2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT) | 978-1-7281-8575-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/ISCTT51595.2020.00068

An Automatic System for Essay Questions Scoring Based on LSTM and Word Embedding

Huang Chimingyang
The Houde Academy
Shenzhen, China
2665680584@qq.com

***Abstract*—Essay scoring is a significant task in education, especially online education. Manual essay scoring is a complex job, which limits the development of large-scale online education. In tradition, essays are all scored by human because computer program cannot understand what text means. Recent advance of natural language processing technology in artificial intelligence provides a way to score essays automatically. In our study, we create an automatic essay scoring (AES) system by using Long-Short Term Memory (LSTM) network and word embedding. We show an automated system that can rate essays in electronic text. We combine manually crafted features and Word2Vec embedding in training the model, which makes it more interpretable. We carefully tune the hyperparameter to improve the precision of our model. The LSTM network reaches a quadratic weighted Kappa score (QWK) of 0.95 ± 0.01, which outperforms many other rating systems. The AES system we design can greatly improve the efficiency of online education.**

Keywords-NLP; LSTM; AES; RNN; word embedding

I. INTRODUCTION

Automation in online education has been a notable topic in recent years. Online education, the use of the internet to access and interact with the learning resource, is a possible substitute for traditional learning associated with its lots of merits such as its unlimited time and place, or the adaptive system to improve student's grade [16]. In addition, during COVID-19, schools all over the world start to adopt online education, which shows the importance of online education in crisis time. By using computer and network, the scale of online education can be very large, while the cost of it is very low since it does not need a real classroom to gather students and it eliminates the commute time. Online education is also usually highly automated, students can receive their learning status more effectively than traditional education. Comparing with traditional education, in which most students sit in a same small class, online education allows much more people to participate in the same time. There are many tasks to be automated in education. Automating teachers' work by using artificial intelligence (AI) can leave them more time to do valuable task such as preparing the class better. One important task in automation is to develop an automatic system for essay scoring. Manual work of evaluating essay is very time-consuming. The teachers have to read through each essay and give score individually. If an automatic essay scoring system (AES) could be utilized, the teachers can improve their work efficiency greatly. Many online learning platforms such as MOOCs have started to test the use of AES system [15]. It turns out that a well-performed AES system is very significant for the whole online learning.

Automatic scoring system has been investigated by a variety of researchers. Most of them built the model by using machine learning approaches. Educational Testing Service (ETS) developed its e-rater to review the essay holistically according to its discourse structure, syntactic structure, and analysis of vocabulary usage in a corpus-based approach [1]. Kaveh Taghipour and Hwee Tou Ng designed a system by using recurrent neural networks. They did not use any manually crafted features. Instead, they directly explored the relationship between essays and scores. They found that long short-term memory (LSTM) network is the best system among convolutional neural network (CNN), recurrent neural network (RNN), gated recurrent unit (GRU) and LSTM networks, and the average quadratic weighted Kappa score (QWK) of their LSTM network model is 0.746 [2]. Dimitrios Alikaniotis created an automated scoring system by using LSTM network as well. Innovatively, they adopted a Collobert and Weston embedding (C&W word embedding) [17] technique that find a representative part in essay to evaluate. C&W word embedding introduces a neural network system that learn the meaning of each word according to its local context. Their model provided a Cohen's kappa score of 0.96 [3]. Jiawei Liu devised an AES system based on a Two-Stage Learning Framework (TSLF). This system solves the problem that some essays are irrelevant with the topic. The irrelevant essay may be scored wrongly because most AES systems didn't rate according to the essay topic. The TSLF includes the merits of both feature-engineer and end-to-end AES systems, which ensure the topic of essay is relative with the contents. Their best system reached an QWK of 0.773 in test without irrelevant essays and 0.709 with irrelevant essays [4]. Danielle S. McNamara developed an AES system by using a hierarchical classification approach. The hierarchical approach differs from other regression models, and the features were selected carefully. Their model achieved 55% exact accuracy and 92% adjacent accuracy [5]. M Ali Fauzi invented an essay scoring system by using cosine similarity in bag of words as features. They tested the best model, which was built on bigrams, to reach a correlation value of 0.67 [6]. Prema Nedungadi aimed to differentiate the meaning of each term by evaluating the semantics of individual essay. He adopted an unsupervised word sense disambiguation (WSD) algorithm to measure the relationship between sentences [7]. Li Bin developed an AES system by using K-nearest neighbor (KNN). They turned the essays into vector space model and expressed the essay in terms frequency–inverse document frequency (TF-

978-1-7281-8575-0/20/$31.00 ©2020 IEEE
DOI 10.1109/ISCTT51595.2020.00068

IDF). Their model showed an accuracy score over 0.76 [8]. Hongbo Chen turned the problem of rating an essay into rating the writing quality. They applied support vector machine in representing the words. To compare the writing quality, they used rank-based algorithms to train an essay rating model and found that native speakers were easy to fail in content, organization and structure while nonnative speakers tended to make mistakes in word choice, grammar error and syntactic consistency [9][10]. Bob Effinger explored the task of AES by using supervised learning models. They tried to use simple regressors, ensemble method and probabilistic graphical models to build a best system of 0.81 in QWK [11].

