International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249 – 8958 (Online), Volume-9 Issue-5, June 2020

Automated Essay Grading System using NLP Techniques

Pramit Shetty, Kaushal Yadav, Prithvi Kunder

Abstract*: The purpose of the study is to develop an automated essay grading system (AES) which can grade students essays based on various factors. Our proposed system performs grading of essays based on two features. Simple features consist of finding syntactic errors such as spelling mistakes, grammatical errors, punctuations and sentence proportions. Complex features consist of finding semantic errors through discourse analysis, thematic analysis and detection of undesirable style of writing. Many existing AES systems fail to consider the semantic parts of the essay which is addressed in this study. Calculation of score would be done based on what is specified in rubrics. The proposed system is evaluated using datasets from kaggle. The accuracy of model and obtained results show an agreement with teachers' grading. This gives us an indication that the model can be deployed for assessment of students' essay, thereby leading to reduction in time, efforts and cost for evaluating an essay.

***Keywords*: discourse analysis, kappa scores, LSTM, thematic analysis**

I. INTRODUCTION

Automated Essay Grading (AES) is a technology where computers can evaluate written work. The computer analyzes written text into some observable components such as the features mentioned above. Automated grading can speed up the grading process and therefore motivate teachers to give more writing assignments without the hassle of going through each of them. However while reviewing existing systems we found out few limitations, which gave us motivation to develop our own system.

Existing essay grading systems can be tricked into assigning a lower or higher score. This can be attributed to the model's inaccuracy in predicting score. Most of the current systems do not consider the theme of the essay or judge the opinion presented by the user. Some of the systems do not remove undesirable written content which can lead to longer processing time. The features to be incorporated must cover all the patterns of evaluation so they are pretty much scalable as opposed to some of the systems which fail to accommodate the small differences. The existing systems work best where they are supposed to work but fail if the data needs or format(rubrics) are changed. And moreover, human grading can take a long time and can also be biased.

Considering the limitations and challenges listed above, we intend to develop a system which automatically scores students essays without human intervention. A simple approach would be to consider only syntactic features (spelling, grammar). But this approach is prone to failure as it does not check the style of writing and relevance of the prompt. We would train the systems in such a way so that it can adapt to changing parameters and can learn on its own if some input parameters or constraints are changed. The system should take user essays as input and display the user scores as output. The grading of the system should come as close to human graders as possible.

Revised Manuscript Received on June 15, 2020.
* Correspondence Author

Mr. Pramit Shetty*, Student Department of Computer Science, K. J. Somaiya College of Engineering, Vidyavihar, Ghatkopar East, Mumbai, Maharashtra, India.
Mr. Kaushal R. Yadav, Student, Department of Computer Science, K. J. Somaiya College of Engineering, Vidyavihar, Ghatkopar East, Mumbai, Maharashtra, India.
Mr. Prithvi Kunder, Student, Department of Computer Science, K. J. Somaiya College of Engineering, Vidyavihar, Ghatkopar East, Mumbai, Maharashtra, India.

II. LITERATURE REVIEW

AES systems have the potential to reduce labour-intensive marking activities, ensuring a consistent application of marking criteria, and facilitating equity in scoring. Although many techniques have been implemented to improve the AES systems, three primary challenges have been concluded:

They lack the sense of the rater as a person. There is no dynamic judging of the essay flow based on each sentence. This is the problem in many AI projects.

They can be tricked into assigning a lower or higher score to an essay than it deserved or not. There can be bugs and loopholes in the system which can be exploited to gain higher grades.

They lack to assess the creativity of the ideas and evaluate their practicality.

Some of the AES systems being used are:

A. Intelligent Essay Assessor (IEA)

The IEA uses a statistical combination of several measures to produce an overall score. It relies on using the Latent Semantic Analysis. IEA can handle students' innovative answers by using a mix of scored essays and the domain content text in the training stage. It uses a procedure in assigning scores in a process that begins with comparing each essay to every other one in a set. On the contrary of other AES systems, IEA requires only 100 pre-scored training essays per each prompt vs. 300-500 on other systems.

It does not take into account the users command of English language and it fails to map punctuations and verbs. The focus exists only on the existence of a certain combination of words in the essay taken as input. There is also no segregation of parts in case marks are to be distributed for conclusion and introduction separately.

B. E-rater

Educational Testing Services (ETS) developed E-rater which relies on using a combination of statistical and NLP techniques to extract the linguistic features from text to start processing, then compares scores with human graded essays. The current version uses 11 features divided into two areas: The first one is the writing quality and the second one is content or use of prompt-specific vocabulary. The E-rater scoring model consists of two stages. The first stage is the model of the training stage, and the other one is the model of the evaluation stage. Human scores are used for training and evaluating the E-rater scoring models.

Our proposed system is more or less similar to the e-rater but it attempts to push the e-rater even further. There is consideration for grammar but the underlying algorithms are not optimized in the e rater system. Also there is no room for customization and all the features are hardcoded. This makes the system applicable to only specific types of examinations and alterations to be made if it is to be used elsewhere. The correlation between human assessors and the system in 2012 ranged from 0.87 to 0.94 which can be made better.

C. Intelli Metric

IntelliMetric is considered as the first AES system that relies on Artificial Intelligence (AI) to simulate manual scoring process carried out by human-raters under the traditions of cognitive processing, computational linguistics, and classification. IntelliMetric relies on using a combination of AI, NLP techniques, and statistical techniques.

The correlation value is found to be too low which means there are issues in mapping. Again not much emphasis on grammar and English literature knowledge of the sets are taken into consideration. There is no room for discourse analysis to grade people based on opinions.

Proposed AES system

To overcome the shortcomings of the above systems we consider both simple and complex features. These will help the system to identify actual meaning expressed in the essay and its correlation with the given topic.

The proposed system will include the following procedure:

Input essay to be graded.

Preprocess the essay to get a preliminary score. This will be based on syntactic features like - spelling errors, grammatical errors, sentence proportion and use of punctuations.

Then the data is fed into the machine learning algorithms in order to get the scores based on semantic features - detection of undesirables, discourse and thematic analysis.

