Ingénierie des Systèmes d'Information
Vol. 29, No. 4, August, 2024, pp. 1573-1579
Journal homepage: http://iieta.org/journals/isi

Application of LSTM for Redundancy Detection in MCTS: Enhancing Test Precision

Najoua Hrich[1, 2*], Mohamed Azekri[3], Mohamed Khaldi[2]

[1] Regional Center for Education and Training Professions, Institutions for Higher Executive Training, Tangier 90000, Morocco
[2] Computer Science and University Pedagogical Engineering Research Team, Higher Normal School, Abdelmalek Essaadi University, Tetouan 93000, Morocco
[3] Regional Academy of Education and Training, Ministry of National Education Preschool and Sports, Tetouan 93000, Morocco

Corresponding Author Email: amine.najoua@gmail.com

https://doi.org/10.18280/isi.290430

Received: 24 April 2024
Revised: 9 July 2024
Accepted: 10 August 2024
Available online: 21 August 2024

Keywords:
deep learning, Long Short-Term Memory (LSTM), Natural Language Processing (NLP), Multiple Choice Tests (MCTs), redundancy, items, distractors

ABSTRACT

In the context of competitive examinations, the number of items can be extremely high. In such situations, the item review process remains essential. It enables designers to consider the complexity and scope of the assessment by reviewing each item and distractor. Identifying redundancies becomes even more critical in this context, as the variety and quality of items are crucial to ensure a fair and equitable assessment of candidates' skills. This article aims to propose an artificial intelligence model specifically designed to efficiently detect and correct these redundancies in multiple-choice tests. By combining the human expertise in item review with the massive data processing capabilities of AI, we aim to improve the quality and reliability of competitive exams, while optimizing the time and resources required for their development.

1. INTRODUCTION

The increasing adoption of multiple-choice tests (MCTs) into examinations and educational competitions has significantly transformed the assessment of student learning. This evolution stems from the need to assess a large number of candidates while evaluating a diverse range of skills in an objective and efficient manner. However, this transition is not without its challenges, particularly when it comes to designing questions that are relevant and free from redundancy.

In the context of competitive examinations, the number of items specified is often very high, sometimes reaching 100 to 120 items, with four distractors for each. This abundance of items increases the risk of redundancy, which can compromise the validity and reliability of assessments.

Faced with this challenge, competition organizers and educators have developed an emerging practice called "test pilot", in which test designers take the same test as the candidates to assess the quality of the questions. Despite this method of validation, there is still a risk that redundancies will escape the designers' vigilance, which could lead to challenges during the examination process.

To address this issue, our research explores an innovative approach based on the use of deep learning techniques to detect and solve redundancy problems in MCTs. In this paper, we present an intelligent solution aimed at overcoming this problem by developing a specific tool capable of automatically analyzing and correcting redundancies in multiple-choice exams. In addition, we propose a detailed methodology and empirical results demonstrating the effectiveness of our approach, particularly in contexts requiring a large number of items, such as competitions and competitive exams. The experimental results of the study illustrate the effectiveness of the proposed model, highlighting its superior performance with a redundancy detection accuracy rate of 93%.

2. BACKGROUND

2.1 Test development process

The assessment test development process involves several crucial stages to ensure the creation of a valid and effective assessment [1-4]. Table 1 displays an overview of these steps.

Each stage of the test development process plays a crucial role in ensuring that the assessment accurately measures the candidate's knowledge, skills, or abilities [5].

2.2 AI-based assessment: Integration into the development test process

The development of assessment tests is a critical process in educational assessment. With the advent of artificial intelligence (AI), there are new opportunities to improve and optimize this process [6]. Integrating AI into each phase of assessment test development offers promising approaches and applications. For example, during the item creation phase, Natural Language Processing (NLP) techniques such as word embeddings (Word2Vec and GloVe) are used to create context-sensitive vector representations of words. Word2Vec

is a neural network model that learns distributed representations of words based on their context in a continuous vector space [7]. GloVe, on the other hand, constructs word embeddings based on global word co-occurrence statistics and factors in a co-occurrence matrix to generate embeddings that capture local and global semantic relationships [8]. BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) are also used to automatically generate items from source texts by capturing semantic and contextual relationships between words [9, 10]. During the pilot testing phase, neural network-based recommender systems analyze participants' performance and recommend the most appropriate items.

