Automated Essay Grading Using LSTM

*Aditi Machhar, Mahitha Umapathy, Reethika Madupu*

**Department** - **Information Technology**

**ABSTRACT**

The recent developments of automation is an inevitable reality. It will have a significant influence in subsequent years. Advanced automation will overpower humans. In the educational field it serves as a great purpose. As we know, writing is considered to be one of the finest ways to express ideas and it can be categorized as an expository, persuasive, narrative and descriptive. Along with the writing skills, proper evaluation of the essay is equally essential. Automated Essay Grading is the tool which is used for evaluation of essays using methods like NLP which are concerned with the interaction of computers with human languages. The primary purpose of this system is to increase efficiency and in addition add to the consistency and accuracy which may be secondary benefits. The system is increasingly becoming popular as human scoring is not only becoming expensive but also cumbersome as the number of examinees grows. Quick feedback is another feature for the increase in the use of the system. Although a few challenges have been faced by the system but it is highly reliable and has a great impact in the field of education.

**1. INTRODUCTION**

**1.1 OVERVIEW**

Essays are a tool for testing the students’ fluency, vocabulary and grammatical correctness in a language. They are also useful to test one’s creativity, originality and articulateness. The highly subjective and diverse nature of an individual in writing an essay makes it difficult to grade the essay uniformly across many human graders. In addition to this there are various other biasing factors in grading an essay. There has been research on automatic essay grading since the 1960s. The first systems based their grading on the surface information from essays. These systems were successful though they failed to capture aspects like grammatical correctness and language fluency. Much research has been conducted in the field most notably by Educational Testing Service (ETS). Clubbing this together with the resurgence in new technologies such as neural networks, deep neural networks, there is a whole new world of possibilities due to their capacity of modeling complex patterns in data. These methods do not depend on feature engineering so they are really useful for solving problems in an end-to-end fashion. With this intuition this project aims to the relevant knowledge in the field of education and try to create an essay grader which can make quality education more accessible. The work also explores methods of improving the quality and usability of the system.

**1.2 PURPOSE**

The human graders unknowingly tend to grade an essay biasing towards the individual subject matter presented. Another major drawback is the time required to grade essays can be significantly high. The present technologies present an excellent opportunity to automate tedious tasks such as essay grading. Availability of powerful Deep Learning libraries is a major push towards the reliance and devising the system. This project will help in maintaining an unbiased and fair approach towards evaluating the written essay for competitive exams, tests, etc. which is in turn beneficial in multiple ways.

**2. LITERATURE SURVEY**

**2.1 PROBLEM STATEMENT**

Our aim is to build a model that can take in an essay and automatically outputs the grade of that essay. Within the scope of this project, we only work with essays written students in grade 7 to grade 10. The grading scale of the essay can vary. Our models are capable of acknowledging the difference in scale and outputting the corresponding grade. There are two kinds of models we are building: regression models and classification models. For regression models, our output will be a continuous value between 0 and 12. For classification models, our output will be a discrete value between 0 and 12.

**2.2 PROPOSED SYSTEM**

The grading system is implemented as a Client – Server Architecture. The essay writers are the clients who can access the app through their web browsers. The model and other implementations as usual reside at the server end. The Essay will be passed to the Server for grading and then will be submitted for grading. After grading, the grade will be shown to the user on the screen.

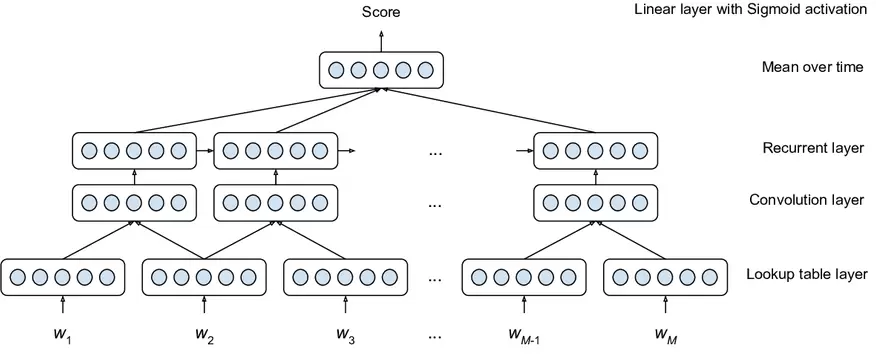
The communication between the LSTM predictor and the Backend script is the most important part. If the libraries are not compatible, it can cause issues due to the threading involved in the scripts.

**3. THEORETICAL ANALYSIS**

**3.1 RNN**

Recurrent neural networks are one of the most successful machine learning models and have attracted the attention of researchers from various fields. Compared to feed-forward neural networks, recurrent neural networks are theoretically more powerful and are capable of learning more complex patterns from data. Therefore, we have mainly focused on recurrent networks in this paper. This section gives a description of the recurrent neural network architecture that we have used for the essay scoring task and the training process.