The output scores are integrated and scaled to get the final score.

III. FUNCTIONAL DESCRIPTION

A. Syntactic features

1. Spell checker and corrector:

Incorrect spellings are identified in this module and counted towards the preliminary score. The words are then replaced with closet correct spellings and passed to the next module. Number of spelling errors is counted and deductions are according to the rubrics given.

2. Grammatical errors:

This module checks for correctness of grammar in the essay. Detailed explanation of the rules is provided. As there is no available open source library for grammar check we implemented it using rules / cfg. If a sentence doesn't follow the rule then it's considered as a grammatical error. One important rule to check here is verb tense agreement, which is a common grammatical mistake among students. Our implementation of grammar checks for such inconsistencies and scores are penalized according to the rubric.

3. Punctuations:

Here we check the number of punctuation errors. We have considered the following types - full stop, comma, question mark, single and double quotes. The algorithm used in each of these is explained in methodology.

4. Lemmatization:

This module takes corrected words from the spell corrector and converts them to its root word. This is done to reduce the complexity of processing while maintaining the meaning of the essay. For grammatical reasons, documents are going to use different forms of a word or maybe related words with similar meanings. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set. For implementing lemmatization we have used the inbuilt NLTK methods which provides the word in its root form based on morphological analysis which is then replaced in the original word in the word vector.

5. Removal of stopwords:

A stop word is a commonly used word such as the, a, an, in which does not add meaning to the sentence. Thus they can be removed safely without altering the overall meaning. This also helps to reduce the complexity due to decrease in the number of words. We have used NLTK library here for POS tagging. We then perform a linear search on the word to identify words with tags as noun, verb, adjective or adverb. All other tags are referring to stopwords and should be purged in the modified word vector.

6. Sentence proportion:

Sentences should be both not too short and not too lengthy. Experts recommend using sentences that consist of 15 to 20 words in academic papers. Based on online research materials and general rules a range of [8, 20] words per sentence is chosen as threshold. Sentence proportions could be further extended to essay length and paragraph length.

B. Semantic features:

1. Thematic analysis

This module checks the correlation of the essay with the essay topic. Essay prompt should not be out of topic just to reach a number of words.

A student can get a high score with just writing correct grammatical sentences and spelled words even if it is out of context. To avoid this thematic module is necessary. Here we find how a student's essay is correlated with the high scored essays from the same topic. All the high scored essays were found to be correlated with each other. Average of the correlation is taken to minimize any error, as there are many ways to get a high score and every essay can have a different style of writing.

2. Discourse analysis

The literature in the teaching of writing suggests that invention, arrangement and revision in essay writing must be developed in order to produce effective writing. If a system can automatically identify the actual text associated with discourse elements in student essays, then feedback like that used in traditional textbook teaching of writing can be directed toward specific text segments in students writing. Writing a system that is capable of segregating essays into their constituent parts is very useful to judge a person’s ability. Structuring is a key component in essay analysis and discourse analysis must be used to achieve segregation.

Cosine similarity of each sentence with its next one is used to segregate the essay into different parts - introduction, body and conclusion. We assumed that in an ideal case, 15% of the essay must be introduction, 10% conclusion and the rest 75% main body. This will be used for implementation

3. Undesirables

A number of other undesirables like off topic, extensive use of indirect speech, number of sentences in passive voice and using additional connectives to illustrate the same point contributes to the candidate securing more grades than deserving. Undesirables also include a paramount understanding of poor structuring of sentences which though grammatically correct may be obscure in relation to the topic at hand.

IV. METHODOLOGY

Proposed system model

In a sequential manner, we perform spell check, then correction. After spell check scores are deducted. After spell corrector, we perform stopword removal followed by lemmatization. The result is a useful cleaned essay which will be used for complex features. In parallel, we perform punctuation checking, grammar checking and identifying sentence proportions all of which will help deduct scores further. These deductions will be combined in the score module and returned to the main application. The product of preprocessing is a clean version of the input essay and a deduced score after considering all features. For clarification check Fig.1 for the flow of syntactic block.

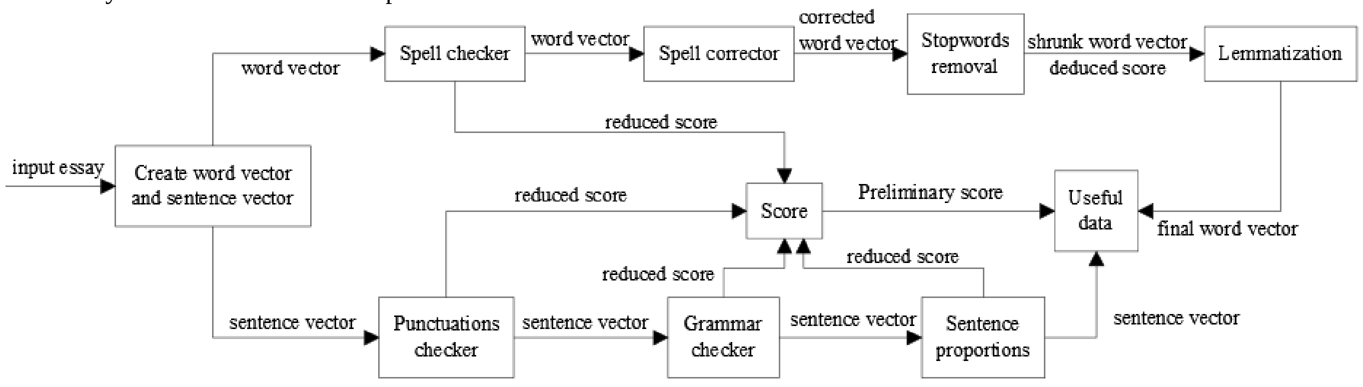


Fig. 1.Syntactic block

Table- 1: Metrics used for evaluating syntactic block

Cases	Acceptable case	Marginally acceptable	Poor
Spelling mistakes (20)	Spelling mistakes are between 1 to 4 (-1). For no mistakes deduct 0	More than four spelling mistakes (-2 for each mistake)	Too many spelling mistakes > 15 (-20)
Grammatical errors (25)	Grammatical errors are Less than or equal to four (-0)	More than four grammatical errors (-1 for each mistake)	Too many grammatical errors > 29 (-25)
Usage of punctuation (25)	Punctuations are used properly when needed with no errors (-0)	Improper usage of punctuation or capitalization after full stop (-1 for each mistake)	Improper usage of punctuations on more than 29 occasions (-25)

