1. **Methodology**
   1. **System Architecture**

The objective of this research is to develop a machine learning-based, intelligent system that can grade essays on its own. A dataset with a significant number of essays on a certain topic should be carefully picked in order to ensure consistency among the raters. Our dataset has enough essays on various topics that have been graded. Pre-processing of the dataset is the next step. Cleaning the data is the first stage in the pre-processing process. The process of cleaning data involves removing any inaccurate, incomplete, duplicate, or other wrong data from the dataset. The removal of all characters from the dataset that aren't alphabets is the second step in the data cleaning process. The stop words are then all eliminated from the text. To get rid of stop words, the text is broken up into words, and those words are eliminated if they appear on the NLTK list of stop words.

Word tokenization is then applied to the words. The phrases, sentences, and paragraphs in this passage are divided into many units. Tokens are the name for these more compact objects. Then, these tokens are further examined in order to categorise or count them according to a specific sentiment. Then, using word embedding, we create a Word2Vec model in which the words or phrases are translated into real-number vectors. When words are embedded, those that share a semantically similar meaning are closer together than those that do not. These are then passed to the 3 layered LSTM model. There are a total three models with different layer combinations but common activation function and evaluation metrics. The activation metrics used is ReLU which is explained in the further section of this paper. We have used Quadratic Weighted Kappa as the Evaluation metric for the Models.

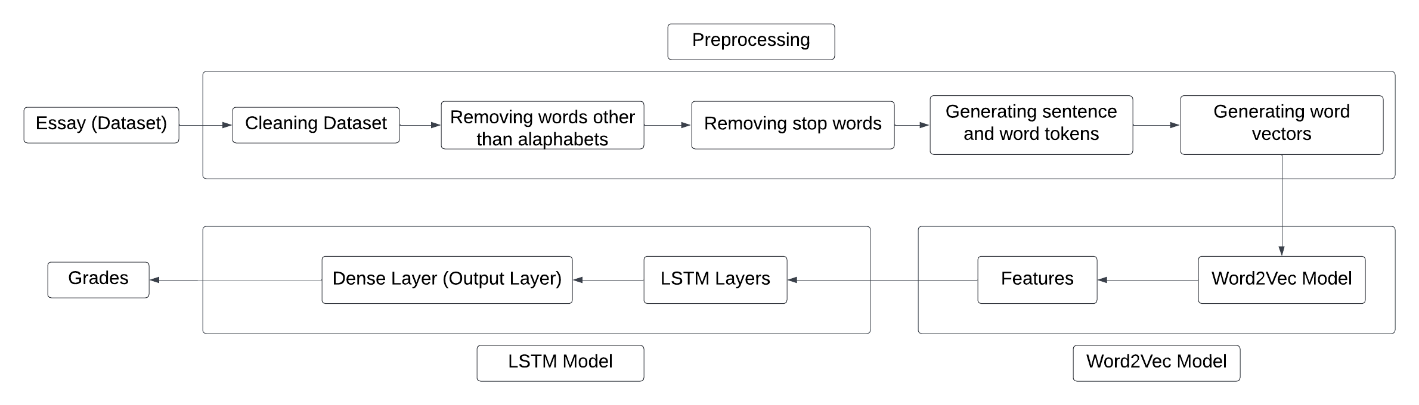


Figure 3.1 : Data Flow in the model

**4.1.1 Word2Vec Model (Word embedding model)**

In the Word2Vec model, each given essay consists of a number of words, and each word is represented by a word embedding according to word2vec. The embedding representations are expected to catch the semantic information carried by each word, i.e. the words with similar meanings will be near to each other in the vector space.

For example, from our corpus, we can find that "computer" is similar to "laptop".

The Neural Network can learn to identify the vector of the input word by using the surrounding word of this input. That means if two different words have the same context, the network tends to give them similar word vectors. Hence, every word in our dataset has a unique vector containing the latent semantic and the vectors of the words in one essay can combine to an essay matrix which is the input unit of our scoring machine.

