Models of binary classification of the semantic colouring of texts

Modelos de clasificación binaria de la coloración semántica de textos

Nataliya Boyko[1*].

Innovaciencia
ISSN: 2346-075X
E- ISSN: 2346-075X
Innovaciencia 2023; 11(1); 1-23
http://dx.doi.org/10.15649/2346075X.3553

ORIGINAL RESEARCH

How to cite this paper:
Boyko N., Models of binary classification of the semantic colouring of texts. Innovaciencia 2023; 11(1): 1-23.
http://dx.doi.org/10.15649/2346075X.3553

Received: 13 November 2023
Accepted: 30 November 2023
Published: 01 December 2023

Keywords:
Binary text classification; Long short-term memory; Convolutional neural network; Gate recurrent node.

ABSTRACT

Introduction. The purpose of the research is to compare different types of recurrent neural network architectures, namely the long short-term memory and gate recurrent node architecture and the convolutional neural network, and to explore their performance on the example of binary text classification. **Material and Methods.** To achieve this, the research evaluates the performance of these two popular deep-learning approaches on a dataset consisting of film reviews that are marked with both positive and adverse opinions. The real-world dataset was used to train neural network models using software implementations. **Results and Discussion.** The research focuses on the implementation of a recurrent neural network for the binary classification of a dataset and explores different types of architecture, approaches and hyperparameters to determine the best model to achieve optimal performance. The software implementation allowed evaluating of various quality metrics, which allowed comparing the performance of the proposed approaches. In addition, the research explores various hyperparameters such as learning rate, packet sizes, and regulation methods to determine their impact on model performance. **Conclusion.** In summary, the study found that recurrent neural networks, particularly the gated recurrent unit (GRU), demonstrated superior performance in binary text classification compared to convolutional neural networks, underscoring the significance of selecting appropriate model architectures and parameter adjustments for specific datasets.

[1] Department of Artificial Intelligence Systems, Lviv Polytechnic National University, 79013, 12 Stepan Bandera Str., Lviv, Ukraine,
* Corresponding author: nataliya.boy@ukr.net

INTRODUCTION

Machine learning, in practice, encompasses a variety of techniques that enable machines to learn from data autonomously, without being explicitly programmed for specific tasks [1,2]. This process involves the machine analysing large datasets to discern patterns, trends, and relationships. The key aspect of machine learning is its ability to adapt and refine its algorithms based on new data, thus constantly evolving and becoming more accurate in its predictions or decisions. This self-learning capability distinguishes machine learning from traditional programming, where tasks are performed based on predefined rules and algorithms [3,4]. In turn, these methods rely on algorithms to learn from a prepared data set and improve their performance over time. Currently, recurrent neural network (RNN) [5] and convolutional neural network (CNN) [6] are two popular deep learning architectures that are commonly used for binary text classification tasks [7]. In addition, using machine learning to recognise handwriting has revolutionised the field of document digitisation, enabling the fast and efficient conversion of handwritten text into digital formats. This technology has transformed industries that rely heavily on document processing, such as banking, insurance, and healthcare [8-12].

The field of machine learning has enormous potential to solve complex problems in various industries, and using deep neural networks, especially RNNs, is a promising area. Due to their ability to analyse sequential data and learn complex patterns, machine learning technologies have transformed areas such as speech recognition, natural language processing and handwriting recognition [13-16], enabling businesses to achieve greater efficiency and productivity. One of the most important developments in the field of machine learning is the emergence of deep neural networks, especially RNNs. RNN are a powerful type of neural network that can be used for a wide range of tasks, such as speech recognition, natural language processing, and even handwriting recognition. They work by taking in sequential data and using feedback loops to store information about previous inputs, allowing them to learn complex patterns over time [17-19].

The model proposed by N.H. Ho et al. [20] uses a RNNs to solve tasks in the field of speech recognition, which ensures efficient and accurate speech transcription. Similarly, S. Chamishka et al. [21] present a technique for detecting emotions in real-time using RNNs and feature modelling. The proposed approach offers an effective solution for accurately detecting emotions from voice recordings, which provides many opportunities for automated translation, sentiment analysis, and text generation. One of the common RNN architectures used for binary text classification is the long short-term memory (LSTM) network. LSTMs are a variant of RNNs that use closed cells to selectively remember or forget information, allowing them to better capture long-term dependencies in data. LSTMs have demonstrated impressive results in tasks such as sentiment analysis and text classification. The work of V. Barzegar et al. [22] presents using conventional RNNs together with LSTM cells for high-speed structural health monitoring. Another example of a neural network architecture is CNNs. These neural networks are designed to process gridded data types, such as images. However, they can be adapted for text classification tasks by treating each word as a one-dimensional input grid. CNNs work by convolving a filter or kernel on the input and extracting local features. It allows them to capture important patterns in the input, such as n-grams of words that often occur together in certain classes of text. The work of L. Yao et.al. [23] presents a new approach to text classification using convolutional graph networks. By introducing the graph structure into the learning process, the proposed model captures the inherent connections between words and ensures competitiveness in text classification tasks.

Recent studies have demonstrated that both CNN and RNN can be effective models for the problem of binary classification of semantic text colouring. A detailed analysis of these networks can be useful for researchers and practitioners working on natural language processing tasks and trying to choose the right model for their needs. A similar analysis was performed in the work of W. Yig et al. [24], where they provide an overview of both CNN and RNN, including their architectures and how they can be used for natural

language processing. RNNs excel at handling sequential data, such as text or time series, due to their memory that captures data order and context. They are used in tasks like language modelling, text generation, and speech recognition but face challenges with vanishing gradients. In contrast, CNNs specialize in processing grid-like data, primarily images, for tasks like image classification and object detection. They excel at pattern recognition by applying filters and capturing spatial hierarchies. RNNs have internal memory, while CNNs focus on spatial learning. These networks are fundamental in deep learning, with RNNs in natural language processing and CNNs in computer vision tasks.

