A

Project Report on

“Automatic Short Answer Grading”

Project submitted
In partial fulfilments of the requirements for the degree of

MASTER OF COMPUTER APPLICATION

By

Ahan Maji

Class Roll No: 001910503015

Examination Roll No: MCA226015

Registration No: 149878 of 2019-20

Under the supervision of

DR. SUDIP KUMAR NASKAR

Department of Computer Science & Engineering
Faculty of Engineering and Technology

Jadavpur University
Kolkata – 700032
India

Jadavpur University

Faculty of Engineering and Technology

Department of Computer Science & Engineering

To Whom It May Concern

I hereby recommended that the thesis entitled "**Automatic Short Answer Grading**" has been carried out by Ahan Maji (Roll No: 001910503015, Registration No: 149878 of 2019-20) under my guidance and supervision and be accepted in partial fulfilment of the requirement for the Degree of Master of Computer Application in Department of Computer Science and Engineering, Jadavpur University.

(Signature of Head of the department)

Dr. Anupam Sinha

Head of the Department of Computer Science & Engineering

(Signature of Project supervisor)

Dr. Sudip Kumar Naskar

(Signature of Dean of the Faculty)

Prof. Chandan Majumdar

Jadavpur University

Faculty of Engineering and Technology

Department of Computer Science & Engineering

CERTIFICATE

This is to clarify that the project entitled "**Automatic Short Answer Grading**" has been completed by Ahan Maji. This work is carried out under the supervision of Dr. Sudip Kumar Naskar in partial fulfilment for the award of the degree of Master of Computer Application of the department of Computer Science and Engineering, Jadavpur University, during the session 2019-2022. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

……

Signature of Examiner 1

Date:

……

Signature of Examiner 2

Date:

Jadavpur University

Faculty of Engineering and Technology

Department of Computer Science & Engineering

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this project contains original work by the undersigned candidate, as part of his Master of Computer Application (MCA) studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material results that are not original to this work.

Name: Ahan Maji
Class Roll No: 001910503015
Project Title: Automatic Short Answer Grading

.....

(Signature of the Candidate)

Acknowledgement

The writing of the thesis additionally because the related work has been a protracted journey with input from many individuals, right from the primary day till the development of the final project.

I would like to express my deepest gratitude to my supervisor, **Dr. Sudip Kumar Naskar, Assistant Professor, Department of Computer Science and Engineering, Jadavpur University** for his admirable guidance, care, and patience and for providing me with a superb atmosphere for doing research. Our numerous scientific discussions and his many constructive comments have greatly improved this work. I feel deeply honored that I got the chance to figure under him.

I would prefer to thank all the faculty members of the Department of Computer Science and Engineering of Jadavpur University for their continuous support.

This thesis would not have been completed without the inspiration and support of a number of wonderful individuals including my batch mates of Master of Computer Application in Jadavpur University — my thanks and appreciation to any or all of them for being a part of this journey and making this thesis possible.

.....

(Signature of the Candidate)

Abstract

The goal of the **Automatic Short Answer Grading** is to assign grades to essays and supply feedback using computers. Automated evaluation is increasingly getting used in classrooms and online exams. The goal of this research is to create and test machine learning models for doing automatic short answer grading. A publicly accessible essay data set was utilised to train and assess the efficacy of the strategies used in this study. The dataset's essays were analysed using natural language processing techniques to extract characteristics. On the specified dataset, three different machine learning techniques were applied. Among all the machine learning models, that are used in this project the sent2vec model performed the best in terms of agreement with human grading as it achieved the highest correlation value for the test dataset.

Keywords: Automatic Short Answer Grading, machine learning, natural language processing.

CONTENTS

LIST OF FIGURES

Title

1. INTRODUCTION

1.1. What is Automatic Short Answer Grading

ASAG (automatic short answer grading) is a difficult activity that provides a grade/marks to a student's response.

The scoring method is more efficient if there is a stronger correlation between students and model answers. Because of the heterogeneity of curriculums, the AS technology had to deal with a range of student answers, including writing, speaking, and arithmetic. Automatic Essay Scoring (AES) and Short Answer Grading are two types of writing evaluation.

1.2. Motivation

The educational community is growing endlessly with a growing number of students, curriculums, and exams. Such a growing community raised the need for scoring systems that ease the burden of scoring numerous numbers of exams and at the same time guarantee the fairness of the scoring process. Automatic Scoring/Grading (AS/AG) systems evaluate students' answers by comparing them to model answer(s).

