computer...
1	2	1	Dear I believe that using computers will benef...	7	Dear I believe using computers benefit us many...
2	3	1	Dear, More and more people use computers, but ...	5	Dear More people use computers everyone agre...
3	4	1	Dear Local Newspaper, I have found that many e...	8	Dear Local Newspaper I found many experts say...
4	5	1	Dear I know having computers has a positive ef...	6	Dear I know computers positive effect people ...

Module – 2

ML algorithms



SVR

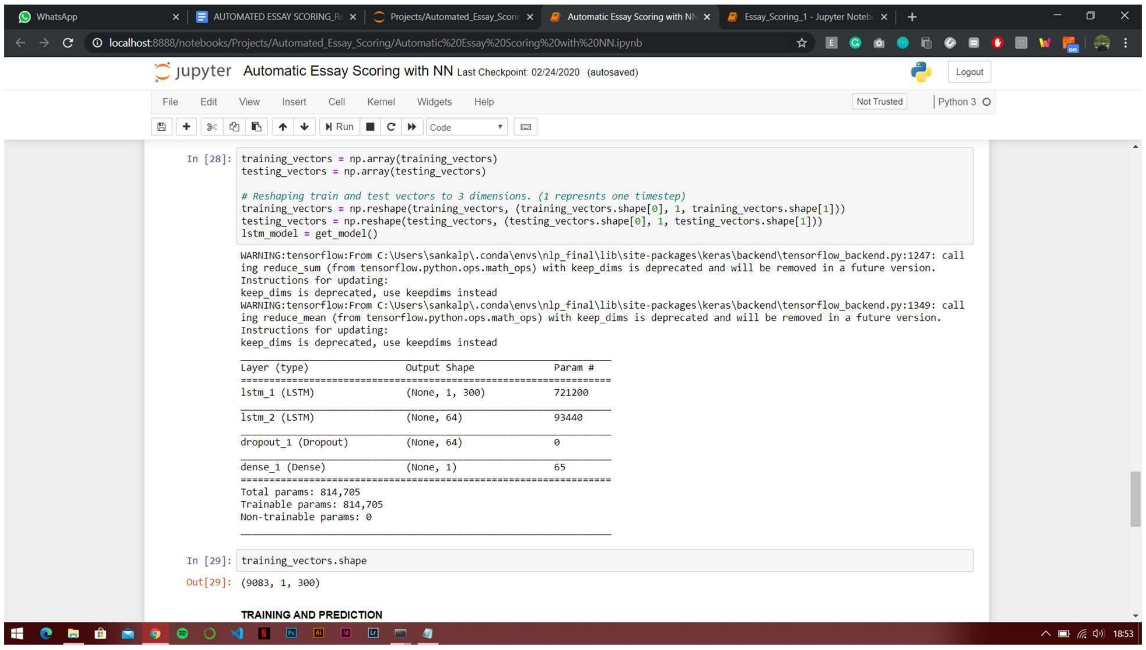
```
In [9]:  #Save Trained Model
         # clf = SVR(C=1.0, epsilon=0.2)
         # clf.fit(X_train, y_train)
         # pickle.dump(clf,open("Saved_Models/SVR_with_pp",'wb'))

         #Use Saved Model
         clf = pickle.load(open('Saved_Models/SVR_with_pp', 'rb'))
         y_pred=clf.predict(X_test)
         print("Mean squared error:%.2f"%mean_squared_error(y_test,y_pred))

         Mean squared error:2.83
```

Random Forest

```
In [13]: #Save Trained Model
         # rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
         # rf.fit(X_train, y_train)
         # pickle.dump(rf, open('Saved_Models/RF_with_PP', 'wb'))

         #Use Saved Model
         rf = pickle.load(open('Saved_Models/RF_wth_pp', 'rb'))
         y_pred = rf.predict(X_test)
         print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))

         Mean squared error: 0.88

In [16]: y_pred

Out[16]: array([5.628, 2.061, 2.635, ..., 3.053, 4.164, 2.895])

In [ ]:
```