Though there are lots of existing work in AES system, there are still challenges in feature engineering and algorithm design. Feature engineering is the first step in building a machine learning model. Selecting appropriate features is fundamental to the success of the model. The feature engineering work in previous AES studies can be mainly classified into two types. The first category uses manually crafted features, which are features selected from essay laboriously and manually. Word length and sentence length are two examples of manually crafted features. The problem associated with such features is that each of them costs lots of time to find and verify. Some features may be useful, but some may be meaningless. Another type of feature is word vector. Computer program cannot process the text directly. Therefore, we need to turn essays into numbers that can be easily input into the program. This is the concept of word embedding. In a word embedding method, a word is converted into a vector. The input in word embedding is a word, and the output is a vector that represents the word. Word embedding introduces the concept of corpus. A corpus is a set of every different words among all essays. Word Embedding is divided into frequency-based embedding and prediction-based embedding. One common frequency-based embedding is count vector. This method sets the length of vector to the length of corpus. Each word is a vector with a 1 in the corresponding position of the token that exists in the word and 0 in other positions. To express an essay in count vector, we simply average the vector of each word in the essay. There are also many other frequency-based embeddings, such as Co-Occurrence Vector [17]. Co-Occurrence vector is constructed in a context window, which is the sequence of words in essays. Then, a window size is decided to search for the co-occurrence word. We can further calculate a co-occurrence matrix, in which the vector of each words is the number of each other words that appears in a same window. After that, we can combine the vector of each words in an essay to get the matrix of that essay. The Co-Occurrence vector method expresses the word by its context information, and hence contains semantic information. Therefore, it is better than the simple count vector. However, Co-Occurrence embedding usually creates a matrix that has a too high dimension. We have to use dimension reduction methods such as Principal Component Analysis (PCA) or Singular Vector Decomposition (SVD) to reduce the dimension. To represent the words by its context more conveniently, we need to use prediction-based embedding. Word2Vec model in prediction-based embedding is one of the most practical models. One method of building Word2Vec is continues bag of words (CBOW). It predicts the word by its context. Another method is Skip-Gram, which is the reversal of CBOW-predicting word by context. Word2Vec model is usually trained by the complex word embedding model but also has its limit. It needs to be built in a two-layer neural network, so it takes a while to train. The features of complex word embedding model are usually the output of hidden layer in neural network. Therefore, the features are hard to interpret by human. This could lead to confusion when the teachers want to understand which aspects of the essay affect the essay score.

In our system, we aim to improve the accuracy and interpretability. We innovate the system by combining manually crafted features and complex word embedding features in building a LSTM model. There are a few innovations in our project. First, the model combines manually features and complex word embedding by using Word2Vec. We can interpret the manually crated features by analyzing the feature importance and correspond each esssay into the ability (how well he/she does in grammar, word choice and etc.) of the student who writes the essay. Second, we use sequential search algorithms in hyperopt package [21] to randomly search the hyperparameter of LSTM network. We have increased the accuracy of the LSTM model by using hyperopt. Hyperparameter tuning is a time-consuming job, as we need to carefully select hyperparameters and tune them one by one. In our LSTM network, there are 4 hyperparameters to tune. The hyperopt package uses sequential search to intelligently maximum the QWK score. After each turning step, it can find the trend between hyperparameters and QWK score and automatically select the next set of hyperparameters to tune. In the end of hyperparameter tuning, it can return an appropriate set of hyperparameters which correspond to the best model on the validation set. Third, we compare the performance of multinomial regression model and LSTM model. The LSTM model performs 0.26 better than multinomial logistic regression in QWK score. Multinomial logistic regression is an analysis used when we are predicting a categorical outcome based on several continuous and categorical variables. LSTM is a special kind of recurrent neural network that takes the sequence of input into account. LSTM advantages in capturing the long-range sequential dependency between words in the text because the optimized structure of LSTM over regular RNNs. Since LSTM has its trait of processing the whole sequence of data set by its feedback connection, it builds the best model for our AES system.

II. Problem and Data Description

In our project, we collect data from dataset in Kaggle competition "The Hewlett Foundation: Automated Essay Scoring" [13]. In real system design, we assume the input essays that are rated must be electronic text. The goal is to devise an automatic essay scoring system that can be applied to every essay. Figure 1 shows what the data looks like. The dataset includes 8 set of essays in different questions. All essays' lengths are ranged from average 150 to 550 words. The essays are written by Grade 7 to Grade 10 students and rated by human. Each essay entry includes the essay_id, essay_set, essay, domain1_score, domain2_score, rater1_domain2, rater2_domain2, rater1_domain1, rater2_domain1, rater3_domain and target score. Essay id is unique for each essay, there are total 12976 essays in the whole data set, so the essay id ranges from 1 to 12976. The score of essays in different

topic is evaluated in a different scale. All scores are evaluated. The rater_domain score describes the score of each essay in one of the two domains rated by each rater. Only the data in set2 has score in domain2. The target score is the addition of 2 domain score. Notice that the score range in each topic is different, so we normalize all target to eliminate the different score range. The normalized scores which range from 0-1, is the target score divided by maximum target score. An automatic scoring task receive essays as its input. It can then generate the score for each essay. In our system, the input is the text of essay, and the output will be a prediction weight ranges from 0-1.

	essay_id	topic	essay	rater1_domain1	rater2_domain1	rater3_domain1	target_score	rater1_domain2	rater2_domain2	topic2_target
0	1	1	Dear local newspaper, I think effects computers have on people are great learning skills/affects...	4	4	NaN	8	NaN	NaN	NaN
1	2	1	Dear @CAPS1 @CAPS2, I believe that using computers will benefit us in many ways like talking and...	5	4	NaN	9	NaN	NaN	NaN
2	3	1	Dear, @CAPS1 @CAPS2 @CAPS3 More and more people use computers, but not everyone agrees that	4	3	NaN	7	NaN	NaN	NaN

Figure 1. First few rows of the dataset

The whole data set is anonymized according to Named Entity Recognizer (NER). The words containing the true information are replaced by identifier such as PERSON", "ORGANIZATION", "LOCATION", "DATE", "TIME", "MONEY", "PERCENT". All identifier starts with a @ symbol. For example, in sentence, Tom attends in sunny school will be replaced to @PERSON attends in @LOCATION. The NER system protect the privacy of essay writer and exclude many useless words in building the AES model.

To evaluate the performance of AES system, we usually compare the score predicted by AES system with score that rated by human. In our experiment, we use Quadratic weighted Kappa [12] and accuracy score to evaluate the model.