Automatic assessment is preferred to Manual Assessment to avoid monotonic, bias errors and conserves the teacher's time for the main activity. Hence automatic assessment is vital for the educational system. The area of Computer-based Assessment Systems has grown exponentially due to the larger intake by the university system, e-learning system as a ubiquitous education platform. Computer Assisted Assessment is an important area of research due to developments in Natural Language Processing (NLP), Information Extraction (IE), and e-learning.

An automatic Scoring system for questions such as Multiple Choice, True-False, Matching, and Fill in the blank is an easy task. AS systems designed for scoring essay questions are a more complicated task as students' answers require text understanding and analysis. This paper is concerned with the automatic scoring for answers to essay questions. This research presents an unsupervised approach that deals with students' answers holistically and uses text-to-text similarity measures.

The proposed model calculates the automatic score by measuring the text similarity between each word in the model answer to all words in the student's answer which saves the time spent by teachers to create predefined patterns and grading rules. Paragraph-based similarity and sentence-based similarity are the two types of text similarity measurements given in this study.

1.3. Statistical Measures for ASAG

Correlation: Correlation is a statistical term that describes the relationship between two variables. The correlation coefficient is used to calculate it. The correlation coefficient might have a value of -1 to +1. As a correlation is positive, it suggests that when one variable rises, the other rises as well. When one variable increases, the other decreases, resulting in a negative correlation. There is no relationship between the variables when the correlation coefficient reaches 0.

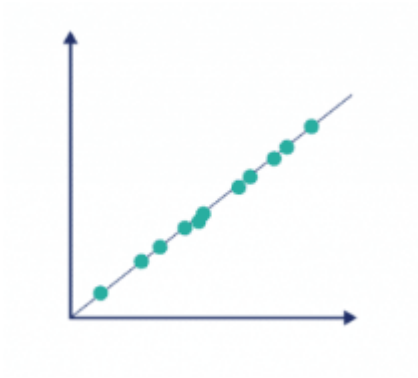

Fig. 1.1 Positive Correlation

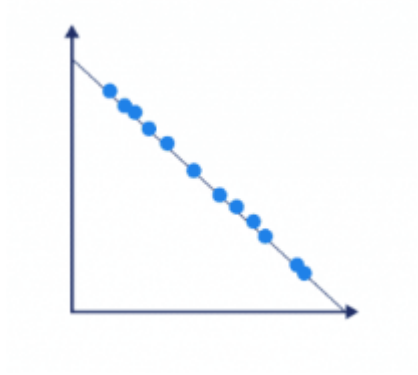

Fig. 1.2 Negative Correlation

Fig. 1.3 Zero Correlation

Here we are using 3 kinds of correlation method

1. **Pearson correlation:** The Pearson coefficient is a form of correlation coefficient that shows how two variables measured on the same interval or ratio scale are related. The Pearson coefficient is a measure of the strength of the association between two continuous variables.

The following formula is used to determine the connection between two variables, X and Y:

$$r = \frac{\Sigma(X-\overline{X})(Y-\overline{Y})}{\sqrt{\Sigma(X-\overline{X})^2}\sqrt{(Y-\overline{Y})^2}}$$

Where, $\overline{X}$ = mean of X variable
$\overline{Y}$ = mean of Y variable

2. **Spearman's rank correlation:** The non-parametric statistical tool used to evaluate the degree of relationship between the two ranked variables is Spearman's Rank Correlation Coefficient. This approach is used with an ordinal set of numbers that may be put in sequence, one after the other, to assign rankings to each.

The rank correlation coefficient approach assigns rankings to each individual based on their quality or quantity, for example, ranking begins with position 1 and ends with position N for the person rated last in the group.

The formula to calculate the rank correlation coefficient is:

$$R = \frac{(1 - 6\sum D^2)}{N\,(N^2 - 1)} = \frac{(1 - 6\sum D^2)}{N^3 - N}$$

Where, R = Rank coefficient of correlation, D = Difference of ranks, N = Number of Observations.

3. **Kendall Rank Correlation:** When the data you're dealing with fails one or more of the test's assumptions, Kendall rank correlation (non-parametric) is an alternative to Pearson's correlation (parametric). When your sample size is small and there are many tied ranks, this is the best option to Spearman correlation (non-parametric).

When data is ranked by quantities, Kendall rank correlation is used to see whether there are any commonalities in the ordering. Kendall's correlation coefficient uses pairs of observations and determines the strength of association based on the pattern of concordance and discordance between the pairs. While other types of correlation coefficients use observations as the basis of the correlation, Kendall's correlation coefficient uses pairs of observations and determines the strength of association based on the pattern of concordance and discordance between the pairs.