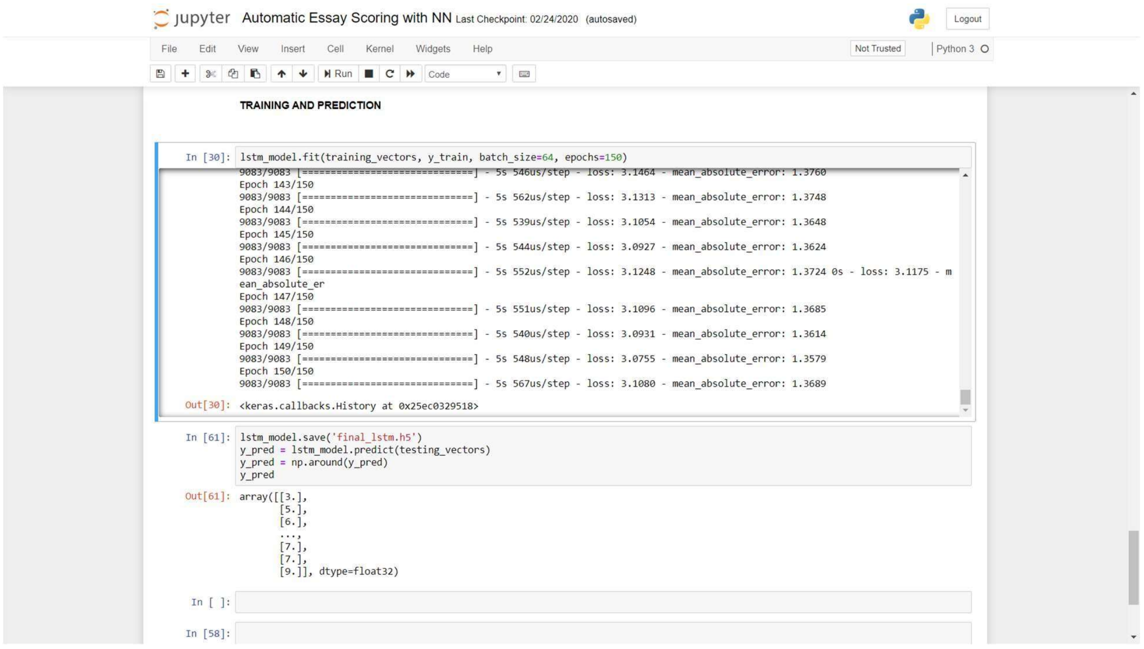
Module – 3

Neural Network summary



```
In [28]: training_vectors = np.array(training_vectors)
         testing_vectors = np.array(testing_vectors)

         # Reshaping train and test vectors to 3 dimensions. (1 represnts one timestep)
         training_vectors = np.reshape(training_vectors, (training_vectors.shape[0], 1, training_vectors.shape[1]))
         testing_vectors = np.reshape(testing_vectors, (testing_vectors.shape[0], 1, testing_vectors.shape[1]))
         lstm_model = get_model()

         WARNING:tensorflow:From C:\Users\sankalp\.conda\envs\nlp_final\lib\site-packages\keras\backend\tensorflow_backend.py:1247: call
         ing reduce_sum (from tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be removed in a future version.
         Instructions for updating:
         keep_dims is deprecated, use keepdims instead
         WARNING:tensorflow:From C:\Users\sankalp\.conda\envs\nlp_final\lib\site-packages\keras\backend\tensorflow_backend.py:1349: call
         ing reduce_mean (from tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be removed in a future version.
         Instructions for updating:
         keep_dims is deprecated, use keepdims instead

         _________________________________________________________________
         Layer (type)                 Output Shape              Param #
         =================================================================
         lstm_1 (LSTM)                (None, 1, 300)            721200
         _________________________________________________________________
         lstm_2 (LSTM)                (None, 64)                93440
         _________________________________________________________________
         dropout_1 (Dropout)          (None, 64)                0
         _________________________________________________________________
         dense_1 (Dense)              (None, 1)                 65
         =================================================================
         Total params: 814,705
         Trainable params: 814,705
         Non-trainable params: 0
         _________________________________________________________________

In [29]: training_vectors.shape

Out[29]: (9083, 1, 300)
```