**4.1.2 3 -Layer LSTM Model**

The paper first introduces the 4-layer LSTM model. It is a sequential model and works better for chosen dataset which is a corpus of essays in text format. The first layer of the model is the word embedding layer. The layer takes 300 as the first argument which is the number of features (output generated from word embedding layer i.e., Word2Vec model), dropout and recurrent dropout as 0.4 respectively and input size from 1 to 300 that is length of each sentence sequence. The next layer takes 64 features as the first argument with recurrent dropout of 0.4. Third layer is also a dropout layer with a dropout value of 0.5. The final layer is a dense layer, it reduces the dimensionality to 1 which is the predicted score. Model have uses ReLU activation function in the dense layer so that the score can be predicted correctly since the values of ReLU function ranges form – ∞ to + ∞. For fitting of training data, models have been passed through batch size of 64 and 100 epochs. These epochs are varied according to the size of the test data. The model produces effective results. Diagrammatic representation is given in figure 4.1. Equations of LSTM are as follows:

Were, is the forget gate, is the input gate, is the output gate, is the cell state, is the hidden state, is sigmoid activation is tanh activation function and ‘.’ is element wise multiplication.

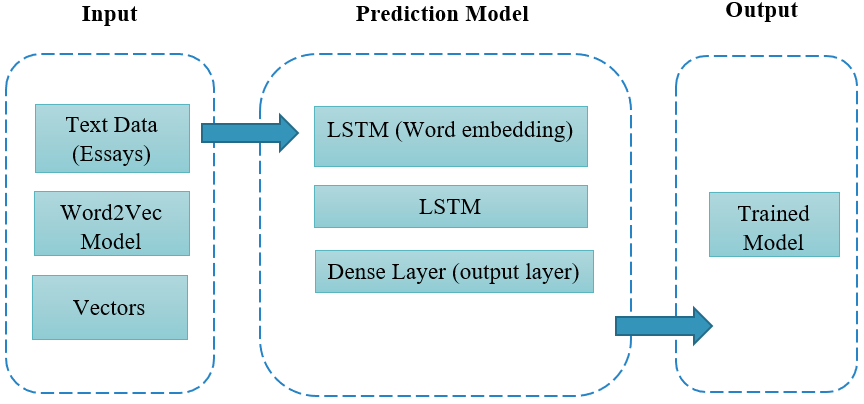


Figure 4.1: LSTM Model

**4.1.3 LSTM with Bi-LSTM**

To improve the performance / results obtained in Unidirectional LSTM, next we implemented the Bi-LSTM model [26]. Unlike standard LSTM, here, the input is allowed to flow in both directions and it is capable of utilizing information from both sides. Again, it is powerful while modelling the sequential data, improving dependencies between words and sentences in both directions. It adds one more LSTM layer to the previous model, the direction of information flow is revered. Then the model combines the output of both LSTM layers to get the final output. Bi-LSTM is considered to produce more meaningful output when both LSTMs are combined. All other parameters are kept the same. Experiments prove that Bi-LSTM model performs better than the standard LSTM model. Diagrammatic representation is given in figure 4.2. Equations of Bi-LSTM are as follows:

Were, output at time t.

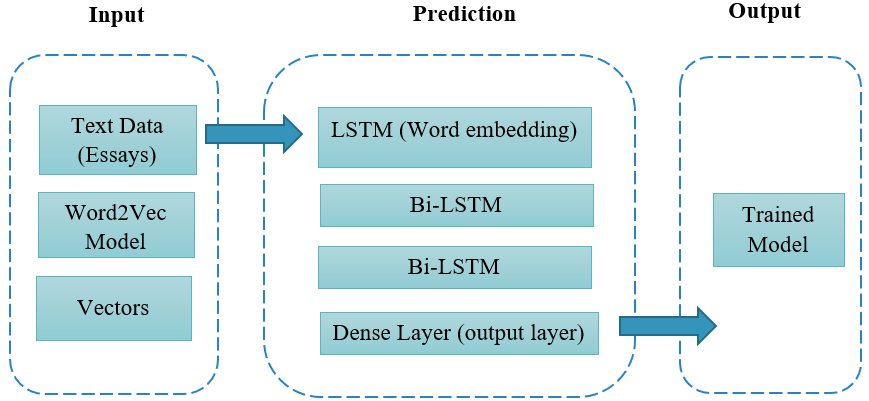


Figure 4.2: LSTM with Bi LSTM Model

**4.1.4 LSTM and Bi-LSTM with attention layer**

Though, using Bi-LSTM gave better results, in order to improve model performance, attention layer [16] is introduced to the model. Sometimes, basic LSTM gets confused between the words and can predict the wrong word. So, in order for the encoder to search for the most relevant information, models have been introduced with an additional layer of attention mechanism. By applying the attention mechanism, the model will be able to effectively extract the information between essays through inter-sentence alignment and gain better performance. Diagrammatic representation is given in figure 4.3

