

# **AUTOMATED ESSAY SCORING USING CLUSTERING**



A Capstone project report  
in partial fulfillment of the requirement for the award of the degree

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE & ENGINEERING**

**and**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**By**

**AMOGH VARSH RAJU AMBATI**

**19K41A0590**

**SAIKIRAN ANUGAM**

**19K41A0592**

**ASMATH FATHIMA**

**19K41A04F3**

Under the guidance of

**D. Ramesh**

Assistant Professor, Department of CSE.



**SR**  
**Engineering**  
**College**  
Innovation . Creativity . Entrepreneurship

# SR ENGINEERING COLLEGE

Ananthasagar, Warangal.



## CERTIFICATE

This is to certify that this project entitled “**AUTOMATED ESSAY SCORING USING CLUSTERING**” is the bonafide work carried out by **Amogh Varsh Raju Ambati, Saikiran Anugam, Asmath Fathima**, bearing Roll No(s) 19K41A0590, 19K41A0592, 19K41A04F3 as a Capstone project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING /Electrical & Electronics Engineering** during the academic year 2022-2023 under our guidance and Supervision.

**Mr. D. RAMESH**

Asst. Professor,  
S R Engineering College,  
Ananthasagar, Warangal

**Dr.M. Sheshikala**

Assoc. Prof. & HOD (CSE),  
S R Engineering College,  
Ananthasagar, Warangal

**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

<b>S.NO</b>	<b>Content</b>	<b>Page No</b>
1	Introduction	6
2	Literature Review	11
3	Design	14
4	Data set	16
5	Pre-processing	18
6	Methodology	19
7	Results	26
8	Conclusion	29
9	References	30

# INTRODUCTION

## 1.1 OVERVIEW

The content which a writer gives in an essay shows the perfection and dedication to the reviewer. Effective writing is readable which should be clear, concise and accurate so, that the people understand them effortlessly. And in some countries, (Ex: India, the United States, and Canada, etc.), essays have become a major part of formal education and entry level test. Secondary students have been taught structured essay formats to improve their writing and ability skills. Even for admissions, essays are used by universities and corporate industries to shortlist and filter the applicants. But evaluating these essays is a tedious and time-consuming task for a human being. Since essays consist of lots of paragraphs and pages (depending on requirement) and evaluating these essays will be a challenge for human beings. Even the number of participants or applicants will be huge in number. Evaluating all these applicants' essays requires a huge number of evaluators and time, around 1-2 months manually. Which will delay the admission process or recruiting process. Solution to such problem is automatic essay grading system. An essay is often a piece of writing in which the author presents their own ideas. It could be the author's expression of recollections, observations of daily life, or reflections. It displays the unique perspective of the author. Essays are used to evaluate a user's conceptual understanding and ability to clearly

express their points of view. Essays are frequently written in a discursive style, fusing ideas, cases, and justifications to address a particular problem or query. The main components of an effective essay are the writer's point of view, an analysis of other writers' viewpoints, and research. The reader is most positively impacted by an essay's straightforward structure, which consists of a brief introduction, body paragraphs, and a conclusion with a reference list.

Essay evaluation software will eliminate human labour while precisely forecasting the writer's abilities. As a result, the diversity in grades brought on by human variables has decreased. In the distant past, a number of systems such as Project Essay Grader (PEG), for example were put forth to forecast the best grade in comparison to manual grading. We have a number of systems, some of which have issues that have been proposed and some of which are still being developed. One of the concerns was the scoring system, which gave lengthier essays better ratings. Additionally, the use of transitional phrases was overemphasized, which raised essay scores. These systems disregard the evolution and coherence of the content. One could get a high score by utilizing those systems.

In our study, we used a data-set that included human rater scores so that we could compare the results and observe how much evaluation time was saved. Our model predicts the best score by learning the meaning of the words, even if we are just utilizing a small data-set to predict scores in accordance with the training set.

## 1.2 EXISTING SYSTEM:

Computers' influence on writing have been extensively studied for a sizable amount of time. Computers can serve as a more useful cognitive tool, according to the research on AES. The PC innovation known as AES is said to evaluate and score composed composition. The creative process necessitates revision and critique. For understudies to improve their writing, they need feedback. However, responding to understudy papers can be a burden for teachers. Giving each candidate their own input on their writings may prove to be a time-consuming and laborious work for the evaluators, especially if there are a lot of applications and regular writing tasks. AES frameworks can be quite beneficial because they can Give the understudy feedback and a score as soon as possible. Technology in development is AES. In order to address the problems of generalizability, cost, and time in preparing evaluations, many AES methods are utilized. The pursuit of excellence in paper scoring by machines is ongoing, and numerous studies are being conducted to assess how adequate the AES frameworks are. Our study shows that a lot of AES programs just employ a few machine learning or deep learning algorithms to forecast the score. However, the accuracy is low and it takes a while to forecast the score because of the limitations of models used.

