Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 23 Issue 1 Version 1.0 Year 2023
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Machine Learning Algorithm to Detect Impersonation in an Essay-Based E-Exam

By Mr. Joseph Kombe Samuel, Dr. Peter Ochieng & Dr. Solomon Mwanjele

TaitaTaveta University

Abstract- Essay-based E-exams require answers to be written out at some length in an E-learning platform. The questions require a response with multiple paragraphs and should be logical and well-structured. These type of examinations are increasingly becoming popular in academic institutions of higher learning based on the experience of COVID-19 pandemic. Since the exam is mainly done virtually with reduced supervision, the risk of impersonation and stolen content from other sources increases. Due to this, there is need to design cost effective and accurate techniques that are able to detect cheating in an essay based E-exam. In this work we develop, train and evaluate real-time LSTM, RNN and GRU algorithms, and then benchmark the performance of the algorithms against other state-of-the-art models in the same study area of detecting cheating in exam in an E-learning environment. Based on a set threshold, the models alert on possible impersonation or stolen content if the discrepancy exceeds the threshold. The evaluation and benchmarking of the algorithms revealed that our GRU model has the highest accuracy of 98.6% compared to other models in similar studies.

Keywords: E-exam, BERT, LSTM, RNN, GRU, Essay, E-learning, machine learning, cheating, education.

GJCST-D Classification: FOR Code: 170203

Strictly as per the compliance and regulations of:

Machine Learning Algorithm to Detect Impersonation in an Essay-Based E-Exam

Mr. Joseph Kombe Samuel [α], Dr. Peter Ochieng [σ] & Dr. Solomon Mwanjele [ρ]

Abstract- Essay-based E-exams require answers to be written out at some length in an E-learning platform. The questions require a response with multiple paragraphs and should be logical and well-structured. These type of examinations are increasingly becoming popular in academic institutions of higher learning based on the experience of COVID-19 pandemic. Since the exam is mainly done virtually with reduced supervision, the risk of impersonation and stolen content from other sources increases. Due to this, there is need to design cost effective and accurate techniques that are able to detect cheating in an essay based E-exam. In this work we develop, train and evaluate real-time LSTM, RNN and GRU algorithms, and then benchmark the performance of the algorithms against other state-of-the-art models in the same study area of detecting cheating in exam in an E-learning environment. Based on a set threshold, the models alert on possible impersonation or stolen content if the discrepancy exceeds the threshold. The evaluation and benchmarking of the algorithms revealed that our GRU model has the highest accuracy of 98.6% compared to other models in similar studies.

Keywords: E-exam, BERT, LSTM, RNN, GRU, Essay, E-learning, machine learning, cheating, education.

I. Introduction

E-learning would be a type of learning that takes place through the use of electronic media (Janelli, 2018). In 1999, it was first used during a seminar on Computer-Based Training (CBT) systems. It's also known as "virtual" or "online" learning. E-learning is becoming a necessary component of modern education, demonstrating the significant role of ICT in the current process of teaching-learning (Soni, 2020). The growth of online devices has facilitated the delivery of information on E-learning platformsto students, wherever and whenever they need it. (Urosevic, 2019) stated that in the year 2017 there were approximately 23 million new online learners, boosting the total number of E-learners to 81 million globally. With the COVID-19 pandemic, most universities and colleges were shut down to stop the virus infection from spreading. This made the universities and colleges think of alternative teaching methods during the lockdown periodand thus increased the use ofE-learning (Almaiah, Al-Khasawneh & Althunibat, 2020).

Author α: Student, Department of Computing and Informatics, TaitaTaveta University. e-mail: joskombe@gmail.com
Author σ: Lecturer, Department of Computer Science and Technology, University of Cambridge. e-mail: po304@cam.ac.uk
Author ρ: Lecturer, Department of Computing and Informatics, TaitaTaveta University. e-mail: smwanjele@ttu.ac.ke

The existence of E-learning environment introduced the aspect of E-assessments. In assessing students, universities and colleges mostly use essay-based E-exams other than choice-based ones because they require students to support their arguments with evidence (Hashim et al., 2018). Essay-based E-exams require answers to be written out at some length in an E-learning platform. The questions require a response with multiple paragraphs and should be logical and well-structured (Frederiks, Derrington & Bartlett, 2021).

Adopting E-learning has made authentication of the identity of students and their work's authenticity during assessment critical to reduce academic dishonesty like impersonation and copy-pasting content from other sources (Okada, Whitelock, Holmes & Edwards, 2019). Authentication is also essential for quality assurance purposes in education, though, implementing e-learning to combat impersonation and plagiarism is a big challenge (AV & Rathi, 2021). Several studies have been done to build E-assessment and E-exam authentication schemes but still have never eliminated the impersonation problem.(Rathgeb et al., 2020) proposed biometric technology as a way of solving impersonation. However, biometric technology has been criticized for the high costs of purchasing special biometric sensors and if the subjects are outside the biometric sensor's capture area or do not make contact with the biometric sensor (Gomez-Barrero et al., 2021). (Tiong & Lee, 2021) proposed the E-cheating intelligent agent that focused on monitoring student's behaviors through their speed of answering questions. The system they proposed is more tailored for choice-based questions, therefore, not effective in essay-based exams. The mouse tracking technique proposed by (Sokout et al., 2020) could detect students' illegal acts during e-exam, but the technique couldn't work in real-time and failed to detect impersonation (Tzafilkou & Protogeros, 2020). Another important students' authentication in e-exam was the utilization of keystroke dynamics (Mattsson, 2020). (Mattsson, 2020) warned against the use of this approach in a production environment due to the hardware limitations and the varying flight times that can occur when capturing keystrokes on an E-learning platform.

The current approaches for curbing impersonation and plagiarism have faced challenges which include; some are tailored for choice-based questions, some cannot detect impersonation in real-

time environments, some require high-cost special biometric sensors, and subjects may fail to interact with the sensors, and some are not recommended for the production environment. In this research, we propose to tackle impersonation in an essay-based E-exam using machine learning technique where students' writing techniques will be analyzed and a unique pattern designed for each candidate. If the machine learning algorithm detects the possibility of impersonation, the student is locked out of the system until authenticated by an administrator. In summary this paper makes the following contributions:

1. We evaluate between RNN, LSTM and GRU algorithms to establish the one with high accuracy in detecting impersonation and plagiarism on an essay-based e-exam platform.
2. We benchmark the performance of the developed LSTM, RNN and GRU algorithms against other state-of-the-art models in the same study area of detecting cheating in exam in an E-learning environment.
3. We identify and propose a model with highest level of accuracy in detecting impersonation and ability to work on real time environment.