Quadratic weighted Kappa describes the agreement between two different rating system. The value of the matric ranges from 0 to 1. 0 means totally random disagreement and 1 means complete agreement between two systems. A kappa score can also be a negative value when two rating systems are conflicted. A Quadratic weighted Kappa κ is calculated by three matrices, W, O and E. Matrix $W_{i,j}$ is calculated by the score i rated by human and j rated by automatic scoring system. Matrix $O_{i,j}$ is the number of essays that are rated i by human and j by AES system. Matrix E is the outer product between the human rating's histogram vector of ratings and the AES predicted rating's histogram vector of ratings, which normalized to the same sum with O matrix. In experiment, the κ is calculated by using cohen_kappa_score method in sklearn.metrics module of python. The parameters are y1, y2: the prediction weight predicted by AES system and human, weights: a "quadratic" string that specify the QWK calculation.

$$W(i,j) = \frac{(i-j)}{(N-1)^2} \tag{1}$$

$$\kappa = 1 - \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}} \tag{2}$$

The accuracy score is the number of correct predictions divided by the total number of essays. A prediction must exactly match the true value to be a correct prediction, that is to say, a human rated score 5 must be predicted as 5 in AES system to be count a correct prediction. In experiment, accuracy score is calculated by using accuracy_score in sklearn.metrics. The parameters are y_true, y_pred: the prediction weight predicted by AES system and human, normalize: a bool variable true that return the fraction of correctly classified samples.

III. Data Exploration

Data exploration is a fundamental step in AES system design, it helps us know about the data better and find relevant features for determining the essay score. After downloading the essay dataset, we do a series of data exploration on the data. During the data exploration, many python modules are used, including matplotlib, seaborn, pandas and numpy. Matplotlib and seaborn are visualization modules and pandas is a data manipulation module.

Figure 2 shows the number of essays in each topic. As can be seen, the number of essays remain the same except for the 8th topic, in which much less essays are provided.

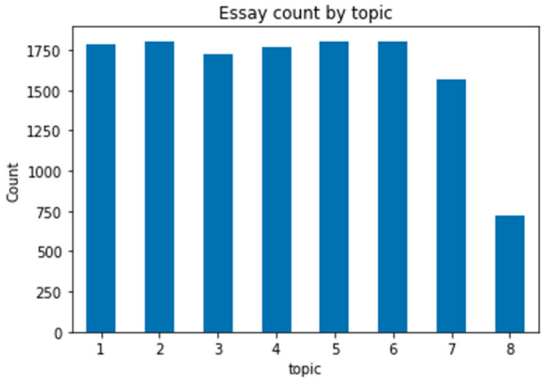

Figure 2. Essay count by topic

Figure 3 shows the number of words in each topic. Each one in the eight boxes shows the word count of essay in that specified topic. In topic 1- 7, most essays distribute from 0 to 500 words. The eighth topic is an exception, in which the most essays are longer and distribute more frequently from 500 to 1000 words.

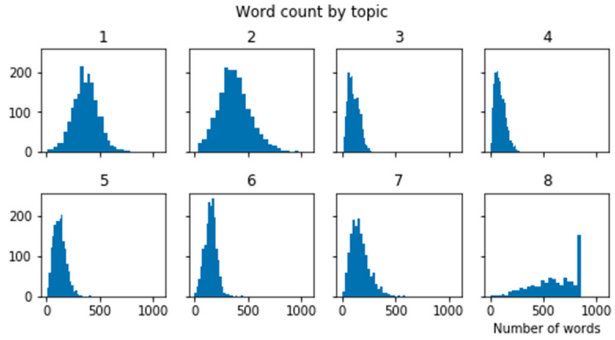

Figure 3. Word count by topic

Figure 4 reveals the relationship between word count and score in each topic. From the graph, we discover that the longer essay receives a higher score in overall trend, and the essays with a higher score also contain longer text. It turns out that the word count can serve a predictive feature in our model.

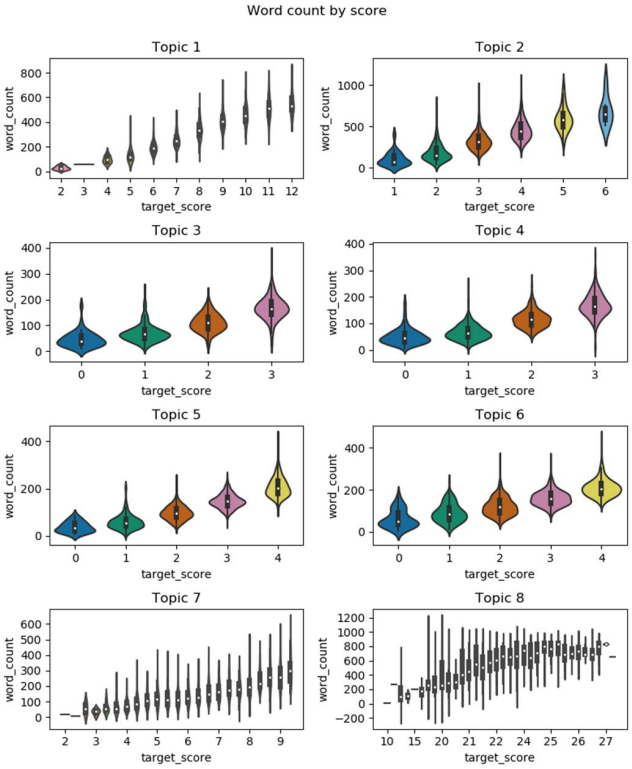

Figure 4. Word count by score

Figure 5 shows the number of different essay score in independent topic. It shows that the most scores are in intermediate range. To analyze the data, we find that the essays that receive an extreme score (too high or too low) is very rare, and the most essays receive a mediocre score. The distribution of essay score follows a pattern of normal distribution, which is very common to see in test grade.