Number of words in a sentence (20)	8-20 words in all sentences (-0)	More than 20 or less than 8 words are used in a sentence (-0.5 for each such sentence)	More than 40 sentences having words < 8 or greater than 20 (-20)

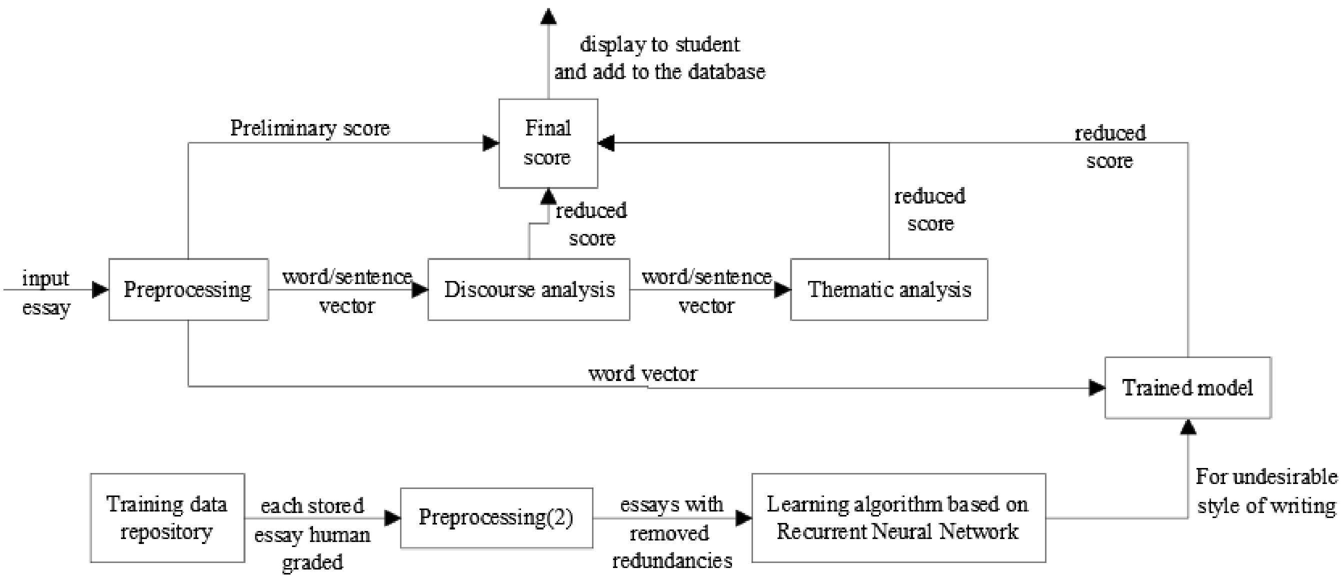


Fig. 2.Main application block

So the preprocessing (syntactic) block mentioned in main application block diagram (refer Fig.2) is the same as the previous figure. The useful data obtained will be fed into thematic analyzer and discourse parser (algorithms of which are explained later) and both return a score for final score computation. We have also trained a model for finding scores based on the style of writing. The RNN based model mentioned above is bidirectional LSTM which will be elaborated on in a later section. The model is trained and we save the container and the weights. When a new essay is to be graded, its word vector will return from the preprocessing block and it will be fed into model to get the scores. The term reduced score implies the score is always less than 100. All the scores are aggregated using the approach of applying weights discussed in the results section and then we get a score out of 100 which will be displayed onto the screen.

The following are the methodologies adopted for both syntactic and semantic features:

Syntactic features

1. Spell checker

The first step is to perform spell checking. For our application, we will need to score a list of almost all available spellings in English dictionary. We could have dynamically used approaches to check spelling by reference to internet but we hypothesized that it will create additional load in the system and out system needs to be as efficient in computationally inexpensive as possible. There is a text file with 3,70,000 words in English.

Hash tables vs Trie

There are two alternatives through which we could implement the spell check operation in our static approach. The first is to create hash tables to perform spelling lookup at runtime and the second is to use trie for lookups. Hash tables are less efficient for retrieval but they take up a lot of space in memory. Trie takes up more time for lookups, but is much faster than hash tables. If we want to compare the worst case complexity of both the approaches, they are as follows:

Data structure	Time complexity (searching)	Space complexity
Trie	O(l)	O(n)
Hash table	O(m)	O(m)

Table 2- : Metrics used for evaluating syntactic block

Notations

l: longest word in the words list
n: Number of nodes created
m: Number of spellings

If we try to compare space complexity, we can see that trie depends on number of nodes created and hash tables depend on number of spellings. For asymptotically large values of m, we can see that n>m as many spellings will share a common prefix. Also O(m) has been shown for hash tables considering the case where the hash function is worst and all the spellings perform collisions. This won't happen and in the best case hash table will take O(1) time for most spellings while O(1) best time of trie never happens unless there is a word of length one in the list of spellings. So hash can be considered slightly better than trie in space complexity while trie is many times better in terms of space complexity.

Based on the above discussion we conclude that trie is the best data structure for the purpose of our application. We will first store all the spellings in a trie data structure. After this, we save that trie.

During runtime, the spellings of all words for the input essay will be checked against the words stored in trie and incorrect words can be rooted out.

2. Spelling correction

Why perform spelling correction? What is the use? Just find incorrect words and then penalize the score to be allotted for the passed essay. This sounds like a best approach but has a major flaw. If we don't change the incorrect word with closest possible correct word then the algorithms which will be used for further correction based on many parameters like conformance with topic relevance, style of writing etc will penalize the candidate for the same mistake again. This will hamper the accuracy of the system. So in order to reduce the effect of repeated penalizing, we must correct those incorrect spellings.