Root Mean Square Error (RMSE): One of the most often used methods for evaluating the validity of predictions is root mean square error, also known as root mean square deviation. It uses Euclidean distance to demonstrate how much forecasts differ from measured true values.

Calculate the residual (difference between prediction and truth) for each data point, the norm of the residual for each data point, the mean of residuals, and the square root of that mean to get the RMSE. Because RMSE employs and requires real measurements at each projected data point, it is extensively utilised in supervised learning applications. Root mean square error can be expressed as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

where N is the number of data points, y(i) is the i-th measurement, and y^(i) is its corresponding prediction.

2. Literature Survey

There have been several approaches to automatic short answer grading proposed in the past, one of which is Uswatun hasanah(2019) from Indonesia, who proposed a study that assesses the sentence similarity of short answers to questions and answers in Indonesian without using any language semantics tool. Case folding, tokenization, stemming, and stopword elimination are among the pre-processing procedures used in this study. The suggested technique is a scoring rubric that is created by comparing phrase similarity using string-based similarity algorithms and a keyword matching mechanism. This dataset included 7 questions, 34 alternative reference answers, and 224 student responses. The experiment findings demonstrate that the suggested technique can obtain Pearson's correlation values ranging from 0.65419 to 0.66383, with Mean Absolute Error-values ranging from 0.94994 to 1.24295. In each technique, the suggested methodology additionally makes use of the correlation value and reduces the error value.

Several approaches have been proposed in the past for automatic short answer grading. Several state-of-the-art short answer graders (Sukkarieh et al., 2004) require manually crafted patterns that, if matched, indicate that a question has been answered correctly. If an annotated corpus is available, these patterns can be augmented semi-automatically by learning more patterns.

Another implementation of the Oxford-UCLES system (Pulman and Sukkarieh, 2005) compares several machine learning algorithms, including inductive logic programming, decision tree learning, and Bayesian learning, to the earlier pattern matching approach with encouraging results.

C-Rater (Leacock and Chodorow, 2003) compares a student response's syntactical properties (subject, object, and verb) to a collection of right replies. While attempting to identify variations in voice ("the guy was bitten by a dog ") the technique notably disregards the bag of words approach to account for the difference between "dog bites man" and "man bites dog".

The job of text similarity is strongly connected to automatic short answer grading. Text similarity is fundamentally the challenge of recognizing and comparing the characteristics of two texts, and it is more general than short answer grading. The vector-space model with a term frequency/inverse document frequency (Salton et al., 1997) (TF. IDF) weighting is one of the early methods of text similarity. For tasks like information retrieval and text categorization, this model, along with the more advanced LSA semantic alternative, has been proven to operate effectively.

A machine learning algorithm has also been used (Hatzivassiloglou et al., 1999). It uses a collection of simple features to create features (e.g., a pair of nouns appear within 5 words from one another in both texts). Synonymy, word order, text length, and word classes are also taken into consideration by this technique.

3. Methodology

Different approaches are used to solve automatic short answer grading problems. The first approach is based on using unsupervised techniques that combine text-text similarity measures to compare student answers and reference answers and predict scores based on a similarity value.

3.1. Word embedding

Word embedding are techniques for learning vector representations for words. Word embedding models are usually trained on large unlabeled corpora to exploit their benefits. Multiple models for word embedding trained on a large corpus of data are publically available. These pre-trained models have the advantage of storing a semantic relationship between words.

The procedure is as follows: given a brief answer (student or reference answer), the goal is to produce the answer's paragraph vector. To acquire a list of the answer's words, it's first tokenized, then any text processing is done. Second, given the word vector model, the corresponding word vector for each word is retrieved. Finally, to get the paragraph vector, the sum operation is applied to all word vectors to get a single vector representing the paragraph vector.