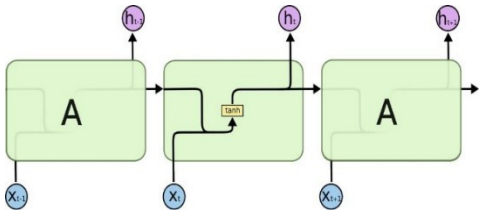
The aim of this study is to evaluate and compare the performance of different neural network architectures for the task of classifying semantic coloring in texts, with the ultimate goal of identifying the optimal architecture for this specific task. This study is important because it compares various neural network architectures for binary text classification using a real-world dataset. It evaluates their performance, explores hyperparameters, and provides practical recommendations. This research contributes to the field by offering insights into architecture selection and hyperparameter tuning, making it valuable and novel.

MATERIALS AND METHODS

ANALYSING THE ARCHITECTURE OF RECURRENT NEURAL NETWORKS

It is known that RNNs have a chain-like architecture consisting of repeating cells. They can be either a single neuron or a sequence of several neurons. For example, a basic RNN cell contains a single layer with an activation function *tahn*, which is quite simple in terms of structure. Suppose there is a sequence of input data $\{x_t\}_{(t=1)}^{T}$. In the context of text analysis, $x_t = x_1,\ldots,x_n$ can represent the vector form of the t-h word in the sequence. For a sequence of elements, T of a RNN are required. Significantly, the output of the t-h cell is fed as input to the $t + 1$ -th cell, establishing a chain-like structure that facilitates processing of the entire input sequence **(Figure 1)**.

Figure 1. Scheme of RNN cells



Source: compiled by the author.

The word vectors are processed by the cells using the ratio specified in **formula (1)**:

$$H_t = \tanh(U_t \cdot x_t + W \cdot H_{t-1} + b). \quad (1)$$

Notably, the activation function used in RNMs should not be limited to *tahn* . Other common options include sigmoid, *ReLu*, *softmax* or other variants. In addition, the weights U_t associated with each element of the sequence t are unique, while the weights Vt remain constant for all elements. During the training of the neural network, the weights are adjusted using the back-propagation algorithm. After the RNN layer, the next output layer can produce a sequence of outputs, denoted as $\{y_t\}_{(t=1)}^{T}$, with the activation function f

applied to each. Thus, the output forecasts are determined by **formula (2)**:

.

$$\hat{y}_t = \mathrm{V} * f(H_t).$$
$$E_t(y_t, \hat{y}_t) = -y_t log\hat{y}_t, \tag{2}$$

The loss function can be defined as **(3)**:

(3)

hence **(4)**:

$$E(y_t, \hat{y}_t) = \sum_{t=1}^{T} E_t(y_t, \hat{y}_t) = -\sum_{t=1}^{T} y_t log\hat{y}_t. \tag{4}$$

To start using gradient descent to updâte the weights of a neural network, it is very crucial to calculate the gradients of the loss function with respect to *U*, *W*, and *V*. In particular, when working with *V*, the error gradient at step *t* depends solely on y_t, y_t, and H_t **(5, 6)**:

$$\frac{\partial E}{\partial V} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial V}, \tag{5}$$

$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial H_t}\frac{\partial H_t}{\partial V}. \tag{6}$$

However, for *W* and *U*, when estimating the error gradient, H_k, $k < t$ **(7)** should be considered:

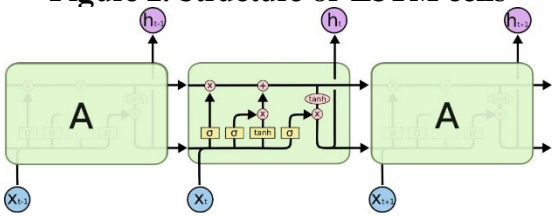
$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial H_t}\frac{\partial H_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial H_t}\frac{\partial H_t}{\partial H_k}\frac{\partial H_k}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial H_t}\left(\prod_{j=k+1}^{t} \frac{\partial H_j}{\partial H_{j-1}}\right)\frac{\partial H_k}{\partial V}. \tag{7}$$

The derivatives of tahn and sigmoid are limited to one, and their values approach zero as the absolute value of the input data increases. It leads to gradients close to zero, and as the distance between the th and th objects increases, the influence of the th object on the weight update decreases. On the other hand, using other activation functions can result in gradients with absolute values greater than one, causing them to grow infinitely. As a result, distant objects may have a greater impact when weights are updated. This problem is known as the gradient damping/explosion problem, and it is not limited to RNN but is present in deep neural networks.

LONG SHORT-TERM MEMORY

Long Short-Term Memory Networks, known as LSTMs, are a unique type of neural network that is capable of identifying both short-term and long-term relationships. Similar to RNN, LSTMs have a circuit-like structure; however, their recurrent cellular structure is more complex. It consists of four neurons connected in a certain way. In **(Figure 2)**, the neurons are marked with yellow blocks, while the pink circles represent the coordinate linear operations.

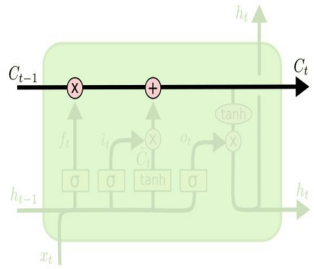
Figure 2. Structure of LSTM cells



Source: compiled by the author.

