Automatic Essay Scoring

A Project Report in partial fulfillment of the degree

Bachelor of Technology

in

Computer Science & Engineering / Electronics & Communication Engineering

By

19K41A0432	Achha Nethaji
19K41A0510	Vamshi Yadav
19K41A0516	Manugonda Ganesh

Under the guidance of

Mr. D. Ramesh

Submitted to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

S.R.ENGINEERING COLLEGE (A), ANANTHASAGAR, WARANGAL

(Affiliated to JNTUH, Accredited by NBA) Nov-2022.

SR Engineering College
Innovation . Creativity . Entrepreneurship

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Project Report entitled “**Automatic Essay Scoring**” is a record of bonafide work carried out by the student NETHAJI ACHHA, JUNKALA VAMSHI YADAV, MANUGONDA GANESH bearing Roll No(s) 19K41A0432, 19K41A0510, 19K41A0516 during the academic year 2022-23 in partial fulfillment of the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Electronics & Communication Engineering** by the S.R. ENGINEERING COLLEGE, Ananthasagar, Warangal.

Supervisor **Head of the Department**

External Examiner

ABSTRACT

Essays are a good medium for evaluating academic competence and the capacity to connect and recall many ideas, but when they are manually evaluated by human assessors, they take a lot of time. One of the popular ways to assess a student's aptitude for learning and intelligence is through an essay. For human judges, scoring essays is a laborious and time-consuming operation. Because manual grading is subjective, it consumes a lot of evaluators' time and is therefore a time-consuming process. Automated essay scoring makes the grading process quicker and more effective. If automated grading is correct, it will not only speed up evaluation when compared to human grades, but it will also produce scores that are precise and accurate. The project's goal is to employ deep learning techniques to create an automated essay evaluation system. In order to evaluate all of the insights from the downloaded data-set, we plan to train models on the provided training set, have them analyse them, and then evaluate the performance of our model by contrasting the results with the values from the data set. By assessing the score using human raters, the data-set values for an automated essay scoring are obtained. We begin by processing the data set and using **Transfer learning** to turn each article into a numerical or vector and to perform embedding form that can be understood by models. Additionally, our system automatically learns the relationship between training essays and their assigned score utilizing **K-means clustering**.

Table Of Contents

1. INTRODUCTION

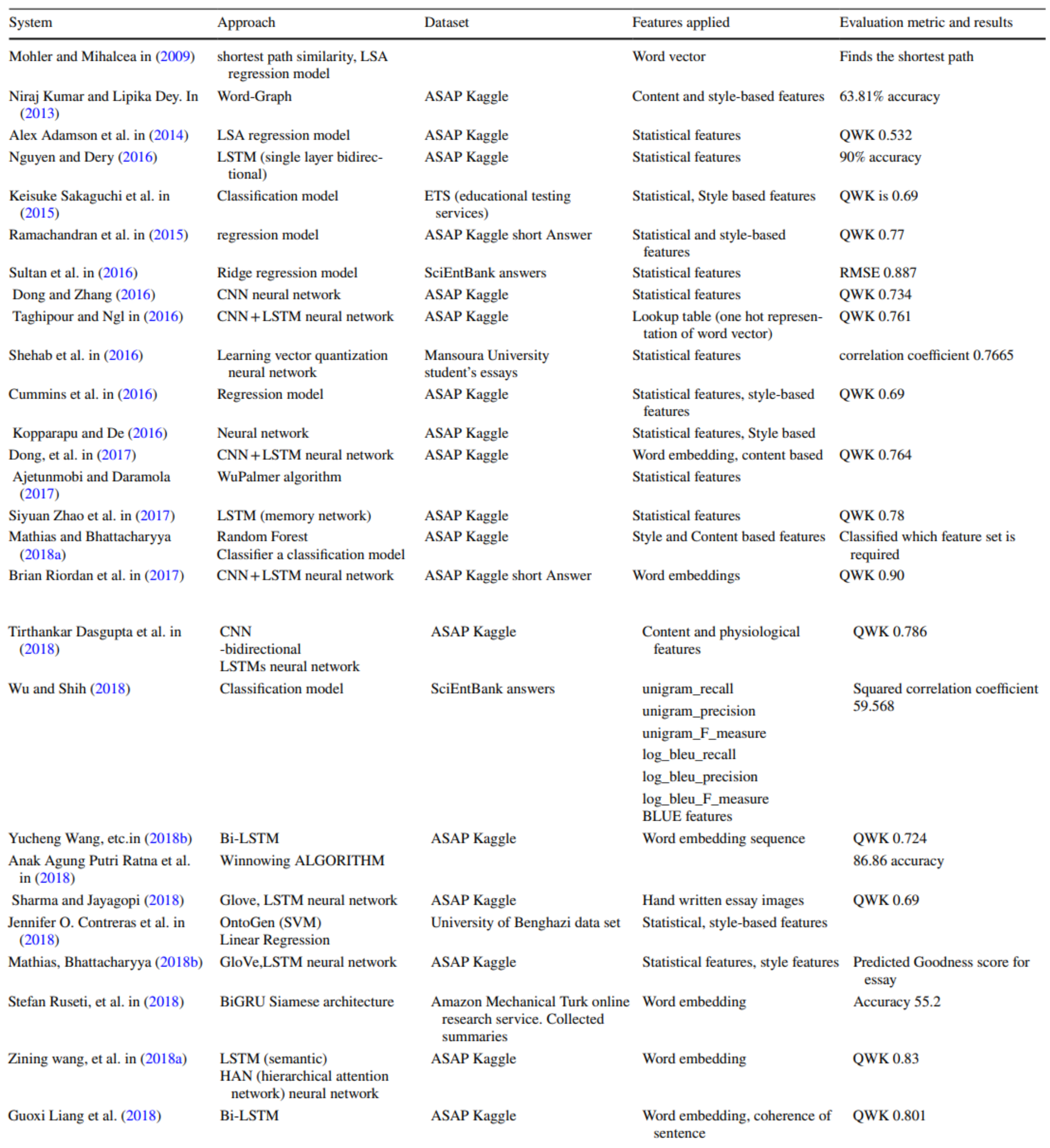
Essays are considered as one of the very important evaluation criteria used by teachers to evaluate student’s performance/Assessments. Manual Essay evaluation is a time consuming process, a teacher denotes a huge amount of time in evaluation of essays because of its subjectivity. Also because of subjective nature of the essay variation in grades usually occurs. Solution to such problem is automatic essay evaluation. Evaluating essays through computer will help reducing teachers load and time as well as reduce the variation in grades as a result of human factors.