So for each word we first have to find all the words close to the given incorrect word. We have used enchant library available in python in order to get all the words close to the given word. The library has a method which returns a list of all words that could be suspected candidates of the missword that was entered at the time of writing the essay. Now we need a metric to find the value that is best suited to replace the incorrect word. Inside the python jellyfish library, there are many such algorithms which can help us get that metric for evaluating. Analyzing all the algorithms, we found that Damerau-Levenshtein Distance as it considers both the missing and interchange of words in the spelling as separate entities while evaluating the closeness between a pair of words. It also evaluates interchange of words as one mistake. Based on these, the best candidate will be the one which has highest score.

Algorithm:

- Create suggestions for each of the incorrect words using suggest() method in pyenchant.
- for each incorrect word and each of the candidates, apply the metric function get the values.
- Take the suggested word from candidates with maximum metric value and put it in new list for corrected word.

The output from this module will be a list of corrected words from the passed incorrect words list. This output will replace the original word in the essay.

2. Lemmatization

We use the lemmatize library in wordnet
Algorithm

```
For each_word in essay:
    word1 = lemmatize each_word to root noun form
    word2 = lemmatize word1 to root verb form
    word3 = lemmatize word1 to root adjective form
    Add word3 to lemmatized list
```

3. Removal of stopwords

Each of the words which have tags that are not useful must be removed.
Useful tags are noun, verb, adjective and pronouns

4. Sentence proportions

- Input is a list of all words in an essay.
- Remove all punctuations in each sentence as we don't need punctuations for counting words in each sentence.
- For each sentence if count is not between 8 and 20 (inclusive) report that sentence as incorrect.

5. Punctuations

Full stop

Note: Checking for capitalization is an important feature in our project which we are incorporating with checking full stops.

Algorithm

On seeing full stop check the following:

- Check for verb in sentence.
- Check next word is uppercase (Capitalization checking).
- If next word is a proper noun then check if there is a verb in the next sentence.
- The last word of the essay list must be a full stop.

If any of the above conditions are not true then return false else not false.

Comma

On seeing comma check the following:

- Check if pos tag of next and previous word is the same.
- Next and previous words are nouns and the next word is not the last word of the sentence.
- Next word is a coordinating conjunction.
- Next word is which or opening quote.
- There is a preposition somewhere near comma (within 3 indexes).

If any of the above conditions are not true then the comma is a mistake.

Question mark

On seeing a question mark, check the following:

- Sentence starts with verb (Check pos tag of first verb of that sentence).
- Sentence starts with a wh word (like what, which, how etc).
- Check question tags (There is a comma three indexes before the question mark).
- Sentence is in direct speech (Check there is a single quote).

If any of the above conditions are not true then the question mark is placed by mistake.

Single and double quotes

Check sentence with open quotes has a closing quote which is of the same type. (Save the opening quote and compare it with closing quote. If closing quote is not present then there is an error.)

6. Grammar checker

Checking verb tense agreement

In a sentence, the tenses of all the verbs must be same. But for sufficiently large sentences, the tenses may differ.

So to relax this strictness, we have introduced a new user defined parameter proximity which is the range within which all the verbs detected must be of same tense.

```
Algorithm:
Traverse the essay.
On seeing a verb, note its tense. Initialize the counter.
On seeing the second verb
    If the counter<proximity
        If tenses different
            Mistakes+=1
        Make the tense of second verb as the referenced tense.
    Else
        Irrespective of the matching result of the tenses, make
the tense of second      verb as the referenced tense.
If we see a full stop anywhere then we reset the counter
```

Error in sentence structures:

We create a bunch of if else rules based on the context free grammar as below:

```
  S -> NP VP
  NP -> DT NP1 VP4 | pronoun NP4 | DT NP6 | DT adjective
noun | propernoun NP3 | noun
  NP1 -> noun | adjective noun
  NP3 -> conjunction NP5 |
  NP4 -> conjunction NP5 | noun |
  NP5 -> noun | pronoun | propernoun
  NP6 -> propernoun NP4 | adjective noun
  VP -> verb VP0 | verb VP3 | verb VP6 | verb VP4 | verb VP8
| verb VP5 | adverb VP6 | 'EX' VP | 'TO' VP
  VP0 -> NP1 VP2 | adverb VP2 | prep NP | pronoun |
  VP1 -> adjective noun
  VP2 -> prep NP |
  VP3 -> verb VP0 | adverb VP6 | pronoun VP1
  VP4 -> verb VP0 | verb VP6 | verb VP8
  VP5 -> verb VP0 | verb VP6
  VP6 -> verb VP0
  VP8 -> verb VP6
  noun -> 'NN' | 'NNS'
  propernoun -> 'NNP' | 'NNPS'
  pronoun -> 'WP' | 'WP$' | 'PRP' | 'PRP$'
  adjective -> 'JJ' | 'JJR' | 'JJS'
  verb -> 'VB' | 'VBG' | 'VBN' | 'VBZ' | 'VBP' | 'VBD' |
  conjunction -> 'IN' | 'CC'
  adverb ->  'RB' | 'RBR' | 'RBS' | 'RP' | 'WRB'
```

The derived rules are given by:

Adjectives

- Next word is a noun (Next tag starts with N)
- There is a pronoun following the adjective (next tag is PRP)
- There is a verb just before the adjective and a noun somewhere preceding the adjective (Check for tags with V and N respectively)
- There is a verb just before the adjective and a pronoun somewhere preceding the adjective
- Noun just before the adjective
- Pronoun just before the adjective
- Previous element or the one before that is an adjective and not in comparative degree

Noun

No conditions can be written for nouns as nouns can appear anywhere in the sentence. We have also written the rules for other tags relative to nouns.

Modal auxiliaries

- There is a verb in that sentence following the modal auxiliary. (Find tags starting with V before full stop)
- There is no 'TO' word following the modal aux (Check for TO tag before end of sentence).