Consider the structure of the LSTM cell in more detail. Its cell consists of two recurrent components – the output vector H_t and the state vector C_t. Unlike other RNNs, the LSTM cell does not use an activation function in component C_t. The state vector C_t is transmitted directly through the entire circuit and is involved in only a few linear transformations. As a result, the resulting value has no blurring in time and the gradient is not lost during the network training process **(Figure 3)**.

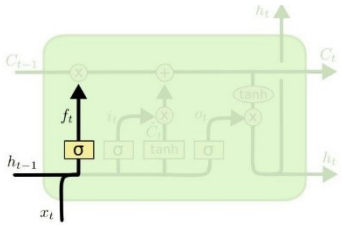
Figure 3. LSTM structure cell



Source: compiled by the author.

The LSTM cell has the ability to discard information from its repetitive components, which is controlled by special structures known as filters. The first step in the LSTM process involves determining what information can be removed from the state vector. This decision is made by a neuron with a sigmoidal activation function called the "forgetting filter layer" **(Figure 4)**.

Figure 4. Schematic representation of the filtering stage of the LSTM cell



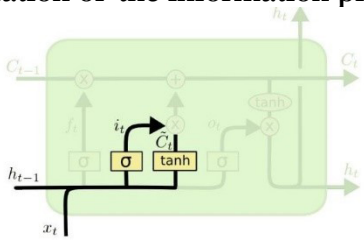
Source: compiled by the author.

This structure takes a vector (H_{t-1}, x_t) as an input parameter and returns another vector $f_t \in [0,1]^m$, where m is the dimension of vector C_t. Each component of C_{t-1} corresponds to its own component of the vector f_t, which can take values from 0 to 1, where 1 corresponds to a full save command, and 0 to a full exclusion command. The appropriate filter is determined from **equation (8)**:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (8)$$

The next step in LSTM involves determining what new information should be stored in the state vector. This process consists of two separate parts. First, the input filter layer is used to identify the components of the state vector that need to be updated in the update estimate. This level uses a sigmoidal activation function **(Figure 5)**.

Figure 5. Schematic representation of the information processing stage of an LSTM cell



Source: compiled by the author.

After that, layer *tanh* establishes a vector of new values to replace the components of the state vector that need to be updated. The input to both layers is a vector $H_{t-1}, x_t)$, while the output is a vector with dimension m. Mathematically, these processes can be described as follows **(9, 10)**:

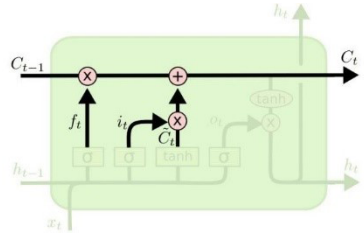
$$i_t = \sigma\ (W_i \cdot [h_{t-1}, x_t] + \ b_i), \tag{9}$$

$$\tilde{C}_t = tanh(\ W_C\ ,\ [h_{t-1}, x_t] + \ b_C), \tag{10}$$

The updated state vector C_t is obtained from **equation (11) (Figure 6)**:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \tag{11}$$

Figure 6. Schematic representation of the stage of updating the state vector



Source: compiled by the author.

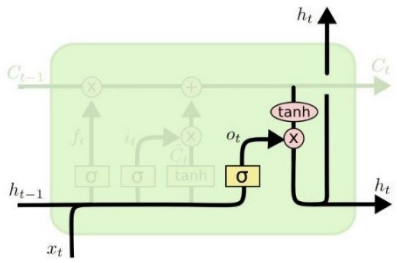
The last step is to update the input vector using the following filters **(12, 13)**:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] \quad b_o), \tag{12}$$

$$h_t = o_{t^*}\ tanh(C_t). \tag{13}$$

First, using the relation (12), the information from the state vector is calculated to be transferred to the output vector. Then, using (13), the result is processed by layer *tanh*. The output values are in the range [-1, 1] and are multiplied with the output values of the sigmoidal layer co-ordinately, eliminating unnecessary information. This process is presented graphically in **(Figure 7)**.

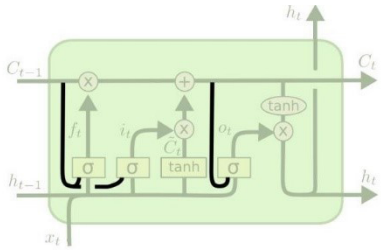
Figure 7. Schematic representation of the input vector update stage



Source: compiled by the author.

Notably, in one embodiment of LSTM, connections within the cell are additionally used, as demonstrated in **(Figure 8)**.

Figure 8. Schematic representation of an LSTM cell by adding so-called "peephole connections"



Source: compiled by the author.

Mathematically, they are described by the following relations **(14-16)**:

$$f_t = \sigma\left(W_f \cdot [C_{t-1}, \ _{t-1}, x_t] + b_f\right), \tag{14}$$

$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t \ + b_i), \tag{15}$$

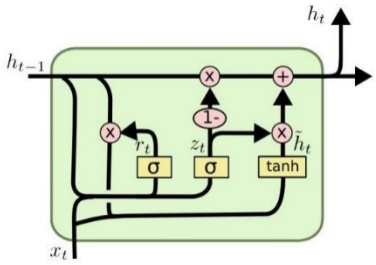
$$o_t = \sigma(W_o \cdot [C_{t-1}, h_{t-1}, x_t] + b_o). \tag{16}$$

As a result, the filter layers have access to the state vector using vector $[C_{t-1}, h_{t-1}, x_t]$ instead of as input.

VALVE RECURRENT ASSEMBLY

Another interpretation of LSTMs is gated recurrent units (GRU), which replace the state vector with an output vector that passes directly through the circuit without applying any activation functions **(Figure 9)**. In addition, instead of separate forgetting and input filters, this modification combines them into a single update filter Z_t.