Recurrent neural networks are used in many different applications, including language translation, speech recognition, time series analysis, etc. They are well described for processing sequential input. In one of the models, the RNN algorithms include a GRU (gate recurrent unit). Regression is used in the e-



rating rater's system. Other ones, such support vector regression, employ conventional regression procedures. Additionally, classification algorithms like the K-nearest neighbour (KNN) and the naive Bayesian mode are employed for AES.

However, because so few models are applying machine learning methods, they are unable to assess the interior views and are therefore unable to anticipate precisely. Using subpar models to translate text to vector representation is another factor. These are examined and resolved in the system we suggest.

### **1.3 PRESENT WORK**

**In** the proposed system we are using Transfer Learning Algorithm to convert the essay into vectors and to perform the embedding operation and then we are using the particular model is that it also helps us to perform tokenization the parts for easy part of the clustering and then for model training and results part K-Means clustering model is being implemented. It will use the human scores present the data-set as the labels and learn the relationship between the essays and the scores and get ready to predict. The following figure illustrates the overview of the system:

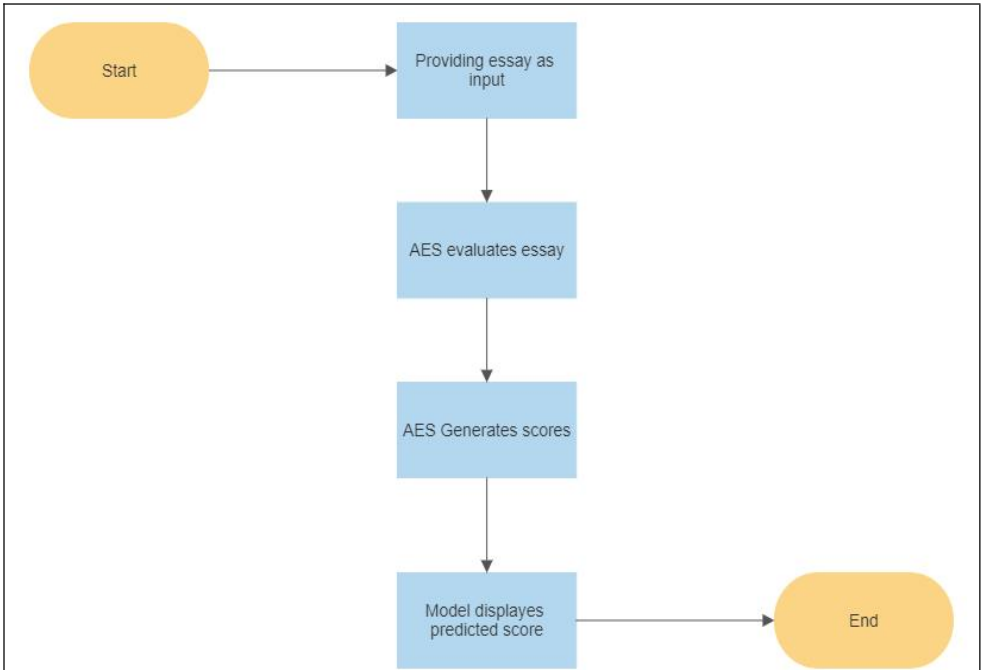


Fig: 1.1 Overview of proposed system

## 2.Literature Review

To assess test takers' responses to essay questions, an automated essay scoring system was created. Many approaches and frameworks for grading expositions have been developed throughout the course of the last many years. The most well-known models found for AES frameworks rely on Latent Semantic Analysis (LSA), Natural Language Processing (NLP), Bayesian text organization, or Neural Networks.

An automated essay scoring system was developed to evaluate test takers' replies to essay questions. Over the past many years, a variety of methods and frameworks for evaluating expositions have been established. The most popular models for AES frameworks

include Bayesian text organisation, Latent Semantic Analysis (LSA), Natural Language Processing (NLP), or Neural Networks.