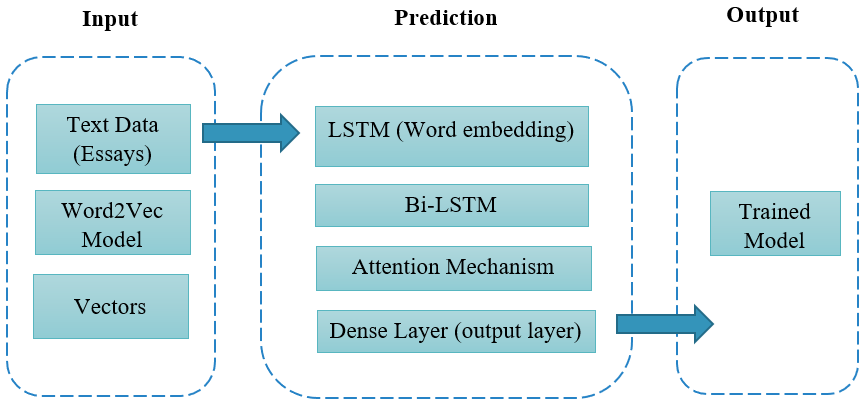
****

Figure 4.3: LSTM and Bi LSTM with Attention

* 1. **Objective Function**

When the essays are given as input the models objective function is to provide minimum error while predicting the scores. We analyse 3 different LSTM models and use following two evaluation metrics to compare the results.

**4.2.1 Quadratic Weighted Kappa**

A set of predictions and a set of multiclass labels are measured by the Quadratic Weighted Kappa index. It attempts to take into account the similarity between the classes, beyond only the class, rather than just focusing on the precision of the match between predictions and labels. As a gauge of agreement between observed raters in cross-classification, Cohen's weighted kappa is frequently used. When ratings are given on nominal scales without an order structure, an appropriate index of agreement is used.Calculation of Cohen’s kappa may be performed according to the following formula:

K = (Pr(a)−Pr(e))/1−Pr(e)

Where Pr(a) represents the actual observed agreement, and Pr(e) represents chance agreement. Generally, a kappa of less than 0.4 is considered poor (a Kappa of 0 means there is no difference between the observers and chance alone). Kappa values of 0.4 to 0.75 are considered moderate to good and a kappa of >0.75 represents excellent agreement. A kappa of 1.0 means that there is perfect agreement between all raters.

**4.2.2 MSE**

MSE stands for Mean Squared Error. It defines the square of the absolute difference between actual and predicted value. Here, squares avoid the cancellation of negative terms.

MSE = 1/n ( y – y`)2

**4.3.1 Approach**

Dataset used in this paper is “The Hewlett Foundation: Automated Essay Scoring” on Kaggle. The dataset includes 8 essay sets. Each of the essay sets was generated from an individual prompt. Average length of essays is in the range of 150 to 550 words per response. All essays were hand graded and were double-scored. The training data is in the format of tab-separated value (TSV) file. There are total 3 scores i.e., rater1 score rater2 score and domain score. The Domain score is an addition of rater 1 and 2 scores. There are some unwanted empty columns also present in the dataset; these columns were dropped using pandas’ libraries. Hence the final dataset has essay\_id, essay\_set, essay, domain1\_score columns.

After generation of the dataset the data needed to be preprocessed before passing on to the model. The essays were first cleaned by removing the stop words, punctuation marks and converting all characters to lowercase. Once the data is cleaned the next step is to generate a feature vector that is to be passed to the word2Vec Model. In order to generate the feature vectors first the cleaned essay was converted into sentence tokens and finally to word tokens. The output of the Word2Vec model was passed as an input to the LSTM layers. For models other than neural network word embeddings were generated.