Preposition

- Preposition not in the start or end.
- There is a noun somewhere (2 or 3 elements later) in the sentence (Check tags starting with N within 2-3 indexes)

Adverbs

We decided not to check for adverbs. According to Oxford, adverbs can occur initially, in the end or before and after an auxiliary. This is just a general and in practice, not all adverbs can appear at all places. This gets problematic in programming practise. This is illustrated by the Fig. 3 given below:

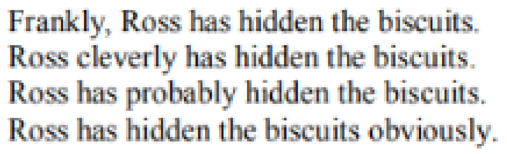

Frankly, Ross has hidden the biscuits.
Ross cleverly has hidden the biscuits.
Ross has probably hidden the biscuits.
Ross has hidden the biscuits obviously.

Fig. 3.Adverb conumdrum problem example

For the sentences given above, the adverbs frankly, cleverly, probably and obviously appear at various positions in the corresponding sentence.

Aside from the above parts of speech, there will be some more POS tags like CD, LS, DT etc but writing a logic for these will make system more complex and students rarely get them incorrect and are often forgiven in real life.

Semantic features

Discourse analysis

- First create sentence vectors from the passed essay
- If there is only 1 sentence return minimum score
- Create a list that will hold similar clusters.
- Add the first sentence by default to the first cluster. (It has to be introduction)
- Loop through sentence2 to end.
- For each sentence apply cosine similarity with each cluster save its value. Make sure both vectors are normalized. Vectors can be normalized by dividing every element with the magnitude of that vector.
- Obtain the index of the cluster with which it forms the highest value of cosine similarity.
- If the value of cosine similarity is greater than the threshold append to that cluster. (Threshold is a hyper-parameter which is used to check the criteria of adding to a cluster. For instance, if the threshold is 0.5 if the cosine similarity value is 0.6 it will be added to that cluster else we add it as a new cluster in the list.)

Else add it to a new cluster.
After applying the above algorithm, it is possible that there are multiple clusters in the list. In that case we assumed first cluster is introduction, last cluster is conclusion and all the other clusters in between constitute the main body so we combine them.

After the above step, we can calculate what percent of the essay is intro, body and conclusion.

Detection of undesirables

a. Bidirectional LSTM model to evaluate style of writing

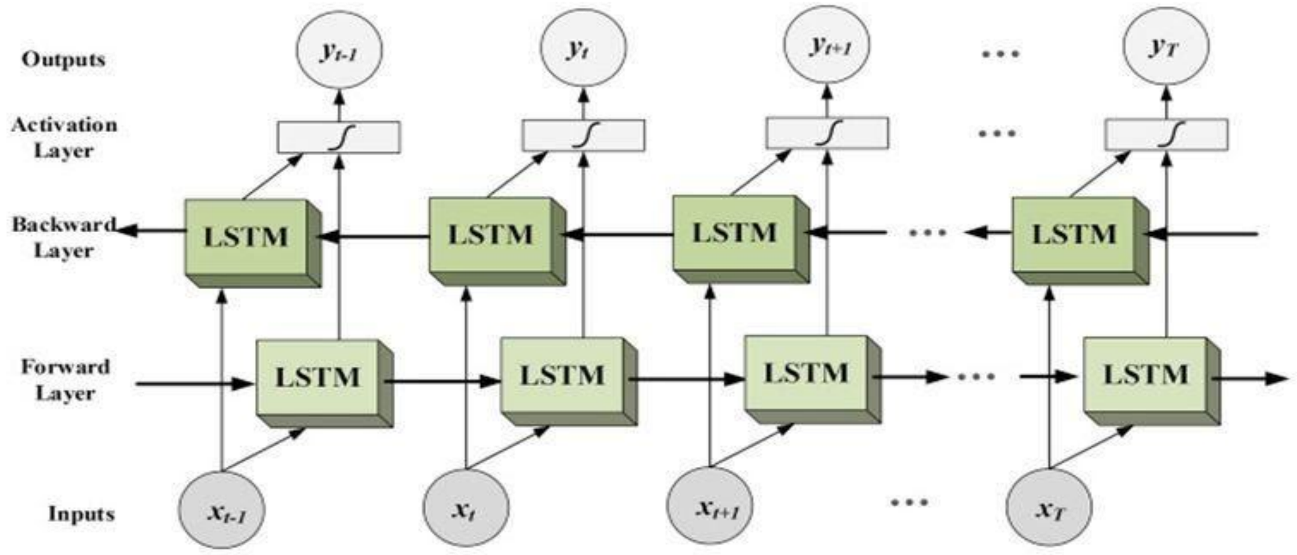


Fig. 4.Proposed system diagram for detecting style of writing

We will be using bidirectional LSTM (Long Short Term Memory) which is a sequential model (Refer fig. 4). This is built on top of a word2vec model and the inputs are passed in two ways. The first is in normal order from left to right and the second is reverse order from right to left. We have used ReLU activation function in order to get the outputs which is a score. The model is compiled on the basis of mean squared error loss. We have used five-fold cross validation in order to train the data. We then trained the model and then saved the weights for the prediction of new essays. We then computed the quadratic cohen's kappa score on prediction of cross validation set i.e. the fifth fold in order to get the accuracy of the model. The fitting of model is done by considering 50 epochs of computation and batch size of the essays taken is 64. It means that the training dataset is parsed 50 times and we consider 64 essays at a time while training i.e. obtaining weights.
This process is repeated five times as that is what the process of cross validation does. When we get new essay, the weights are loaded and then the score for that essays are obtained.

b. Passive voice detection

If the root form of a word (after lemmatization) is be and the next word is a verb but not a gerund, the sentence is in passive voice. 10 marks are assigned to passive voice detection and 1 mark is deducted per mistake.

c. Indirect speech detection

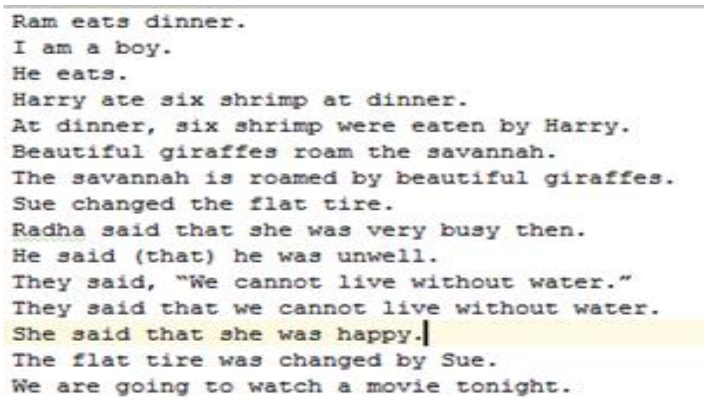
If there are specific words in a sentence like say, says, said, told etc then it is compulsory that there must be quotes used. In the absence of quotes, the sentence will be indirect speech else direct speech. 10 marks are assigned to passive voice detection and 1 mark is deducted per mistake.