II. Related Work

Several studies have been recommended to detect impersonation in E-exams, (Okada et al., 2019). In their study, (Rathgeb et al., 2020) recommended biometric technologies to be integrated in to e-learning platforms to assure the presence of the actual student during e-exam. However, (Gomez-Barrero et al., 2021) stated that in the COVID-19 era, the use of surgical masks that cover the nose and mouth, as well as the indirect effects of strict hygiene measures taken to prevent the spread of the virus have negatively affected biometrics technology. In addition, biometric technology has been criticized for the high costs of purchasing special biometric sensors and if the subjects are positioned outside of a biometric sensor's capture area or cannot get in touch with the biometric sensor (Gomez-Barrero et al., 2021).

(Tiong & Lee, 2021) proposed an e-cheating intelligent agent that integrates IP detector and behavior detector protocols. The IP detector monitored the IP addresses of the students' devices. The behavior detector assessed the speed at which the students were answering questions, they were labeled as 'abnormal' if they moved too quickly or too slowly; otherwise were considered as 'normal'. It was tested in four deep learning algorithms, which were the DNN, DenseLSTM, LSTM and RNN. (Sokout et al., 2020) proposed a model to track the students' behavior using a mouse-tracking approach which utilizes Support Vector Machine (SVM) to classify and predict illegal activities committed by students. The classification resulted in the hidden reconstruction of mouse activity and clear space, which resulted to detection of actions like in-activity, copy or cut, paste, double-click, opening new tab and scrolling. These actions determined whether the student was cheating or not. During the online mid-term exam, the model correctly predicted that 94% of the students would cheat. Nonetheless, it is not possible to detect impersonation and does not notify the examiner of any potential cheating in real-time, (Tzafilkou & Protogeros, 2020).

(Mattsson, 2020) proposed a method that utilizes keystroke dynamics for student authentication in online examinations. A GMM-UBM was used to test and evaluate the method. This approach resulted in an Equal Error Rate (ERR) of 5.4% and an accuracy of 94.5%. Despite its high rate of accuracy, the author recommended that this solution not to be used in a production environment because of other factors like hardware limitations and the inconsistent flight times that can occur when capturing keystrokes on an e-learning platform. (Javed & Aslam, 2013) conducted a study that used the Visual Eye Tracking Algorithm that uses cameras to track a learner's eye movement. Visual Eye Tracking Algorithm detects online exam cheating by reviewing visual attention of an examinee depending on their concentration to the screen. The algorithm utilizes an intelligent alarm system to be used in examination environments to significantly reduce cheating based on the eye movement of the student. It is developed in a way that it can detect a human figure in the exam environment and use face detection and visual eye tracking recognition for authentication. The intelligent visual eye tracking algorithm ensures that examination is free and fair by capturing the face of the examinee and monitoring their eye movement. The system has the capability of edges detection and analysis of eye movement detection using Kalman filtration algorithm while human face is detected using Viola Jones algorithm due to excellent results produced even when using low resolution cameras. Every frame captured is processed and analyzed by comparing it with the previous frame, if there is a significant difference between the frames that satisfies the threshold outlined, an alarm is raised of noted cheating incident. The results showed that the algorithm can follow eye movement of an exam taker and performs pupil analysis with a success rate of 93% and a processing time of 0.9 seconds. This shows its effectiveness in maintaining eye movement and pupil analysis compared to other methods like Tree Classifier, SVM, and EOG algorithms. More research is recommended to include other factors like voice detection to ensure that the exam environment has unauthorized individuals to help in handling the exam.

Another study that used the visual analysis approach was by (Bawarith, Basuhail, Fattouh & Gamalel-Din, 2017). The approach combined a

fingerprint reader for authentication as well as an eye tribe tracker for visual analysis during exam sessions. The system worked in such a way that, the examinee was supposed to use fingerprint scanner to allow access into the system as well as tracker for eye tribe to ensure the authentic examinee is sticks throughout the assessment period. If the system noticed the examinee was absent it locked and required authentication using fingerprint. The study had 30 participants who were divided into two groups of 15 cheating and 15 non-cheating participants. Every participant was supposed to undertake three exam sessions so that the data sample equaled 90. The results showed that Sensitivity which assessed the proportion true positive rate to show that the system correctly identified the participants was 100 percent successful. The Specificity which measured the true negative rate to mean the correctly identified cheating instances was 95.56 %. Also, the Precision which measured the fraction of relevant retrieved instances which is the positive predictive value was 95.74% and the Accuracy which is the proximity of obtained results to the true value was 97.78 %. Overall the system had a success rate of 97.83%. The authors recommended that the research could be expanded by implementing it over the internet for distributed systems and include other features like voice detectors to improve accuracy.

Another study by (Chen & Chen, 2017) used Data Mining algorithms to identify cheating cases in exam rooms by observing patterns in answers provided by students. The study used multivariate statistics tools to observe association pattern in the answer sheets. Moreover, the Hierarchical Clustering and Dendrogram Tree algorithm were employed for clustering affinity behavior identified in the dataset. Heat Map was used to recognize patterns in the scores through visual analysis. The top 20 percent of the most difficult questions of the 25 multiple-choice queries were considered to improve the cheating detection power. The research study involved 75 students who were required to sit in groups of 3 per table to form 25 different small tables in a very limited classroom environment. The exam sheet was modified by the instructor to form three different versions, each student in a table was to handle a different version. This was done to identify synchronization attempts among the student when handling hard questions by marking similar answers for those questions, hence, providing evidence of cheating patterns. In addition, cell phones or laptops were not allowed during the exam session to limit communication among the students. The results after analysis using data mining strategies showed that data mining algorithms were able to identify with high prediction accuracy similar patterns displayed by the answers of students who were seated in the same table during the exam session. The data mining algorithms that played a critical role in obtaining the results were Heat Map, Principal Component Analysis (PCA), and Clustering Analysis. Multivariate Correlation had chances of wrong cheating detection since students can have similar answers, but their answering patterns should differ. (Cavalcanti, Pires, Cavalcanti & Pires, 2012) in their research study employed text mining methodology and algorithms to detect academic dishonesty (cheating) by evaluating open-ended college assessments using document classification techniques. The authors note that cheating in Brazilian public universities is a prevalent behavior that lacks a concrete solution, (Cavalcanti et al., 2012).