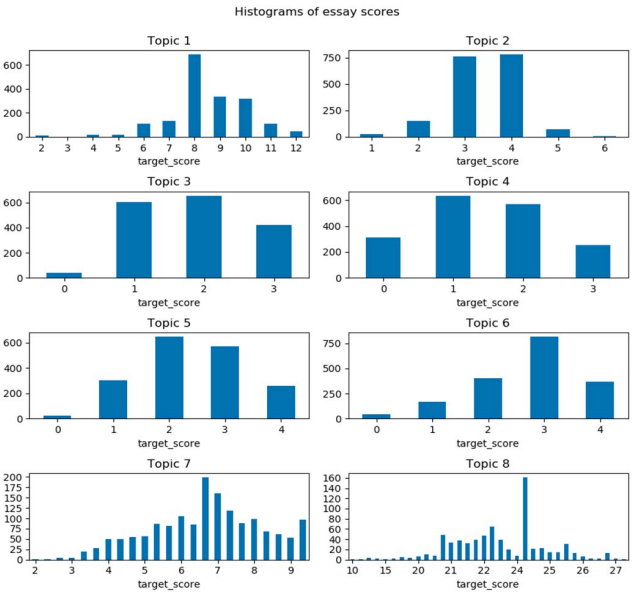

Figure 5. Histograms of essay scores

Figure 6 shows the relationship between essay scores and average sentence length. It is noteworthy that there are some exceptions in topic 1, 7, 8 where the essays with a low score have longer average sentence length. By reading the essays in the low score range we find that those essays usually have other serious problem, such as too many grammar errors or spelling mistakes. From graph we know that most essays have an average sentence length from 10 to 20, and it turns out that the essays with a longer average sentence length receive higher scores. Though the exceptions exist, the average sentence length can still be used as an important feature since the most essay follow the trend. So average sentence length is an appropriate feature for model training.

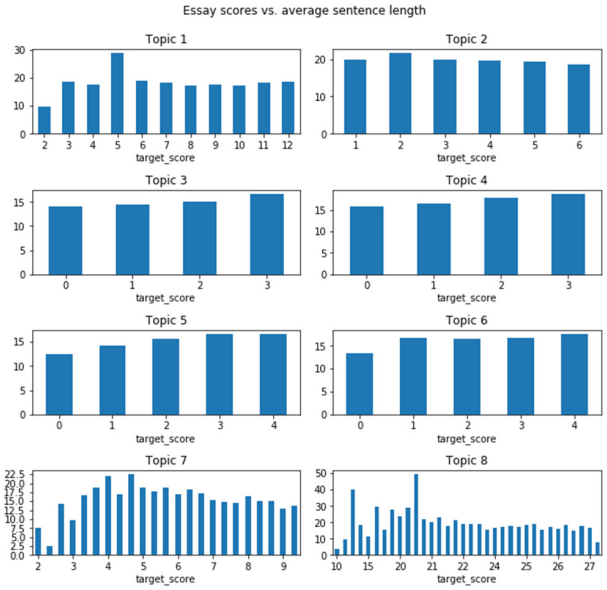

Figure 6. Essay scores vs. Average sentence length

Figure 7 draws the average error ratio in each topic. The result shows that the most grammar error rate distributes over 0.00 to 0.10. It is easy to observe that the essays with a higher score usually have lower error ratio. Therefore, Error ratio is an excellent feature for training the model.

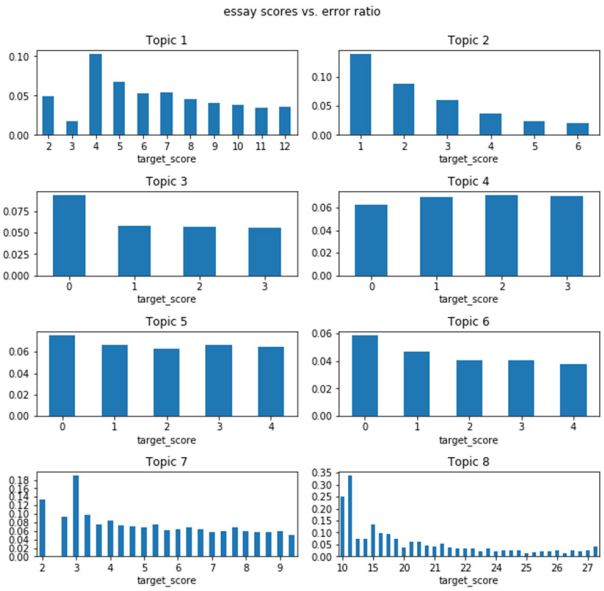

Figure 7. Essay scores vs. Error ratio

Another feature we adopt is average word length. As can be seen on figure 8, the essays with higher score also have a greater word length.

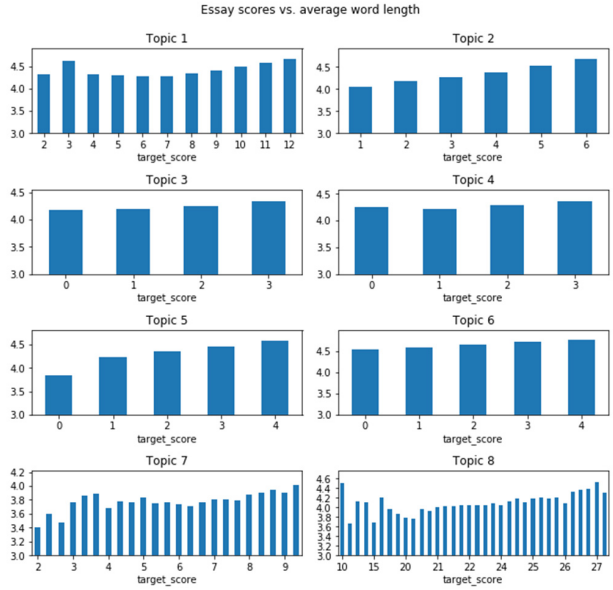

Figure 8. Average word length of each essay score

IV. Data Preprocessing

Data preprocessing is significant for the whole experiment because it helps us to clean irrelevant data and filtrate useful information.