Figure 9. Schematic representation of the GRU-based architecture



Source: compiled by the author.

In the GRU-based system, a cell contains a set of three neurons. Assuming that the input to the sigmoid layer is vector $[h_{t-1}, x_t]$, the output is the updated vector Z_t, whose dimension is the same as the initial vector **(17)**:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]). \tag{17}$$

Then, using another sigmoidal layer, the reset vector r_t **(18)** is constructed in a similar way:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]). \tag{18}$$

It is necessary to determine the components of vector h_{t-1} and their concentration required to establish an update of the original vector. The last neuron, in turn, contains the activation function *tanh* **(19)**:

$$\tilde{h}_t = tanh(W \cdot [r_t * h_{t-1}, x_t]). \tag{19}$$

The new value of the state vector can be obtained from equation **(20)**:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t. \tag{20}$$

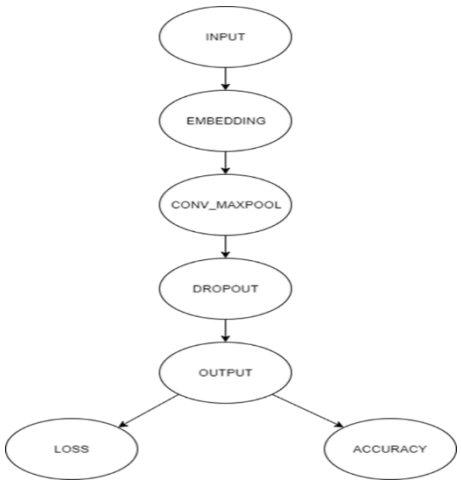
As a result, a fairly simpler model can be obtained compared to the usual Global Consumer Community Platform (GCCP).

CONVOLUTIONAL NEURAL NETWORK

CNNs are known to have made significant progress in image classification and are a fundamental component of various computer vision systems, such as self-driving cars and automatic tagging of Facebook photos. However, they are now used in natural language processing applications. Typically, a CNN consists of convolutional, pooling (sub-sampling), and fully connected output layers in any order. A network can contain a combination of all three types of layers. In the convolutional layer, neurons that use the same weights are combined into feature maps, and each neuron in the feature map is connected to a part of the neurons in the previous layer. When the network is calculated, it appears that each neuron performs a convolution of some area of the previous layer (which is determined by the set of neurons associated with this neuron). Unlike a fully connected convolutional layer, in which a neuron is connected to only a limited number of neurons in the previous layer, a convolutional layer is similar to a convolutional operation, where only a small

weight matrix (convolution kernel) is used and migrates throughout the processed layer. Another feature of the convolutional layer is that it slightly reduces the image due to edge effects. Image pixels, which are conventionally used as CNN inputs, can be replaced by sentences or documents represented in a matrix. In this case, each row of the matrix will correspond to one word or character. A diagram of this architecture is presented in **(Figure 10)**.

Figure 10. An example of CNN visualization for natural language processing



Source: compiled by the author.

As a rule, the input data is a vector representation of a word, using the Word to Vector (word2vec) or Global Vectors for Word Representation (GloVe) methods. Word2Vec, based on neural networks, learns word representations by predicting context words from target words or vice versa, resulting in dense vectors where similar words are closer in the vector space. GloVe, on the other hand, leverages global word co-occurrence statistics to generate word embeddings by factorizing a co-occurrence matrix, leading to vectors that capture semantic relationships between words.

For example, for a 10-word sentence, using 100-dimensional embedding, the output is a 10×100 matrix. The above-mentioned architectures, including RNNs, CNNs and other models, have been implemented using software tools and frameworks. During the implementation process, various parameters such as speed and accuracy were carefully measured and evaluated to determine the performance of these architectures when applied to practical scenarios. By making careful comparisons of these architectures in terms of speed and accuracy, valuable insights were gained into their performance characteristics and suitability for different applications. This empirical analysis helped to select the most appropriate model for a particular problem domain, considering factors such as computational efficiency and prediction accuracy.

RESULTS

DEMONSTRATION OF RNN OPERATION

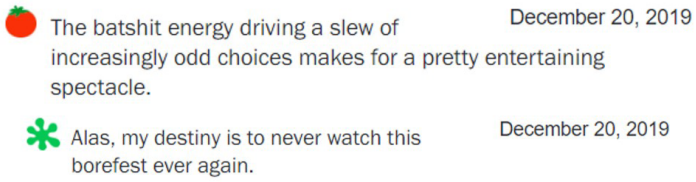
The Python programming language and the TensorFlow machine learning library were used as development tools. Python is an interpreted high-level programming language that has become one of the most popular programming languages for machine learning due to its simplicity, readability, and large number of available libraries. Its syntax is intuitive and easy to learn, making it accessible to developers of all skill levels. In

addition, Python is an open-source language, which means that anyone can contribute to its development and improvement. TensorFlow – is an open-source software library developed by Google for machine learning applications such as neural networks. It provides an easy-to-use interface for creating, training, and deploying machine learning models. TensorFlow allows establishing of complex models with many layers, making it a popular choice for deep learning applications. It can run on a variety of devices, including CPUs, GPUs, and even mobile devices. In addition, TensorFlow supports distributed computing, which allows training and scaling large models across multiple machines. The library is based on C++ but has interfaces for Python, C++, Java. In general, TensorFlow – is a powerful and flexible tool for creating machine learning models in Python.