The vector in BoW contains the frequency of words that appear in the essay. Based on the occurrences of words in the essay, the vector signifies 1 or more and 0 for absence. Therefore, in BoW, the vector only applies to single words and does not retain the link with adjacent words. In word2vec, the vector depicts the multidimensional relationships between words and other words as well as the prompts for phrases. Word2vec is unable to locate semantic vectors when a word has more than one meaning and those meanings depend on one another because it prepares vectors in a unidirectional, rather than a bidirectional, manner. A comparison of machine learning models and feature extraction techniques is shown in Table 2.

<b>AES System</b>	<b>Approach</b>	<b>Data-set</b>	<b>Features applied</b>	<b>Evaluation Metrics &amp; results</b>
<b>Pedro Uria Rodriguez et al. (2019)</b>	BERT, Xlnet	ASAP Kaggle	Error correction.	QWK 0.755
<b>Jiawei Liu et al. (2019)</b>	CNN, LSTM, BERT	ASAP Kaggle	semanticdata, handcrafted features like grammar correction,essay length etc	QWK 0.709
<b>Darwish and Mohamed (2020)</b>	Multiple Linear Regression	ASAP Kaggle	Style and content-based features	QWK 0.77
<b>Jiaqi Lun et al. (2020)</b>	BERT	SemEval-2013	Student Answer, R	Accuracy 0.8277 (2-way)
<b>Süzen, Neslihan, et al. (2020)</b>	Text mining	Introduction to computer science in UNT, Assignments	Sentence similarity	Correlation score 0.81
<b>Wilson Zhu and Yu Sun in (2020)</b>	RNN (LSTM, Bi-LSTM)	ASAP Kaggle	Word embedding, grammarcount, word coun	QWK 0.70
<b>Salim Yafet et al. (2019)</b>	XGBoost machine learning	ASAP Kaggle	Word Count,POS, parse tree,coherence,cohesion,type token ration	Accuracy 68.12
<b>Andrzej Cader (2020)</b>	Deep Neural Network	University of Social Sciences in	asynchronous feature	Accuracy 0.99
<b>Tashu TM,</b>	Rule based	ASAP Kaggle	Similarity based	Accuracy 0.68

<b>Horváth T (2019)</b>	algorithm, Similarity based algorithm		feature applied	
<b>Masaki Uto(B) and Masashi Okano (2020)</b>	Item Response Theory Models (CNN- LSTM,BER T)	ASAP Kaggl		ASAP Kaggl

Table 2.1: State of the art

## **3.Design**

### **3.1 REQUIREMENT SPECIFICATION(S/W & H/W)**

#### **Hardware Requirements**

- ✓ **System** : Pentium 4, 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** : Jupiter Notebook
- ✓ **Program Language** : Python

### 3.2 Flow chart

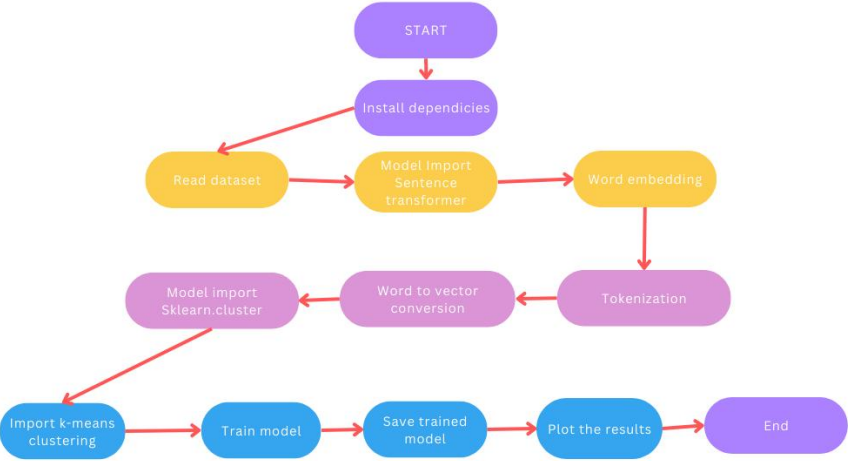


Figure:3.1 Workflow of the project

## 4. DATA-SET

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 ??? from cellular phones and video gam			
A	B	C	D	E	F
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(f	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 betw	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 sc	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**