The data is imported by using pandas. We then prepossess the data. The first step is to clean the grammar and spelling error in essays. The grammar and spelling errors cause a word to be identified wrongly, which impedes the training of word vector. We will recover all essays into errorless forms without grammar

and spelling mistake. The removal is done by the a language check tools at https://languagetool.org/. It has a wrapper in python to correct the spelling and grammar mistake in essay. First, the language check tool records match, an intermediate storage of a list of errors. Then, it returns a corrected essay by rectify each mistake in match. By using language check tool, the essays are not free of grammar and spelling error.

The second step is to tokenize the essays. This done by using SpaCy. SpaCy is python natural language processing module. We use SpaCy to tokenize the sentence and classifications the tokens. Tokenization means to separate each word in an essay. The text of essay is a one string, which isn't suitable for learning. Tokenization helps to build a new data column that contains the word list for each essay. Then, we process the essays into lemmatization, sentence, part of speech (POS) and Named Entity Recognition (NER). It takes us 5 mins and 44 secs to finish the whole tokenization process. Lemmatization is the simplest morphology of a word. For example, the words "is, are" are lemmatized to be, which is the simplest form. The sentence is the sentence in each essay that separated by period symbol. The POS is part of speech of all words in sequence in each essay. The NER is the specified entities in a body of text, which is very similar to the anonymized entities existing mentioned before. In feature engineering, we can then use that information from SpaCy to create appropriate features for training. Finally, we store the data in a pickle file, which saves a DataFrame object with new information generated by SpaCy and the error free version text. The prepossessing in spacy takes 5 mins time.

	tokens	pos	sents	ner
0	[Dear, local, newspaper, ,, I, think, effects,...	[ADJ, ADJ, NOUN, PUNCT, PRON, VERB, NOUN, NOUN...	[Dear local newspaper, I think effects compute...	[@ORGANIZATION2, @CAPS1, Facebook, @DATE1, all...
1	[Dear, @CAPS1, @CAPS2, ,, I, believe, that, us...	[ADJ, PROPN, PROPN, PUNCT, PRON, VERB, SCONJ, ...	[Dear @CAPS1 @CAPS2, I believe that using comp...	[Facebook and MySpace, millions, @NUM1, one, M...
2	[Dear, ,, @CAPS1, @CAPS2, @CAPS3, More, and, m...	[ADJ, PUNCT, PROPN, PROPN, PROPN, ADJ, CCONJ, ...	[Dear, @CAPS1 @CAPS2 @CAPS3, More and more peo...	[@CAPS1 @CAPS2 @CAPS3, today, one, @CAPS4, one...
3	[Dear, Local, Newspaper, ,, @CAPS1, I, have, f...	[PROPN, PROPN, PROPN, PUNCT, PROPN, PRON, AUX,...	[Dear Local Newspaper, @CAPS1, I have found th...	[@CAPS1, @PERSON1, @PERSON2, @CAPS3, @CAPS2 @C...
4	[Dear, @LOCATION1, ,, I, know, having, compute...	[ADJ, PROPN, PUNCT, PRON, VERB, VERB, NOUN, AU...	[Dear @LOCATION1, I know having computers has ...	[First, one, Mae, Secondly, @LOCATION2, one, o...

Figure 9. SpaCy's preprocessing result

Figure 9 shows the result of spaCy's prepocessing. It contains tokens, pos, sents and ners in each essay as mentioned above.

V. Feature Engineering

After prepossessing the data, we conduct a list of feature engineering steps. Feature engineering is an important task in building machine learning models. In feature engineering, we extract useful features from the raw data, which is the essays. We want to select features that are directly related to the quality of essays. Therefore, we also draw some graphs to observe the trend of essay score with certain features.

A. Manually Crafted Features

We adopt some manually crafted features in our mode and write function to extract those features and save it in the original tsv data file.

In multinomial logistic regression, the features are total error ratio, average sentence length, average word length and count vector. All manually created features are extracted and then stored in dataaframe's form, which can be easily used in further experiment.

The first feature is grammar error. When we were doing data preprocessing, the language tools return a match objects which contains the information of all grammar errors in an essay. Most errors are morphologic errors, which are spelling mistakes. There are also other errors, such as punctuation or case sensitive error. Therefore, we write a function to calculate the error rate of two types of errors as features. The average word length, average number of words per sentence and total number of words are easy to get by using language check model and spaCy. The count vector is built using CountVectorizer in sklearn.feature_extraction.text. It counts the number of each word's appearance in each article. The initial count vector is a large sparse matrix with more than 30000 columns. The data is too big for training the logistic regression model and caused memory error when running the program. Therefore, we use Principal Component Analysis (PCA) to reduce the dimension of the count vector. [20] Finally, we get a new Dataframe with these new features to be used in building the model. To explore the relationship between features and score, we draw more graphs in data exploration section before. In LSTM model, the count vector is replaced by the Word2Vec model trained vector, which improves the precision.

B. Word2vec vector

Besides the use of manually crafted features such as grammar error and average word length, we also train a Word2Vec model to generate word vector.

Word2Vec is a model to represent words, invented by mikolov in 2013.[18][19] Word2Vec is trained by neural network to learn the associations of text in the whole corpus. In a Word2Vec model, a word is represented by a vector, we can compare the similarity of words or predict words in a sentence by the word vector. Word2Vec model trains a two-layer neural network. There are two builds of Word2Vec, continuous bags of words (CBOW) and skip grams. In CBOW, the model predicts the word in window based on its context. In skip grams, the model predicts the word in context based on its window.