The study focuses on showing how text mining algorithms are a promising technique for finding a solution that not only detects cheating, but it also estimates cheating on open-ended exams. There are two types of classification techniques namely supervised classification when the information of the classes is already available and non-supervised classification when the information is absent. In this research study for detecting cheating on scholar exams was developed and administered to Business Management and Computer Science students at the Federal University of Campina Grande in Brazil. The case study had thirty scholar exams and each exam had four open-ended questions on administration and marketing written in the Portuguese language. The exams were handled by the selected students and answers were stored electronically in plain text format. Strong evidence of cheating was detected every time the program identified documents with high similarity index due to a large number of identical words, (Cavalcanti et al., 2012).

Decision Tree was used to detect and assess cheating on examinations by applying two classification models namely co-sine based and overlap based models in the supervised algorithm. Results showed that overlap based model performed better by attaining accuracy of approximately 99.43 percent which is an excellent inference quality for cheating detection and evaluation. The results of an overlap that considers two answers from the same question were defined to show that an overlap score of less than 0.22 indicated no cheating, between 0.22 and 0.2 indicated low cheating, a score between 0.3 and 0.38 meant intermediate cheating while a score of over 0.38 showed high cheating. This research study showed that text mining can be used for educational purposes to curb cheating by detection and evaluation mechanisms in a manner that can help a teacher to identify exam malpractices in labor-intensive evaluation tasks, (Cavalcanti et al., 2012).

Another text mining study by (Pertile, Moreira & Rosso, 2016) to analyze cheating in academic papers using plagiarism detector tools, 85 pairs for PubMed and 96 for ACL were considered. The case study employed 10 human assessors who were tasked with

the responsibility of detecting and reporting similarity cases noted. Pooling method had been used select the pairs from the huge pool of dataset in the PubMed and ACL databases. After assessing the evaluations from the assessors, the agreement rate was noted to be 84 percent for ACL and 80 percent for PubMed journals. ParsCit which uses supervised machine-learning method was employed to extract information from the selected scientific papers for content and reference analysis.

Results showed that the intersection of text in the documents from ACL collection ranged between 15 percent and 46 percent. On the other hand, intersection between the content and reference-based metrics was higher reporting a range of 23 to 61 percent. The larger intersection does not translate to higher plagiarism in the documents, but it shows common content and citations were used by the authors. The selected pairs were tested with machine-learning techniques and the findingsrevealed that hybrid decision-table/Naïve Bayes classifier had better results for ACL and a decision-tree classifier, J48 produced better results for PubMed. It was concluded that with a CF-Score of 0.8 for ACL and 0.9 for PubMed, there was high probability of plagiarism cases among PubMed than ACL journals, (Pertile et al., 2016).

Another study by (Kuin, 2018) sought to explore three convolutional neural networks to identify fraudulent behavior among students in digital platforms like Canvas or BlackBoard using screen recordings. The study conducted by (Kuin, 2018) proposes the creation of a framework that permits students to handle an assessment using resources of their choice that includes search engines such as Bing or Google. The proposed framework has three parts namely an interface, frame classification, and a video processor. The interface captures and sends student's screen recordings as videos to a pipeline with a series of classification methods. The pipeline executes video processing by shortening long videos into several thousand frames while frame classification creates a series of methods to receive the processed videos. The frame classification categorizes these frames, compiles the results, and sends to the supervisor's interface indicating instances of fraudulent behavior, (Kuin, 2018).

The study used screen recordings of three digital exam sessions of two hours long in an environment that gave student the freedom to chat on social media and write notes. The videos recorded are then converted to frames that are labeled either fraud or not fraud using ANVIL tool for the collection of images for training the neural network. The total number of frames used was 25,000 images that were divided into 3 categories namely train, validate and test set and allocated with 50 percent, 25 percent, and 25 percent respectively. The model was able to categorize the 25,000 photographs into 12,000 images identified as fraud and 13,000 images categorized as not fraud, according to the results (Kuin, 2018). The results show that VGG16 yields 96.8% accuracy on traditional approach while the cross-validation technique produces 67.1% accuracy. Likewise, Inception-v4 produces results that show 96.0% accuracy when using the traditional approach while showing 46.8% accuracy for the cross-validation method. Lastly, MobileNets which produced the underwhelming results in comparison with the other two by showing a precision of 48.8% when using a conventional approach and 48.2 percent using the cross-validation method (Kuin, 2018).

Hence, the created framework for detecting fraudulent behavior in online platforms was VGG16 and Inception-v4 approaches due to their high accuracy levels of over 96.8 percent. However, more study is recommended so that the framework can be able to guarantee that the right student is undertaking the tests and monitor the exam environment to ensure that the student is handling the exam alone, (Kuin, 2018).

Most of the existing studies have approached the issue using machine learning approaches that analyze the behavior of the users to ascertain their authenticity while taking the exams, (Hu, Gingrich & Sentosa, 2008). Most approaches use computer features such as webcams, screenshots, video recording, and text analysis. However, although all online learners use keyboards for input, few studies use keystroke approach in finding solution to the cheating challenges in E-learning platforms (Shilton & Greene, 2019). Hence, there is room for advancing the existing researches by developing new models that would effectively handle cheating behaviors in online platforms with a higher accuracy and effectiveness. Recurrent neural networks approaches can be employed in developing an adaptive algorithm to predict essay-based e-exams because they are capable of guessing the next symbol in a series of symbols (Brownlee, J., 2018). They can learn the sequences of a problem and then build totally new plausible sequences for the problem area, in addition to providing predictions (Pérez-Ortiz, J. A., Calera-Rubio, J., &Forcada, M. L., 2001). Therefore, this research study wishes to employ RNN, LSTM and GRU networks as unique methods for detecting cheating during online exams.

III. Methodology

a) Experimental Setup

i. *Dataset*

A collection of words in a form of essay was required for training the models in this research. This set of words represents writing of essay by students in an actual e-exam. Therefore sample data was retrieved from a pool of existing data (at http://www.statmt.org/wmt14/training-monolingual-news-crawl/). Data was in text form in a text file of size 286 MB with English words. This dataset represents dummy writings of students.

The dataset was used to train the models in two scenarios; in the first one it was utilized in the word-level RNN, LSTM as well as GRU models and the other one in character-level RNN, LSTM as well as GRU models. Therefore, the same dataset was used in both cases, where in the first instance, the data for training was divided into words, so, the dataset contained 453668 unique English words. In the second instance the training data gotdivided to 285579163 English characters. These datasets provided us with enough words and characters to evaluate the effectiveness of our models.

ii. *Data pre-processing*

In data preprocessing, we use spaCy (https://spacy.io/) for natural language processing because the data must be represented in a computer-readable format. SpaCy is a Python-based NLP library that comes with a lot of capabilities in-built within.