2.3 Deep learning models for text classification

The use of deep learning models to detect redundancy in texts represents a significant advance in the field of Natural Language Processing (NLP [11]. These models exploit deep neural architectures to extract relevant features from text data, enabling them to identify similarities and repetitions within text content. Architectures commonly used for this task include recurrent neural networks (RNNs), convolutional neural networks (CNNs) and transformers.

-RNNs: RNNs adapt to model sequential data, making them relevant for detecting repetitive patterns in texts. For example, an RNN model can be used to identify sentences or passages that are repeated throughout a document [12].

-CNNs: CNNs extract local features from text data. They can detect similarities between text passages by analyzing local patterns. For example, a CNN can be used to identify similar sentences in a set of documents [13].

-Transformers: Transformers have revolutionized the field of NLP by introducing attention mechanisms. They can consider relationships between words across the whole text, making it easier to detect redundancy across long distances. For example, a transform model can be used to identify text passages that are repeated at different points in a document [14].

These deep learning models offer different yet complementary approaches to detecting redundancy in texts, by analyzing word sequences, local features, and long-distance relationships between words [15, 16].

In this paper, we have used LSTMs because they are efficient at detecting redundancies that depend on long-term dependencies present in the data. LSTMs, as an advanced variant of RNNs, are designed to better capture and retain relevant information over long sequences, which makes them suitable for complex tasks such as redundancy detection in multiple-choice questions with a large number of items and distractors. In comparison, CNNs are less efficient for this specific task due to their inability to perceive the global information needed to detect textual redundancy. While CNNs can traditionally deal with sequences of data, they are often ineffective at dealing with data characterized by long-term dependencies due to vanishing gradient problems, making them less suitable for detecting highly complex redundancies that require deep contextual understanding. Consequently, the use of LSTMs in our application is based on its efficient processing of long-term information and some of the strongest solutions in redundancy detection that can eventually improve test accuracy.

Interest in the use of LSTMs in redundancy detection has grown considerably, stimulating research in this field. Researchers have explored various applications aimed at improving the accuracy and efficiency of this task. For example, LSTM models have been deployed for real-time detection of actions in human motion streams, based on sequences of 3D skeletal configurations [17]. These models outperformed traditional methods in terms of efficiency, enabling annotation at up to 10,000 frames per second. Similarly, LSTM classifiers were exploited to detect exploit kit traffic by analyzing the sequential structure of HTTP redirects, offering promising results in terms of accuracy and performance [18]. By optimizing the hyperparameters of the LSTM model, they obtained an F1 score of 0, demonstrating the effectiveness of their approach.

In short, LSTMs continue to be an active research area for redundancy detection, and advances in this field open up new prospects for practical applications.

Table 1. Test development process

Stage	Description
Stage 1: Specifications	This initial phase involves defining the purpose of the examination, identifying the target audience, delimiting the content areas to be covered, and determining the types of items to be included.
Stage 2: Item edition	Experienced editors develop questions or tasks according to the specifications defined in the previous phase. Items are rigorously reviewed and edited.
Stage 3: Pilot test	A preliminary version of the exam is administered to a small group of candidates to assess item performance and collect data for further analysis and improvement.
Stage 4: Revision	Based on feedback from pilot tests and expert assessments, items may be revised or modified to address any problems or concerns identified.
Stage 5: Final test	This phase involves the selection of items meeting predetermined criteria based on the results of the item analysis. The test is analyzed to ensure accuracy and consistency prior to administration.

3. METHODS AND MATERIALS

In this research, we introduce a novel approach designed to enhance test quality by identifying and removing redundancies in both items and distractors. The procedure for our proposed method is illustrated in Figure 1.

To contribute to improving item quality by identifying and eliminating duplications, we initially developed a deep-learning model using Python for detecting redundancy in items and distractors. This model was designed to identify duplicate items within an MCT, using a LSTM neural network, a specific variant of RNNs. LSTMs are particularly suited to capturing long-term dependencies in data sequences, making them an ideal choice for our redundancy detection task [19]. then, we have created a training dataset using MS Word. To optimize the quality of the information provided to the model, the items in this dataset are numbered from Q1 to Q1000, and each item is composed of 4 distractors A, B, C, and D. Finally, we used our model for 10 tests of 100 items each, to evaluate its effectiveness in detecting duplications and its practical application in a real assessment context.