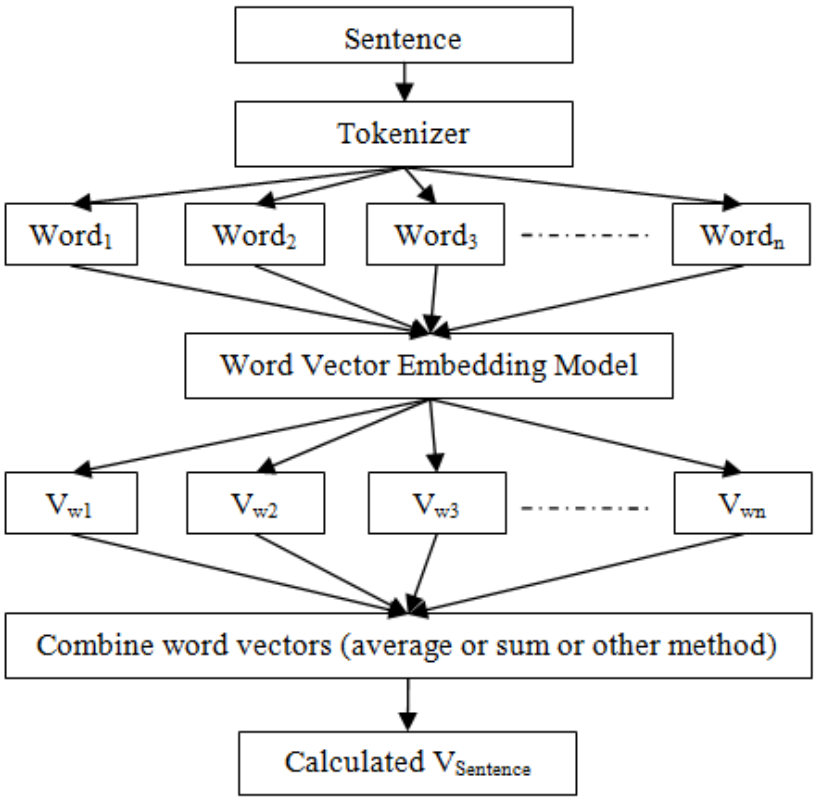

Fig. 3.1. Generating Paragraph Vectors from Pre-Trained Word Vector Models.

3.2. Paragraph embedding

In paragraph embedding models, the deep learning model is trained on sequences of text to directly learn and infer vector representation of the variable-size sequence.

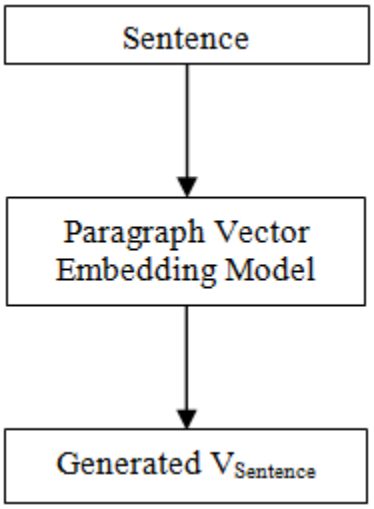

Fig. 3.2. Generating Paragraph Vectors from Paragraph Embedding Models.

Advantages of paragraph vectors: Paragraph vectors have the benefit of being able to be learned from unlabelled data, therefore they can be used for tasks when there isn't enough labelled data. Bag-of-words models have several flaws, which are addressed via paragraph vectors. For starters, they inherit one of the word vectors' most significant properties: the semantics of the words. "Powerful" is closer to "strong" than "Paris" in this context. The paragraph vectors' second benefit is that, at least in a limited context, they handle word order in the same manner as an n-gram model with a big n would. This is significant since the n-gram model keeps a lot of the paragraph's information, including the word order.

3.3. Doc2vec Model

Doc2vec model is a neural network classifier with a stochastic gradient descent algorithm trained on a fixed-width sliding window over words of paragraphs. Paragraph vectors are learned along with word vectors so the trained model can be used to infer paragraph vectors or word vectors.

Gensim Doc2vec is a python library that provides an implementation of the paragraph vectors model. It is designed to model word sequences ranging from n-gram sentences, paragraphs, or documents. On our chosen dataset for solving the automatic short answer grading problem, we trained the 300-dimensional doc2vec model.

3.4. Doc2vec Model Architecture

Doc2Vec is designed to be an extension of Word2Vec, in which Word2Vec learns to project words into a latent d-dimensional space, whereas Doc2Vec learns to project a document into a latent d-dimensional space.

Now somewhere we heard terms like paragraph2vec (paragraph vector). All are the same thing. Mikolov and Le used the term Paragraph vector in their paper and gensim implemented this in a package called doc2vec.

Words now have a logical (grammatical) structure, but documents do not. Another vector (Paragraph ID) must be added to the word2vec model to overcome this problem. This is the one and only difference between word2vec model and doc2vec model.

Now like word2vec there are two flavors of doc2vec are available:

1. Distributed Memory Model of Paragraph Vectors (PV-DM)
2. Distributed Bag of Words version of Paragraph Vector (PV-DBOW)

3.4.1. PV-DM

Word2Vec influenced the core concept of PV-DM. In Word2Vec's CBOW model, the machine learns to anticipate a centre word from context. Given the sentence "The cat sat on the sofa," the CBOW model would learn to predict the word "sat" based on the surrounding terms — the, cat, on, the, sofa. Similarly, the core idea behind PV-DM is to randomly choose successive words from a paragraph and predict a centre word from the randomly chosen collection of words using the context words and the paragraph id as input.