When the dataset is passed into spaCy, the following tasks are performed:

Sentence Detection. The beginning and ending of sentences in the dataset are defined here, allowing the text to be divided into linguistically units of meaning.

After sentence identification, the next stage is tokenization. It enables you to recognize the text's basic units. Tokens are the fundamental units. Tokenization is beneficial since it divides a text into logical components.

The other step is Lemmatization. This is when a word's inflected forms are reduced while still guaranteeing that the reduced form is linguistically acceptable. Organized, organizes, as well as organizing, for example, all are synonyms for organize. Organize is the lemma in this.

Part of speech (POS) tagging is the next step. Each token is given a POS tag based on how it is used in the phrase. The interjection, conjunction, preposition, adverb, verb, pronoun, adjective as well as noun are the eight components of speech. Each word can be assigned a syntactic category using POS tags.

One of the phases in extracting data from a dataset is rule-based matching. It was used to discover and extract tokens and trends based on grammatical characteristics and patterns, like lowercase, as part of speech.

Another step in extracting a sentence's dependency parse to describe its grammatical structure is dependency parsing. It establishes the relationship between headwords and their subordinates. Dependency parsing reveals a word's role in a text as well as how different words are related to one another.

Another process performed is Named Entity Recognition (NER). This is the act of finding identified entities inside an unstructured dataset as well as classifying them into pre-defined classes like organizations' names, people's names, percentages, places, time expressions, and monetary amounts, among other things.

Finally, the dataset is represented as a sequence of integer values, with each word in the text file having its own integer value. This process was achieved using neural network embedding layer. This allowed the text data to be consumed in the neural networks.

iii. *Model Parameters*

This involved setting up necessary parameters required in machine learning modelling. They include batch size which refers to the size of each batch of data that is fed into the models. This is set to 64. Batch size of 64 was established as an optimum batch size that led to faster convergence of the training. Embedding size is the other parameter which represents the number of features in the dataset and is set to be 256. Sequence length parameter corresponds to the number of iterations the dataset is run through our model, it is set to 50. The buffer size parameter is set to be 10,000. The last parameter is neurons which corresponds to the size of the hidden state of the models and is set at 1024. The learning rate of 0.001 was defined for this research.

b) Baseline Models

We make comparisons of our model with other state-of-the-art models in this section. We refer to our model as Impersonation Detector. The performance of Impersonation Detector will be compared with the performance of:

1. E-cheating intelligent agent that integrates IP detector and behavior detector protocols. Its performance was tested using DNN, Dense LSTM, LSTM and RNN (Tiong & Lee, 2021).
2. Mouse-Tracker that uses a mouse-tracking technique to track students' behavior and uses a SVM to categorize as well as predict pupils who engage in illegal behavior (Sokout et al., 2020).
3. Keystroke dynamics for student authentication in online examinations. This model was tested and evaluated using GMM-UBM (Mattsson, 2020).

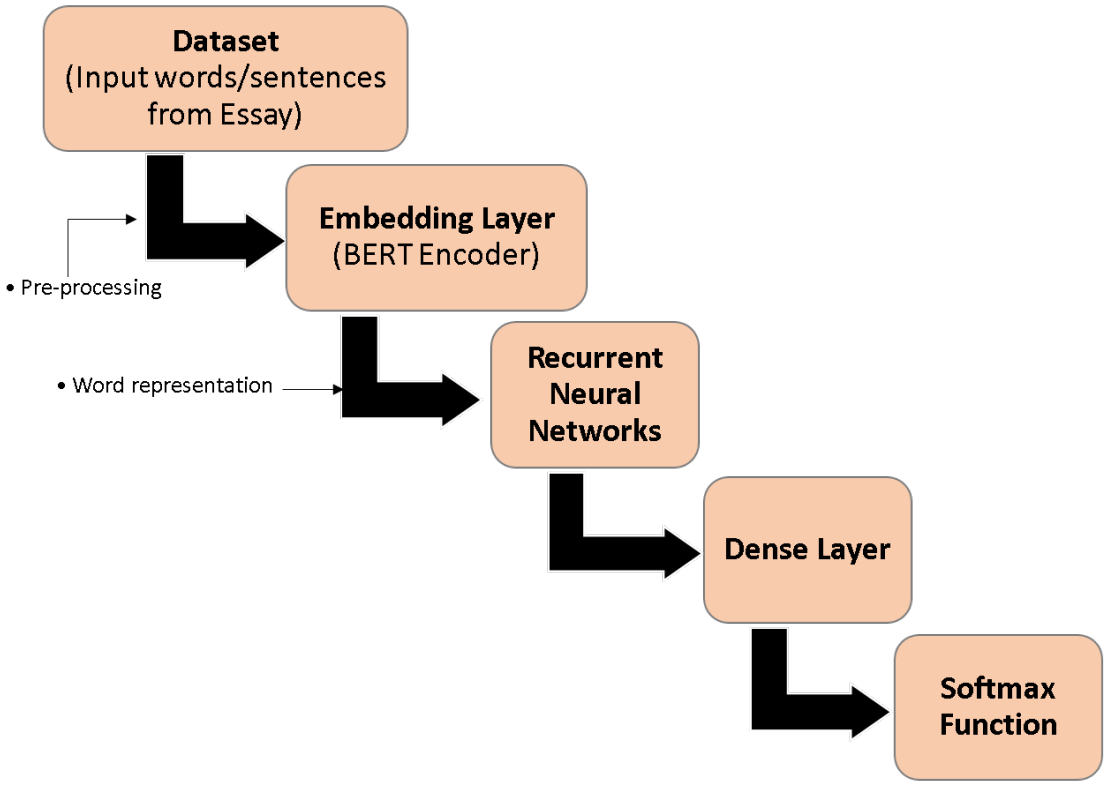


Figure 1: Proposed Model

i. *Dataset Layer*

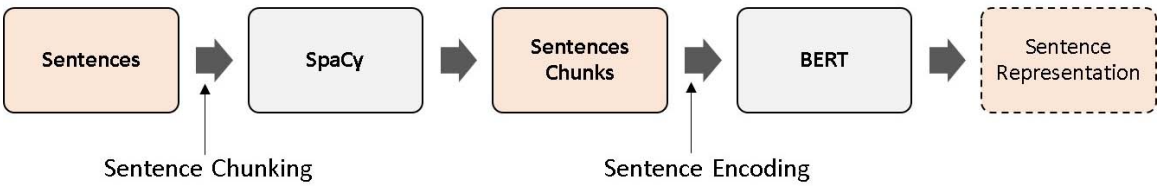
This is the first layer in the model. It provides the model with set of words or sentences in a form of an essay for training and testing.