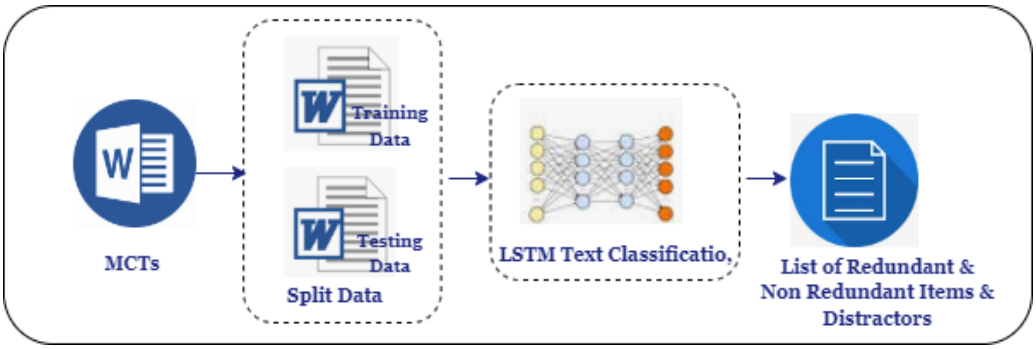

Figure 1. Overview of LSTM model for detecting tests items & distractors redundancies

To train our deep learning model, we employed a standard method of splitting data into training and testing sets. initially, we separated our training dataset, which comprises MCT questions and their corresponding distractors, into two parts: one for training the model and the other for validating the model. We allocated approximately 80% of the data for training and the remaining 20% for testing. This process is described in more detail in the next sections.

3.1 Data preprocessing

The data set used this study was collected from 10 MCTs from various disciplines, specifically designed for the recruitment exams of future teachers. Each test was meticulously reviewed to extract the questions and their associated four distractors, labeled A, B, C, and D. Each item in the dataset consisting of the question and its distractors, was systematically organized with consecutive numbering, facilitating efficient data management and subsequent analysis. This rigorous approach ensures that the dataset accurately reflects the types of questions posed in teacher recruitment exams, providing a solid foundation for analysis and development of the redundancy detection model. Figure 2 display an extract of the dataset used in this study.

3.2 LSTM algorithm

In this study, we utilize the LSTM algorithm, which operates through four key components: memory cells, input gates, forget gates, and output gates. The input gate manages the selection of values to be updated. Following this, the forget gate filters out irrelevant information. The remaining data is then processed by the output gate, which generates the final output [20].

LSTM is engineered to retain information from previous cells, allowing it to uncover hidden layers within each cell. This approach involves classifying long-term data by leveraging storage in memory cells.

3.3 Model description

The procedure involves managing questions and distractors in a structured dataset. Initially, questions and distractors are extracted from a DOCX file and pre-processed using tokenization and padding to align the textual sequences. The model begins with an embedding layer that transforms each word into a dense vector of size 100, facilitating vector representation of the words. Next, an LSTM layer with 64 units is used to capture the complex temporal dependencies in the text sequences, enabling the model to maintain and use long-term information. The output of the LSTM layer is fed into a Dense layer with sigmoid activation, suitable for binary classification that decides whether a question and a distractor are redundant (1) or not (0). The model is compiled with the Adam optimizer and the binary cross-entropy loss function, while evaluating accuracy as a metric. Trained on data prepared during 30 epochs with mini-lots of size 32, the model aims to learn to generalize and effectively predict redundancy in new questions and distractors. Once trained, the model is saved for future use, offering a robust solution for automatic semantic similarity analysis in multiple-choice questions and their options.

3.4 Model implementation

The model is implemented under the Python programming language, with the TensorFlow and Keras libraries for neural network development. TensorFlow is an open-source machine learning platform developed by Google, while Keras is a high-level neural network API that facilitates efficient model building and training.

For the initial processing of text data extracted from DOCX files, the library is used to read and extract document content. The tokenizer and 'pad_sequences' classes of the 'tensorflow.keras.preprocessing.text' library are then used to tokenize and pad the text sequences. The LSTM model itself is built using Keras' Sequential class to define the sequential structure of the network layers. The embedding layer is implemented with Keras Embedding, followed by an LSTM layer. Next, a dense layer with sigmoid activation is added using Dense for binary classification of question-distractor pairs. For model optimization, the Adam optimizer is used with the binary cross-entropy loss function specified by binary_crossentropy in 'model.compile'. Finally, model performance is evaluated using the accuracy metric accuracy.

