AUTOMATED ESSAY GRADING

PROJECT REPORT

INTRODUCTION:

- .Essays are paramount for of assessing the academic excellence along with linking the different ideas with the ability to recall but are notably time consuming when they are assessed manually. Manual grading takes significant amount of evaluator’s time and hence it is an expensive process
- 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 by classifying a corpus of textual entities into small number of discrete categories, corresponding to possible grades.
- Lstm technique will be utilized for training the model along with making the use of various other classifications and clustering techniques. We intend to train classifiers on the training set, make it go through the downloaded dataset, and then measure performance our dataset by comparing the obtained values with the dataset values. We have implemented our model using python
- ### Purpose:

◎ Our aim is to build a model that can take in an essay and automatically outputs the grade of that essay

◎ In this automated essay grading project ,we have developed a lstm based model to grade the essay submitted instantly.

LITERATURE SURVEY:

Existing problem:

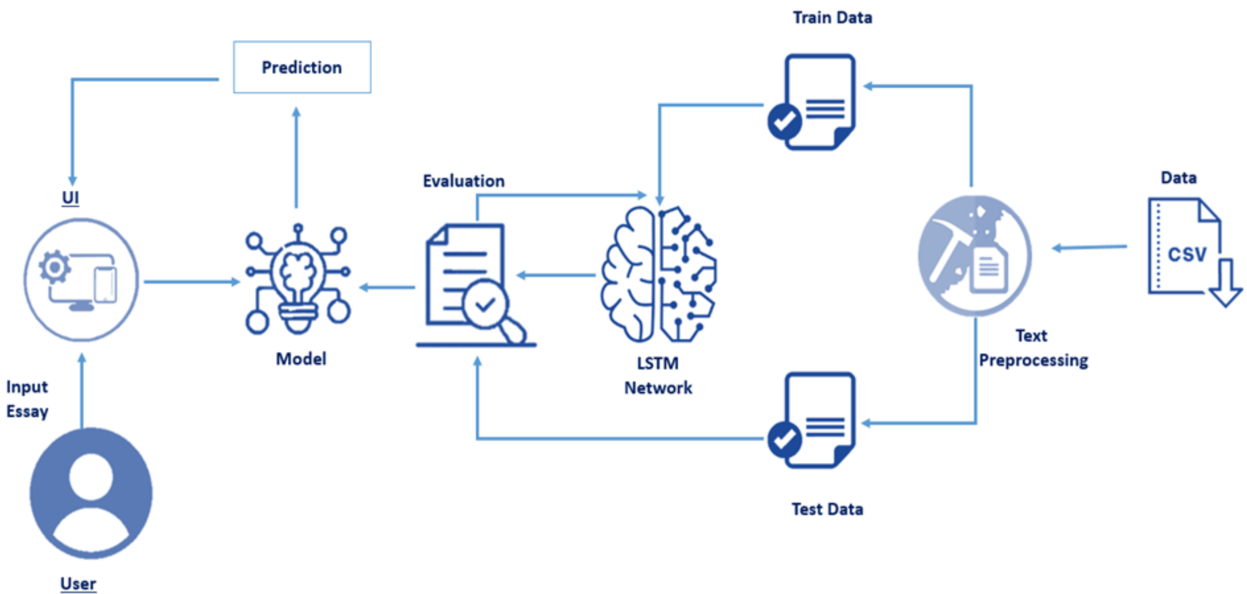
One of the difficulties of grading essays is represented by the perceived subjectivity of the grading process. Many researchers claim that the subjective nature of essay assessment leads to variation in grades awarded by different human assessors, which is perceived by students as a great source of unfairness.

proposed solution:

we propose to use Long Short Term Memory (LSTM) neural networks to create an essay grader to undertake this daunting task. The grader will give an essay a grade which would be expected from a human grader. ... The grader will be able to grade an essay without any bias and high accuracy.

THEORITICAL ANALYSIS:

BLOCK DIAGRAM:



SOFTWARE DESIGNING:

In order to develop this project we need to install the following software/packages:

Anaconda Navigator :

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder

To build Deep learning models you must require the following packages

Tensor flow: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML-powered applications.

Keras: Keras leverages various optimization techniques to make high-level neural network API easier and more performant. It supports the following features:

•Consistent, simple, and extensible API.
•Minimal structure - easy to achieve the result without any frills.
•It supports multiple platforms and backends.
•It is a user-friendly framework that runs on both CPU and GPU.
•Highly scalability of computation.