ii. *Embedding Layer*

This is the second layer in the model in all the neural networks designs (i.e. RNN, LSTM and GRU). It involved word embedding, which is the process of learning a representation for text in which words with related meanings are represented similarly (Brownlee, 2017). To perform embedding we exploited Bidirectional Encoder Representations from Transformers (BERT) framework (Guo et al., 2020).

iii. *Sentence Encoding*

Here, we encode the whole sentence using BERT. Given a document described in section II, we use SpaCy (https://spacy.io/) for sentence chunking. For each sentence we fine-tune BERT to generate the sentence representation. For each sentence BERT generates a vector of dimension i.e. R^ddimension. BERT uses a dimension of 512.



iv. *Word Encoding*

Here, we encode words as opposed to the whole sentence using BERT. Given a document described in section II, SpaCy (https://spacy.io/) is used to extract sentences. The sentences are then fed into BERT. BERT now configured to generate the word level representation of a given sentence. In this case BERT generates a matrix of dimension dxn where d is a word's vector representation and n is the words' number in a sentence i.e. R^dxn dimension. BERT uses 512.

Recurrent Neural Networks Layer

v. *RNN Layers*

A RNN is a feed-forward neural network with internal memory. It is recurrent in nature as it performs the same operation for every data input, and also the outcome of the current input is determined by the prior computation (Mittal, 2020). When the output is formed, it is duplicated then relayed back into the recurrent network. When determining a decision, it considers the current input as well as the output it has learned from previous input.

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}) \end{aligned}$$

(Goodfellow, Bengio & Courville, 2016)

where *h(t)* is the network's "memory" and reflects a hidden state at time *t*, *o(t)* denotes the output of the network and *y(t)* represents the network targets, at step of time*t*, the input of the network is *x(t)*. The bias vectors b and c, along with the weight matrices *U* for input-to-hidden, *V* for hidden-to-output, as well as*W* for hidden-to-hidden connections, are the remaining parameters.

In this research, the RNN model was implemented with three hidden layers. The first and second layers had 400 neurons each and the third layer had 224 neurons.

LSTM Layer

RNNs have a problem with short-term memory.If the series is lengthy enough, they will have difficulties transmitting information from earlier time steps into the later ones. During back propagation, the vanishing gradient challenge impacts RNNs. The vanishing gradient challenge occurs whenever a gradient decreases as it propagates backwards in time. Whenever a gradient value goes below a specific level, it ceases to be useful in learning (Nguyen, 2018). In rnns, layers that receive a minor gradient increase stop learning. All these are usually the first layers to appear. Since this layers don't learn, rnndo not remember what they've been through in lengthier sequences, leading to a short-term memory.

LSTMs are a more refined kind of rnns which make it much easier to remember prior eventsby introducing into the network a memory unit known as a cell (Yan, 2016). LSTM networks have internal unitsreferred to as gates which control the flow of information. The gates can determine which information inside a series should be retained as well as which should be deleted. It has the ability to send pertinent data along a long chain of series in this way, allowing it to make predictions (Nguyen, 2018). Here, the RNN's vanishing gradient problem is solved. LSTM is highly adapted to categorize, analyze, and forecast unexpected time gaps in time series. The model is trained via back-propagation.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

Where it stands for input gate, f_t for forget gate, o_t for output gate, a for sigmoid function, and w_x for weight for the gate(x) neurons, ht-1 represents the preceding LSTM block's output (at timestamp t-1), x_t represents the current timestamp's input, and b_x represents the relevant gates(x)'s biases (Olah, 2015).

The LSTM model was developed with two instances, with one layer and with two layers. This was done to determine the effectiveness of the LSTM when the depth of the network is increased by one layer. In the first instance, the LSTM layer had one hidden layer with 1024 as number of nodes within the LSTM cell. The second instance, the LSTM layer had two hidden layers. There were 800 nodes in the first hidden layer and 224 nodes in the second hidden units.

GRU Layer

(Cho et al., 2014) suggested a gated recurrent unit that enables every recurring unit to gather relationships throughout temporal scales in an adaptable manner. The GRU, just like LSTM, features gating units which control the information flow within the unit, without the discrete memory cells. Both GRU and LSTM networks are capable of capturing long and short term dependencies in sequences, however GRU networks have fewer parameters and are hence faster to train.

A reset and update gate is a concept in a GRU network that helps guarantee memory isn't taken over by tracking short-term dependencies.

The following formula is used for calculating updating gate z_t for timestep t

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

Update gate, (Kostadinov, 2017)

Whenever x_t is connected to a network layer, $W^{(z)}$ which is it's own weight is used to be multiplied with x_t. The same is true for h_{t-1}, which stores data for prior t-1 units as well as being multiplied by its weight $U^{(z)}$. By using sigmoid activation function, the values are combined and the outcome is squeezed between 0 and 1.

This reset gate is utilized from the model to decide how much past data to forget and is calculated using the formula:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

Reset gate (Kostadinov, 2017)

This is the same formula as for the update gate. The weights and how the gate is used are the key differences. $h_{(t-1)}$ and x_t are connected to the model, multiplied by their respective weights, summed, and the sigmoid function is applied.

To calculate a fresh content in the memory whichutilizes the reset gate to retain pertinent historical data, this formula is used:

$$h_t^{'} = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

Current memory content (Kostadinov, 2017)

Finally, the network calculates the h_t vector, which carries the information for the current layer and feeds it downwards to the next network unit using the updatinggate. It is done as follows:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t^{'}$$

At the current time step in the final memory (Kostadinov, 2017)

In its implementation, the GRU layer had two instances as well. The first instance was where we had the GRU layer with one hidden layer and we used 1024 neurons or nodes within the GRU cell. In the second instance we had GRU layer with two hidden layers. The first hidden layer had 800 neurons and the second one had 224 neurons.

Dense Layer

This is the last layer in RNN model. The dense layer learns a weight matrix, where the first dimension of the matrix is the input data's dimensionality, and the second one is the output data's dimension. The layer utilizes activation functions to convert input signal of nodes in a neural network to an output signal (Walia, 2018).

Here, softmax action function was used. The Softmax function converts numbers into one-to-one probabilities and returns a vector that describes the probability distributions of a set of possible outcomes (Kouretas & Paliouras, 2019). The function is typically used to calculate predicted losses when training a dataset. The input shape of the layer was (453668,) with the number of neurons was 1024.