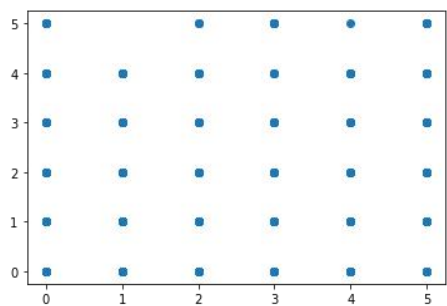


Fig:Scattering of reviewer 1 clusters

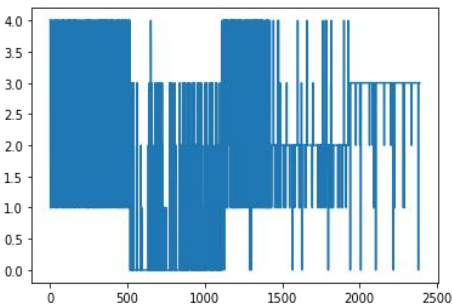


Fig: Cluster\_assignment

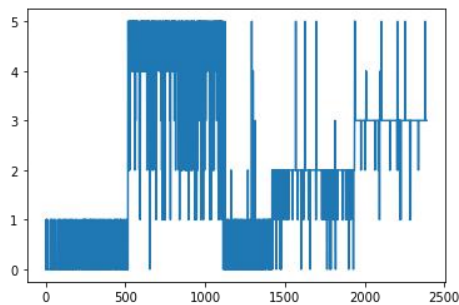


Fig:Clusters

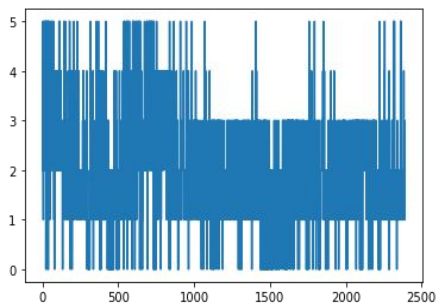


Fig: Reviewer 1

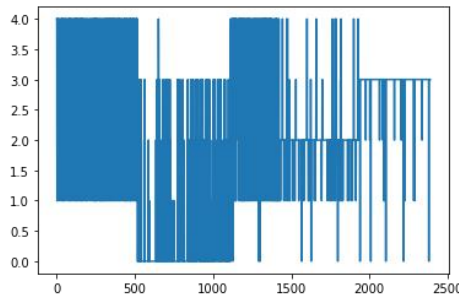


Fig:cluster assignment

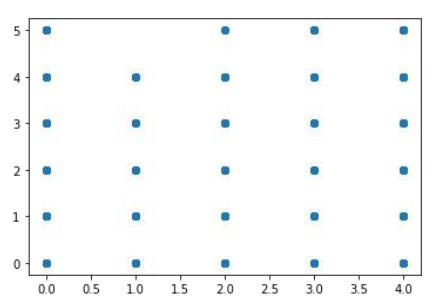


Fig:cluster assignment,list

## 5. 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

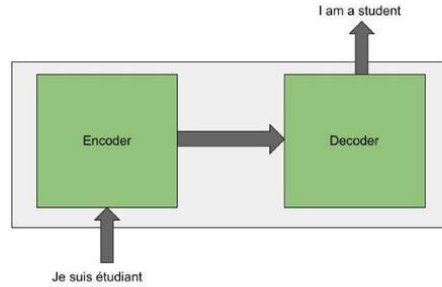
The initial stage in any NLP pipeline is tokenization. It significantly affects the remainder of your pipeline. Tokenization is the process of dividing unstructured data and natural language text into units of data that can be regarded as discrete pieces. A document's token occurrences can be directly 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. Methodology**

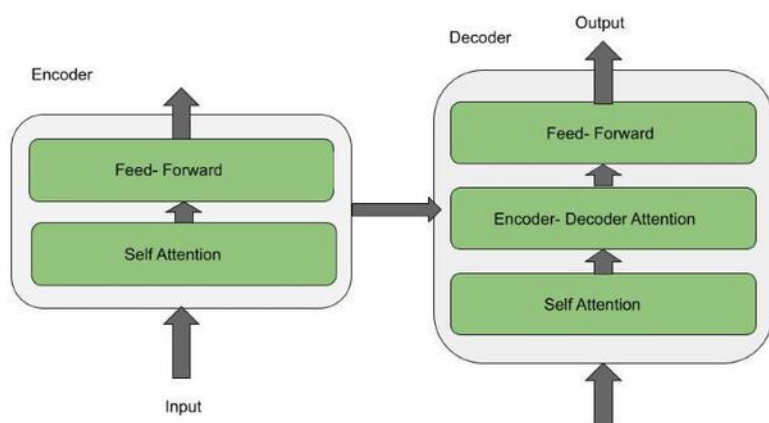
### **1. Transfer learning:**

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.6.2 Encoder 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.