The development of a neural network involved several key steps, including data pre-processing, model design and architecture, training, evaluation, and deployment. The process of developing a neural network using Python and TensorFlow begins with data preparation. The dataset is cleaned and pre-processed. The data is then divided into training, validation, and testing sets for evaluation purposes. During the forward propagation phase, the input data is fed into the network and computations are performed layer by layer. Each layer applies its activation function to establish an output that serves as an input to the next layer. To evaluate the performance of the network, a loss function is calculated to measure the discrepancy between the predicted outputs and the actual targets. Based on the evaluation results, the model can be tuned by making adjustments to the architecture, and hyperparameters, or using regularisation techniques to prevent overfitting. Commonly used loss functions include mean square error or cross-entropy loss. Backpropagation is then used to compute gradients of the loss function for the network parameters. These gradients are used to update the weights and offsets to minimise losses and improve model performance. Once the training process is complete, the model is evaluated using a validation or testing suite. Performance metrics such as precision, accuracy, recall, and F1 score are calculated to evaluate the model's performance. Throughout the entire development process, Python and TensorFlow provide a powerful framework for establishing and training neural networks. They offer a wide range of tools, libraries, and application programming interfaces (APIs) that simplify implementation and ensure efficient computation.

The dataset used in this machine learning task is derived from film review data obtained from RottenTomatoes, a popular online film review platform. This dataset contains 10662 reviews with an even distribution of positive and negative opinions. Specifically, half of the opinions are positive and the other half are negative. To classify the opinions as positive or negative, the snippets of opinions from the RottenTomatoes webpages that were labelled "fresh" were considered positive, while those labelled "rotten" were considered negative. Using this dataset, machine learning models can be trained to accurately classify the sentiment of film opinions **(Figure 11)**.

Figure 11. Excerpts of opinions from the Rotten Tomatoes website



Note: The first opinion contains a positive review (marked "fresh"), the second – a negative one (marked "rotten").

Source: compiled by the author.

The data are split into 2 files – rt-polarity.pos contains 5331 positive excerpts, rtpolarity.neg contains 5331 negative excerpts **(Table 1)**.

Table 1. Dataset fragments

rt-polarity.pos	rt-polarity.neg
therockisdestinedtobethe 21st century'snew "conan" andthathe'sgoingtomake a splashevengreaterthanarnold-schwarzenegger, jean-claudvandammeorstevensegal.	simplistic, sillyandtedious.
thegorgeouslyelaboratecontinuationof "thelordoftherings" trilogyissohugethat a columnofwordscannotadequate-lydescribeco-writer/directorpeterjackson'sexpandedvisionof j. r. r. tolkien'smiddle-earth.	it'ssoladdishandjuvenile, onlyteenageboyscouldpossiblyfin-ditfunny.
effectivebuttoo-tepidbiopicifyousometimesliketogoto-themoviestohavefun, wasabiis a goodplacetostart. emerge-sassomethingrare, anissuemoviethat'ssohonestandkeenlyob-servedthatitdoesn'tfeellikeone.	exploitativeandlargelydevoidofthedepthorsophisticationthat-wouldmakewatchingsuch a graphictreatmentofthecrimes-bearable.
thefilmprovidessomegreatinsightintotheneuroticmindseto-fallcomics -- eventhosewhohavereachedtheabsolutetopoft-hegame.	[garbus] discardsthepotentialforpathologicalstudy, exhuminginstead, theskewedmelodramaofthecircumstantial-situation.
offersthatrarecombinationofentertainmentandeducation.	a visuallyflashybutnarrativelyopaqueandemotionallyvapidex-erciseinstyleandmystification.
	thestoryisalsoasunoriginalastheycome, alreadyhavingbeenre-cycledmoretimesthani'dcaretocount.
	abouttheonlythingtogivethemoviepointsforisbravado -- totakeanentirelystaleconceptandpushitthroughtheaudi-ence'smeatgrinderonemoretime.

Source: compiled by the author.

The dataset contains approximately 20000 unique words, which is quite challenging to process and analyse. In addition, the dataset did not have a predefined split between training and test data. Thus, to make sure that the model is high-performing and generalizable, 10% of the data was randomly selected for testing, while the remaining 90% was used for training. The main purpose of the neural network is to classify each opinion as positive or negative, which is a binary classification problem. It allows determining the sentiment of the text and understanding how viewers perceive films. This task is crucial in the film industry, as it allows filmmakers to assess the audience's reaction to their work and make informed decisions accordingly. Data pre-processing steps:

1. Loading positive and negative sentences from raw data files.

2. Data cleaning. Remove special characters and double quotes. Regular expressions are used to do this, and the pattern search method in the feed replaces all unnecessary characters with an empty string. In addition, the feed must be reduced to lowercase letters. To do this, use the methods of converting characters to a specific case.

3. Limiting the length of each sentence to the maximum possible length of 59 words. Adding special tokens <PAD> to shorter sentences, as they need to be of the same length for efficient data processing.
Indexing each word with a specific integer from 0 to 18765. Each sentence is converted into a vector of integers **(Table 2)**.