IV. Results and Discussion

a) Data Analysis

The operating system used for this experiment was Windows 10 64bit Professional Edition running on a computer hardware with a processing power of Intel(R) Core(TM) i5-6400 CPU @ 2.70 GHz, 2712 MHz, 4 Core(s), 4 Logical Processor(s). The hardware had physical random access memory of 8 GB and 9.74 GB of total virtual memory.

The RNN, LSTM and GRU models were developed using Tensorflow (https://www.tensorflow .org/).With these working environment specification, the time taken to preprocess data was 12.95 seconds.

Table 4.1 below shows percentage of detection accuracy of the three models, i.e. RNN, LSTM as well as GRU, after we trained and tested them for the same period of time. The epochs for each model in the specified period of time was not considered since our interest was in detection accuracy against the time taken for the model to get to optimum performance.

Table 4.1: Evaluated results of RNN. LSTM and GRU Models

Model Name	Accuracy at					
	5 minutes	20 minutes	40 minutes	60 minutes	80 minutes	100 min.
RNN models	8%	42.6%	67.7%	78.8%	82.9%	83.9%
LSTM models	21.8%	54.9%	77.6%	86.3%	92.2%	92.3%
GRU models	32.4%	56.4%	82.5%	86.6%	97.7%	98.6%

The initial detection accuracy percentage for each model was recorded after five minutes and the subsequent accuracy percentage were taken in the intervals of 20 minutes during the evaluation, as per the above table. There was no significant change in detection accuracy percentage after 100th minute in all the three models (RNN, LSTM as well as GRU).

In the first 5 minutes the RNN recorded accuracy of 8%, the LSTM recorded 21.8% accuracy and the GRU model achieved 32.4% accuracy. The highest detection accuracy percentage for all the three models were obtained after 100 minutes. With a precision of 98.6%, the GRU model was the most accurate. The LSTM model was the second highest and demonstrated a comparable accuracy of 92.3% in detecting impersonation in an essay-based e-exam. In contrast, the performance of the RNN model was the lowest with accuracy score of 83.9%.

V. Discussion of Results

The performance accuracy of RNN model was the lowest at 83.7% detection accuracy compared to other models because it has short-term memory problem. Whenever a series gets too long, the RNN model struggles to carry information from earlier to later time steps. The Vanishing Gradient Problem is the name for this weakness. To train an RNN model, you back-program the network through time step, then calculate the gradient at every time step. This gradient is being used to update training weights of the network. If the

previous effect of the layer on the current layer is low, then the gradient value would be low, and conversely. Whereas if gradient of the previous layer gets smaller, then gradient of the following layer will become even smaller. These gradients would decrease drastically while back-propagating. A lesser gradient would have no impact on weight updating. Therefore, the network doesn't really remember how prior inputs affect it,and due to this, the short-term memory lossoccurs. The equation used to calculate the weight at any given time as show below:

$$h_t = \tanh\left(W_{hh}\, h_{t-1} + W_{xh} x_t\right)$$

Where h_t refers to the state of the memory at time$t$, and x_t refers to the impute at time t.

The best detection accuracy of GRU and LSTM modelsachieved at 98.6% and 92.3% respectively is attributed to the presence of memory cells which allow retention of any data without a lot of loss. They also have gates, which aid in the regulation of information flow to the cell state, and the gates can learn which data in a sequence is relevant and which is not.

The cell state $h_t$is equal to the output at time$t$, which is the first thing we see in a GRU cell.

$$\tilde{h}_t = \tanh(W\,.\,[r_t * h_{t-1}, x_t])$$

The updated value or candidate that can replace the cell state at time t is shown in the preceding equation. It is determined by the cell state at the previous time step h_{t-1} and a relevance gate calledr_t, which determines the significance of the previous cell state in the calculation of the present cell state.

$$r_t = \sigma\left(W_r\,.[h_{t-1}, x_t]\right)$$

As can be seen, the relevance gate r_t has a sigmoid activation with a value of 0 to 1, which determines how relevant previous information is, and is subsequently employed in the candidate for the updated value.

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

The updated candidate $\tilde{h}_t$ is a filtered mixture of the prior cell state h_{t-1} and the current cell state h_t. The update gate z_t determines how much updated candidate is required to calculate the current cell state, as well as how much of the prior cell state is preserved.

$$z_t = \sigma(W_z\,.[h_{t-1}, x_t])$$

The update gate, like the relevance gate, is a sigmoid function that aids the GRU in retaining the cell state for as long as it is required. This is how a GRU stores memory, avoiding the Vanishing Gradient Problem that RNN models suffer from.

While the basic concept is the same, an LSTM is a more complicated network. The LSTM has three gates: the forget gatef_t, the update gate i_t, and the output gateo_t. The GRU has two gates: the update gate and the relevance gate. The cell state was equivalent to the activation state/output in GRU, but they aren't quite the same in the LSTM. $h_t$represents the output at time t, and C_t represents the cell state.

$$\tilde{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C)$$

The cell state at time t has a candidate value $\tilde{C}_t$ that depends on the previous outputh_{t-1} and the input x_tt, just like in GRU.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

In LSTM, the current cell state C_t is a filtered version of the previous cell state and candidate value, just like in GRU. The filter is determined by two gates, the update gate and the forget gate, in this case. The value of $(1 - \text{updateGate}_t)$ in GRU is quite similar to the forget gate. Sigmoid functions are used in both the forget gate and the update gate.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

The forget gate determines how much of the previous cell state's information is necessary in the current cell state.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$

In the current cell state, the update gate calculates how much of the candidate value $\tilde{C}_t$ is necessary. Between 0 and 1 is the value of both the update and forget gates.

$$O_t = \sigma(W_O.[h_{t-1}, x_t] + b_O)$$

$$h_t = O_t * \tanh(C_t)$$

Finally, we must determine what we will produce. This is a filtered representation of our current cell state. As a result, we run the cell state via a $tanh$layer to push the values between -1 and 1, then multiply it by a sigmoid activation output gate to ensure that we only output what we want.

GRU models are considerably simpler and require less computing power, thus they can be used to construct incredibly deep networks, however LSTM models are more powerful since they have a larger number of gates, but they demand a lot of computer effort.