The neural network architecture that we have used is illustrated in the below figure.



**3.2 LONG SHORT-TERM MEMORY(LSTM) AND GATED RECURRENT UNITS (GRU)**

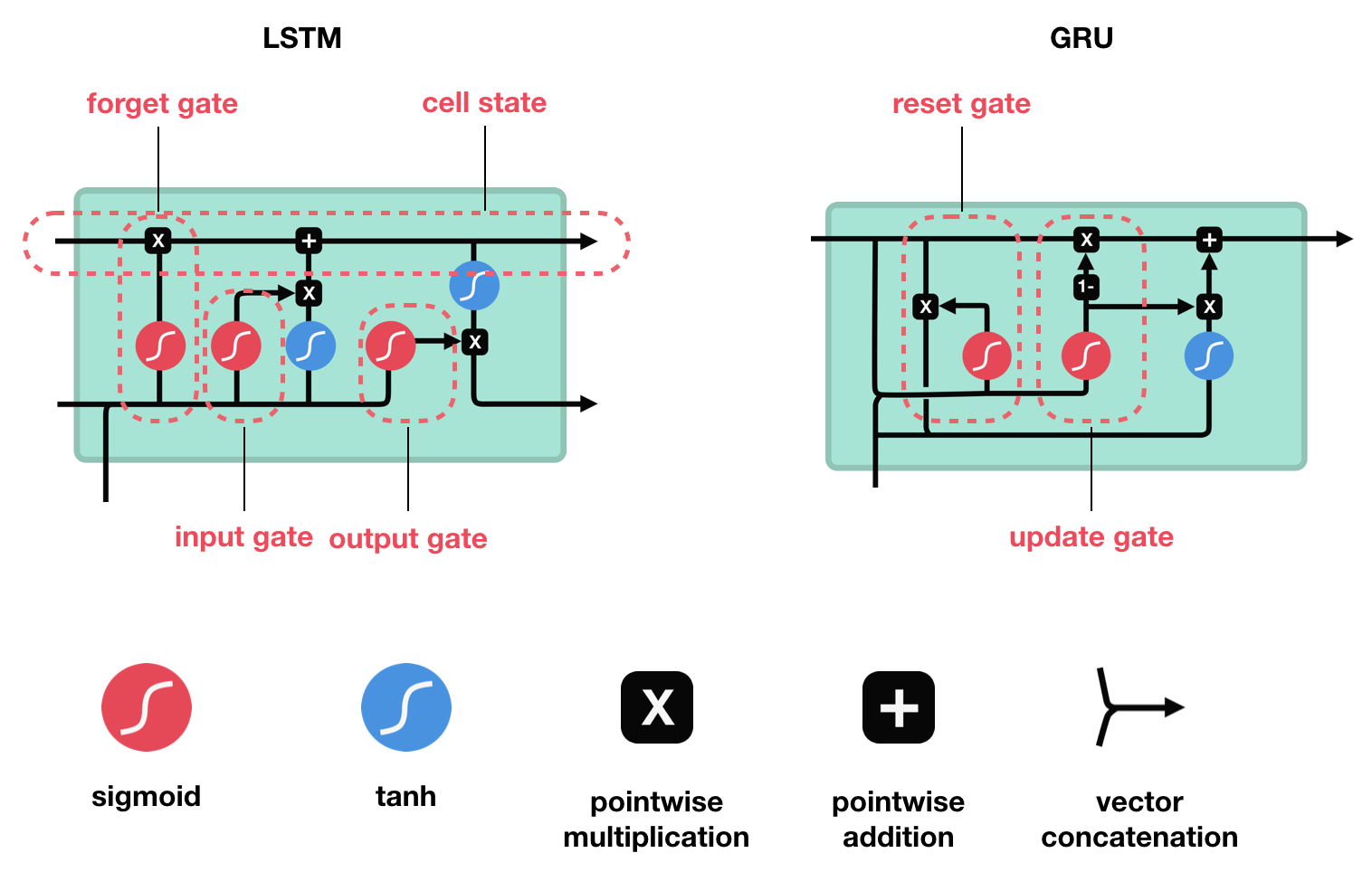
* **Long Short-Term Memory (LSTM)**

LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems). A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.

LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications. The advantage of an LSTM cell compared to a common recurrent unit is its cell memory unit. The cell vector has the ability to encapsulate the notion of forgetting part of its previously stored memory, as well as to add part of the new information. To illustrate this, one has to inspect the equations of the cell and the way it processes sequences under the hood.

* **Gated Recurrent Units (GRUs)**

GRUs are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate. GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be similar to that of LSTM. GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets.