Table 2. Data processed

Cleaned data	Data after indexing
therockisdestinedtobethe 21st century's newconanandthathe 's goingtomake a splashevengreaterthanarnoldschwarzenegger, jeanclaudvandammeorstevensegal	[1 2 3 4 5 6 1 7 8 9 10 11 12 13 14 9 15 5 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
thegorgeouslyelaboratecontinuationofthelordoftheringstrilogyissohugethat a columnofwordscannotadequatelydescribecowriterdirectorpeterjackson 's expandedvisionof j r rtolkien 's middleearth	[1 31 32 33 34 1 35 34 1 36 37 3 38 39 13 17 40 34 41 42 43 44 45 46 47 48 49 9 50 51 34 52 53 53 54 9 55 56 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
effectivebuttootepidbiopic effectivebuttootepidbiopic	[57 58 59 60 61 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
ifyousometimesliketogotothemoviestohavefu, wasabiis a goodplacetostart	[62 63 64 65 5 66 5 1 67 5 68 69 70 3 17 71 72 5 73 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
emergesassomethingrare, anissuemoviethat 's sohonestandkeenlyobservedthatitdoesn'tfeellikeone	[74 75 76 77 78 79 80 13 9 38 81 12 82 83 13 84 85 86 87 65 88 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
thefilmprovidessomegreatinsightintotheneuroticmindsetofallcomicseventhosewhohavereachedtheabsolutetopofthegame	[1 89 90 91 92 93 94 1 95 96 34 97 98 19 99 100 68 101 1 102 103 34 1 104 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
offersthatrarecombinationofentertainmentandeducation	[105 13 77 106 34 107 12 108 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
perhapsnopictureevermadehasmoreliterallyshowedthattheroadtohellispavedwithgoodintentions	[109 110 111 112 113 114 115 116 117 13 1 118 5 119 3 120 121 71 122 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
steersturnsin a snappyscreenplaythatcurlsattheedgesit 's socleveryouwanttohateitbuthesomehowpullsitoff	123 124 125 17 126 127 13 128 129 1 130 84 9 38 131 63 132 5 133 84 58 14 134 135 84 136 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Source: compiled by the author.

The analysis of the above data on the example of one training step of a conventional RNN was performed using the formulas described in the second section. First, the weights *U*, *W*, *V* of the neural network were initialised randomly using the standard normal distribution **(21-23)**:

U=[[-2.00456907e-03, -8.34052357e-04, 6.10279870e-04,
2.98218282e-04, 9.35577816e-04, -2.73085004e-03,
1.13186633e-03, 2.93236646e-04, 1.92821343e-04,
1.45058171e-03, -3.73766472e-04, -4.42388945e-04],
[-2.70303440e-04, 3.22807853e-04, 6.87949363e-04,
1.24186513e-03, 7.46562625e-04, 1.86563700e-04,
-9.20280693e-04, 9.53312541e-04, -1.10615129e-03,
-1.10598617e-03, -8.35331015e-04, -1.87544232e-03],
[5.01363176e-04, -9.24946704e-04, 2.13172721e-03,

-2.30123171e-03, 1.18675087e-03, 8.67312143e-04,
4.67885259e-04, -1.90070662e-03, -1.07344439e-04,
-6.60950010e-04, -9.38031675e-04, 1.20613743e-03],
[-1.04895964e-03, -3.54182117e-05, 1.20204007e-03,
-1.49670460e-03, -1.78542349e-05, 5.55857297e-04,
4.86410704e-05, 5.80639118e-04, 4.02617033e-04,
-4.98332578e-04, -6.88800337e-04, -2.04081040e-03],
[5.98590384e-04, -7.36782837e-04, 8.56020828e-04,
-4.00714846e-04, -1.07904359e-03, -5.26005845e-05,
-2.67968309e-04, 2.74468294e-04, -2.34090581e-04,
1.11609294e-03, -1.00539672e-03, -4.84710832e-04],
[-1.51924959e-03, 1.64434894e-03, 1.56213856e-04,
-1.35289399e-03, 4.88548883e-05, 8.84324689e-04,
-6.45691326e-04, -8.06647654e-04, 6.94644673e-04,
3.83385077e-05, -1.08716131e-03, -5.18897641e-04],
[4.86007242e-04, 3.82236344e-04, -5.03780216e-04,
-5.38758226e-04, -6.90489313e-04, 8.36563848e-04,
-2.34637708e-03, 1.44254311e-03, -4.16121256e-05,
4.07545900e-04, 9.08219628e-04, -1.38770469e-03],
[9.26889193e-04, -9.10883671e-04, -1.24136589e-03,
6.46345510e-04, -1.08363613e-04, 3.11623071e-04,
1.44847470e-03, 1.75157354e-03, -1.41342266e-03,
-3.16462417e-04, 5.04558895e-05, 1.29151023e-03],
[6.76194441e-04, -7.30576819e-04, 1.17405100e-03,
-5.22469350e-04, 1.42111622e-03, -9.51555785e-04,
-2.03822068e-03, -4.65060799e-04, -2.67421725e-03,
1.98944518e-03, -9.24845568e-05, -5.08209844e-04],
[8.84437086e-04, 4.74557193e-04, -8.22125491e-04,
-5.05219400e-05, 1.45807120e-03, 6.08441881e-04,
-2.18019454e-04, 1.24282506e-03, 6.48709211e-05,
-1.49215696e-03, -8.20555645e-04, 3.76936320e-04],
[-1.66276520e-06, 7.58396673e-04, -8.92426805e-04,
9.28591574e-04, -8.14924826e-04, 7.81197309e-04,
3.07809045e-04, -3.29337032e-03, -9.40217371e-04,
-1.00550139e-03, -3.65265185e-04, -2.24458795e-04],
[-2.44486783e-04, 1.23370027e-04, -7.45227268e-04,
-5.77545648e-04, 4.89536133e-04, 5.91256474e-04,
-1.78993055e-03, -2.76947331e-03, -7.93885838e-06,
1.94844011e-04, -1.99312866e-04, 3.38654328e-04]], (21)