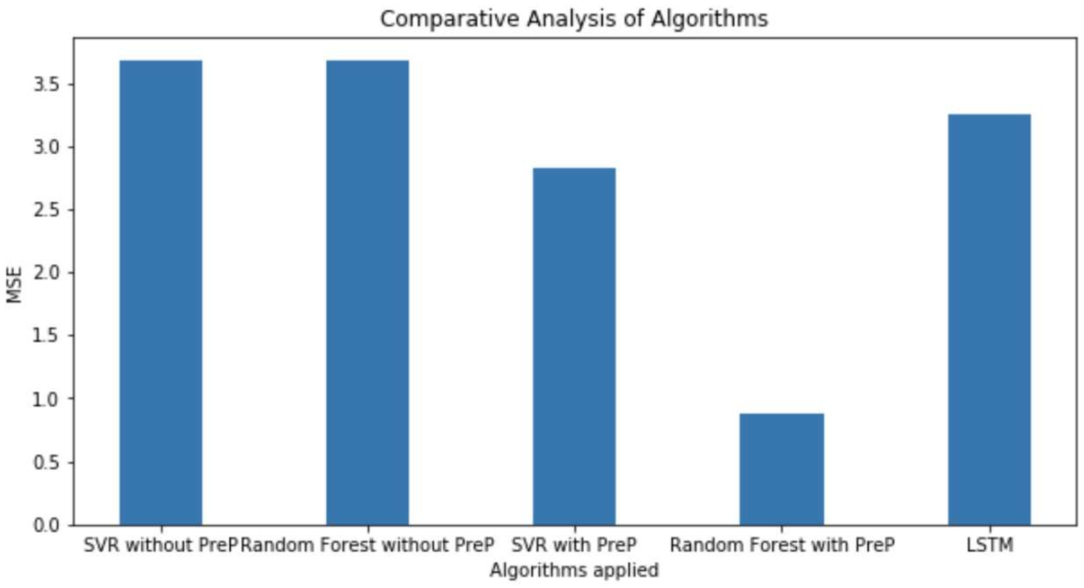
TRAINING AND PREDICTION


jupyter Automatic Essay Scoring with NN Last Checkpoint: 02/24/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

TRAINING AND PREDICTION

```
In [30]: lstm_model.fit(training_vectors, y_train, batch_size=64, epochs=150)
         9083/9083 [==============================] - 5s 546us/step - loss: 3.1464 - mean_absolute_error: 1.3760
         Epoch 143/150
         9083/9083 [==============================] - 5s 562us/step - loss: 3.1313 - mean_absolute_error: 1.3748
         Epoch 144/150
         9083/9083 [==============================] - 5s 539us/step - loss: 3.1054 - mean_absolute_error: 1.3648
         Epoch 145/150
         9083/9083 [==============================] - 5s 544us/step - loss: 3.0927 - mean_absolute_error: 1.3624
         Epoch 146/150
         9083/9083 [==============================] - 5s 552us/step - loss: 3.1248 - mean_absolute_error: 1.3724 0s - loss: 3.1175 - m
         ean_absolute_er
         Epoch 147/150
         9083/9083 [==============================] - 5s 551us/step - loss: 3.1096 - mean_absolute_error: 1.3685
         Epoch 148/150
         9083/9083 [==============================] - 5s 540us/step - loss: 3.0931 - mean_absolute_error: 1.3614
         Epoch 149/150
         9083/9083 [==============================] - 5s 548us/step - loss: 3.0755 - mean_absolute_error: 1.3579
         Epoch 150/150
         9083/9083 [==============================] - 5s 567us/step - loss: 3.1080 - mean_absolute_error: 1.3689

Out[30]: <keras.callbacks.History at 0x25ec0329518>

In [61]: lstm_model.save('final_lstm.h5')
         y_pred = lstm_model.predict(testing_vectors)
         y_pred = np.around(y_pred)
         y_pred

Out[61]: array([[3.],
                [5.],
                [6.],
                ...,
                [7.],
                [7.],
                [9.]], dtype=float32)

In [ ]:

In [58]:
```