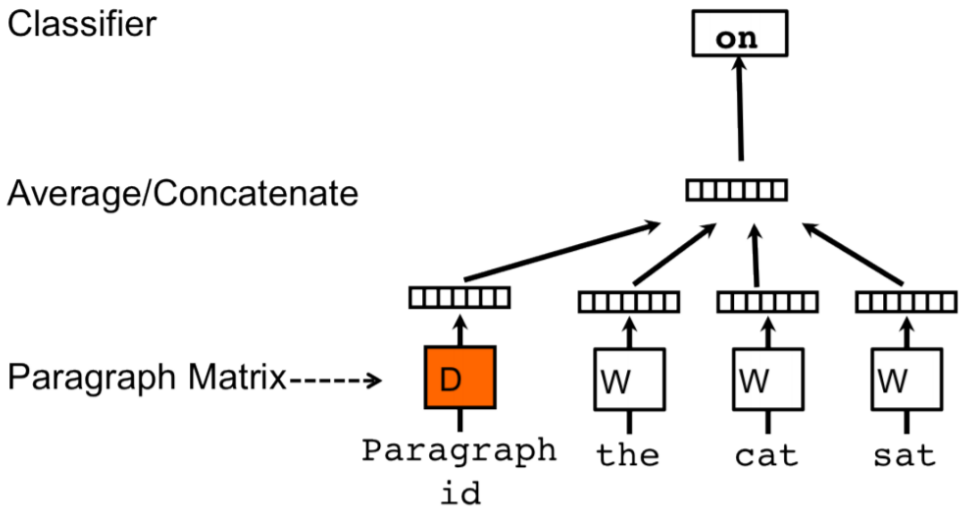

Fig.3.3. Distributed Memory Model of Paragraph Vectors (PV-DM)

Let's take a closer look at the model diagram for greater clarity. The presented model has parts for Paragraph Matrix, Average/Concatenate, and Classifier. The paragraph matrix is a matrix in which each column represents a paragraph's vector. If the word vectors and paragraph vectors are averaged or concatenated, this is referred to as Average/Concatenate. Finally, the Classifier portion predicts the centre word using the hidden layer vector (the one that was concatenated/averaged) as input. In the same

way that Word2Vec models learn embedding for words, Matrix D contains embedding for "seen" paragraphs (i.e. arbitrary length texts). To infer a document vector for unread paragraphs, the model is run through gradient descent (5 or so iterations).

3.4.2. PV-DBOW

The DBOW (Distributed Bag Of Words) model differs somewhat from the PVDM model. The DBOW model "ignores the context words in the input and drives the model to predict words randomly picked from the paragraph in the output," according to the researchers. Let's pretend that the model is learning by predicting two sampled words in the example above. As illustrated in the picture, two words from the phrase "the cat, sat, on, the, sofa" are sampled in order to learn the document vector.

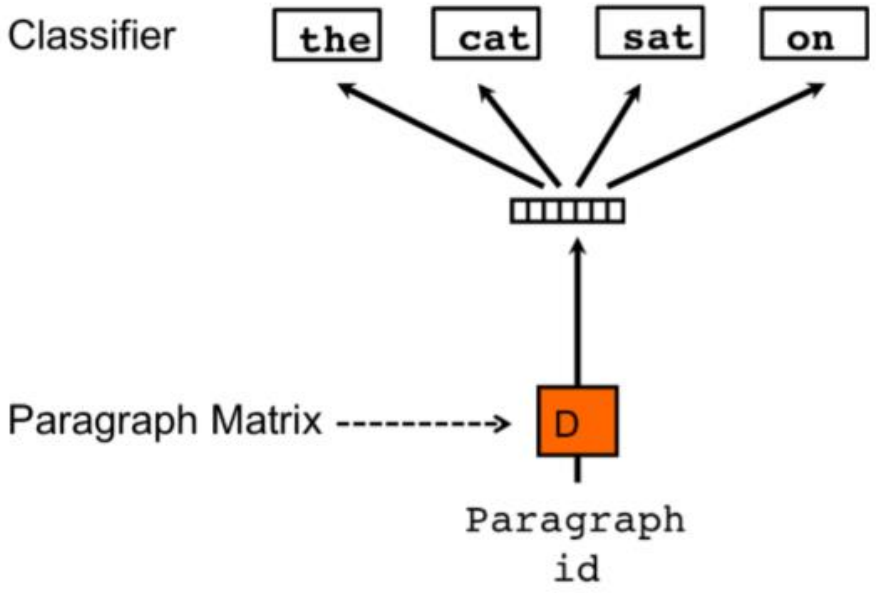

Fig.3.4 Distributed Bag of Words version of Paragraph Vector (PV-DBOW)