Thematic analysis

1. Make a list (L) of high scored essay from training set
2. Append each essay from test set to L
3. Find cosine similarity of the test essay with all other essays
4. Take average of all the cosine similarities and multiply with 100
5. A high value will indicate that the essay is written according to the topic and hence a high score will be given and vice versa.

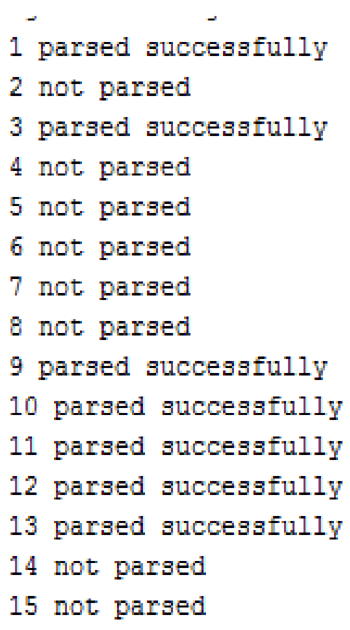
V. RESULTS AND EVALUATION

For grammar checker our first approach was to create a recursive descent parser for evaluating sentence structures. The results were terrible



```
Ram eats dinner.
I am a boy.
He eats.
Harry ate six shrimp at dinner.
At dinner, six shrimp were eaten by Harry.
Beautiful giraffes roam the savannah.
The savannah is roamed by beautiful giraffes.
Sue changed the flat tire.
Radha said that she was very busy then.
He said (that) he was unwell.
They said, "We cannot live without water."
They said that we cannot live without water.
She said that she was happy.
The flat tire was changed by Sue.
We are going to watch a movie tonight.
```

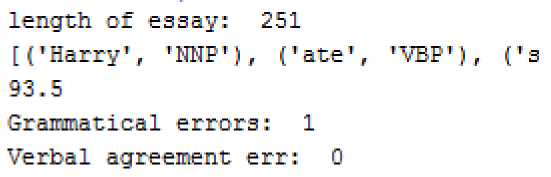
Fig. 5.Examples for grammar checking



```
1 parsed successfully
2 not parsed
3 parsed successfully
4 not parsed
5 not parsed
6 not parsed
7 not parsed
8 not parsed
9 parsed successfully
10 parsed successfully
11 parsed successfully
12 parsed successfully
13 parsed successfully
14 not parsed
15 not parsed
```

Fig. 6.Results of checking using the grammer given in methodology section for Fig. 5

In Fig. 5 are all the sentences which should have been parsed successfully but through the grammar only 50% are parsed (Refer Fig. 6) which means the accuracy in the first approach was only 50%.



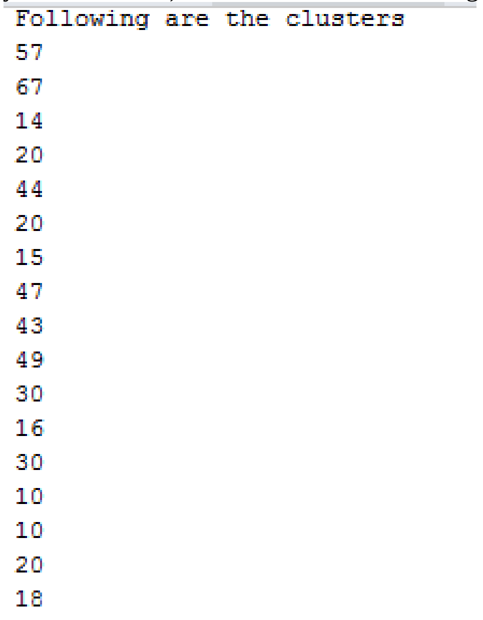
```
length of essay:  251
[('Harry', 'NNP'), ('ate', 'VBP'), ('s
93.5
Grammatical errors:  1
Verbal agreement err:  0
```

Fig. 7.Results using the second approach using rules

In our second approach, for the same input, only 1 error is detected (Refer Fig. 7) in the sentences which makes and accuracy of 93.33% in grammatical error detection.

Discourse analysis

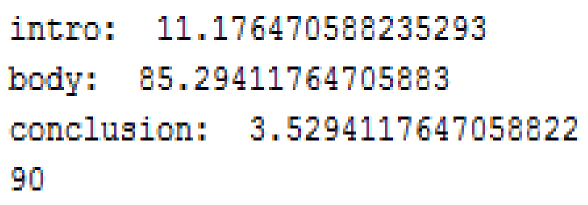
For discourse analysis, we took a sample essay.
The results of first pass for threshold = 0.05 (very small value as the sentences closely related show very low value of cosine similarity between them) were a number of clusters given by:



```
Following are the clusters
57
67
14
20
44
20
15
47
43
49
30
16
30
10
10
20
18
```