Flask: Web framework used for building Web applications

EXPERIMENTAL INVESTIGATIONS:

While working on the solution we have investigated the following topics

•Supervised and unsupervised learning

•Regression Classification and Clustering

•Artificial Neural Networks

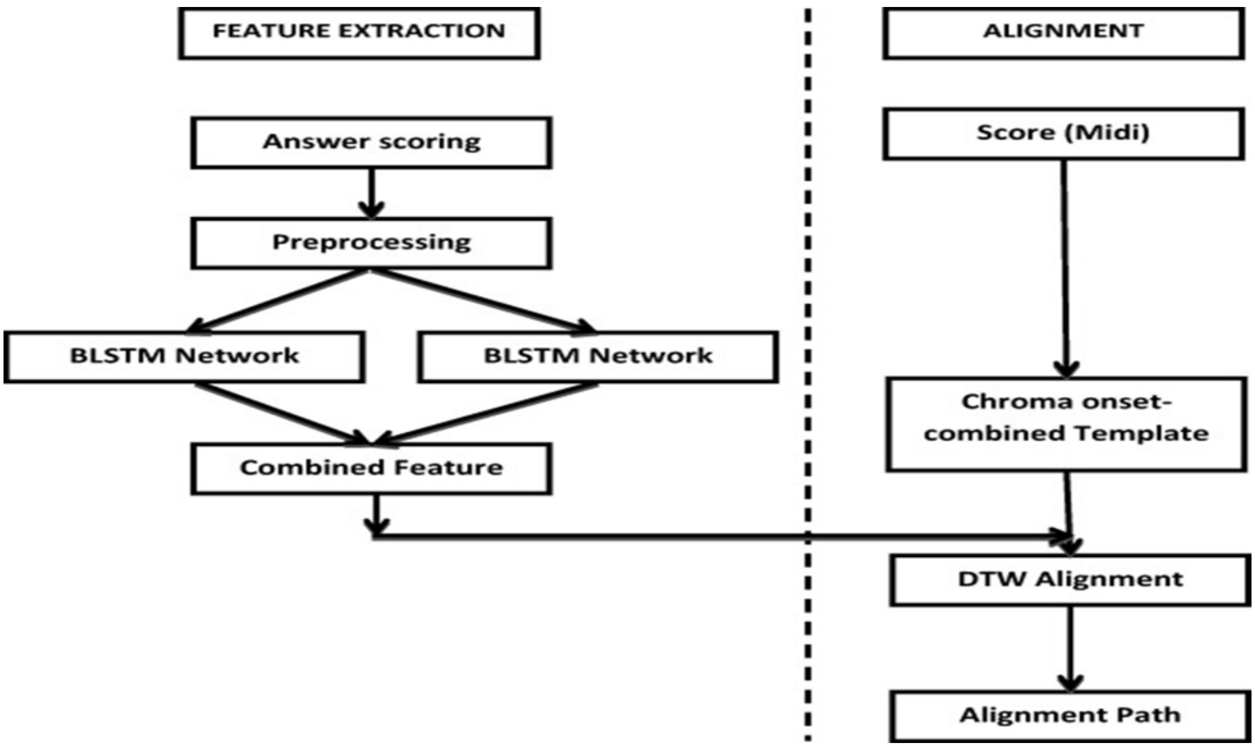
•Natural Language Processing

•Flask app

•LSTM and GRU

WE HAVE COLLECTED DATASET FROM KAGGLE,WHICH CONSISTS OF MAMY SAMPLE ESSAYS AND THEIR GRADES.

FLOW CHART:

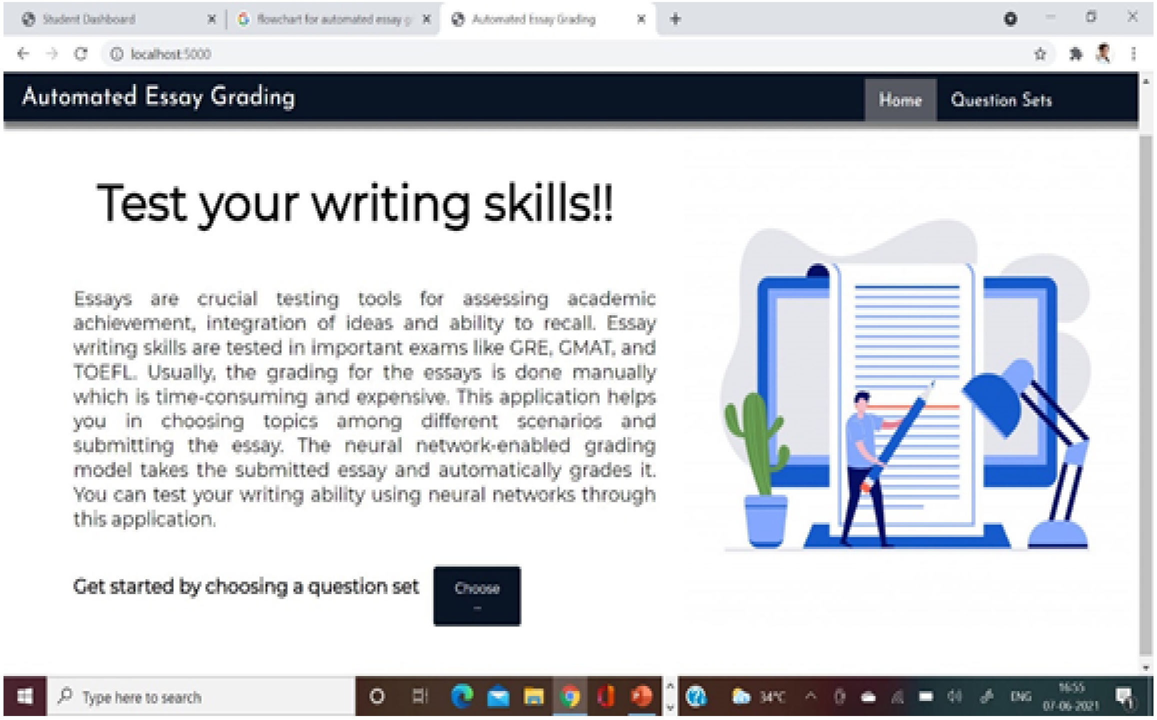


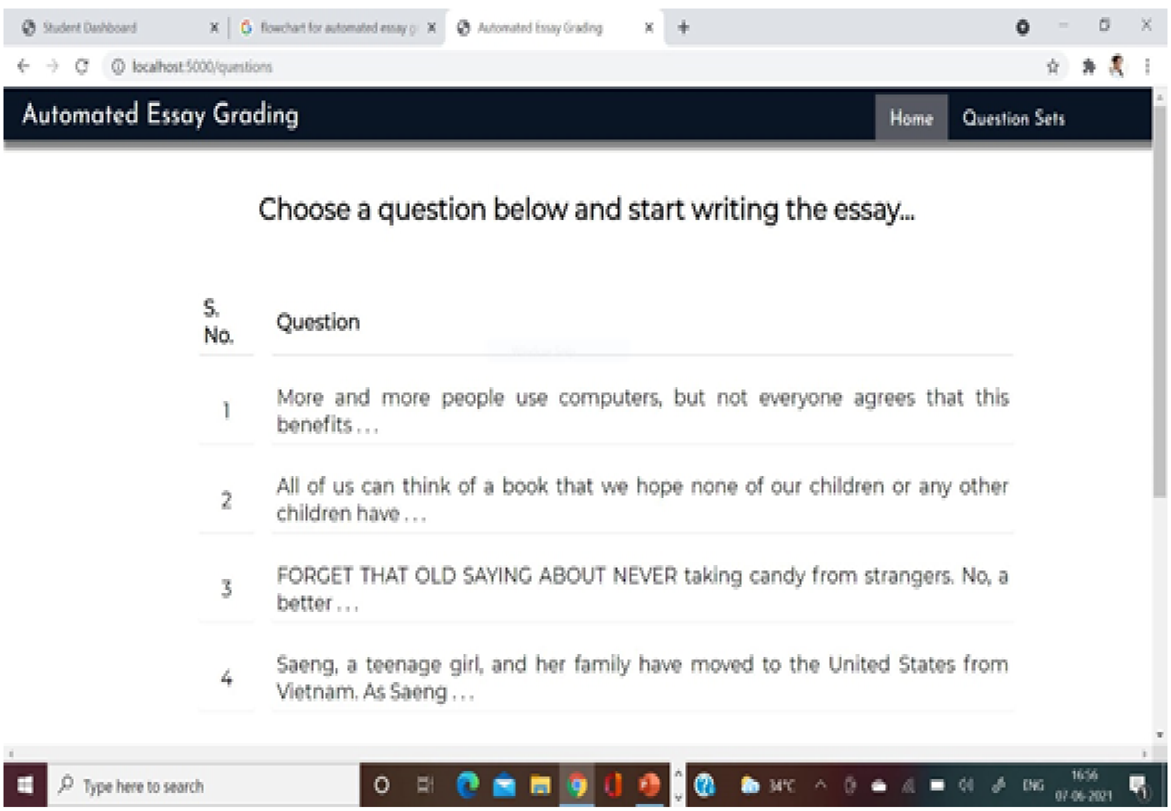
RESULT:

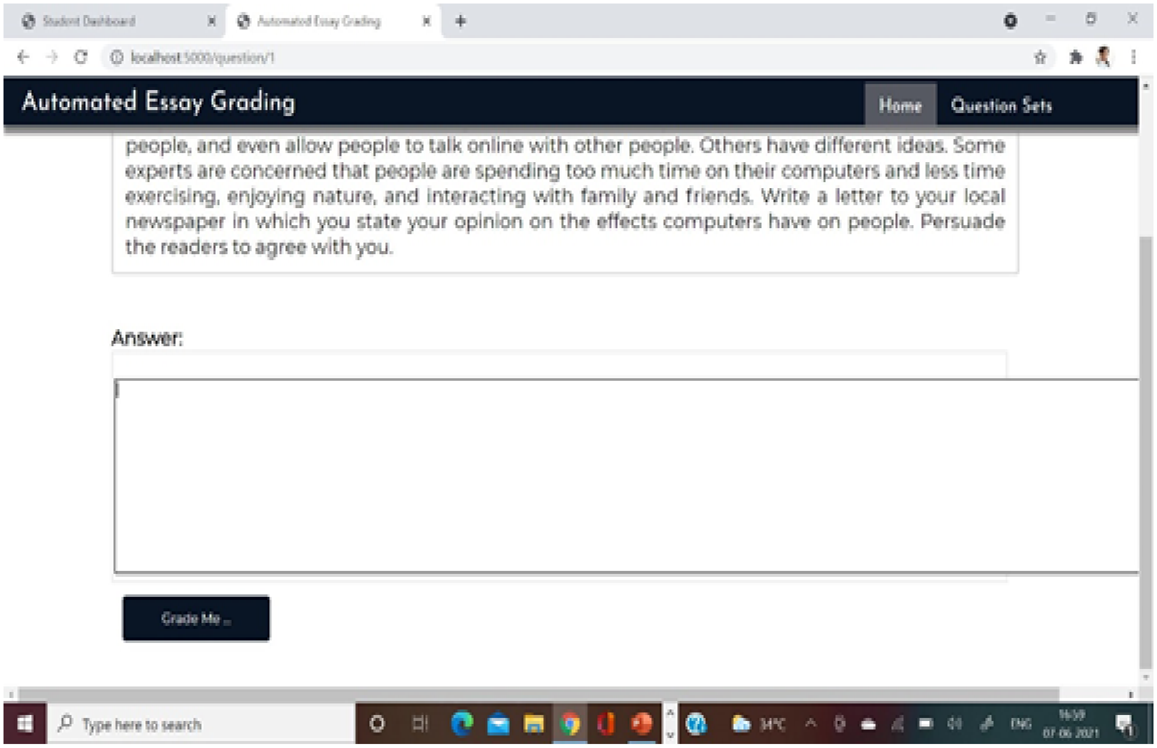
```
In [85]:  model.fit(trainDataVecs, y_train, batch_size=64, epochs=11)

Epoch 1/11
10380/10380 [==============================] - 5s 516us/step - loss: 5.5559 - mean_absolute_error: 1.2767
Epoch 2/11
10380/10380 [==============================] - 5s 516us/step - loss: 5.5874 - mean_absolute_error: 1.2809
Epoch 3/11
10380/10380 [==============================] - 5s 529us/step - loss: 5.6758 - mean_absolute_error: 1.2694
Epoch 4/11
10380/10380 [==============================] - 5s 523us/step - loss: 5.7715 - mean_absolute_error: 1.2878
Epoch 5/11
10380/10380 [==============================] - 5s 525us/step - loss: 5.7667 - mean_absolute_error: 1.2733 0s - loss: 5.7280
- mean_absolute_err
Epoch 6/11
10380/10380 [==============================] - 5s 528us/step - loss: 5.3691 - mean_absolute_error: 1.2545
Epoch 7/11
10380/10380 [==============================] - 6s 535us/step - loss: 5.0935 - mean_absolute_error: 1.2280
Epoch 8/11
10380/10380 [==============================] - 6s 533us/step - loss: 5.3537 - mean_absolute_error: 1.2550
Epoch 9/11
10380/10380 [==============================] - 6s 544us/step - loss: 5.1854 - mean_absolute_error: 1.2490
Epoch 10/11
10380/10380 [==============================] - 6s 539us/step - loss: 5.3082 - mean_absolute_error: 1.2433
Epoch 11/11
10380/10380 [==============================] - 6s 535us/step - loss: 5.4899 - mean_absolute_error: 1.2660

Out[85]: <keras.callbacks.History at 0x2142abc29e8>
```