The assessment plays a significant role in measuring the learning ability of the student. Most automated evaluation is available for multiple choice questions, but assessing short and essay answers remain a challenge. The education system is changing its shift to online-mode, like conducting computer-based exams and automatic evaluation. Here the problem is for a single question, we will get more responses from students with a different explanation. So, we need to evaluate all the answers concerning the question. Humans has various deviations so they are on not same mood every time sometimes they are happy or sad or angry etc it will effect on evaluation and grading of an essay.

It is a computer-based assessment system that automatically scores or grades the student responses by considering appropriate features. Its objective is to classify a large set of textual entities into a small number of discrete categories, corresponding to the possible grades, for example, the numbers 1 to 6. Automated scoring means using machines to evaluate things typically evaluated by people.

It takes much less time to score, ensuring that results and feedback can be provided instantly. This is especially important with university and college classes that are so large it would be almost impossible to give each student frequent, detailed, individualized feedback. AES grades each essay based on its own merits, and similar papers will receive the same grade. It is beneficial for teachers they no need correct papers manually it decreases the burden. Students no longer need to wait for their score.

2. LITERATURE SURVEY



System	Approach	Dataset	Features applied	Evaluation metric and results
Mohler and Mihalcea in (2009)	shortest path similarity, LSA regression model		Word vector	Finds the shortest path
Niraj Kumar and Lipika Dey. In (2013)	Word-Graph	ASAP Kaggle	Content and style-based features	63.81% accuracy
Alex Adamson et al. in (2014)	LSA regression model	ASAP Kaggle	Statistical features	QWK 0.532
Nguyen and Dery (2016)	LSTM (single layer bidirectional)	ASAP Kaggle	Statistical features	90% accuracy
Keisuke Sakaguchi et al. in (2015)	Classification model	ETS (educational testing services)	Statistical, Style based features	QWK is 0.69
Ramachandran et al. in (2015)	regression model	ASAP Kaggle short Answer	Statistical and style-based features	QWK 0.77
Sultan et al. in (2016)	Ridge regression model	SciEntBank answers	Statistical features	RMSE 0.887
Dong and Zhang (2016)	CNN neural network	ASAP Kaggle	Statistical features	QWK 0.734
Taghipour and Ngl in (2016)	CNN + LSTM neural network	ASAP Kaggle	Lookup table (one hot representation of word vector)	QWK 0.761
Shehab et al. in (2016)	Learning vector quantization neural network	Mansoura University student's essays	Statistical features	correlation coefficient 0.7665
Cummins et al. in (2016)	Regression model	ASAP Kaggle	Statistical features, style-based features	QWK 0.69
Kopparapu and De (2016)	Neural network	ASAP Kaggle	Statistical features, Style based	
Dong, et al. in (2017)	CNN + LSTM neural network	ASAP Kaggle	Word embedding, content based	QWK 0.764
Ajetunmobi and Daramola (2017)	WuPalmer algorithm		Statistical features	
Siyuan Zhao et al. in (2017)	LSTM (memory network)	ASAP Kaggle	Statistical features	QWK 0.78
Mathias and Bhattacharyya (2018a)	Random Forest Classifier a classification model	ASAP Kaggle	Style and Content based features	Classified which feature set is required
Brian Riordan et al. in (2017)	CNN + LSTM neural network	ASAP Kaggle short Answer	Word embeddings	QWK 0.90
Tirthankar Dasgupta et al. in (2018)	CNN -bidirectional LSTMs neural network	ASAP Kaggle	Content and physiological features	QWK 0.786
Wu and Shih (2018)	Classification model	SciEntBank answers	unigram_recall unigram_precision unigram_F_measure log_bleu_recall log_bleu_precision log_bleu_F_measure BLUE features	Squared correlation coefficient 59.568
Yucheng Wang, etc.in (2018b)	Bi-LSTM	ASAP Kaggle	Word embedding sequence	QWK 0.724
Anak Agung Putri Ratna et al. in (2018)	Winnowing ALGORITHM			86.86 accuracy
Sharma and Jayagopi (2018)	Glove, LSTM neural network	ASAP Kaggle	Hand written essay images	QWK 0.69
Jennifer O. Contreras et al. in (2018)	OntoGen (SVM) Linear Regression	University of Benghazi data set	Statistical, style-based features	
Mathias, Bhattacharyya (2018b)	GloVe,LSTM neural network	ASAP Kaggle	Statistical features, style features	Predicted Goodness score for essay
Stefan Ruseti, et al. in (2018)	BiGRU Siamese architecture	Amazon Mechanical Turk online research service. Collected summaries	Word embedding	Accuracy 55.2
Zining wang, et al. in (2018a)	LSTM (semantic) HAN (hierarchical attention network) neural network	ASAP Kaggle	Word embedding	QWK 0.83
Guoxi Liang et al. (2018)	Bi-LSTM	ASAP Kaggle	Word embedding, coherence of sentence	QWK 0.801

3. METHODOLOGY:

3.1 - REQUIREMENT SPECIFICATION (S/W & H/W) (DESIGN):

Hardware Requirements:

✓ System : Intel Core i3, i5, i7 and 2GHz Minimum

✓ RAM : 4GB or above

✓ Hard Disk : 10GB or above

✓ Input : Keyboard and Mouse

✓ Output : Monitor or PC

Software Requirements:

✓ OS : Windows 8 or Higher Versions

✓ Platform : Jupyter Notebook/Colaboratory

✓ Program Language : Python

- **3.2 Flowchart**

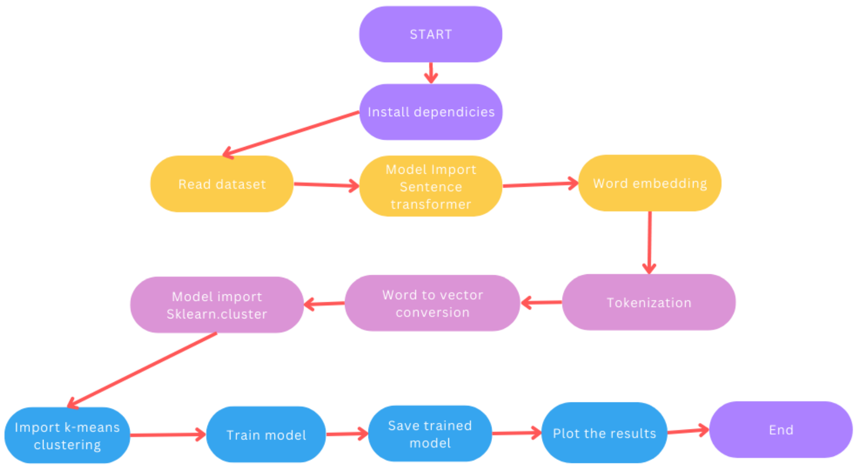

START
Install dependicies
Read dataset
Model Import Sentence transformer
Word embedding
Tokenization
Word to vector conversion
Model import Sklearn.cluster
Import k-means clustering
Train model
Save trained model
Plot the results
End