W=[[-7.43501058e-04, 1.13945632e-03, 2.71596508e-03,
6.83213773e-04, 3.67918030e-04, -1.24179678e-03,
4.21751712e-04, 1.50503537e-03, -8.56542758e-04,
1.00762307e-04, 7.17365791e-04, 1.51315637e-03,
6.09443791e-04, -1.01827016e-03, -3.78503815e-04,
-1.70011665e-03, -8.91450449e-04, 1.94388497e-04],
[-1.72146197e-03, 5.67313869e-04, 7.92918946e-04,
-9.29384449e-04, -1.62428072e-04, 3.48908579e-04,
-2.39202090e-04, 5.67652365e-04, -2.59817590e-04,
1.63069431e-03, -2.10457961e-03, 7.18617451e-04,
1.13670752e-03, 5.54216340e-04, -2.06734280e-03,
6.09246538e-04, -7.71383392e-04, 3.68883349e-04],
[-1.74287629e-04, 2.14087598e-03, 1.35637737e-03,
-7.34809974e-04, -9.11367887e-04, 1.57182967e-04,
-1.53899671e-03, 3.40326624e-04, -1.60058378e-04
-1.56729154e-03, 4.25455264e-04, 5.87848314e-05,
-5.00407737e-05, -4.05929107e-04, -1.62461135e-03,

1.20197458e-03, -1.76014949e-03, -8.34966591e-04],
[6.09153394e-04, 3.73298147e-04, 7.59758625e-04,
3.76298435e-04, -5.20334544e-04, -7.35782230e-04,
-1.04365545e-03, 2.56117182e-04, -2.03382281e-03,
1.67316466e-03, -1.00568158e-03, -8.29733445e-04,
-1.02387087e-03, -4.35958028e-04, 3.68695265e-04,
5.15849235e-04, 1.36106059e-03, 4.06744080e-04],
[8.75547059e-05, 1.00182012e-03, 8.29474157e-04,
-1.47439758e-04, 2.36613439e-05, -6.89341343e-04,
5.83526975e-04, -2.43163973e-05, -1.67937825e-04,
1.75877924e-04, 1.65482455e-03, -5.83321893e-04,
-1.20862307e-03, -1.13386203e-03, -1.04186466e-03,
-6.91273610e-04, -1.14799489e-04, 4.86630674e-04],
[-1.30327635e-03, -1.61791617e-03, -5.67636990e-04,
6.46376907e-04, 8.88009287e-04, -1.29191249e-03,
1.41325038e-03, 2.17118457e-03, 1.01552883e-03,
1.48617835e-03, -1.48429087e-03, -2.03796678e-04,
6.52084473e-04, 1.31820340e-03, -1.83054445e-03,
1.39339255e-03, -1.80003356e-04, -5.04561851e-04],
[9.68814534e-04, -4.25911385e-04, -7.15609093e-04,
1.55393186e-03, 1.14034155e-03, -8.26087630e-04,
-2.67735732e-04, 6.87901233e-04, 1.16043724e-03,
-5.63456218e-04, -1.54130581e-05, 7.02852308e-04,
3.45119565e-04, 1.49392689e-04, 1.97596737e-03,
-2.30287667e-05, 8.42035270e-04, -1.34766097e-03],
[-1.41206943e-03, 5.03993973e-05, -6.73991833e-04,
-1.45240979e-03, 8.62530937e-05, -1.17570864e-03,
-3.96391720e-04, -7.78602663e-05, 1.35673320e-04,
-1.68444191e-04, -2.88154553e-04, 2.60281519e-04,
-5.44921102e-04, 9.19103268e-04, 8.37950658e-04,
-1.91479886e-03, 7.28326869e-04, -1.87211398e-03],
[1.27126127e-03, -7.94735334e-04, -6.09872172e-04,
7.88531050e-04, -5.31216003e-04, 3.65503364e-04,
-7.20699580e-04, -1.92067965e-03, -8.26947417e-04,
-5.24688360e-04, -2.32615824e-04, 7.92341691e-04,
1.80021848e-03, 1.57361850e-03, 6.23740703e-04,
1.14223262e-03, -7.36898731e-04, -1.72218062e-03],
[1.24967947e-03, -3.99473435e-04, 2.02734792e-03,
-3.76105023e-04, -1.74571177e-04, -5.20932309e-04,
1.10343809e-03, -2.17150734e-04, -1.65242278e-05,
7.72270978e-04, -1.38699112e-04, -1.07715531e-03,
-9.66706468e-05, -1.17833491e-03, -7.70796009e-04,
-1.30135852e-03, -1.46655035e-04, -2.15277291e-04],
[3.62839955e-04, 1.42670053e-03, 1.20191656e-03,
-3.75126260e-04, 1.64015236e-04, -9.07869068e-04,
2.77180986e-03, -2.10635378e-04, -1.72432980e-04,
7.83437578e-04, -1.14828276e-03, -8.62950857e-04,
1.60885201e-03, 7.71301617e-04, 1.38947992e-03,
7.78340340e-05, -1.03855478e-03, -8.72731154e-04],
[-5.37873976e-04, -7.92708678e-04, -7.76586129e-04,
-1.89621443e-03, 1.90244359e-03, -1.53229409e-03,
-9.95312560e-04, -3.34126850e-04, -5.95603171e-04,
4.39441070e-04, -1.27540524e-04, 1.16436721e-03,
-1.13342838e-04, -7.81671085e-04, 1.28612465e-03,
-3.85553064e-04, 1.84815410e-03, 4.05814493e-04]], (22)

V=[[2.45870624e-04, -3.87342067e-05, -4.04186783e-04,

1.28670309e-03, 2.51081193e-04, 8.24950409e-04,
-1.54277976e-03, -5.44293303e-04, -1.49104413e-03,
1.46696367e-04, 3.75921175e-03, 1.65118961e-03],
[-1.02563196e-03, 1.51516964e-03, -1.03714671e-04,
1.45839393e-03, -6.99300673e-04, -1.14206067e-03,
-2.67702479e-05, -1.86604511e-06, 2.22941322e-04,
-2.71639587e-04, 3.69203145e-04, -1.51000287e-03]]. (23

In addition, initialise $\{x_t\}_{(t=1)}^{T}$ – the representation vector of the input sentence. For example, the sentence "Moviewasgood" will be indexed as [0 1 2]. At each step of the calculation, the unitary vector of each word will be used as , the length of which will correspond to the number of unique words in the dataset. In the abovecase, there are three of them, thus unitary vectors for each word:

- word "Movie" has an index of 0, and its unitary vector is [1, 0, 0];
- word "was" having an index of 1, and its unitary vector is [0, 1, 0];
- word "good" has an index of 2, and its unitary vector is [0, 0, 1].