**3.3 PYTHON FLASK**

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. In the case of Flask, its dependencies are:

* + - Werkzeug a WSGI utility library
    - jinja2 which is its template engine

**4. EXPERIMENTAL ANALYSIS**

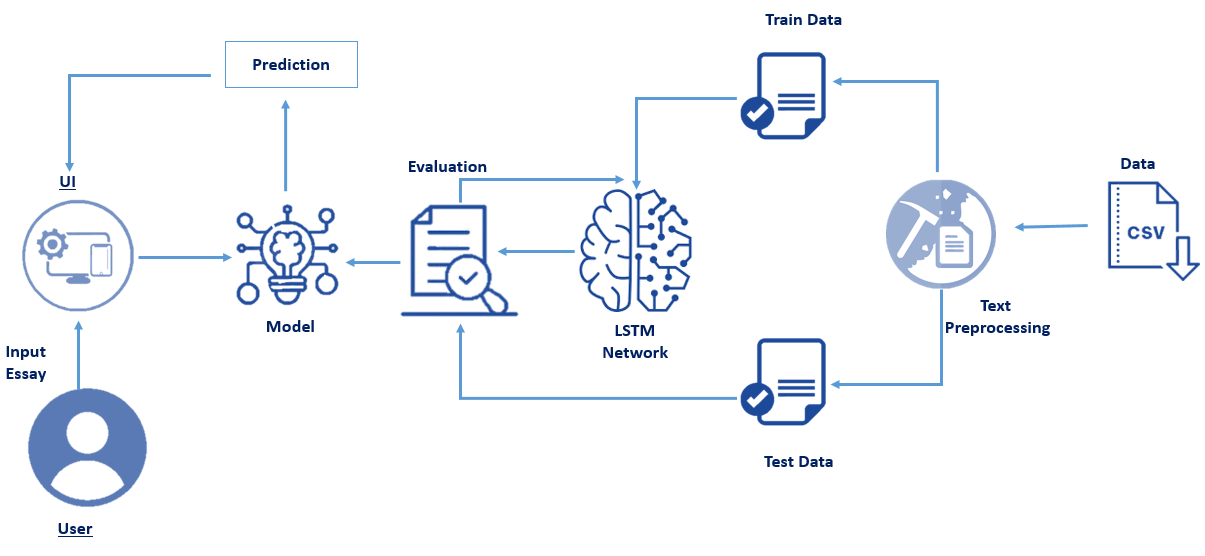
In this section, we describe our experimental setup and present the results. We train the model for a fixed number of epochs and then choose the best model based on the development set. We tokenize the essays using the NLTK5 tokenizer, lowercase the text, and normalize the gold-standard scores to the range of [0, 1]. During testing, we rescale the system-generated normalized scores to the original range of scores and measure the performance. In order to evaluate the performance of our system, we compare it to a publicly available opensource6 AES system called ‘Enhanced AI Scoring Engine’ (EASE). The features that are extracted by EASE can be categorized into four classes:

* + - Length-based features
    - Parts-of-Speech (POS)
    - Word overlap with the prompt
    - Bag of n-grams

After extracting the features, a regression algorithm is used to build a model based on the training data. We use these two regression methods as our baseline systems. Our system has several hyper-parameters that need to be set. We use the RMS optimizer with decay rate (ρ) set to 0.9 to train the network and we set the base learning rate to 0.001. The vocabulary is the 4,000 most frequent words in the training data and all other words are mapped to a special token that represents unknown words. We regularize the network by using dropout and we set the dropout probability to 0.5. During training, the norm of the gradient is clipped to a maximum value of 10. We set the word embedding dimension (dLT) to 50 and the output dimension of the recurrent layer (dr) to 300. If a convolution layer is used, the window size (l) is set to 3 and the output dimension of this layer (dc) is set to 50. Finally, we initialize the lookup table layer using pre-trained word embeddings8 released by Zou et al. (2013). Moreover, the bias value of the linear layer is initialized such that the network’s output before training is almost equal to the average score in the training data. We have performed several experiments to identify the best model architecture for our task. These architectural choices are summarized below:

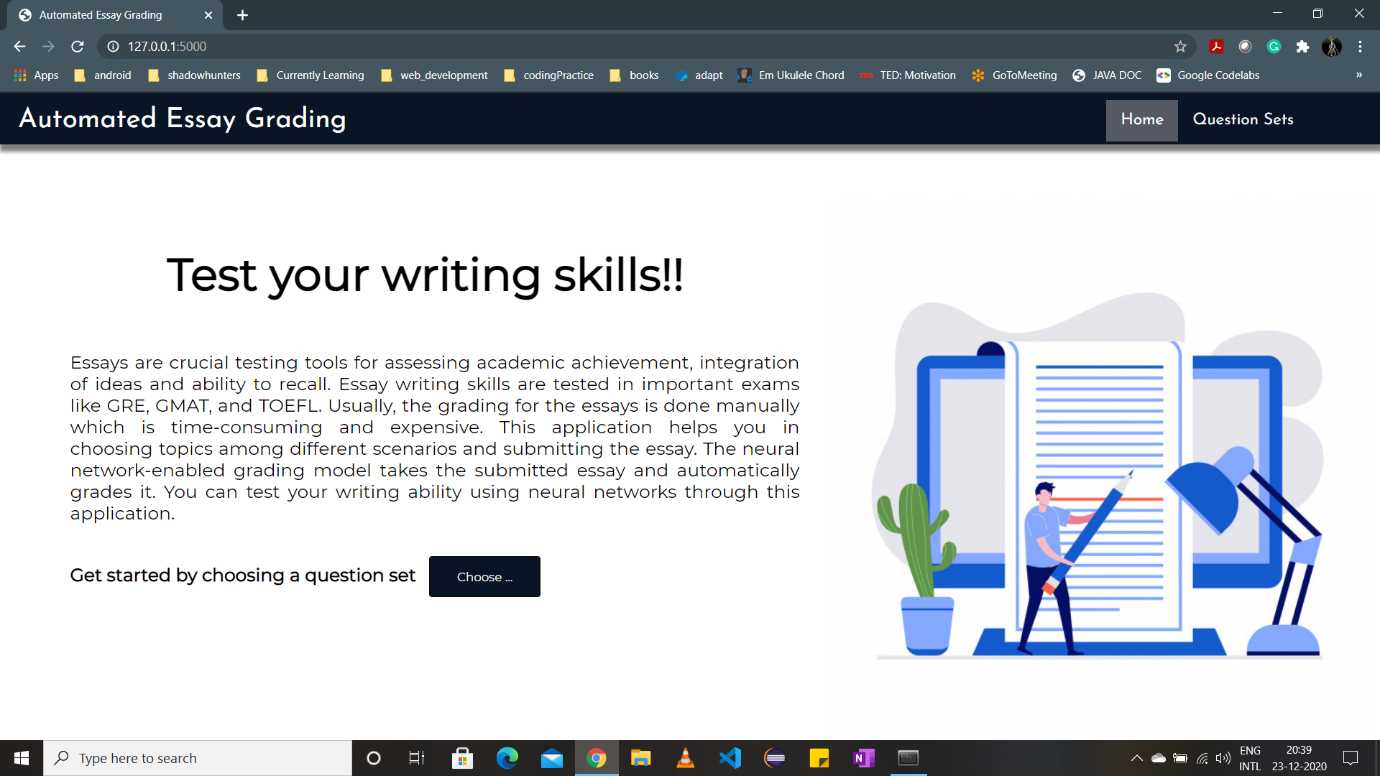
* + - Convolutional vs. recurrent neural network
    - RNN unit type (basic RNN, GRU, or LSTM)
    - Using mean-over-time over all recurrent states vs. using only the last recurrent state
    - Using mean-over-time vs. an attention mechanism
    - Using a recurrent layer vs. a convolutional recurrent layer
    - Unidirectional vs. bidirectional LSTM