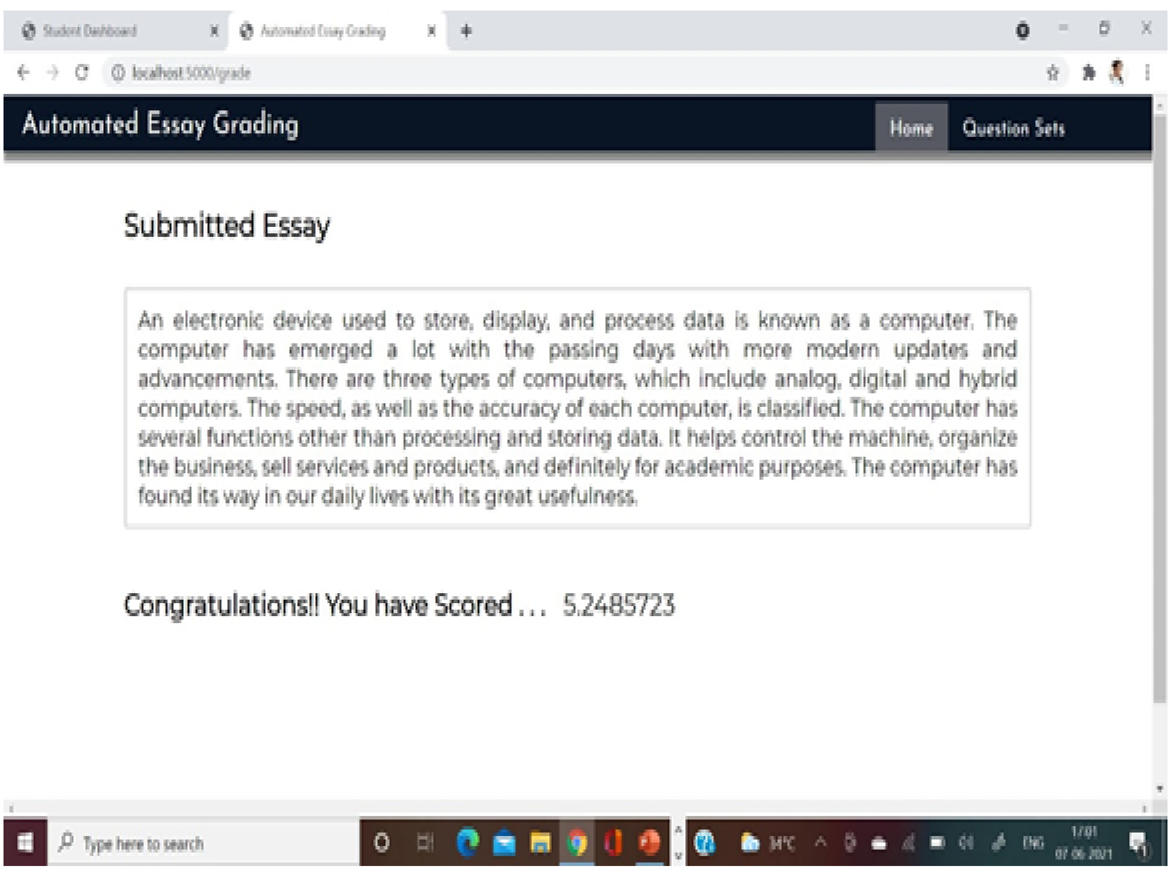
```
In [90]:  from sklearn.metrics import r2_score
          accuracy = r2_score(y_test,y_pred)
          accuracy

Out[90]:  0.941444519660382
```