Figure:3.1 Workflow of the project

4. DATASET

We have used OS data-set for this project, the data set consists of 3 independent variables and 2 dependent feature they are as follows

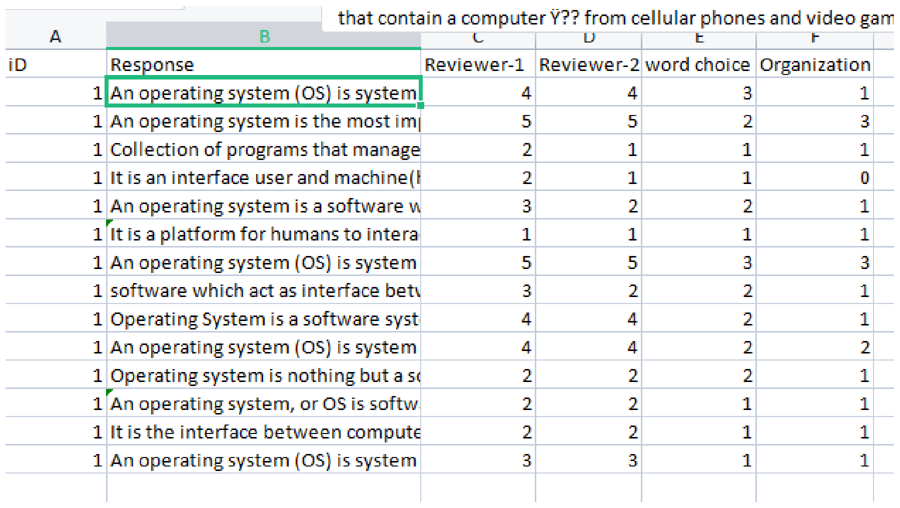
Response: Whole essay (sentences format)

Review 1: Score from first review

Review 2: Score from second review

Word choice: Score for type of words picked and vocab.

Organization: Score for the organization of the sentences



that contain a computer Ÿ?? from cellular phones and video gam

iD	Response	Reviewer-1	Reviewer-2	word choice	Organization
1	An operating system (OS) is system	4	4	3	1
1	An operating system is the most im	5	5	2	3
1	Collection of programs that manage	2	1	1	1
1	It is an interface user and machine(	2	1	1	0
1	An operating system is a software w	3	2	2	1
1	It is a platform for humans to intera	1	1	1	1
1	An operating system (OS) is system	5	5	3	3
1	software which act as interface bet	3	2	2	1
1	Operating System is a software syst	4	4	2	1
1	An operating system (OS) is system	4	4	2	2
1	Operating system is nothing but a s	2	2	2	1
1	An operating system, or OS is softw	2	2	1	1
1	It is the interface between compute	2	2	1	1
1	An operating system (OS) is system	3	3	1	1

DATA Visualization

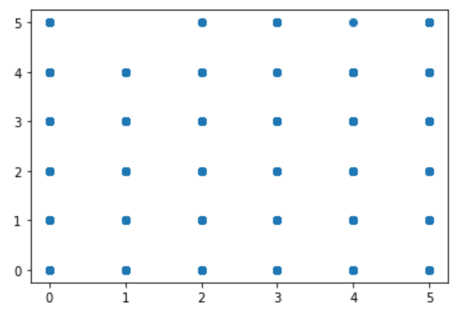


Fig:Scattering of reviewer 1 clusters

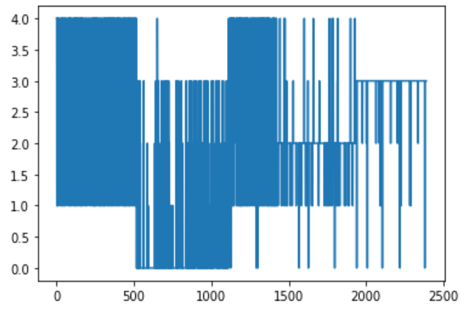


Fig: Cluster_assignment

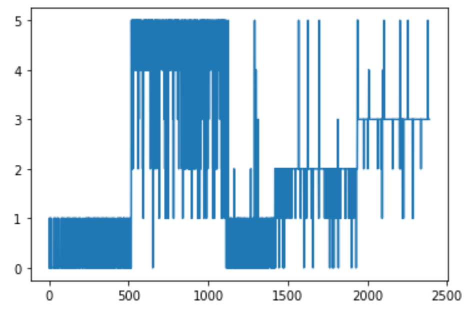


Fig:Clusters

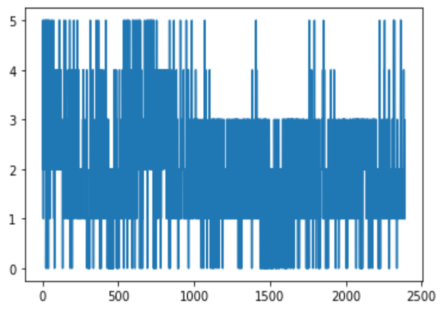


Fig: Reviewer 1

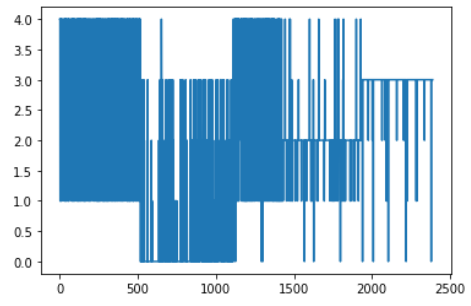


Fig:cluster assignment

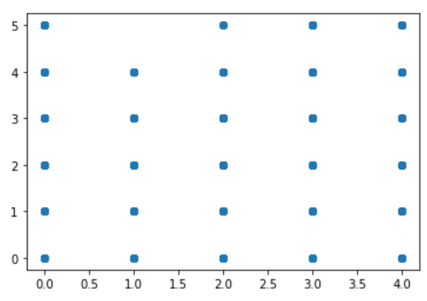


Fig:cluster assignment,list

5. DATA PRE-PROCESSING

We have used a transformer to complete this process where the main goal is to convert the data into vector, perform embedding on it and tokenise it.