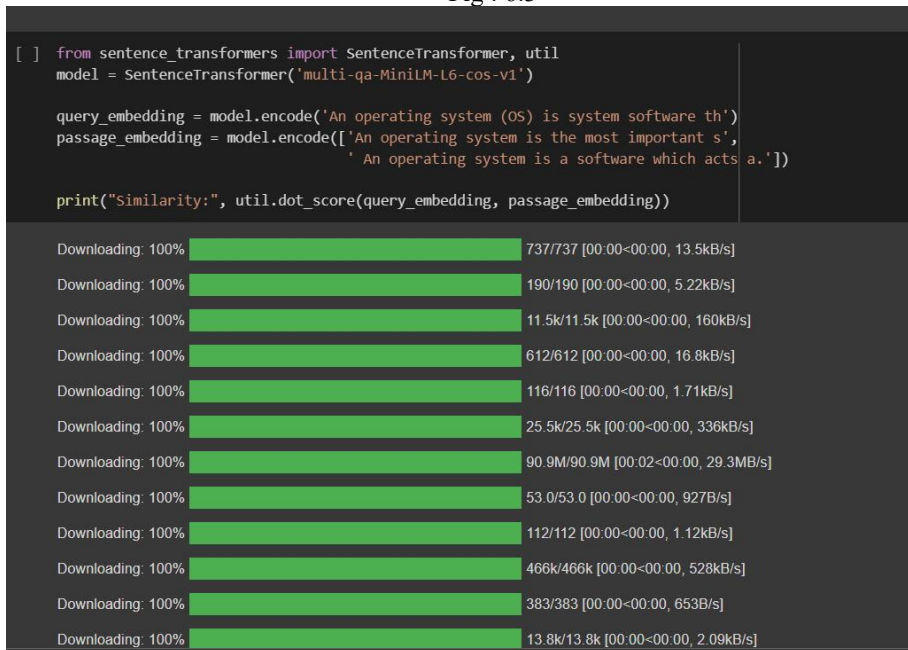
The very last layer, known as Softmax, assigns a probability to each word in the lexicon, and all of these probabilities add up to 1.

Code snippet:

```
[ ] from sentence_transformers import SentenceTransformer

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

Fig : 6.3



The screenshot shows a Google Colab notebook titled 'Untitled4.ipynb'. The code in the cell is as follows:

```

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

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

```

The output of the code is displayed below the code cell. It shows the contents of four clusters, each labeled 'Cluster ' followed by its index (1, 2, 3, 4). Each cluster contains a list of sentence IDs and their corresponding feature vectors (represented as lists of numbers). For example, Cluster 1 contains sentence IDs 2389, 1, 2, 3, 4 and their respective feature vectors.

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  $[0,3]$ ).

### **Classify Items:**

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.

### **Find means:**

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`.

### **Find clusters:**

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  $d = \frac{\langle \mathbf{X}, \mathbf{Y} \rangle}{\|\mathbf{X}\| * \|\mathbf{Y}\|}$

$$\frac{\langle \mathbf{X}, \mathbf{Y} \rangle}{\|\mathbf{X}\| * \|\mathbf{Y}\|}$$

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.

$$d = \left( \sum_n |X_i - Y_i|^p \right)^{\frac{1}{p}}$$

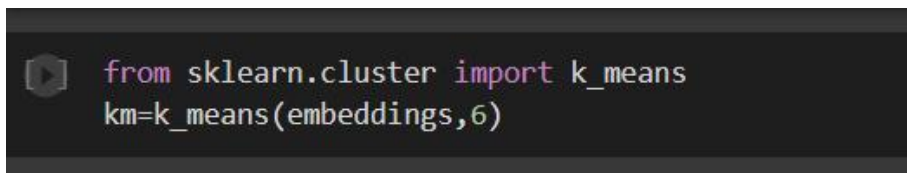
```
A dark-themed code block with a play button icon on the left. It contains two lines of Python code: 'from sklearn.cluster import k_means' and 'km=k_means(embeddings,6)'.  
from sklearn.cluster import k_means  
km=k_means(embeddings,6)
```

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
Name: Reviewer-1, Length: 2390, dtype: int64, 0      4
```

## 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]
```



## **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.

## 9. References

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. wyld et al. (eds): mlnlp, bdiot, itecma, 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 in Intelligent 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