The results of the RNN, LSTM, and GRU models' performance are distributed in curved graph to show their trend as shown in Figure 4.1 below:

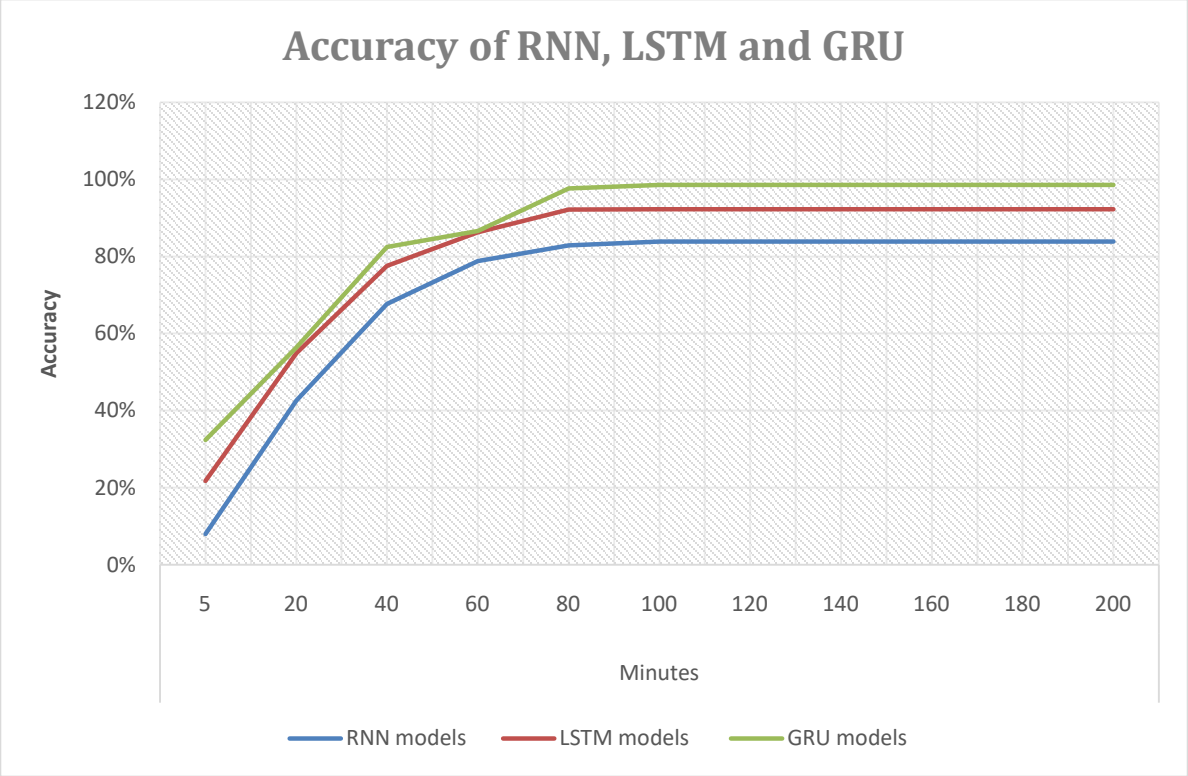


Figure 4.1: Performance comparison of RNN, LSTM and GRU Models

a) Comparison with other state of art models

The performance of the RNN, the LSTM and the GRU models in detecting impersonation in an essay-based e-exam was benchmarked against similar studies on detecting cheating in online examinations. The comparison is based on performance accuracy of the models used in each study. A table comparing the accuracy of different studies in detecting cheating in e-exams is provided below.

Table 4.2: Comparison Models Accuracy in Similar Studies

Tool	Reference	Model Name	Best accuracy (%)
Impersonation Detector	This study	RNN models	83.9
		LSTM models	92.3
		GRU models	98.6
E-cheating intelligent agent	(Tiong & Lee, 2021)	Dense LSTM	95.32
Mouse-Tracker	(Sokout et al., 2020)	Support Vector Machine (SVM)	94
Keystroke dynamics	(Mattsson, 2020)	Gaussian Mixture Models with Universal Background Model	94.5

From this benchmarking, we find out that our GRU model has the highest accuracy of 98.6% compared to other models in similar studies. This makes our tool, Impersonation Detector, perform the best in detecting cheating in an online exam.

VI. Conclusion

The primary goal of this study was to develop a machine learning algorithm to detect impersonation and plagiarism during an essay-based exam in an E-learning environment. To achieve this, essays written by students were required to train the models. A text file with a predefined collection of English words was retrieved from an online pool of existing data to represent the students' inputted words. Then, a real-time LSTM, RNN and GRU models were developed to detect impersonation and plagiarism in an essay-based exam using students' inputted words and characters extracted from those words.

The models were tested and evaluated using the time taken by each epoch in order to determine which model between RNN, LSTM and GRU resulted into the best detection performance. Finally, the best performing model from this was compared to other state-of-the-art best performing models in the same research area. This study found out that GRU our best performing model turned out to have achieved the highest performance accuracy of 98.6% in detecting

impersonation and plagiarism in e-exam. We could, therefore, conclude that the developed and trained machine learning algorithms are able to detect impersonation in an essay-based e-exam with detection accuracy of 98.6%.

In this research, we used a pre-existing dataset for training the models retrieved from an online pool of predefined text files with collection of words. Future studies can look into using data typed by actual students in a real online exam. This will allow researchers in this field to assess the performance of various approaches while avoiding errors caused by biases in the training data.