The data set consists of 2391 * 6 rows and columns respectively. These are the steps taken for data Pre-processing.

Word Embedding:

Word embedding is a method for translating words to vectors of real numbers in language modelling. It represents words or sentences in a multidimensional vector space. Numerous techniques, including neural networks, co-occurrence matrices, probabilistic models, etc., can be used to create word embedding. Word embedding generation models are part of Word2Vec. With one input layer, one hidden layer, and one output layer, these models are shallow two-layer neural networks. Word2Vec uses two different architectures.

Tokenization:

Tokenization is the process of dividing text into a list of tokens from a string of text. Tokens can be viewed as components, similar to how a word functions as a token in a phrase and how a sentence functions as a token in a paragraph:

- Text into sentences tokenization
- Sentences into words tokenization
- Sentences using regular expressions tokenization

The initial stage in any NLP pipeline is tokenization. It significantly affects the remainder of your pipeline. Tokenization is the process of dividing unstructured data and natural language text into units of data that can be regarded as discrete pieces. A document's token occurrences can be directly used as a vector to represent that document. This instantly converts a text document or unstructured string into a numerical data format appropriate for machine learning. They can also be directly employed by a computer to initiate helpful answers and actions. Alternatively, they could be employed as features in a machine learning pipeline to initiate more complicated actions or behaviour.

6. MODEL

1. Transfer learning:

The Vanishing Gradient problem, which impairs long-term memory, affected RNN. RNN processes text in a sequential manner, therefore if a sentence is long like "XYZ visited France in 2019 during a time when there were no cases of cholera," it will be processed as "XYZ visited France in 2019 during a time when there were no cases of cholera." Now, if we inquire as to which location is meant by "that country" here? The fact that the country was "France" won't be remembered by RNN because it has already heard the term "France" many times. The model was trained at the word level, not at the sentence level, due to the sequential nature of processing. When the gradient shrinks, no true learning occurs because the gradients convey information used in the RNN parameter update.

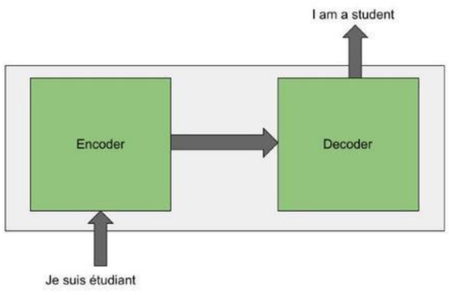


Fig.6.1 Architecture

The issue of long-term memory loss was partially handled by adding a few extra memory cells and addressing the vanishing gradients problem. However, because RNN was unable to process the entire sentence at once, the issue with sequential processing persisted. It processed words sequentially as opposed to in concurrently. Due to their sequential architecture, LSTMs cannot resolve this problem. We employ the static embedding strategy in LSTMs, which proposes that we embed a word into an n-dimensional vector without beforehand understanding its context. However, the meaning also changes if the context does.

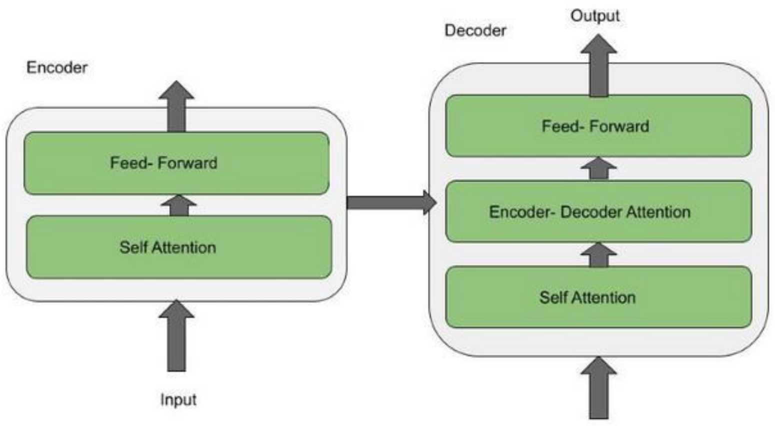


Fig.6.2Encoder and decoder

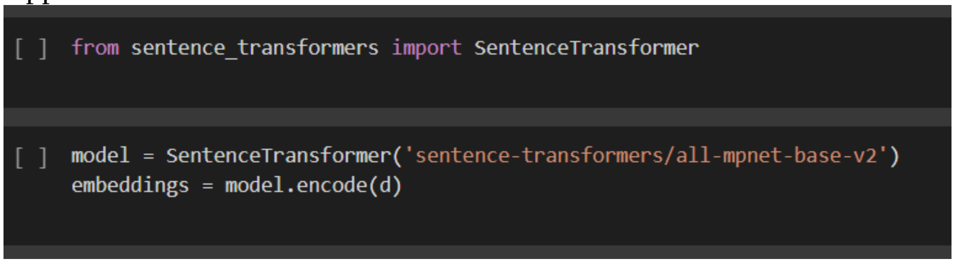
Self Attention and Feed Forward are the two layers of the encoder architecture. The outputs of the self-attention layer are supplied to a feed-forward neural network once the encoder's inputs have passed through it. Sequential data possesses temporal properties. It means that each word has a certain place in relation to the others. Take the line, "The cat didn't chase the mouse because it was not hungry," as an example. Here, it is clear that "it" refers to the cat, but it is more difficult to understand for an algorithm. Self-attention enables the model to connect the word "it" with the word "cat" when it is processing the word "it". Self-attention is the approach to reformulate the representation depending on all other words of the sentence.

The Self Attention, Encoder-Decoder Attention, and Feed Forward layers make up the decoder architecture. In addition to the self-attention and feed-forward layers found in the encoder, the decoder also features an attention layer that aids in focusing on key elements of the input sentence.

In the Transformer architecture, there are six layers of encoders and decoders. Word embeddings are carried out at the bottom encoder, where each word is converted into a 512-byte vector. The output of the encoder directly below would serve as the input to the other encoders. The encoder's many levels are used to identify the NLP pipeline. As an example, part of speech tags are employed in the first layer, constituents in the second, dependencies in the third, semantic roles in the fourth, coreference in the fifth, and relations in the sixth.

The very last layer, known as Softmax, assigns a probability to each word in the lexicon, and all of these probabilities add up to 1.