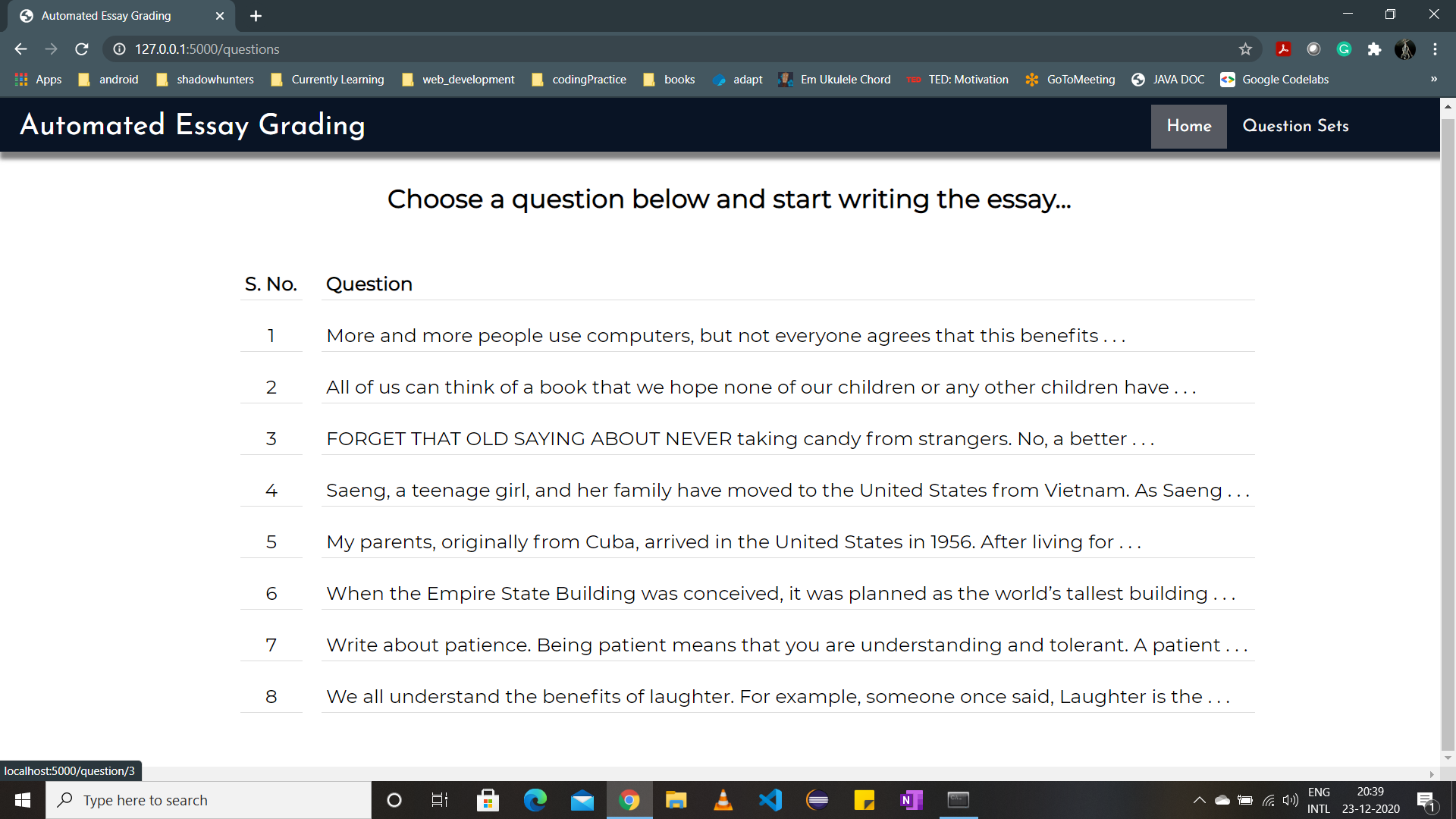
**5. FLOW CHART**

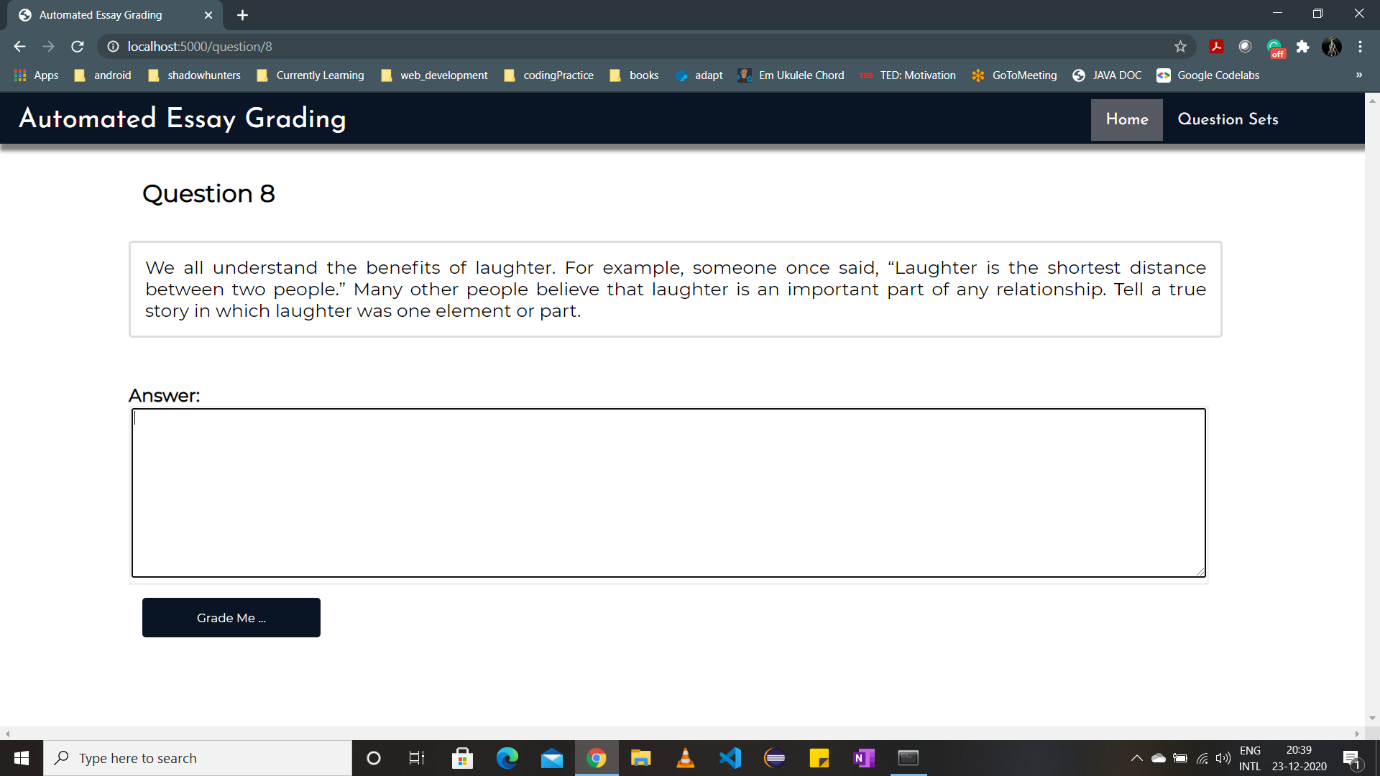


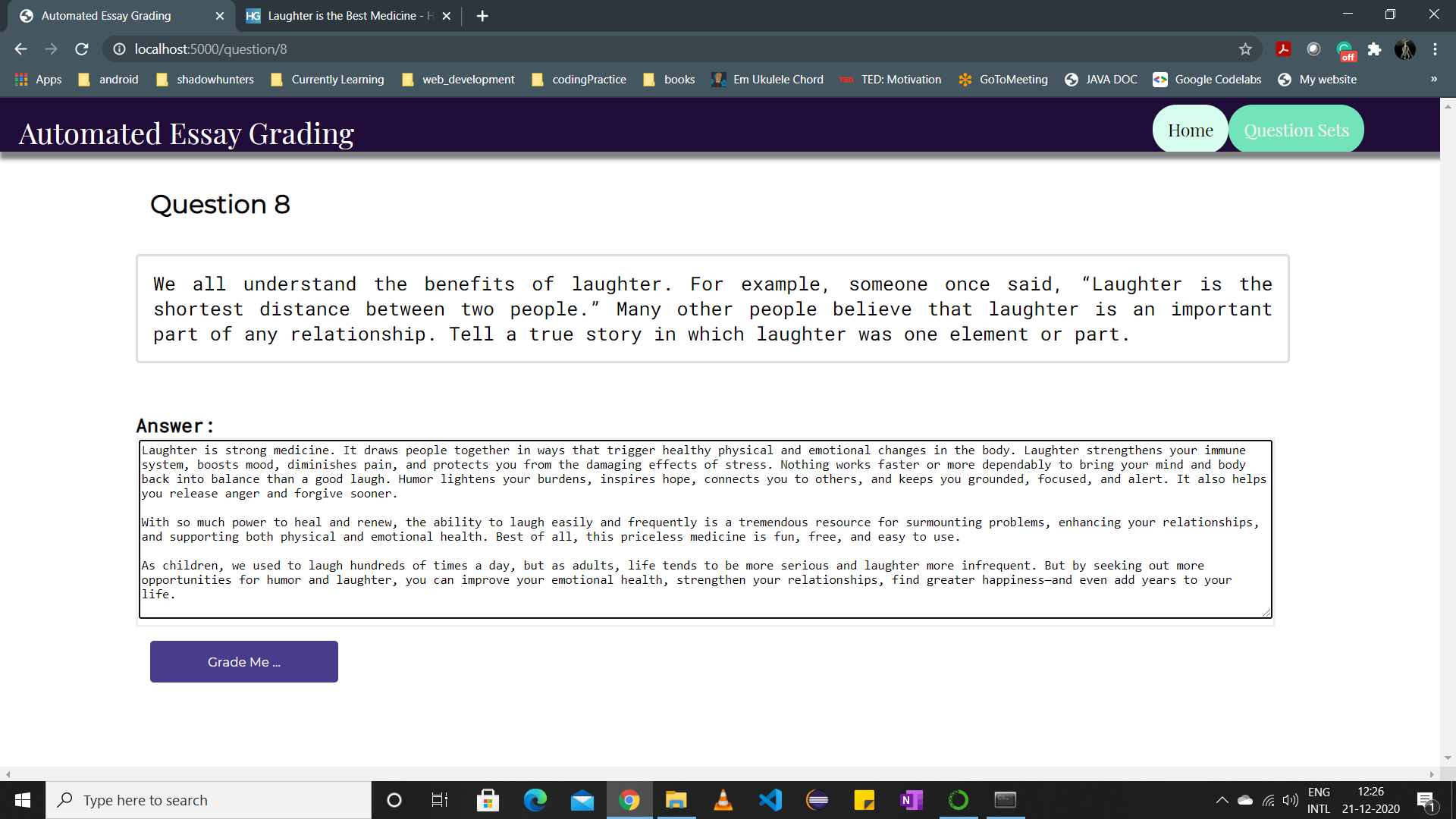
**6. RESULT**

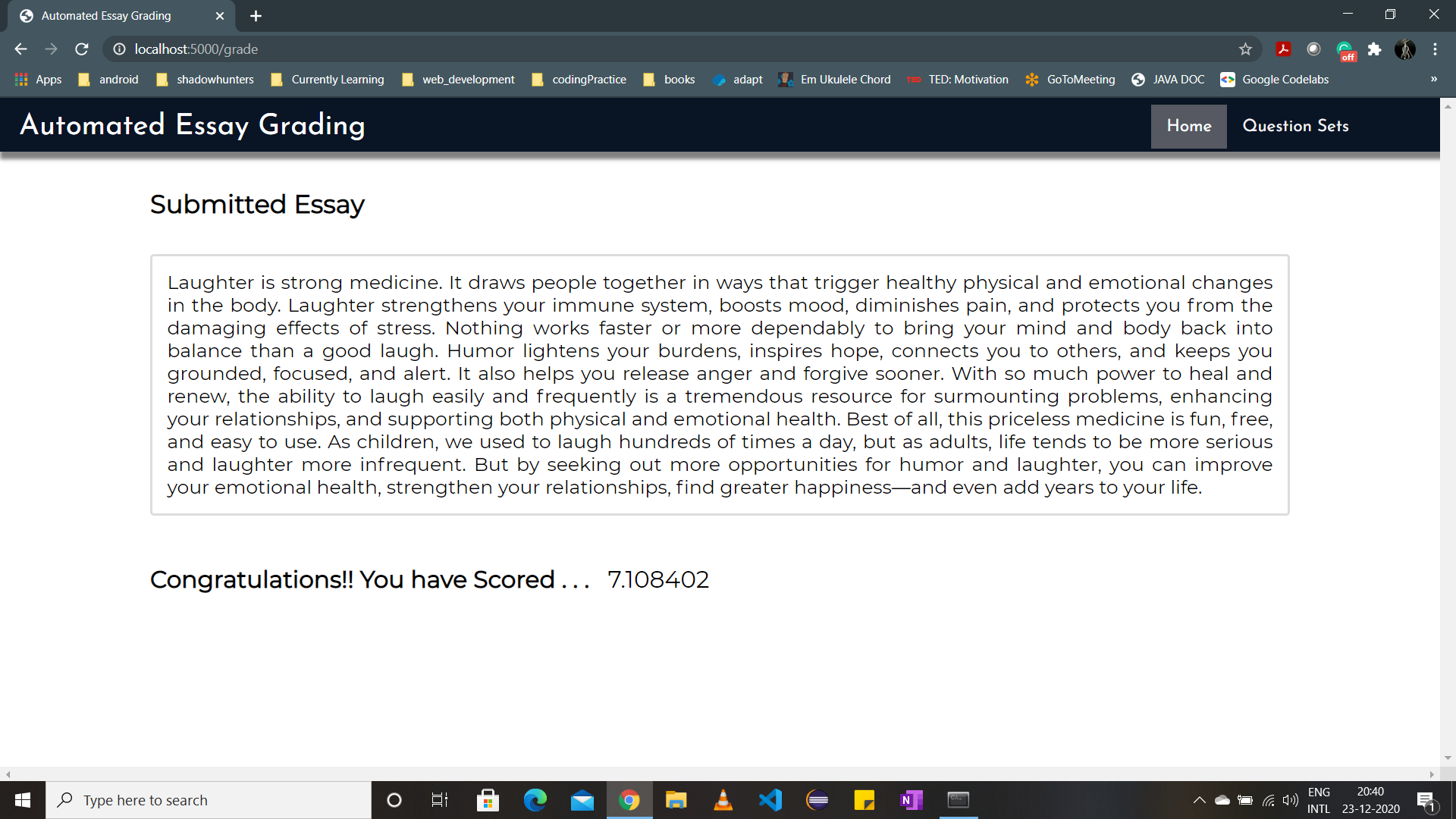
Automated essay scoring systems are used in evaluating and scoringstudent essays written based on a given prompt. The performance of thesesystems are assessed by comparing their scores assigned to a set of essaysto human-assigned gold-standard scores. Since the output of AES systemsis usually a real valued number, the task is often addressed as a supervisedmachine learning task (mostly by regression or preference ranking). Deeplearning algorithms are used to learn the relationship between the essaysand reference scores.