In our experiment, we use the Word2Vec in genism package to train our Word2Vec model. Before we input the essays data, each essay is tokenized into single word without stopwords and punctuation. The tokenized essays are our corpus. Then, we use the trained model to get the average vector of each essay. The

average vector of each essay is the addition of all single word vectors in an essay divided by the number of words. Finally, we divide the word vector into train set, validation set and test set in figure 12 prepared for training the LSTM model.

VI. Model Training and Evaluation

A. Multinomial Logistic Regression

Multinomial (multiclass) logistic regression generalizes logistic regression to solve multiclass problems. It's simpler than LSTM model. In our experiment, we train it by using count vector and manually crafted features: grammar error rate, average sentence length and average word length as input. We implement it by using LogisticRegression in sklearn.linear_model. Then, we train it to predict the score.

B. Hyperparameters Tuning In Multinomial Logistic Regression

We have performed hyperparameter tuning during the experiment in multinomial logistic regression by using GridSearchCV and RandomizedSearchCV in sklearn.model_selection. GridSearchCV searches the given hyperparameters one by one and calculate the best model. RandomizedSearchCV search randomly through the given hyperparameters and also give the best searched models. The advantage of using RandomizedSearchCV is that RandomizedSearchCV can search through a large number of hyperparameter in a short time, while GridSearchCV takes too long time to search each match of hyperparameters. However, RandomizedSearchCV may not yield the best model, but it still performs well since it can search much more effectively than GridSearchCV. In our experiment, we search two hyperparameters for multiclass logistic regression: tol and max_iter. Tol is the tolerance for stopping criteria, which shows to what degree the model is stopped. Max_iter is the maximum iterations times for the model to converge.

C. LSTM

LSTM is a special recurrent neural network that can process the data in sequence [14]. RNN is widely used in predictions on time series data. It can remember the information in previous part in the data and apply it in processing the later part of data. The common RNN has a problem of vanishing gradient, which means that the gradient gets small and only updates with respect to near effect. LSTM network solves the problem of vanishing gradient. It introduces a cell state $c(t)$ that stores long term information and gates that select the process of information in cell state. Forget gates decides what is kept or forgotten from previous cell state. Input gates control which part of new cell content is written to the cell. Output gates controls what parts of cells are output to hidden states. New cell contents the new contents that are going to be written in cell. Cell state erases some old contents from last cell state and write some of the new cell content. Hidden state read some content from the cell. Supposed there is a sequence of input x^t, the sequence of hidden states h^t and cell states c^t can be computed by following formulae:

$$\text{Forget gate: } f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f) \quad (3)$$

$$\text{Input gate: } i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i) \quad (4)$$

$$\text{Output gate: } o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o) \quad (5)$$

$$\text{New cell content: } \tilde{c}^{(t)} = \tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c) \quad (6)$$

$$\text{Cell state: } c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} \quad (7)$$

$$\text{Hidden state: } h^{(t)} = o^{(t)} \circ \tanh c^{(t)} \quad (8)$$

Our model has two LSTM layers. In general, a two layers LSTM model is enough to process the features in our dataset and more layers increase the difficulty of training rapidly. Each LSTM layer is defined by hidden units and dropout rate. Hidden units carry out the calculation of hidden state. Dropout rate prevents the model from overfitting by regularize the input data. Those hyperparameters are tune by using validation set and test set. The model uses RMSProp algorithms [22] to minimize the mean squared error (MSE).

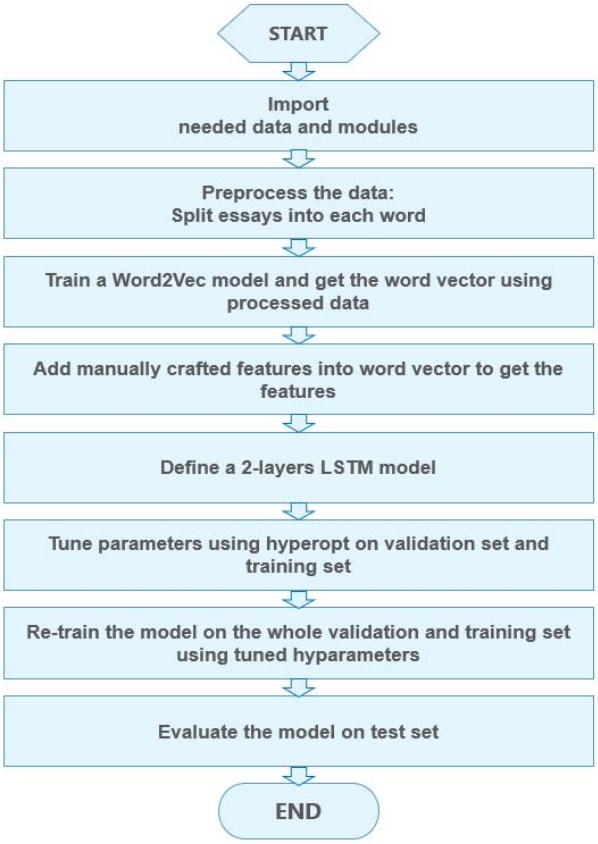

Figure 10. Flow chart for training LSTM network

The whole process of training the LSTM model can showed on figure 10. We build our LSTM model by using keras package in python. First, we add a two layers LSTM network. Then, we combine our manually crafted features and word vector generated by Word2Vec model to get the input of LSTM network. Then we split the data into train set, validation set and

test set. The data is split in fixed proportion in a same random state. 20% of the data is test set, 64% of the data is training set and 16% of the data is validation set. By using hyperopt, we find the best hyperparameters on validation set, which are hidden units of the two layers, dropout_rate, batch_size and epochs. Hyperopt is an optimized hyperparameter algorithm package that performs hyperparameter search by greedy sequential methods. The table shows the range for tuning hypserparameters.