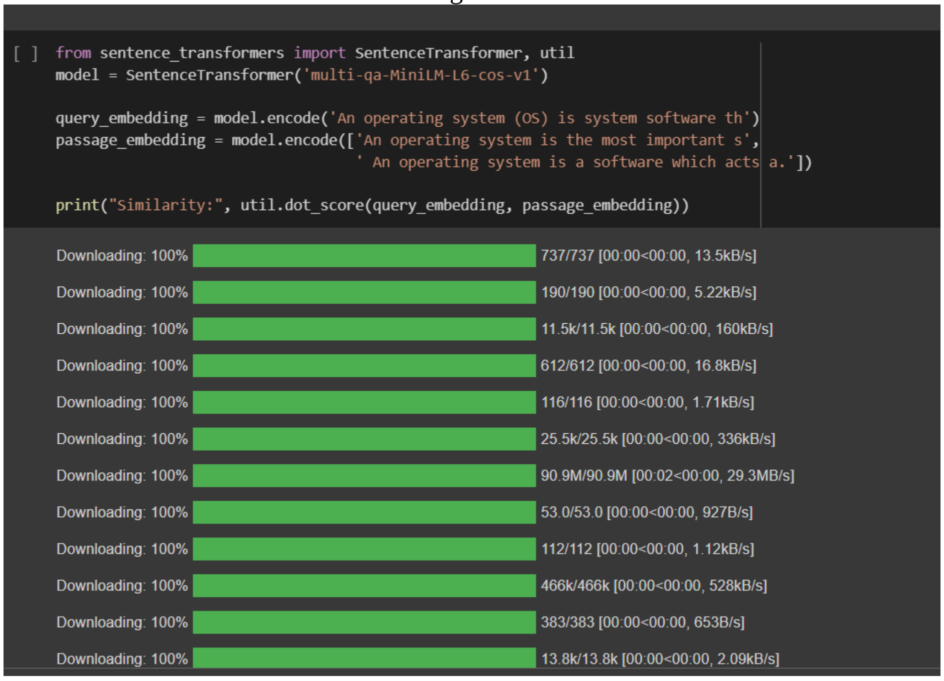
Code snippet:



```
[ ]  from sentence_transformers import SentenceTransformer
```

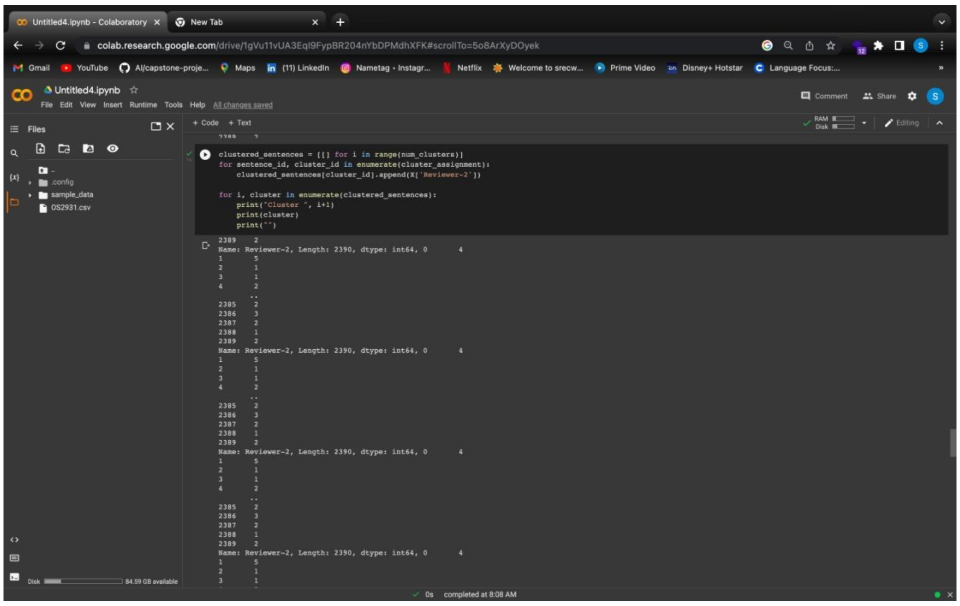
```
[ ]  model = SentenceTransformer('sentence-transformers/all-mpnet-base-v2')
     embeddings = model.encode(d)
```

Fig : 6.3



```
[ ]  from sentence_transformers import SentenceTransformer, util
     model = SentenceTransformer('multi-qa-MiniLM-L6-cos-v1')

     query_embedding = model.encode('An operating system (OS) is system software th')
     passage_embedding = model.encode(['An operating system is the most important s',
                                       ' An operating system is a software which acts a.'])

     print("Similarity:", util.dot_score(query_embedding, passage_embedding))
```

```
Downloading: 100% 737/737 [00:00<00:00, 13.5kB/s]
Downloading: 100% 190/190 [00:00<00:00, 5.22kB/s]
Downloading: 100% 11.5k/11.5k [00:00<00:00, 160kB/s]
Downloading: 100% 612/612 [00:00<00:00, 16.8kB/s]
Downloading: 100% 116/116 [00:00<00:00, 1.71kB/s]
Downloading: 100% 25.5k/25.5k [00:00<00:00, 336kB/s]
Downloading: 100% 90.9M/90.9M [00:02<00:00, 29.3MB/s]
Downloading: 100% 53.0/53.0 [00:00<00:00, 927B/s]
Downloading: 100% 112/112 [00:00<00:00, 1.12kB/s]
Downloading: 100% 466k/466k [00:00<00:00, 528kB/s]
Downloading: 100% 383/383 [00:00<00:00, 653B/s]
Downloading: 100% 13.8k/13.8k [00:00<00:00, 2.09kB/s]
```



colab.research.google.com/drive/1gVu11vUA3EqI9FypBR204nYbDPMdhXFK#scrollTo=5o8ArXyDOyek

Untitled4.ipynb

```
clustered_sentences = [[] for i in range(num_clusters)]
for sentence_id, cluster_id in enumerate(cluster_assignment):
    clustered_sentences[cluster_id].append(X['Reviewer-2'])

for i, cluster in enumerate(clustered_sentences):
    print("Cluster ", i+1)
    print(cluster)
    print("")
```

```
2389    2
Name: Reviewer-2, Length: 2390, dtype: int64, 0       4
1       5
2       1
3       1
4       2
       ..
2385    2
2386    3
2387    2
2388    1
2389    2
Name: Reviewer-2, Length: 2390, dtype: int64, 0       4
1       5
2       1
3       1
4       2
       ..
2385    2
2386    3
2387    2
2388    1
2389    2
Name: Reviewer-2, Length: 2390, dtype: int64, 0       4
1       5
2       1
3       1
4       2
       ..
2385    2
2386    3
2387    2
2388    1
2389    2
Name: Reviewer-2, Length: 2390, dtype: int64, 0       4
1       5
2       1
3       1
```

0s completed at 8:08 AM

Fig:6.4 :Code snippets

2. K-Means Clustering

(Imagining objects as points in an n-dimensional space will assist.) The items will be divided up into k groups or clusters of resemblance by the algorithm. We will use the euclidean distance as a unit of measurement to calculate that similarity.