Fig. 8.Result of first parse for discourse

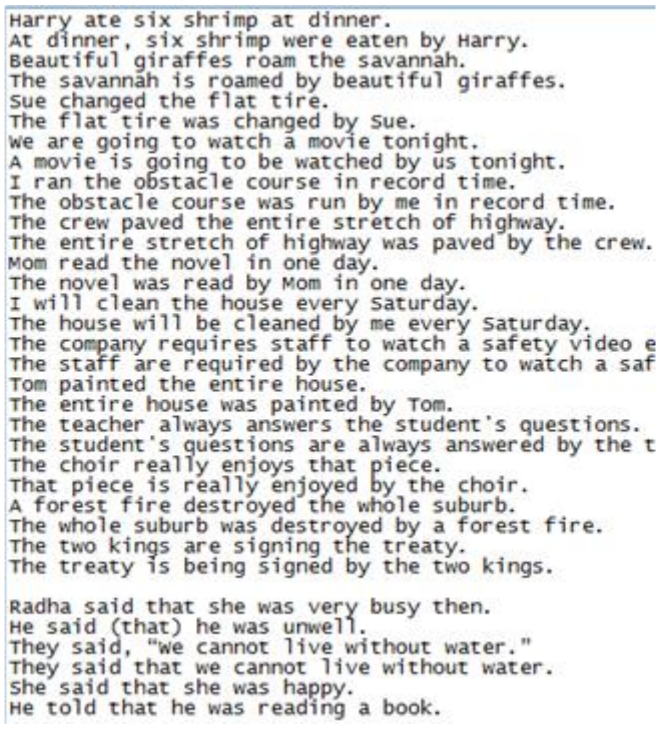
In the second pass, we get percentages and then we can get a score based on the percentages mentioned in functional description above. Output given below



```
intro:  11.176470588235293
body:  85.29411764705883
conclusion:  3.5294117647058822
90
```

Fig. 9.Final result for discourse

So score is calculated as follows:
11.17% of essay is introduction so we take 11/15 for intro
85.3% is main body. But as stated earlier maximum marks that can be obtained with main body is 75, so we take 75/75
3% is conclusion, so we take 3/10
Total = 11+75+3 = 90 which is the score generated and returned. This is confirmed in Fig. 9.
Detection of passive voice and indirect speech
Consider the input given in Fig. 10:

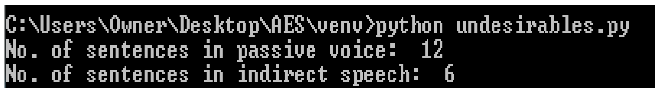


```
Harry ate six shrimp at dinner.
At dinner, six shrimp were eaten by Harry.
Beautiful giraffes roam the savannah.
The savannah is roamed by beautiful giraffes.
Sue changed the flat tire.
The flat tire was changed by Sue.
We are going to watch a movie tonight.
A movie is going to be watched by us tonight.
I ran the obstacle course in record time.
The obstacle course was run by me in record time.
The crew paved the entire stretch of highway.
The entire stretch of highway was paved by the crew.
Mom read the novel in one day.
The novel was read by Mom in one day.
I will clean the house every Saturday.
The house will be cleaned by me every Saturday.
The company requires staff to watch a safety video e
The staff are required by the company to watch a saf
Tom painted the entire house.
The entire house was painted by Tom.
The teacher always answers the student's questions.
The student's questions are always answered by the t
The choir really enjoys that piece.
That piece is really enjoyed by the choir.
A forest fire destroyed the whole suburb.
The whole suburb was destroyed by a forest fire.
The two kings are signing the treaty.
The treaty is being signed by the two kings.

Radha said that she was very busy then.
He said (that) he was unwell.
They said, "We cannot live without water."
They said that we cannot live without water.
She said that she was happy.
He told that he was reading a book.
```

Fig. 10. Examples of active/ passive voice and speech detection

According to theory 14 sentences are in passive voice and 4 sentences are in indirect speech. For the sake of demonstration on writing the code for getting the number



```
C:\Users\Owner\Desktop\AES\venv>python undesirables.py
No. of sentences in passive voice:  12
No. of sentences in indirect speech:  6
```

Fig. 11. Output for Fig. 10 using code

On observing the output, 12 of the 14 passive sentences are detected and 2 more in passive list in addition to the 4 for indirect have been detected which demonstrates higher accuracy (Refer Figs 10 and 11).

Thematic analysis

Essay written according to the given topic:
Note - the last row and column gives correlation of given essay with high scored ones.

```
[[1.         0.71629889 0.64824723 0.71917506 0.64542933 0.67173744]
 [0.71629889 1.         0.69302907 0.66061521 0.67895626 0.6688513 ]
 [0.64824723 0.69302907 1.         0.6698983  0.6354324  0.64750049]
 [0.71917506 0.66061521 0.6698983  1.         0.6214009  0.62695021]
 [0.64542933 0.67895626 0.6354324  0.6214009  1.         0.65575007]
 [0.67173744 0.6688513  0.64750049 0.62695021 0.65575007 1.        ]]
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d4348bb70>
```

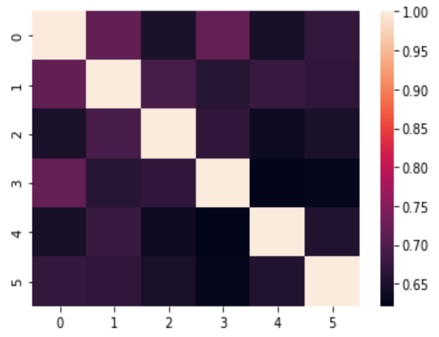


Fig. 12. Example of good essay

Out of context essay:
Note the difference in correlation.

```
[[1.         0.71629889 0.64824723 0.71917506 0.64542933 0.18080073]
 [0.71629889 1.         0.69302907 0.66061521 0.67895626 0.24575217]
 [0.64824723 0.69302907 1.         0.6698983  0.6354324  0.26705847]
 [0.71917506 0.66061521 0.6698983  1.         0.6214009  0.16075581]
 [0.64542933 0.67895626 0.6354324  0.6214009  1.         0.28194877]
 [0.18080073 0.24575217 0.26705847 0.16075581 0.28194877 1.        ]]
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d4074a7b8>
```

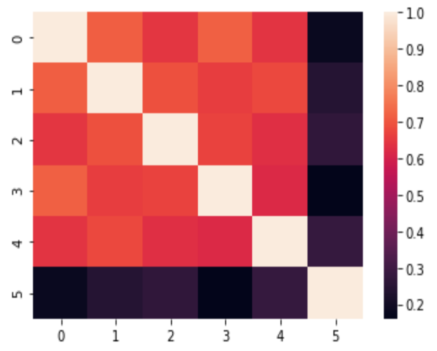


Fig. 13. Example of bad essay

Training the model

We used 5 fold cross validation for understanding the performance of the of the model on the part of training data (one-fifth) being equivalent to test data. So when performed the operation using google colab, the kappa score obtained can be seen in Fig. 14:

```
Kappa scores of all folds with stopwords removed
0.95
0.962
0.959
0.953
0.959
```

Fig. 14. Kappa scores after each validation of 5 fold cross validation

The average score comes close to 0.96. A kappa score greater than 0.9 is said to be in perfect agreement. So here we can say the model has been trained. We use this model to get score of essay.