TABLE I. HYPERPARAMETER SEARCH SPACE

hidden_units_1	np.arange(100,1001,10)
hidden_units_2	np.arange(20,501,10)
dropout_rate	rand(10)
batch_size	16,32,64,128,256
epochs	np.arange(1,11,1)

Finally, we train the final LSTM model and evaluate the final model on the test set.

VII. FEATURE IMPORTANCE

In our LSTM model, we want to explore whether the manually crafted features are useful in evaluating the score. Therefore, we do another experiment of training the LSTM network without manually crafted feature and with only Word2Vec vector. By comparing the performance of these two models, we can understand whether the manually crafted features are helpful. The result turns out that the QWK of LSTM withs only word vector is 0.945 and accuracy score is 0.321, which is a bit lower than the QWK of LSTM with both word vector and manually crafted features, which are 0.948 and 0.340. The improvement of QWK score in model with manually crafted features shows that those manually crafted features are useful for rating Furthermore, we explore the importance of each manually crafted features. We use three manually crafted features, which are average word length, average sentence length and grammar error ratio. To explore how each feature affects the score, we add Gaussian noise to each feature separately, and compare the mean of the absolute value of the difference of each rated score before and after adding Gaussian noise.

Suppose the original input test data is $x = (x_1, x_2, ..., x_m)$, where $x_i (1 \le i \le m)$ is the i-th feature of this data point. Given this input data, the essay scoring model will output a predicted score $f(x)$, where f is a function that describes the model. To study the importance of a certain feature x_i, we perturb the original feature by adding a Gaussian noise $\delta \sim N(0, \sigma)$ where 0 is the mean and σ is the standard deviation of the Gaussian distribution. Now given the perturbed input $\tilde{x} = (x_1, x_2, ..., x_i + \delta, ..., x_m)$, the model will generate a new predicted score $f(\tilde{x})$. Notice that we only perturb one feature at a time, while keeping other features unchanged. This allows us to measure the importance of a single feature. Then, we use the following equation to measure the feature importance of the i-th feature in x

$$FI_i(x) = \frac{\left| f(\tilde{x}) - f(x) \right|}{\sigma} \tag{9}$$

If we have multiple test data point, $x^{(j)}$, $1 \le j \le n$, where j is the index of the j-th test sample, and n is total number of test samples. For each test data point, we can measure the feature importance of the i-th feature. To evaluate the overall importance of this feature, we compute the arithmetic mean of $FI_i(x^{(j)})$ across all the test samples $1 \le j \le n$ as follows

$$FI_i(x^{(j)}) = \frac{\sum_{j=0}^{n} \left| f(\tilde{x}^{(j)}) - f(x^{(j)}) \right|}{n \cdot \sigma} \tag{10}$$

We add Gaussian noise with mean of 0 and standard deviation of 0.1. In each test, only one features are added Gaussian noise. We repeat the test for ten times to account the randomness of the Gaussian noise. Finally, we determine the feature importance based on the feature importance score of each feature.

VIII. RESULTS AND DISCUSSIONS

In this section, we show the results of the experiments and discuss these results. The results include the experiments with multinomial logistic regression and the experiments with LSTM model. We also demonstrate the results of hyperparameter tuning and compare experiments of LSTM models with and without the manually crafted features. Moreover, we measure the feature importance of each manually crafted features by adding perturbations to each feature.

A. Multinomial Logistic Regression

We repeat 5 times to train the multinomial logistic regression model, the train and test data in each time are split by using a random state from 0 to 4. In each model, 80 percent of data is train set and 20 percent of data is test set. The input of logistic regression model includes manually crafted features and count vector. In the multiclass logistic regression model trained by manually crafted features and count vector. We find the best hyperparameters in RandomizedSearchedCV test. The best parameters are tuned to be a tolerance of 1e-06 and a max iteration number of 60. The model trained with the best hyperparameter reaches a QWK of 0.65 ± 0.03 in five repetition tests.

B. LSTM

The LSTM model with word2vec vector and manually crafted features reach a QWK of 0.95 ± 0.01 in test set. As can be seen in table 2, LSTM perform much better than multiclass logistic regression because its advantages in processing sequential data.

TABLE II. QWK RESULTS OF THE MULTICLASS LOGISTIC REGRESSION AND LSTM MODELS

	Multiclass logistic regression	LSTM
QWK run 1	0.67	0.95
QWK run 2	0.60	0.95
QWK run 3	0.69	0.95
QWK run 4	0.62	0.95
QWK run 5	0.66	0.93
QWK mean and standard deviation of 5 runs	0.65± 0.03	0.95±0.01

In figure 11, we graph the loss change in training the LSTM model. As can be seen in the graph, the loss decreases in each epoch, and finally reaches a small loss in the end. It shows that the iteration of our model successfully decreases the loss and the training is effective.

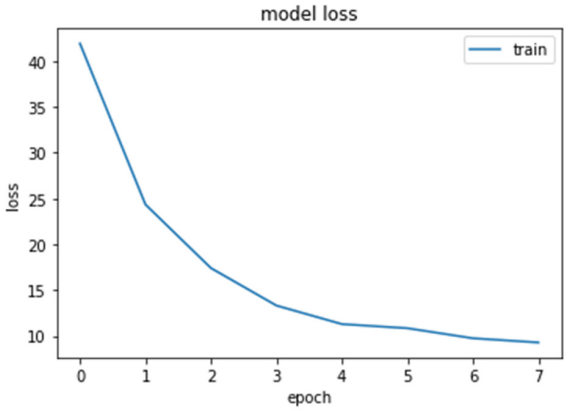

Figure 11. Loss Change in LSTM model

C. Hyperparameter Tuning

In both multiclass logistic regression and LSTM models, we tune the hyperparameter tuning before training the model to improve the precision.