This is how the algorithm operates:

First, we randomly initialise k locations, also known as cluster centroids or means. Each item is categorised according to the nearest mean, and the coordinates of that mean, which are the averages of the items categorised in that cluster thus far, are updated.
After a specific amount of iterations, we repeat the process until we get our clusters.

Since the things described in the "points" mentioned above have mean values, they are known as means. We can initialise these means in a variety of ways. Initializing the means at random elements in the data set is a natural approach. Another approach is to put the means' beginning values at arbitrary ranges between the data set's boundaries (if for a feature x the items have values in [0,3], we will initialise the means with values for x at [0,3]).

<u>Classify Items:</u>

Now we need to create a function to group or cluster an item. We will compare the similarity of the provided item to each mean before classifying it with the closest one.

<u>Find means:</u>

We will loop through every item to classify them into the closest cluster and update the cluster's mean before actually determining the means. The procedure will be repeated a certain number of times. If no item's classification changes during the course of two rounds, the procedure is terminated because the algorithm has found the best course of action.
The function below accepts as inputs the items, the maximum number of iterations, and k (the number of desired clusters), and outputs the means and the clusters. An item's classification is kept in the array belongs To, while the size of a cluster is kept in cluster-sizes.

<u>Find clusters:</u>

Finally, given the means, we wish to identify the clusters. Each item will be categorized into the nearest cluster when we have gone through all of the objects.

The other widely used similarity metrics include:

1. Cosine distance: It establishes the angle's cosine between two locations in an n-dimensional space using the formula d = frac

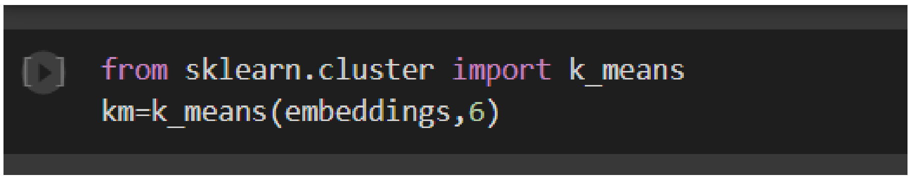
{X.Y}{||X||*||Y||}

2. Manhattan distance: The total of the absolute differences between the coordinates of the two data points is computed.

d = "sum n" (X i-Y i)

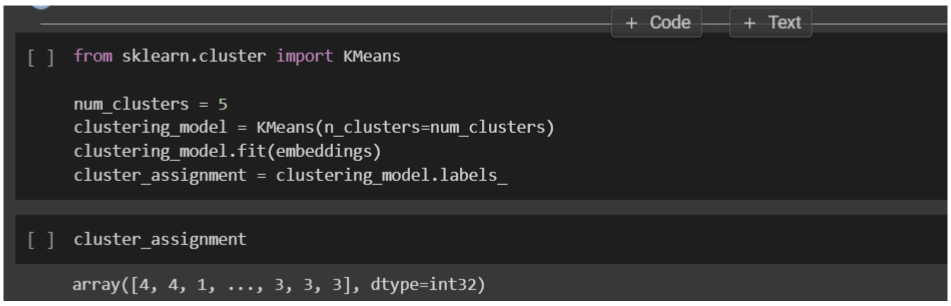
3. The generalised distance metric is another name for the Minkowski distance. Both ordinal and quantitative variables may be used with it.

d = (sum _n|X i-Y i|frac1p)p



```
from sklearn.cluster import k_means
km=k_means(embeddings,6)
```

Fig.6.5

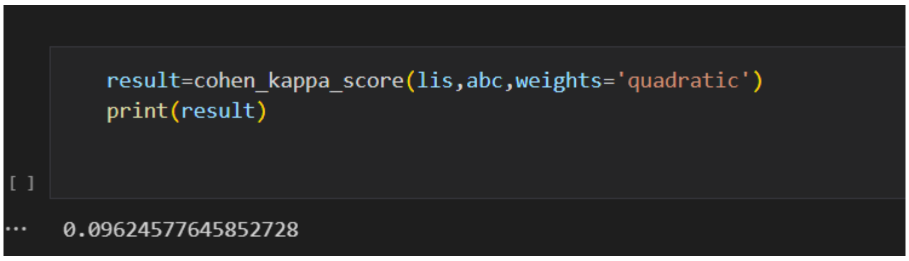


```
[ ]  from sklearn.cluster import KMeans

     num_clusters = 5
     clustering_model = KMeans(n_clusters=num_clusters)
     clustering_model.fit(embeddings)
     cluster_assignment = clustering_model.labels_

[ ]  cluster_assignment

     array([4, 4, 1, ..., 3, 3, 3], dtype=int32)
```

Fig 6.6

7.Results

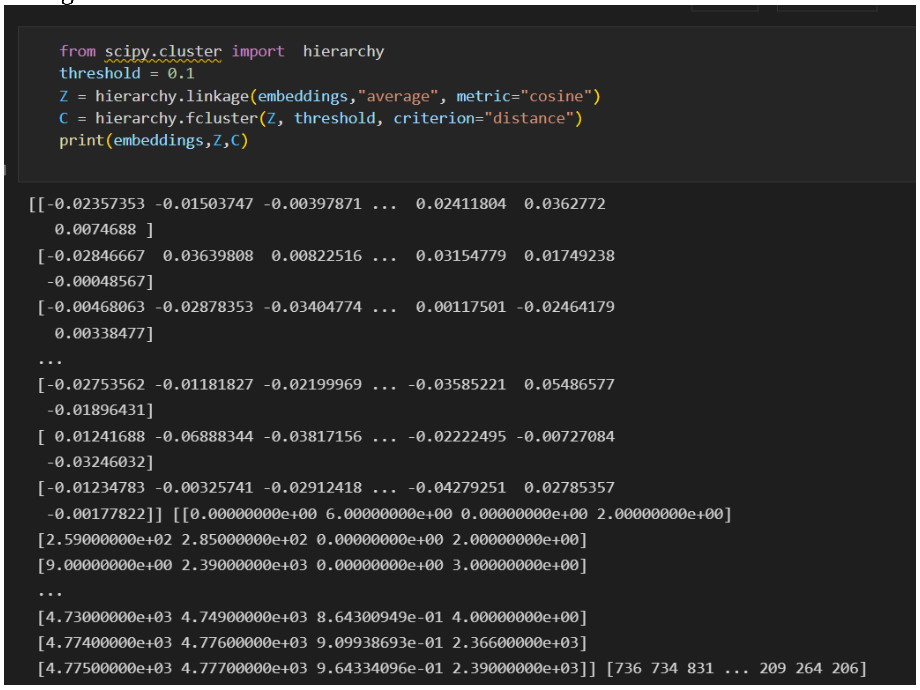
7.1: Kappa score:



```
result=cohen_kappa_score(lis,abc,weights='quadratic')
print(result)
```

```
0.09624577645852728
```