References Références Referencias

1. Aaron, L. S., & Roche, C. M. (2013). Stemming the tide of academic dishonesty in higher education: It takes a village. Journal of Educational Technology Systems, 42(2), 161-196.
2. Almaiah, M. A., Al-Khasawneh, A., & Althunibat, A. (2020). Exploring the critical challenges and factors influencing the E-learning system usage during COVID-19 pandemic. Education and Information Technologies, 25, 5261-5280.
3. Amidi, A., & Amidi, S. (2019). Recurrent Neural Networks cheatsheet: Cited 28-03-2019. Available at: https://stanford. edu/% 7Eshervine/teaching ….
4. AV, S. K., & Rathi, M. (2021). Keystroke Dynamics: A Behavioral Biometric Model for User Authentication in Online Exams Research Anthology on Developing Effective Online Learning Courses (pp. 1137-1161): IGI Global.
5. Bawarith, R., Basuhail, A., Fattouh, A., & Gamalel-Din, S. (2017). E-exam cheating detection system. Int. J. Adv. Comput. Sci. Appl, 8, 176-181.
6. Britz, D. (2017). {nd}. WILDML, Recurrent Neural Networks Tutorial, Part1-Introduction to RNNs.
7. Brownlee, J. (2017). How to use word embedding layers for deep learning with keras. 4 October 2017.
8. Cavalcanti, E. R., Pires, C. E., Cavalcanti, E. P., & Pires, V. F. (2012). Detection and evaluation of cheating on college exams using supervised classification. Informatics in Education, 11(2), 169-190.
9. Chalapathy, R., Menon, A. K., & Chawla, S. (2018). Anomaly detection using one-class neural networks. arXiv preprint arXiv:1802.06360.
10. Chen, M., & Chen, C. (2017). Detect Exam Cheating Pattern by Data Mining. Paper presented at the FSDM.
11. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
12. Frederiks, A., Derrington, K., & Bartlett, C. (2021). Types of Exams. Academic Success.
13. Gomez-Barrero, M., Drozdowski, P., Rathgeb, C., Patino, J., Todisco, M., Nautsch, A., . . . Busch, C. (2021). Biometrics in the Era of COVID-19: Challenges and Opportunities. arXiv preprint arXiv:2102.09258.
14. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning: MIT press.
15. Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM networks. Paper presented at the Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.
16. Guo, W., Liu, X., Wang, S., Gao, H., Sankar, A., Yang, Z., . . . Chen, B.-C. (2020). Detext: A deep text ranking framework with bert. Paper presented at the Proceedings of the 29th ACM International Conference on Information & Knowledge Management.
17. Hashim, H., Yunus, M. M., Yusuf, N. S. M., Zanzuri, N. A. H., Ruslee, M. H., & Fakhruddin, S. M. (2018). Factors Influencing Students' Selection of Different Types of Essay in Examination. Creative Education, 9(14), 2334.
18. Hernándeza, J.-A., Ochoab, A., Muñozd, J., & Burlaka, G. (2006). Detecting cheats in online student assessments using Data Mining. Paper presented at the Conference on Data Mining| DMIN.
19. Hu, J., Gingrich, D., & Sentosa, A. (2008). A k-nearest neighbor approach for user authentication through biometric keystroke dynamics. Paper presented at the 2008 IEEE International Conference on Communications.
20. Janelli, M. (2018). E-learning in theory, practice, and research. **Вопросы образования**(4 (eng)).
21. Javed, A., & Aslam, Z. (2013). An intelligent alarm based visual eye tracking algorithm for cheating free examination system. International Journal of Intelligent Systems and Applications, 5(10), 86.
22. Kahiigi Kigozi, E., Vesisenaho, M., Hansson, H., Danielson, M., & Tusubira, F. (2012). Modelling a peer assignment review process for collaborative e-learning. Journal of Interactive Online Learning, 11(2), 67-79.
23. Kasliwal, G. (2015). Cheating Detection in Online Examinations.
24. Kausar, S., Huahu, X., Ullah, A., Wenhao, Z., & Shabir, M. Y. (2020). Fog-assisted secure data exchange for examination and testing in e-learning system. Mobile Networks and Applications, 1-17.
25. Kopun, D. (2018). A Review of the Research on Data Mining Techniques in the Detection of Fraud in Financial Statements. Journal of Accounting and Management, 8(1), 1-18.

26. Korman, M. (2010). Behavioral detection of cheating in online examination.
27. Kostadinov, S. (2017). Understanding GRU networks. Towards Data Science. Towards Data Science, Towards Data Science, 16.
28. Kouretas, I., & Paliouras, V. (2019). Simplified Hardware Implementation of the Softmax Activation Function. Paper presented at the 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST).
29. Kuin, A. (2018). Fraud detection in video record-ings of exams using Convolu-tional Neural Networks.
30. Le, N. Q. K., Yapp, E. K. Y., & Yeh, H.-Y. (2019). ET-GRU: using multi-layer gated recurrent units to identify electron transport proteins. BMC bioinformatics, 20(1), 377.
31. Mattsson, R. (2020). Keystroke dynamics for student authentication in online examinations.
32. Mittal, A. (2020). Understanding RNN and LSTM. URL: https://towardsdatascience. com/understanding-rnn-and-lstm-f7cdf6dfc14e. Accessed, 01-08.
33. Nguyen, M. (2018). Illustrated Guide to LSTM's and GRU's: A step by step explanation. URL https://towardsdatascience.com/illustrated-guide-to-lstms-and-grusa-step-by-step-explanation-44e9eb85bf21.
34. Nicholson, C. (2018). A Beginner's Guide to Neural Networks and Deep Learning: The Artificial Intelligence Wiki https://skymind. ai/wiki/neural-network ….
35. Okada, A., Whitelock, D., Holmes, W., & Edwards, C. (2019). e-Authentication for online assessment: A mixed-method study. British Journal of Educational Technology, 50(2), 861-875.
36. Olah, C. (2015). Understanding lstm networks.
37. Pertile, S. d. L., Moreira, V. P., & Rosso, P. (2016). Comparing and combining C ontent-and C itation-based approaches for plagiarism detection. Journal of the Association for Information Science and Technology, 67(10), 2511-2526.
38. Rathgeb, C., Pöppelmann, K., & Gonzalez-Sosa, E. (2020). Biometric Technologies for eLearning: State-of-the-Art, Issues and Challenges. Paper presented at the 2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA).
39. Shilton, K., & Greene, D. (2019). Linking platforms, practices, and developer ethics: Levers for privacy discourse in mobile application development. Journal of Business Ethics, 155(1), 131-146.
40. Sokout, H., Purnama, F., Mustafazada, A. N., & Usagawa, T. (2020). Identifying potential cheaters by tracking their behaviors through mouse activities. Paper presented at the 2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE).
41. Soni, V. D. (2020). Global Impact of E-learning during COVID 19. Available at SSRN 3630073.
42. Tiong, L. C. O., & Lee, H. J. (2021). E-cheating Prevention Measures: Detection of Cheating at Online Examinations Using Deep Learning Approach--A Case Study. arXiv preprint arXiv:2101.09841.
43. Tzafilkou, K., & Protogeros, N. (2020). Monitoring Mouse Behavior in E-Learning Activities to Diagnose Students' Acceptance Items of Perceived Usefulness and Ease of Use. European Educational Researcher, 3(1), 21-27.
44. Ullah, A., Xiao, H., & Barker, T. (2019). A dynamic profile questions approach to mitigate impersonation in online examinations. Journal of Grid Computing, 17(2), 209-223.
45. Urosevic, A. (2019). Student authentication framework for Online exams outside of school.
46. Walia, A. S. (2018). Activation functions and it's types-which is better?(2017).
47. Yan, S. (2016). Understanding LSTM and its diagrams. MLReview.com.