The Python code shown above provides the implemented model (Figure 3).

3.5 Training procedure

The results displayed during training show an impressive progression of the LSTM model over the epochs. Right from the start, at epoch 3, the model shows an accuracy of 66.67% with a loss of 0.6826, indicating a moderate initial performance (Figure 4).

This accuracy remains constant over the first few epochs, but from epoch 25 onwards, we see a clear improvement. Accuracy rises rapidly to 91.67% at epoch 26, then reaches a remarkable 100% accuracy from epoch 27 onwards, with a steadily decreasing loss to 0.1412 at epoch 30 (Figure 5).

Q1 **Soit l'adresse 172.16.5.10/28. Quel est le masque réseau correspondant ?**

A 255.255.255.240
B 255.255.255.248
C 255.255.224.28
D 255.255.224.0

Q2 **Soit l'adresse 192.16.5.133/29. Le nombre de bits utilisés pour identifier la partie réseau est :**

A 29
B 3
C 28
D 2

.
.
.

Q294 **Quel protocole permet de trouver l'adresse MAC associée à une adresse IP?**

A DNS
B HTTP
C ARP
D ICMP

Figure 2. Sample from dataset

```python
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense

# Définir la taille de l'embedding, la taille du vocabulaire et la longueur maximale de la séquence
embedding_size = 1000
vocab_size = 100000
max_length = 1000

# Créer un modèle séquentiel
model = Sequential()

# Couche d'embedding
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_size))

# Couche LSTM
model.add(LSTM(units=64, dropout=0.2, recurrent_dropout=0.2))

# Couche Dense pour la prédiction binaire
model.add(Dense(units=1, activation='sigmoid'))

# Compiler le modèle
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 3. LSTM model construction

```
Epoch 1/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 0.3333 - loss: 0.6969
Epoch 2/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 0.8333 - loss: 0.6896
Epoch 3/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.6667 - loss: 0.6826
Epoch 4/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - accuracy: 0.6667 - loss: 0.6757
Epoch 5/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.6667 - loss: 0.6686
Epoch 6/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - accuracy: 0.6667 - loss: 0.6612
Epoch 7/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.6667 - loss: 0.6534
Epoch 8/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.6667 - loss: 0.6450
Epoch 9/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 0.6667 - loss: 0.6360
```

Figure 4. Evolution of accuracy and loss from epoch 1 to 9

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - accuracy: 0.8333 - loss: 0.3433
Epoch 26/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step - accuracy: 0.9167 - loss: 0.3046
Epoch 27/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - accuracy: 1.0000 - loss: 0.2579
Epoch 28/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 1.0000 - loss: 0.2087
Epoch 29/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 1.0000 - loss: 0.1695
Epoch 30/30
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 1.0000 - loss: 0.1412
```

Figure 5. Evolution of accuracy and loss from epoch 26 to 30

This trend suggests that the model is learning efficiently from the training data, adjusting its weights to perfectly match the labels provided. An accuracy of 100% indicates that the model is able to correctly predict redundancy between questions and distractors, which is a very promising result for its ability to generalize and perform well on new data.

3.6 Testing procedure

Once the model had been trained, we tested it on the test dataset. This dataset contains questions and distractors that were not used during training, allowing us to assess the model's ability to generalize to new data. We ran the questions and distractors from the test set through the model and evaluated its performance by measuring its ability to detect duplications.

Data splitting involves dividing the dataset into two distinct subsets: training data and testing data. Specifically, 80% of the dataset is allocated for training purposes, where it is used to develop and train the classification model. The remaining 20% is designated as testing data, which is employed to assess the model's performance and generalizability. This separation ensures that the model is trained on one portion of the data and its effectiveness is evaluated on an independent portion, providing a robust measure of its accuracy and reliability.

3.7 Evaluation procedure

The purpose of model evaluation is to measure its performance in distinguishing between redundant and non-redundant items in the data set, which constitutes a binary classification. To assess this performance, several metrics are used, of which the confusion matrix is the most commonly used.