3.5. Sentence embeddings

The input to many machine learning algorithms must be expressed as a fixed-length feature vector. Word embedding are a representation of words in an N-dimensional vector space that, depending on the training technique, brings semantically similar (e.g. "king" — "monarch") or semantically related (e.g. "bird" — "fly") words closer (using words as context or using documents as context). When it comes to texts, the bag-of-words is one of the most popular fixed-length feature. But this method neglects a lot of information like the ordering and semantics of the words. For example, two sentences can have identical final representations but entirely different meanings; like 'I'm going to study Math instead of English' and 'I'm going to study English instead of Math'.

3.6. Sent2vec Model

Unsupervised Learning of Sentence Embedding using Compositional N-Gram Features, a new model for sentence embedding called Sent2Vec. Sent2Vec neural network that transforms a traditional GloVe embedding into a sentimental embedding representation. Sent2vec can be thought of as an extension of FastText model and word2vec (CBOW) to sentences. The average of the source word embeddings of its component words is the sentence embedding. This model is further enhanced by learning source embeddings for both unigrams and n-grams of words in each phrase, and averaging the n-gram embeddings alongside the words.

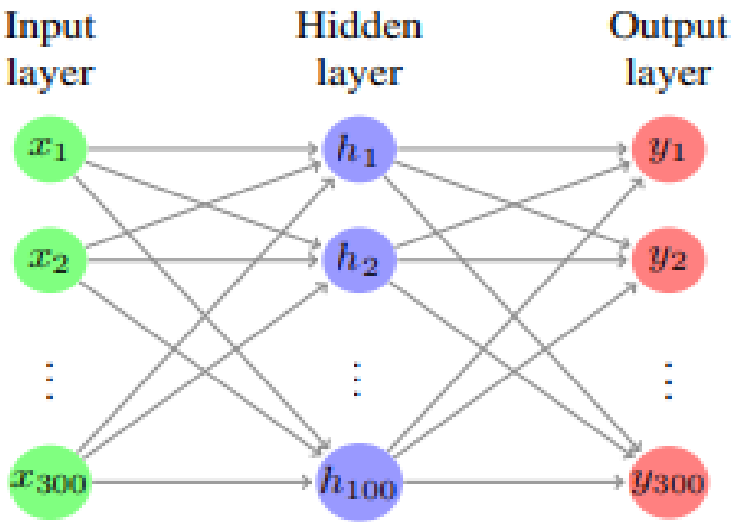

Fig.3.5 Sent2vec Model

The model receives a 300-dimension word embedding vector X as the input, where each dimension in X is a numeric value denoted by xi and 1 < i < 300:

$$\mathcal{X} \triangleq \{x_1, x_2, \dots, x_{300}\}$$

The network turns X into a sentimental 300-dimension vector Y where yi corresponds to a numeric value in the sentimental vector. We refer to Y as the true vector.

$$\mathcal{Y} \triangleq \{y_1, y_2, \dots, y_{300}\}$$

We set the hidden layer as a fully connected dense layer with 100 neurons after several configuration evaluations. We used relu and sigmoid activation functions for the hidden and the output layers, respectively. This can be explained as below:

$$\mathcal{H} = \mathcal{F}_{relu}(\mathcal{W}_1\mathcal{X} + \beta_1)$$

$$\mathcal{Y} = \mathcal{F}_{sigmoid}(\mathcal{W}_2\mathcal{H} + \beta_2)$$

where H is the output of the hidden layer, W and ß are weight matrices and bias vectors respectively. We have:

$$\mathcal{F}_{relu}(z) = \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases}$$

$$\mathcal{F}_{sigmoid} = \frac{1}{1 + e^{-z}}$$

The loss function to be optimized is the cross-entropy for the 300 dimension output, defined as below:

$$Loss = -\frac{1}{300}\sum_{i=1}^{300} y_i \,.\log \hat{y}_i + (1 - y_i).\log(1 - \hat{y}_i)$$

$\hat{y}_i$ is the i-th predicted scalar value by the model and yi is a numeric value in the output vector Y. Accordingly, the output of the neural network Y, is a vector of numbers between 0 and 1. The example below illustrates a hypothesis case that a sentimental embedding vector Yˆ is predicted by Sent2Vec neural network model for an input sentence embedding vector X.