RESULT AFTER WEB APPLICATION:






Student Dashboard
Automated Essay Grading
localhost:5000/question/1
Automated Essay Grading
Home
Question Sets
people, and even allow people to talk online with other people. Others have different ideas. Some experts are concerned that people are spending too much time on their computers and less time exercising, enjoying nature, and interacting with family and friends. Write a letter to your local newspaper in which you state your opinion on the effects computers have on people. Persuade the readers to agree with you.
Answer:
Grade Me ..
Type here to search


Student Dashboard
Automated Essay Grading
localhost:5000/grade
Automated Essay Grading
Home
Question Sets
Submitted Essay
An electronic device used to store, display, and process data is known as a computer. The computer has emerged a lot with the passing days with more modern updates and advancements. There are three types of computers, which include analog, digital and hybrid computers. The speed, as well as the accuracy of each computer, is classified. The computer has several functions other than processing and storing data. It helps control the machine, organize the business, sell services and products, and definitely for academic purposes. The computer has found its way in our daily lives with its great usefulness.
Congratulations!! You have Scored . . . 5.2485723
Type here to search

ADVANTAGES:

- Since feedback is immediate, students are able to submit work at any stage of the **writing** process, receive feedback, make improvements, and keep **writing**. They no longer need to wait the customary two weeks for a teacher to comment and suggest corrections
- The **benefits of Automated essay grading** is that **essay** will get **graded** faster, saving time for students and allowing students to receive their papers fasterso they can revise them if necessary.

DISADVANTAGES:

- When there are many students and time is short, feedback detail is reduced, assessment quality may be compromised, and in extreme cases a 'tick and flick' approach to **grading** may seem a tantalizing option

APPLICATIONS:

- educational institutions to reduce work burden on teachers
- online essay competitions
- online essay writing tutorials

CONCLUSION:

As we hypothesized, features from each category contribute towards a good prediction. Our nal model contains features across all the categories we tried to test for. Our model works relatively better on noncontext specic essays. Performance on content specic and richer essays can be improved by incorporating content and advanced NLP features.

our model performs relatively well on the persuasive essays. It suffers on sets where context is central to the essay. This suggests that counting the presence of top words of the essay set in the essay is not a suffciently complete measure of content,This indicates we may need to incorporate features for testing complex sentence structure, like N-grams

FUTURE SCOPE:

Even though LSTM worked as a good predictor for essay scores we are did not test if this is the best model for text assessment machine learning problems. There is scope for further exploration and evaluation of alternative models in this area (for example logistic model trees). We could also further improve the model by using more complex features. This would be particularly constructive for context specic essays in the data set. We believe that more advanced NLP features (N-grams, k-nearest neighbors in bag of words) and features that are grammar and usage specic can further enhance the prediction model.

BIBILOGRAPHY:

https://github.com/Guided-Projects/Automated-Essay-Grading

APPENDIX

SOURCE CODE:

```
In [ ]:  import nltk
         nltk.download('punkt')
         nltk.download('stopwords')
         from nltk.corpus import stopwords
```

```
In [ ]:  import pandas as pd
         import numpy as np
         import re
         from gensim.models import Word2Vec
         from sklearn.model_selection import train_test_split
```

```
In [ ]:  data= pd.read_csv('./data1/training_set_rel3.tsv', sep='\t', encoding='ISO-8859-1')
```

```
In [ ]:  data.head()
```

```
In [ ]:  data.isnull().any()
```

```
In [ ]:  data= data.dropna(axis=1)
         data= data.drop(columns=['rater1_domain1', 'rater2_domain1'])
```

```
In [ ]:  data.head()
```

```
In [ ]:  x=data.iloc[:,0:3]
         y=data.iloc[:,3]
```

```
In [ ]:  min_scores = [-1, 2, 1, 0, 0, 0, 0, 0, 0]
         max_scores = [-1, 12, 6, 3, 3, 4, 4, 30, 60]
```

preprocessing

```
In [ ]:  #removing the extra characters other than alphabets and stopwords and tokenizing the words
         def essay_to_wordlist(essay_v):
             essay_v = re.sub("[^a-zA-Z]", " ", essay_v)
             words = essay_v.lower().split()
             stops = set(stopwords.words("english"))
             words = [w for w in words if not w in stops]
             return (words)

         #Tokenize the senstences and call essay_to_wordlist() for word tokenization.
         def essay_to_sentences(essay_v):
             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))
             return sentences
```