Comparative Analysis



CONCLUSION

In this project, we introduced a deep neural network model capable of representing both local and contextual usage of information by essay scoring. This model yields score-specific word embeddings used later by a recurrent neural network in order to form essay representations. We have shown that this kind of architecture is able to surpass similar state-of-the-art systems. We also introduced a novel way of exploring the basis of the network's internal scoring criteria and showed that such models are interpretable and can be further explored to provide useful feedback to the author.

It was satisfying that our neural network model using 300-dimensional LSTM as initialization to the embedding layer was our most successful model. We believe that a more extensive hyperparameter search with our LSTM based models could outperform this result. There are many ideas moving forward. Trying out the models in Ensemble mode is also an extension we wish to try out in the near future.

REFERENCES

[1] Rowtula, V., Oota, S.R. and Jawahar, C.V., 2019, September. Towards Automated Evaluation of Handwritten Assessments. In 2019 International Conference on Document Analysis and Recognition (ICDAR) (pp. 426-433). IEEE.

[2] . Sienčnik, S.K., 2015, May. Adapting word2vec to named entity recognition. In Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015) (pp. 239-243).

[3] Mahana, M., Johns, M. and Apte, A., 2012. Automated essay grading using machine learning. Mach. Learn. Session, Stanford University.

[4] Contreras, J.O., Hilles, S. and Abubakar, Z.B., 2018, July. Automated Essay Scoring with Ontology based on Text Mining and NLTK tools. In 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE) (pp. 1-6). IEEE.

[5] Kapoor., Nagpure., 2020, February. An Analysis of Automated Answer Evaluation Systems based on Machine Learning. In Proceedings of the Fifth International Conference on Inventive Computation Technologies (pp. 439-443). IEEE

[6] Cai, C., 2019, March. Automatic essay scoring with recurrent neural network. In Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications (pp. 1-7).

[7] Taghipour, K. and Ng, H.T., 2016, November. A neural approach to automated essay scoring. In Proceedings of the 2016 conference on empirical methods in natural language processing (pp. 1882-1891).

[8] Phandi, P., Chai, K.M.A. and Ng, H.T., 2015, September. Flexible domain adaptation for automated essay scoring using correlated linear regression. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 431-439).

[9] Janda, H.K., Pawar, A., Du, S. and Mago, V., 2019. Syntactic, Semantic and Sentiment Analysis: The Joint Effect on Automated Essay Evaluation. IEEE Access, 7, pp.108486-108503.

[10] Wang, Z., Liu, J. and Dong, R., 2018, November. Intelligent Auto-grading System. In 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS) (pp.430-435). IEEE.

[11] Hussein, M.A., Hassan, H. and Nassef, M., 2019. Automated language essay scoring systems:A literature review. PeerJ Computer Science, 5, p.e208.

[12] Nguyen, H. and Dery, L., 2018. Neural networks for automated essay grading.

[13] Chang, Chia-Yang, Shie-Jue Lee, and Chih-Chin Lai. "Weighted word2vec based on the distance of words." 2017 International Conference on Machine Learning and Cybernetics (ICMLC). Vol. 2. IEEE, 2017.

[14] Palma, Diego, and John Atkinson. "Coherence-based automatic essay assessment." IEEE Intelligent Systems 33.5 (2018): 26-36.

[15] B. S. J. Kapoor, S. S. Kolhatkar, P. G. Chanore, M. M. Vishwakarma and R. B. Kokate, "An Analysis of Automated Answer Evaluation Systems based on Machine Learning," 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2020, pp. 439-443, doi: 10.1109/ICICT48043.2020.9112429

[16] Y. Yang, L. Xia and Q. Zhao, "An Automated Grader for Chinese Essay Combining Shallow and Deep Semantic Attributes," in IEEE Access, vol. 7, pp. 176306-176316, 2019, doi: 10.1109/ACCESS.2019.2957582.