3.7. The Dataset

The benchmark dataset used is short answer grading consists of seven assignments between four to six questions each. The total number of questions is 40. The experiments in this paper also were applied only to the 40 short answer questions. The number of students' answers per question ranges from 24 to 31 with an average of 25 answers and the total number of short answers is 1000.

These assignments/exams were provided at an introductory computer science class at the University of North Texas. Elements of the dataset are questions' texts, the reference answer for each question, and students' answers. The answers were graded by a grader. All three types of grades are in the range of 0 to 5. This research works on average grades following other researchers.

Sample Question 1: What is the role of a prototype program in problem-solving?

Reference Answer: To simulate the behavior of portions of the desired software product

Student 1 Answer: High-risk problems are addressed in the prototype program to make sure that the program is feasible. A prototype may also be used to show a company that the software can be possibly programmed.

Student 2 Answer: To simulate portions of the desired final product with a quick and easy program that does a small specific job. It is a way to help see what the problem is and how you may solve it in the final project.

Student 1 Marks: 4

Student 2 Marks: 5

Sample Question 2: What does a function signature include?

Reference Answer: The name of the function and the types of the parameters.

Student 1 Answer: It includes the name of the function and the types of its arguments.

Student 2 Answer: Name, parameters, scope, and other general function information

Student 1 Marks: 4

Student 2 Marks: 4

3.8. Experimental Results, Comparison

Evaluation Tasks: We evaluate doc2vec in two task settings. For all tasks, we split the dataset into 2 partitions: training and testing. Here in the case of our dataset first we take the reference answers as our training data and all the student's answers as our testing data. We use all documents in the training data and test set to train doc2vec. Our reasoning is that the doc2vec training is entirely unsupervised, i.e. the model takes the only raw text and uses no supervised or annotated information, and thus there is no need to hold out the test data, as it is unlabelled. At first, we tokenized every document/answer both reference answer and student's answer, then we convert all the tokenized reference answers into tagged document format where every document has a tag value starting from 0.

The value of all the hyper parameters are **(vector_size=40, min_count=2,epochs=30)** where

vector_size – Dimensionality of the feature vectors.

min_count – Ignores all words with a total frequency lower than this.

epochs – Number of iterations (epochs) over the corpus. Defaults to 10 for Doc2Vec.

After training doc2vec, document embeddings are generated by the model we calculate the similarity score for every tokenized student's answer and based on the similarity score we assign grades between1 and 5. But later we think all reference answers taken as training data and assigning a grade to every individual answer based on that is not the right way. Then we take all the student's answers as our training data and all the reference answers as our testing data. Now we don't take all the student's answers as a whole for training. In our dataset for every question, there are 25 different students' answers are available. So for a particular question, we take 25 students' answers at a time for training the model and calculate the similarity score for that particular tokenized reference answer of that question, and based on that we assign the grade between 1 to 5.

Now to evaluate the result we have all the assigned grades in a list and in our dataset, there are marks given by human graders. So we take all the human grader's marks in a list and calculate the correlation between two lists as a whole and for each question we calculate the correlation value and take the average of all 40 different correlation values.

Here we use 4 different methods for evaluation. The results are:

Pearsons correlation	**0.05103785627900118**
Spearmans-rank-correlation	**0.056668571956125034**
Kendall Rank correlation	**0.010366929842873301**
RMSE	**2.071687046340028**

Average correlation value using doc2vec model

In the sent2vec model for all tasks, we split the dataset into 4 partitions: each part containing 10 different reference answers and 250 student answers. Our rationale for this is that the doc2vec training is completely unsupervised, i.e. the model takes the only raw text and uses no supervised or annotated information, and thus there is no need to hold out the test data, as it is unlabelled. At first, we take all the reference answers in a list and all the student's answers inside another list and then we create an object of vectorizer. Now with the help of the run function, we pass the list of reference answers, and using the function of the vector we get vectors of all the reference answers. Then we do the same job for the student's answers to the vectors for all student's answers. Now for each question, we calculate the cosine distance of 25 different student answers with particular reference answers. Then we subtract the cosine distance value from1 to get the cosine similarity value answer and based on the cosine similarity score we assign a grade between1 and 5.

Now to evaluate the result we have all the assigned grades in a list and in our dataset, there are marks given by human graders. So we take all the human grader's marks in a list and calculate the correlation between two

lists as a whole and for each question we calculate the correlation value and take the average of all 10 different correlation values.

We do the same process for all the 4 different sets of data and we take the average of 4 results as the final result of the model.