**7. CONCLUSION**

We have proposed an approach based on recurrent neural networks to tackle the task of automated essay scoring. Furthermore, an analysis of the network has been performed to get an insight of the recurrent neural network model and we show that the method effectively utilizes essay content to extract the required information for scoring essays. The essay grading task is a laborious task which can be automated with the help of Deep Learning. The grader is very good in differentiating between ambiguous sentence formations and the new vector representation is very good in implementing the NLP translations. The performance on context and sentiment rich essays can be made better by better training our model with larger and more complex datasets and advanced NLP features. It is also found that the LSTM model proves to be a good model to predict scores for essays that include relatively long sequences of words which is consistent with the nature of the LSTM models.

**8. FUTURE SCOPE**

The future directions of this work may be to highlight the words and sentences that made the AES system give a specific score for further analysis and adaptive feedbacking, in addition to training and testing the model on a larger dataset. The essay grader can be used by teachers for grading student essays. It can also be used by testing agencies for test of English writing to take the burden off the human graders. The concept can be extended to other languages too if the dataset is available. Lastly, the accuracy of the model can still be increased if more data is feed to the LSTM network while training. More sophisticated models can be developed using transfer learning at the cost of using more resources required to train the model.

**9. APPENDIX**

**Pre-processing the Data**

We will pre-process all essays and convert them to feature vectors so that they can be fed into the RNN. These are all helper functions used to clean the essays.

|  |
| --- |
|  |
|  | import nltk |
|  | import re  import numpy as np |
|  | from nltk.corpus import stopwords |
|  | from gensim.models import Word2Vec |
|  |  |
|  | def essay\_to\_wordlist(essay\_v, remove\_stopwords): |
|  | """Remove the tagged labels and word tokenize the sentence.""" |
|  | essay\_v = re.sub("[^a-zA-Z]", " ", essay\_v) |
|  | words = essay\_v.lower().split() |
|  | if remove\_stopwords: |
|  | stops = set(stopwords.words("english")) |
|  | words = [w for w in words if not w in stops] |
|  | return (words) |
|  |  |
|  | def essay\_to\_sentences(essay\_v, remove\_stopwords): |
|  | """Sentence tokenize the essay and call essay\_to\_wordlist() for word tokenization.""" |
|  | tokenizer = nltk.data.load('tokenizers/punkt/english.pickle') |
|  | raw\_sentences = tokenizer.tokenize(essay\_v.strip()) |
|  | sentences = [] |
|  | for raw\_sentence in raw\_sentences: |
|  | if len(raw\_sentence) > 0: |
|  | sentences.append(essay\_to\_wordlist(raw\_sentence, remove\_stopwords)) |
|  | return sentences |
|  |  |
|  | def makeFeatureVec(words, model, num\_features): |
|  | """Make Feature Vector from the words list of an Essay.""" |
|  | featureVec = np.zeros((num\_features,),dtype="float32") |
|  | num\_words = 0. |
|  | index2word\_set = set(model.wv.index2word) |
|  | for word in words: |
|  | if word in index2word\_set: |
|  | num\_words += 1 |
|  | featureVec = np.add(featureVec,model[word]) |
|  | featureVec = np.divide(featureVec,num\_words) |
|  | return featureVec |
|  |  |
|  | def getAvgFeatureVecs(essays, model, num\_features): |
|  | """Main function to generate the word vectors for word2vec model.""" |
|  | counter = 0 |
|  | essayFeatureVecs = np.zeros((len(essays),num\_features),dtype="float32") |
|  | for essay in essays: |
|  | essayFeatureVecs[counter] = makeFeatureVec(essay, model, num\_features) |
|  | counter = counter + 1 |
|  | return essayFeatureVecs |

* **getAvgFeatureVecs**: This function accepts 3 parameters: essays, model, num\_features. It internally calls **makeFeatureVec** function to convert essays into FeatureVector.
* **makeFeatureVec:** This function accepts 3 parameters: words, model, num\_features. Using Word2Vec index2word function and np.divide it gives ultimately average feature vectors for the passed model.
* **essay\_to\_sentense**: This function accepts 2 parameters: essay\_v, remove\_stopwords. it internally calls essay\_to\_wordlist and converts essays to sentences.
* **essay\_to\_wordlist**: This function accepts 2 parameters: essay\_v, remove\_stopwords. It removes the stopwords and returns words.

Whenever you are working with NLP Machine Learning and Deep Learning tasks the above-mentioned steps are almost necessary because machine understands numbers or we can say that computation is very easy when we use numbers here, we refer to vectors. We are trying to convert essay or corpus to first sentences and then to words which can also be called are tokens and then convert them to vectors.