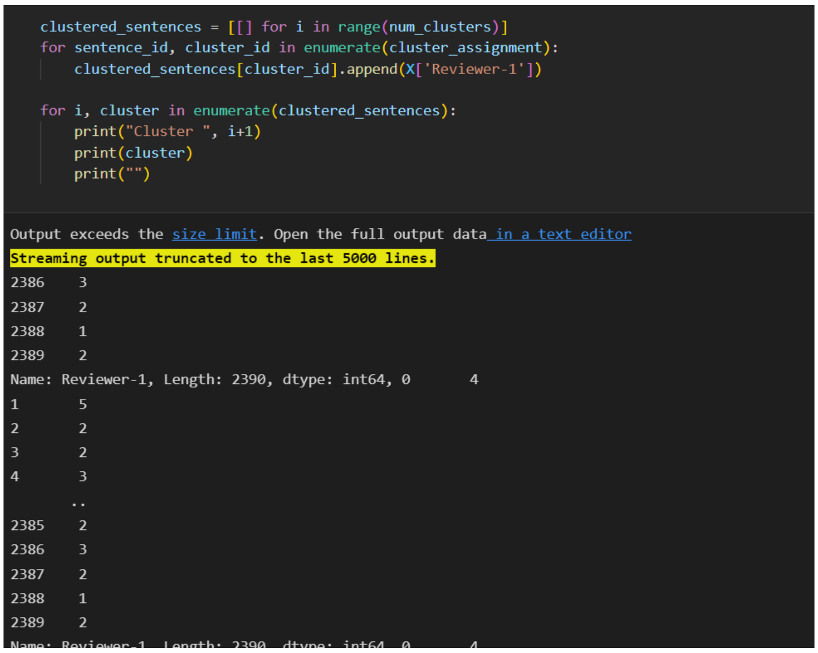
7.2: Embeddings



```
from scipy.cluster import  hierarchy
threshold = 0.1
Z = hierarchy.linkage(embeddings,"average", metric="cosine")
C = hierarchy.fcluster(Z, threshold, criterion="distance")
print(embeddings,Z,C)
```

```
[[-0.02357353 -0.01503747 -0.00397871 ...  0.02411804  0.0362772
   0.0074688 ]
 [-0.02846667  0.03639808  0.00822516 ...  0.03154779  0.01749238
  -0.00048567]
 [-0.00468063 -0.02878353 -0.03404774 ...  0.00117501 -0.02464179
   0.00338477]
 ...
 [-0.02753562 -0.01181827 -0.02199969 ... -0.03585221  0.05486577
  -0.01896431]
 [ 0.01241688 -0.06888344 -0.03817156 ... -0.02222495 -0.00727084
  -0.03246032]
 [-0.01234783 -0.00325741 -0.02912418 ... -0.04279251  0.02785357
  -0.00177822]] [[0.00000000e+00 6.00000000e+00 0.00000000e+00 2.00000000e+00]
 [2.59000000e+02 2.85000000e+02 0.00000000e+00 2.00000000e+00]
 [9.00000000e+00 2.39000000e+03 0.00000000e+00 3.00000000e+00]
 ...
 [4.73000000e+03 4.74900000e+03 8.64300949e-01 4.00000000e+00]
 [4.77400000e+03 4.77600000e+03 9.09938693e-01 2.36600000e+03]
 [4.77500000e+03 4.77700000e+03 9.64334096e-01 2.39000000e+03]] [736 734 831 ... 209 264 206]
```