In [ ]:

```
#Feature vector is made from the words list of an essay.
def makeFeatureVec(words, model, num_features):
    featureVec = np.zeros((num_features,),dtype="float32")
    num_words = 0.
    index2word_set = set(model.wv.index_to_key)
    for word in words:
        if word in index2word_set:
            num_words += 1
            featureVec = np.add(featureVec,model.wv[word])
    featureVec = np.divide(featureVec,num_words)
    return featureVec

#Word vectors are generated for Word2Vec model
def getAvgFeatureVecs(essays, model, num_features):
    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
```

In [ ]:

```
#the dataset is split to training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state = 0)
```

In [ ]:

```
x_train.shape
```

In [ ]:

```
x_test.shape
```

In [ ]:

```
train_essays = x_train['essay']
test_essays = x_test['essay']
```

In [ ]:

```
sentences = []
# Obtaining all sentences from the training essays.
for essay in train_essays:
    sentences += essay_to_sentences(essay)
```

In [ ]:

```
sentences
```

In [ ]:

```
#build the vectorizer with maximum featuroes of 300
num_features = 300
min_word_count = 40
num_workers = 4
context = 10
downsampling = 1e-3
model = Word2Vec(sentences, workers=num_workers, vector_size=num_features, min_count = min_word_count, window = context,
                 sample = downsampling)
```

In [ ]:

```
#save the vectorizer in .bin file
model.wv.save_word2vec_format('word2vecmodel.bin', binary=True)
```

In [ ]:

```
#get the training vectors
clean_train_essays = []
for essay_v in train_essays:
        clean_train_essays.append(essay_to_wordlist(essay_v))
trainDataVecs = getAvgFeatureVecs(clean_train_essays, model, num_features)

#get the testing vectors
clean_test_essays = []
for essay_v in test_essays:
    clean_test_essays.append(essay_to_wordlist( essay_v))
testDataVecs = getAvgFeatureVecs( clean_test_essays, model, num_features )

#convert the vectors to numpy array
trainDataVecs = np.array(trainDataVecs)
testDataVecs = np.array(testDataVecs)

# Reshaping train and test vectors to 3 dimensions. (1 represnts one timestep)
trainDataVecs = np.reshape(trainDataVecs, (trainDataVecs.shape[0], 1, trainDataVecs.shape[1]))
testDataVecs = np.reshape(testDataVecs, (testDataVecs.shape[0], 1, testDataVecs.shape[1]))
```

model building

In [ ]:

```
from keras.layers import Embedding, LSTM, Dense, Dropout, Lambda, Flatten
from keras.models import Sequential, load_model, model_from_config
import keras.backend as K
```

In [ ]:

```
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'))
```

In [ ]:

```
model.compile(loss='mean_squared_error', optimizer='rmsprop', metrics=['mae'])
```

In [ ]:

```
model.fit(trainDataVecs, y_train, batch_size=64, epochs=20)
```

In [ ]:

```
testDataVecs.shape
```

In [ ]:

```
y_pred = model.predict(testDataVecs)
```

In [ ]:

```
y_pred
```

In [ ]:

```
model.save('final_lstm.h5')
```

In [ ]:

```
from sklearn.metrics import r2_score
accuracy = r2_score(y_test,y_pred)
accuracy
```

model testing

```
In [ ]: import gensim.models.keyedvectors as word2vec
```

```
In [ ]: testsen='''Dear local newspaper, I\'ve heard that not many people think computers benefit society. I disagree with that. Comp
```

```
In [ ]: testsen
```

```
In [ ]: model = word2vec.KeyedVectors.load_word2vec_format('word2vecmodel.bin', binary=True)
        index2word_set = set(model.index_to_key)
```

```
In [ ]: testsen2 = re.sub("[^a-zA-Z]", " ", testsen)
        testsen2 = essay_v.lower()
        featureVec = np.zeros((300,),dtype="float32")
        for word in testsen2:
                if word in index2word_set:
                    featureVec = np.add(featureVec,model[word])
```

```
In [ ]: featureVec.shape
```

```
In [ ]: avc=featureVec.reshape(1,1,300)
```

```
In [ ]: from keras.models  import load_model
        model1 = load_model("final_lstm.h5")
```

```
In [ ]: y_pred = model1.predict(avc)
```

```
In [ ]: y_pred
```

-----------------------------------THE END--