TABLE III. HYPERPARAMETERS TUNING RESULTS FOR LSTM MODEL

Batch size	Dropout rate	epochs	Hidden units1	Hidden units2
4	6	8	72	48

In multiclass logistic regression models, best tuned hyperparameters are shown in table 4.

TABLE IV. HYPERPARAMETERS TUNING RESULTS FOR LOGISTIC REGRESSION USING RANDOMIZEDSEARCHCV

Tolerance	Max Iterations
1e-05	50

To show that hyperparameter tuning is efficient in improving the precision, we also do a set of experiment on training LSTM model without tuning hyperparameters. We find out that the QWK of LSTM model without tuning hyperparameters is 0.86, which is much lower than the model with tuning hyperparameters, which is 0.95 ± 0.01. This shows that hyperparameter tuning is useful to improve precision.

The experiment runs on computer with i7-8750H CPU and WDC PC SN720 SDAPNTW-512G-1101 hard disk.

D. Feature Importance

The result of feature importance in LSTM analysis is shown in figure 12, the result shows that importance of average sentence length, error ratio and average word length are 0.40, 1.29, and 0.22. As we can see, the error ratio affects the score most profoundly, which has an importance of 1.29. Average sentence length and average word length also affect the score with slightly lower feature importance scores 0.40 and 0.22 respectively. By analyzing feature importance, we can interpret the model better and understand what features represent and how important they are in the model.

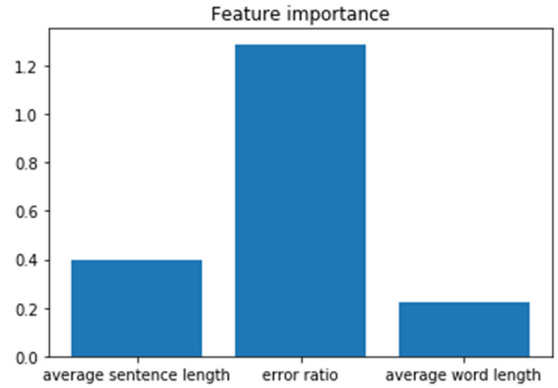

Figure 12. Feature importance

In addition, the feature supports what ETS's essay points out, organization sentence structure and content.

E. Task Significance

Our LSTM model perform well and reaches a QWK score of 0.94 with human rated score. It shows that our models are highly similar to the human rated system. The model can be used to evaluate essay score in practical. It can help online education solves the problems of rating students' essays in large scale. With a few steps of data preprocessing, the model can evaluate the score in high accurate.

F. Feature Work

In future, we want to design a system that can display students' different ability of writing essay in each area. By analyzing the students' data in each feature and the feature importance, we can understand whether the students' vocabulary, grammar or sentence pattern affects their score the greatest and gives them custom suggestions on how to improve their essay score. The system would be more helpful because students can not only get the score but also learn their weakness in writing.

IX. CONCLUSION

In our paper, we train a AES model based on LSTM and Word2Vec vector. We explore many manually created features

and add them to word vector to improve the precision of the model. Before training the LSTM model, we use hyperopt to tune hyperparameter carefully on validation set. The LSTM model reaches a quadratic weighted Kappa of 0.94 and accuracy score of 0.32 in test set that independent from training set. We also analyze the feature importance, or how the features in essay affect the score performance in the model, which increase the interpretability of the model.

REFERENCES

[1] Williamson, D. M., Bennett, R. E., Lazer, S., Bernstein, J., Foltz, P. W., Landauer, T. K., ... & Sweeney, K. (2010). Automated scoring for the assessment of common core standards. White Paper.Taghipour, K., & Ng, H. T. (2016, November).

[2] A neural approach to automated essay scoring. In Proceedings of the 2016 conference on empirical methods in natural language processing (pp. 1882-1891).

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

[4] Liu, J., Xu, Y., & Zhu, Y. (2019). Automated essay scoring based on two-stage learning. arXiv preprint arXiv:1901.07744.

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

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

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

[8] Bin, L. , Jun, L. , Yao, J. M. , & Zhu, Q. M. . (2008). Automated Essay Scoring Using the KNN Algorithm. International Conference on Computer Science & Software Engineering. IEEE.

[9] Chen, H. , Xu, J. , & He, B. . (2014). Automated essay scoring by capturing relative writing quality. Computer Journal, 57(9), 1318-1330.

[10] Chen, H., He, B., Luo, T., & Li, B. (2012). A Ranked-Based Learning Approach to Automated Essay Scoring. In 2012 Second International Conference on Cloud and Green Computing (pp. 448–455).

[11] Effinger, B., Merrell, D., Ordman, J., Truskowski, J., & Vinitsky, S. (2018). Grading Essays with Supervised Learning.

[12] Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. Psychological Bulletin, 70(4), 213–220.

[13] https://www.kaggle.com/c/asap-aes/

[14] Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

[15] Reilly, E. D., Stafford, R. E., Williams, K. M., & Corliss, S. B. (2014). Evaluating the validity and applicability of automated essay scoring in two massive open online courses. International Review of Research in Open and Distributed Learning, 15(5), 83-98.

[16] Ally, M. (2004). Foundations of educational theory for online learning. Theory and practice of online learning, 2, 15-44.

[17] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of machine learning research, 12(ARTICLE), 2493-2537.

[18] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[19] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[20] Jolliffe, I. (2014). Principal Component Analysis. Wiley StatsRef: Statistics Reference Online

[21] Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., & Cox, D. D. (2015). Hyperopt: a Python library for model selection and hyperparameter optimization. Computational Science & Discovery, 8(1), 14008.

[22] Dauphin, Y. N., Vries, H. de, Chung, J., & Bengio, Y. (2015). RMSProp and equilibrated adaptive learning rates for non-convex optimization. ArXiv Preprint ArXiv:1502.04390.

[23] s design with FPGAs and CPLDs[M]. Newnes, 2011