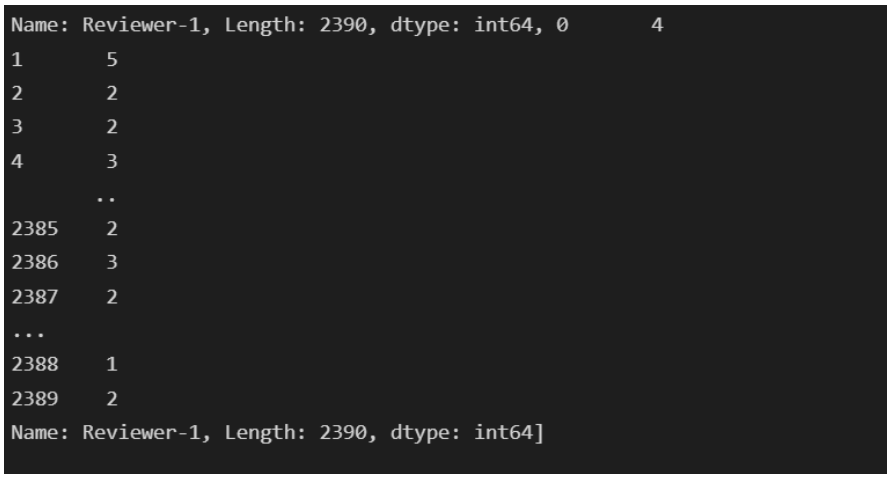
7.3 Clustered sentence



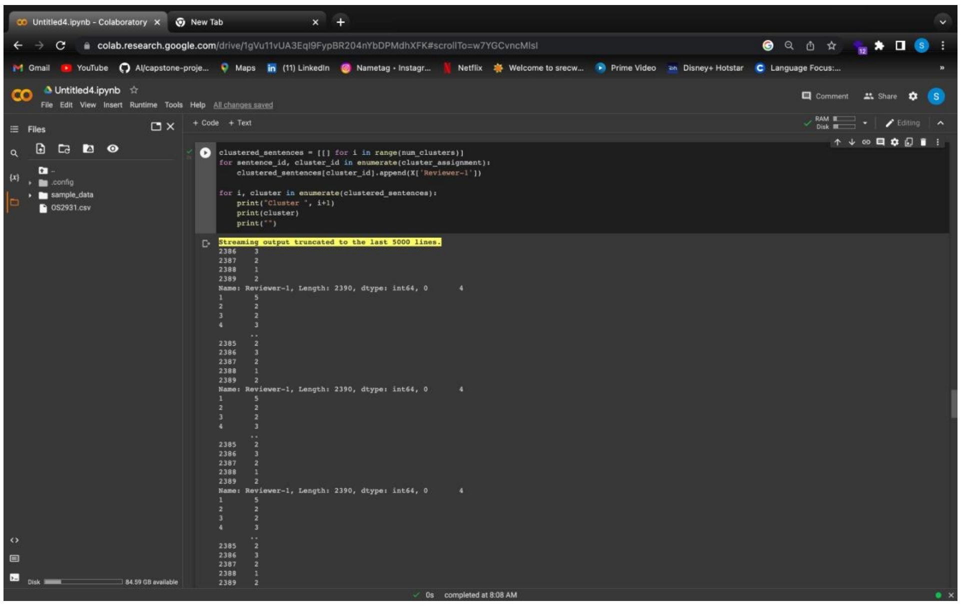
```
clustered_sentences = [[] for i in range(num_clusters)]
for sentence_id, cluster_id in enumerate(cluster_assignment):
    clustered_sentences[cluster_id].append(X['Reviewer-1'])

for i, cluster in enumerate(clustered_sentences):
    print("Cluster ", i+1)
    print(cluster)
    print("")
```

```
Output exceeds the size limit. Open the full output data in a text editor
Streaming output truncated to the last 5000 lines.
2386    3
2387    2
2388    1
2389    2
Name: Reviewer-1, Length: 2390, dtype: int64, 0       4
1       5
2       2
3       2
4       3
       ..
2385    2
2386    3
2387    2
2388    1
2389    2
```

7.4:



```
Name: Reviewer-1, Length: 2390, dtype: int64, 0       4
1       5
2       2
3       2
4       3
       ..
2385    2
2386    3
2387    2
...
2388    1
2389    2
Name: Reviewer-1, Length: 2390, dtype: int64]
```



```
clustered_sentences = [[] for i in range(num_clusters)]
for sentence_id, cluster_id in enumerate(cluster_assignment):
    clustered_sentences[cluster_id].append(X['Reviewer-1'])

for i, cluster in enumerate(clustered_sentences):
    print("Cluster ", i+1)
    print(cluster)
    print("")
```

```
Streaming output truncated to the last 5000 lines.
2386    3
2387    2
2388    1
2389    2
Name: Reviewer-1, Length: 2390, dtype: int64, 0       4
1       5
2       2
3       2
4       3
       ..
2385    2
2386    3
2387    2
2388    1
2389    2
Name: Reviewer-1, Length: 2390, dtype: int64, 0       4
1       5
2       2
3       2
4       3
       ..
2385    2
2386    3
2387    2
2388    1
2389    2
Name: Reviewer-1, Length: 2390, dtype: int64, 0       4
1       5
2       2
3       2
4       3
       ..
2385    2
2386    3
2387    2
2388    1
2389    2
```

8. CONCLUSION

Essays are collections of sentences and paragraphs that are useful to analyze the writing, communication, and grammatical skills of users or applicants. Essays became a standard evaluation criterion in several fields like secondary education, academics, software recruitment's etc. As there are huge number of applicants or participants, it’s a hurdle for human evaluators to assess each essay and score it. It will kill huge amount of time and delay the process.

We have created a new way of analyzing the essay and scoring them based on clustering which helps the data to be easily be classifies into categories and the kind of scoring being offered will be easily understood based on the group of answers and their similarity.

The clustering through K-Means clustering helps the categorization but the sentence transformer takes the embedding very seriously and helps in easy access of the data to to be converted into vectors and helps in tokenization too.

This model has a deep understanding of sentences and the similarities between the sentences, hence more useful to create vectors with perfect meaning. We are trying to reduce the burden of essay evaluators and make the work automated. This project will doesn’t show any bias in generating the score.

REFERANCES

1. Sharma A., & Jayagopi D. B. (2018). Automated Grading of Handwritten Essays 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2018, pp 279–284. https://doi.org/10.1109/ICFHR-2018.2018.00056

2. Agung Putri Ratna, A., Lalita Luhurkinanti, D., Ibrahim I., Husna D., Dewi Purnamasari P. (2018). Auto- matic Essay Grading System for Japanese Language Examination Using Winnowing Algorithm, 2018 International Seminar on Application for Technology of Information and Communication, 2018, pp. 565–569. https://doi.org/10.1109/ISEMANTIC.2018.8549789.

3. Wu, S. H., & Shih, W. F. (2018, July). A short answer grading system in chinese by support vector approach.In Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications (pp. 125-129).

4. Kumar, N., & Dey, L. (2013, November). Automatic Quality Assessment of documents with application to essay grading. In 2013 12th Mexican International Conference on Artificial Intelligence (pp. 216– 222). IEEE.

5. Zhu W, Sun Y (2020) Automated essay scoring system using multi-model Machine Learning, david c. wyldet al. (eds): mlnlp, bdiot, itccma, csity, dtmn, aifz, sigpro

6. Tashu TM, Horváth T (2020) Semantic-Based Feedback Recommendation for Automatic Essay Evaluation.In: Bi Y, Bhatia R, Kapoor S (eds) Intelligent Systems and Applications. IntelliSys 2019. Advances inIntelligent Systems and Computing, vol 1038. Springer, Cham

7. Tashu TM, Horváth T (2019) A layered approach to automatic essay evaluation using word-embedding. In: McLaren B, Reilly R, Zvacek S, Uhomoibhi J (eds) Computer Supported Education. CSEDU 2018. Communications in Computer and Information Science, vol 1022. Springer, Cham

8. Tashu TM (2020) "Off-Topic Essay Detection Using C-BGRU Siamese. In: 2020 IEEE 14th International Conference on Semantic Computing (ICSC), San Diego, CA, USA, p 221–225, doi: https://doi.org/ 10.1109/ICSC.2020.00046

9. Rodriguez P, Jafari A, Ormerod CM (2019) Language models and Automated Essay Scoring. ArXiv, abs/1909.09482

10. Parekh S, et al (2020) My Teacher Thinks the World Is Flat! Interpreting Automatic Essay Scoring Mechanism.” ArXiv abs/2012.13872 (2020): n. pag

.