Combining all the scores

So far we have studies five modules for grading and have obtained individual scores. The next problem was the integration of these features. What percentage of scores generated from each module must be taken. Based on common perception and understanding, we had proposed a metric earlier for grading. The quadratic weighted kappa metric for them was calculated. In addition we used dirichlet function available in numpy library to generate a list of five random numbers for weights which add upto 1. For the purpose of finding best fit, we will use 10, 100 and 1000 iterations respectively and get the metric score along with weights (Refer Fig. 15 and 16). Some of the results are:

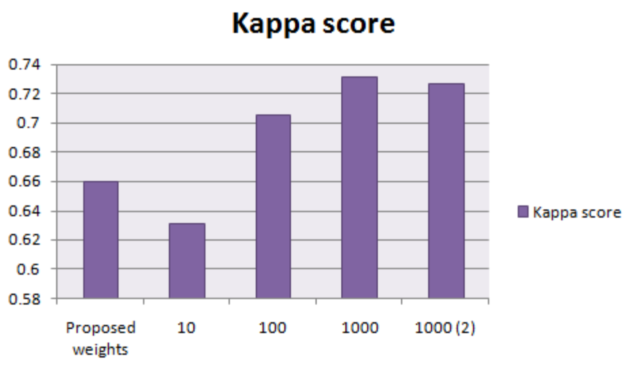


Fig. 15. Kappa scores over n iterations on the sample essay with random weights

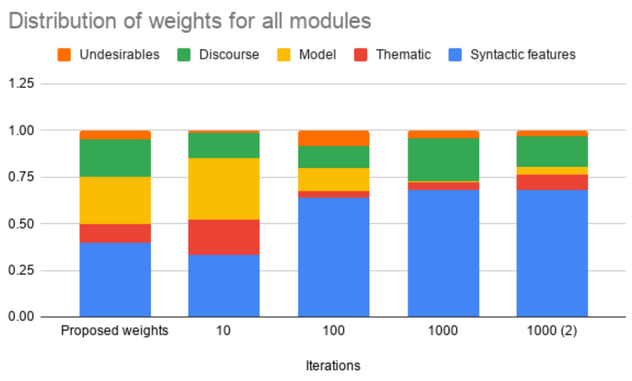


Fig. 16. Representing the weights of all modules in each case of Fig. 15

VI. OBSERVATIONS

The following are some of the observations with respect to the graphs in Fig. 15 and Fig. 16:

1. The kappa score increases with increase in the weightage of preprocessing. This is because the dataset taken is for school essays and for schools more weightage is given to spellings, grammar and punctuations than theme or discourse analysis.

2. The weights of undesirables (here indirect and passive) are always very low signifying their poor contribution.
3. The best performance is 0.73 for any weights on our scores. We can be sure about it because inspite of taking 1000 random combinations twice, the best is 0.73. An agreement of 0.73 for the metric falls under the category of substantial agreement which is good.
4. The weights are not a clear sign and were only taken in this way to understand the potential of the proposed system. In real deployment, weights for each module could be taken from the author/teacher.

VII. CONCLUSION AND FUTURE WORK

Potential AES systems in the past have failed to incorporate evaluation of essays by thematic analysis, detection of undesirable features such as presence of passive voice, indirect speech etc. This has led to lower accuracies in grading essays and the scores often didn't come in agreement with the human scores. We have laid groundwork to this argument and implemented the same in our version. The system is flexible enough for instructors/evaluators to incorporate any changes in rubrics, without affecting the accuracy of grading essays. This makes the system more scalable as opposed to some systems which fail to incorporate small changes. The system fails considerably on new data and although an agreement of 0.73 is good, it simply is not acceptable while deploying in real world. A blame could be based on the dataset being not detailed enough but such fantasized datasets are hard to find. A better dataset could be compiled together but it is very tedious and can take longer time. The system can grade any essay. However, more emphasis is laid on basic level simple features because the system is deployed for grading school essays. Weighted average score needs to be adjusted if the system is deployed anywhere else. The system can be made more robust, whenever change in customer requirements arises. While implementing modules, we have considered some risks as acceptable because we don't wish to penalize accuracy. Mitigation measures would be taken if such risks affect the system.

REFERENCES

1. A Hybrid Scheme for Automated Essay Grading Based on LVQ and NLP Techniques, 2016 IEEE, shehab2016_essay grading
2. A Machine Learning Approach for Identification of Thesis and Conclusion Statements in Student Essays, Kluwer Academic Publishers, Burnstein_Marcu_Chum_discourse analysis
3. A New Linguistic Feature for Automated Essay Scoring, 2010 IEEE
4. Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.
5. Rules for grammar and punctuations - https://www.grammarbook.com/

AUTHORS PROFILE

Mr. Pramit Shetty is a student in his final year at K.J. Somaiya College of Engineering, Vidyavihar and his currently pursuing his bachelor's degree in computer science. He has been a member of CSI KJSCE and has interests in competitive programming, cloud computing, deep learning and machine learning paradigms. He has a certificate for teaching concepts in machine learning and is an active contributor to the ML and deep learning community through platforms like github.

Mr. Kaushal R. Yadav - Currently a final year student at KJ Somaiya College of engineering, vidyavihar, mumbai. His main area of interest is in machine learning, NLP, cloud computing, application development, data structures, algorithms. He has also worked on multiple projects on application and web development.

Mr. Prithvi Kunder, currently a final year Computer Engineering student from K.J. Somaiya College of Engineering. His interest includes machine learning, artificial intelligence, natural language processing, and web development with keen interest in MEAN stack.