Here we use 4 different methods for evaluation. The results are:

Pearsons correlation	**0.15712906804521312**
Spearmans-rank-correlation	**0.1819130703896988**
Kendall Rank correlation	**0.1401217629633598**
RMSE	**1.5026775433700141**

Average correlation value using sent2vec model

If we compare the results of our model with the previous work on automatic short answer grading, we will see that in some papers their proposed method consists of the following steps: pre-processing, dependency parsing, lexical alignment, categorization of tokens, dependency graph representation, graph matching and assignment of scores, calculation of grade. They got 0.627 (Assignment basis) and 0.366 (Overall basis)

In another paper, they Sentence BERT model for Automatic short answer grading, and to evaluate the effect of the model they evaluate our ASAG system using 3 measures.

Accuracy rate (ACC). Percentage of correctly scored answers. Macro Average F1 Score (M-F1). Calculate and discover the unweighted mean of each label's metrics. Weighted average F1 score (W-F1): Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance.

We can see that Sentence BERT obtained impressive results which reached an accuracy of more than 70 percent on the dataset, indicating that the pre-trained model has an excellent adaptation to the short answer scoring area. Using BERT model they got accuracy rate(ACC 0.73), Macro Average F1 Score(M-F1 0.61), Weighted Average F1 Score(W-F1 0.54).

3.9. Analysis of the Result

From the above results of our model, we see that using the doc2vec model we get a very low correlation value but using the sent2vec model we are getting better results. But then also the average of 18% is not so good. So to find the reason behind that result we observe that there are some of the questions in the dataset for which the correlation value is high but also there is some question for which the correlation value is low even though some of them are in negative value also. Now in this situation, if we observe the dataset we see that all the questions and answers present in the dataset are related to computer science. Here we are using doc2vce and sent2vec models which are giving very good results for English paragraphs and good similarities between English sentences. But in our dataset, there is a lot of question whose answers are in 2 or 3 sentences or 1 sentence for those the correlation values are good but there are some answers where the answers are one word or one phrase or some special keyword that have special meaning in programming aspect so these kind of question correlation values are very low. One more important thing is that both of our models are focusing on similarity scores. For example, there is a question like “What type of data structures are stack?” and an answer to the question like “Stack is LIFO data structures” but some student writes that “Stack is FIFO data structures” which is wrong but these two sentences have 85-90% similarity which is another big reason for getting low correlation value.

4. Conclusion

In this paper, we explored unsupervised techniques for automatic short answer grading. We believe the paper made some important contributions. First, there are several words and text similarity measures that have been proposed for the task of short answer grading.

This work shows that a pre-trained neural network model with a proper task function can achieve promising results in automatic short answer grading using only a small amount of data. Specifically, the Sent2vec model has a better performance in ASAG than doc2vec model. Since our model achieves a better result on the shorter answer dataset, we believe limiting the length of answers plays an important role in the model's performance. However, due to the black-box nature of deep learning, we are unable to explain the reason for scoring. It means that the model cannot interpret the process of identifying errors in answers. Additionally, we only tested the short answer question dataset in the computer science domain. In future research, we expect to validate our model by using datasets from other domains. It is also possible to further explore how to improve the performance of the model on longer textual short-answer question datasets.

REFERENCES

- C. Leacock and M. Chodorow. 2003. C-rater: Automated Scoring of Short-Answer Questions. *Computers and the Humanities*, 37(4):389–405.

- J.Z. Sukkarieh, S.G. Pulman, and N. Raikes. 2004. Auto-Marking 2: An Update on the UCLES-Oxford University research into using Computational Linguistics to Score Short, Free Text Responses. *International Association of Educational Assessment, Philadephia*.

- Uswatun hasanah from STMIK Amikom Purwokerto,Indonesia.

- S.G. Pulman and J.Z. Sukkarieh. 2005. Automatic Short Answer Marking. *ACL WS Bldg Ed Apps using NLP*.

- G. Salton, A. Wong, and C.S. Yang. 1997. A vector space model for automatic indexing. In *Readings in Information Retrieval*, pages 273–280. Morgan Kaufmann Publishers, San Francisco, CA.

- V. Hatzivassiloglou, J. Klavans, and E. Eskin. 1999. Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

- Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features by Matteo Pagliardini , Prakhar Gupta, Martin Jaggi

- Sent2Vec: A New Sentence Embedding Representation With Sentimental Semantic by Mahdi Naser Moghadasi, Computer Science Department Texas Tech University, Lubbock,TX, USA, Mahdi.Moghadasi@ttu.edu and Yu Zhuang, Computer Science Department Texas Tech University, Lubbock,TX, USA, Yu.Zhuang@ttu.edu

- An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation Jey Han Lau and Timothy Baldwin IBM Research and Dept of Computing and Information Systems, The University of Melbourne