The confusion matrix includes parameters such as True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). From these parameters, several evaluation metrics are derived, including accuracy, precision, recall, and the F1 score, as detailed below:

Accuracy: it measures the ratio of correct predictions to the total number of predictions, offering an overall view of the model's ability to classify both positive and negative instances. Although it indicates how often the model's predictions are correct, accuracy may be less meaningful in situations with class imbalance, where classes are not evenly distributed.

Precision: it evaluates the accuracy of positive predictions by dividing the number of true positives by the total number of positive.

Recall: It assesses the model's capacity to detect all positive class instances.

F1-Score: it represents the harmonic mean of precision and recall and is commonly employed in situations with imbalanced classes.

We have used the Scikit-learn libraries to calculate these metrics with a view to evaluating the performance of our neural network model. The results are present-ed in the following section.

4. RESULTS AND DISCUSSION

```python
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import matplotlib.pyplot as plt

#294 questions with 4 distractors each.
y_true = [0, 1, 1, 0, 1, 1, 0, 1,0, 1, 1, 0, 0, 1, ..., 1,0, 1, 1, 1, 1, 1, 0, 1,0]
y_pred = [1, 1, 1, 0, 1, 1, 0, 1,0, 1, 1, 0, 0, 1,..., 0, 1,0, 1, 1, 0, 0, 1, 0, 1]

#Confusion Matrix
cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:")
print(cm)

# Classification Report
print("\nClassification Report:")
print(classification_report(y_true, y_pred))

# Accuracy
accuracy = accuracy_score(y_true, y_pred)
print("\nAccuracy:", accuracy)
```

Figure 6. Script for calculating metrics

```
Confusion Matrix:
[[499  24]
 [ 55 598]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.95      0.93       523
           1       0.96      0.92      0.94       653

    accuracy                           0.93      1176
   macro avg       0.93      0.93      0.93      1176
weighted avg       0.93      0.93      0.93      1176


Accuracy: 0.9328231292517006
```

Figure 7. Classification report

The model's evaluation involves several critical metrics that provide a deep insight into its performance. The confusion matrix offers a comprehensive summary of both correct and incorrect predictions, reflecting the model's proficiency in differentiating between accurate and erroneous classifications.

To calculate these metrics, a Python script was developed using the 'sklearn. metrics' library, as illustrated in Figure 6.

The results provided by this script are shown in Figure 7. The results show that the classification model performs remarkably well, with high accuracies for both classes (90% for non-redundant distractors and 96% for redundant distractors), as well as high recall (95% for non-redundant distractors and 92% for redundant distractors), demonstrating its ability to correctly identify the actual occurrences of each class. The F1-Scores, which combine precision and recall, are also high (93% for non-redundant distractors and 94% for redundant distractors), highlighting good harmony in the model's ability to predict accurately. With an overall accuracy of 93%, the model consistently predicts the class of observations correctly in the majority of cases. In conclusion, these results confirm that the classification model is effective and accurate in predicting classes on this specific dataset.

5. LIMITATIONS AND FURTHER WORK

Although this study revealed significant results on the model's ability to detect items and distractor redundancy, several limitations were identified. Firstly, the limited sample size. Indeed, the limited number of questions may restrict the generalizability of the results obtained. In addition, it should be noted that this research was conducted within a single discipline and that the questions were written in French. Consequently, the results obtained may be specific to this field of study and may not apply to other languages and subjects. To address this limitation, we will expand the experimental sample to other subjects and use a larger sample size, which may enhance the generalizability of the findings. These improvements would contribute to strengthening the robustness and applicability of AI-based assessment frameworks in various educational and professional contexts.

6. CONCLUSIONS

In conclusion, the article explored the innovative application of artificial intelligence models to improve the quality of assessment tests, focusing on the detection of redundancies in items and distractors. By integrating these models into the test development process, designers can not only speed up the piloting process, but also guarantee greater validity and reliability of assessments. This approach also frees up valuable time and resources by automating tasks that were previously tedious and prone to human error. Ultimately, the integration of AI into test development marks a significant evolution, paving the way for more efficient and accurate methods of assessing learners' knowledge and skills.

REFERENCES

[1] Reynolds, N., Diamantopoulos, A., Schlegelmilch, B. (1993). Pre-testing in questionnaire design: A review of the literature and suggestions for further research. Market Research Society. International Journal of Market Research, 35(2): 7. https://doi.org/10.1177/147078539303500202

[2] Gierl, M.J., Bulut, O., Guo, Q., Zhang, X. (2017). Developing, analyzing, and using distractors for multiple-choice tests in education: A comprehensive review. Review of Educational Research, 87(6): 1082-1116. https://doi.org/10.3102/0034654317726529

[3] Tests statistiques élémentaires. https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-l-inf-tests.pdf.

[4] Hrich, N., Lazaar, M., Khaldi, M. (2019). Problematic of the assessment activity within adaptive E-learning systems. International Journal of Emerging Technologies in Learning, 14(17): 133-142. https://doi.org/10.3991/ijet.v14i17.10675

[5] Haladyna, T.M., Rodriguez, M.C. (2013). Developing and Validating Test Items. Routledge. https://doi.org/10.4324/9780203850381

[6] Peng, C., Zhou, X., Liu, S. (2022). An introduction to artificial intelligence and machine learning for online education. Mobile Networks and Applications, 27(3): 1147-1150. https://doi.org/10.1007/s11036-022-01953-3

[7] Yilmaz, S., Toklu, S. (2020). A deep learning analysis on question classification task using Word2vec representations. Neural Computing and Applications, 32(7): 2909-2928. https://doi.org/10.1007/s00521-020-04725-w

[8] Pennington, J., Socher, R., Manning, C.D. (2014). Glove:

Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1532-1543. https://doi.org/10.3115/v1/D14-1162

[9] Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. https://doi.org/10.48550/arXiv.1810.04805

[10] Zhang, H., Song, H., Li, S., Zhou, M., Song, D. (2023). A survey of controllable text generation using transformer-based pre-trained language models. ACM Computing Surveys, 56(3): 64. https://doi.org/10.1145/3617680

[11] Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., Gao, J. (2021). Deep learning-based text classification: A comprehensive review. ACM Computing Surveys, 54(3): 62. https://doi.org/10.1145/343972

[12] Fu, X., Ch'ng, E., Aickelin, U., See, S. (2017). CRNN: A joint neural network for redundancy detection. In 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, China, pp. 1-8. https://doi.org/10.1109/SMARTCOMP.2017.7946996

[13] Wang, Z., Li, C., Wang, X. (2021). Convolutional neural network pruning with structural redundancy reduction. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, pp. 14908-14917. https://doi.org/10.1109/CVPR46437.2021.01467

[14] Dai, Z., Lai, G., Yang, Y., Le, Q.V. (2020). Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. arXiv preprint arXiv:2006.03236. https://doi.org/10.48550/arXiv.2006.03236

[15] Hrich, N., Azekri, M., Khaldi, M. (2024). An ai educational tool for detecting redundancy in distractors and items within multiple-choice tests. In 18th International Technology, Education and Development Conference, Valencia, Spain, pp. 6454-6458. https://doi.org/10.21125/inted.2024.1691

[16] Hrich, N., Azekri, M., Elhaddouchi, C. (2024). Efficient redundancy detection in large-scale examinations. International Research Journal of Computer Science, 11(5): 459-462. https://doi.org/10.26562/irjcs.2024.v1105.04

[17] Carrara, F., Elias, P., Sedmidubsky, J., Zezula, P. (2019). LSTM-based real-time action detection and prediction in human motion streams. Multimedia Tools and Applications, 78: 27309-27331. https://doi.org/10.1007/s11042-019-07827-3

[18] Burgess, J., O'Kane, P., Sezer, S., Carlin, D. (2021). LSTM RNN: Detecting exploit kits using redirection chain sequences. Cybersecurity, 4: 25. https://doi.org/10.1186/s42400-021-00093-7

[19] Yang, S., Yu, X., Zhou, Y. (2020). LSTM and GRU neural network performance comparison study: Taking yelp review dataset as an example. In 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI), Shanghai, China, pp. 98-101. https://doi.org/10.1109/IWECAI50956.2020.00027

[20] Imaduddin, H., Kusumaningtias, L.A., A'la, F.Y. (2023). Application of LSTM and GloVe word embedding for hate speech detection in indonesian twitter data. Ingénierie des Systèmes d'Information, 28(4): 1107-1112. https://doi.org/10.18280/isi.280430