Applying formula (1) for each unitary vector , taking the zero vector as the initial value of H_{t-1} , obtained **(24)**:

[[-0.00013509]
[0.00013733]
[0.00045064]
[-0.00079666]
[-0.00062898]
[0.00120057]
[-0.00044855]
[0.00047876]
[-0.00038644]
[0.00113644]
[0.00036988]
[-0.00076792]]. (24)

The vector of initial values is obtained **(25)** using expression (2):

[[-0.01000172]
[0.01000092]]. (25)

Using function *softmax*, the final predictions were calculated **(26)**:

[[0.49499951]
[0.50500049]]. (26)

To improve the results of the neural network, the weights are recalculated using the back-propagation method. This technique plays a crucial role in adjusting the network parameters based on the mismatch between the predicted outputs and the actual goals. Different optimisation algorithms can be used to improve the optimisation process. These algorithms, such as stochastic gradient descent or its variants such as Adam or RMSprop, introduce additional methods to improve the weight updates. They often include momentum, adaptive learning rate, or other enhancements to speed up convergence and avoid getting stuck in local minima.

IDENTIFICATION AND EVALUATION OF QUALITY INDICATORS OF DIFFERENT TYPES OF ARCHITECTURE

To determine and evaluate the qualitative indicators of different architectures, a software implementation of RNNs for binary classification of the processed dataset is performed. The Python programming language and the TensorFlow machine learning library are used as development tools. The software performed a training cycle of each type of RNN for 100 epochs (15000 steps). The qualitative indicators of each neural network (training time in seconds, accuracy) obtained after the training process are presented in **Table 3**.

Table 3. Recurrent neural network training results

Name	Time, s	Number of steps	NuAccuracy on test datamber of steps
Recurrent neural network	1337	15000	0.67893
Long short-term memory	2136	15000	0.722326
Gated recurrent units	1815	15000	0.729193

Source: compiled by the author.

Comparing the data in Table 3, it can be concluded that the fastest training time for a simple RNN is 1337 seconds, but its accuracy is inferior to LSTM and GRU. Therewith, the accuracy rates of LSTM and GRU are almost the same (72.2% and 72.9%, respectively), but GRU is trained faster (1815 c, versus 2136 c for LSTM). Subsequently, the hyperparameter of the output vector dimension was changed to better analyse the qualitative performance of different architectures. When establishing an instance of the RNN class in the training cycle, the value of the hidden_size variable, which contains information about the output vector of each layer, was changed. **Table 4** presents the results of training a conventional RNN when the dimension of the output vector of the recurrent layer of the network is 12, 64, 128, and 512.

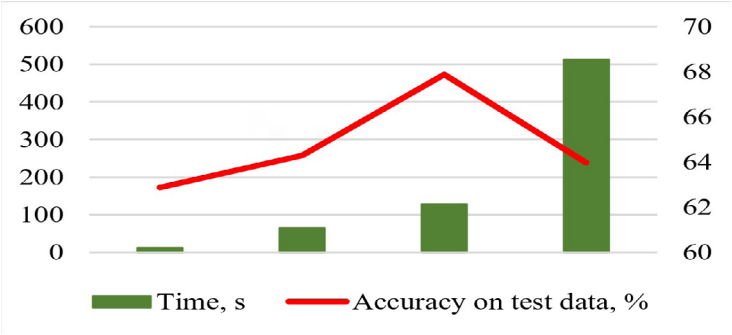
Table 4. Training results with different dimensions of the output vector for RNN

Size	Time, s	Number of steps	NuAccuracy on test datamber of steps
12	615	15000	0.62879
64	960	15000	0.64276
128	1337	15000	0.67893
512	4679	15000	0.6398

Source: compiled by the author.

The dependence of training time and accuracy on the dimensionality of the output vector for RNNs is presented in **(Figure 12)**.

Figure 12. Dependence of training time and accuracy on the dimension of the output vector for RNN



Source: compiled by the author.

A similar analysis was conducted for other types of architecture. Thus, **Table 5** presents the results of training LSTM and GRU, respectively. The values of the dimension of the output vector of the recurrent network layer are similar to those for the RNN case.

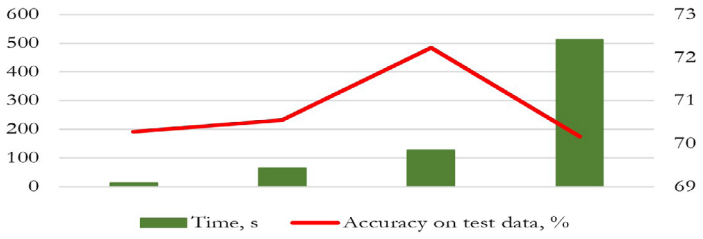
Table 5. Training results with different dimensions of the output vector for LSTM and GRU

Size	Time, s	Number of steps	NuAccuracy on test datamber of steps
Long short-term memory			
12	869	15000	0.70275
64	1136	15000	0.70541
128	2136	15000	0.722326
512	8330	15000	0.70172
Gated recurrent units			
12	763	15000	0.69439
64	916	15000	0.71341
128	1815	15000	0.729193
512	6716	15000	0.70452

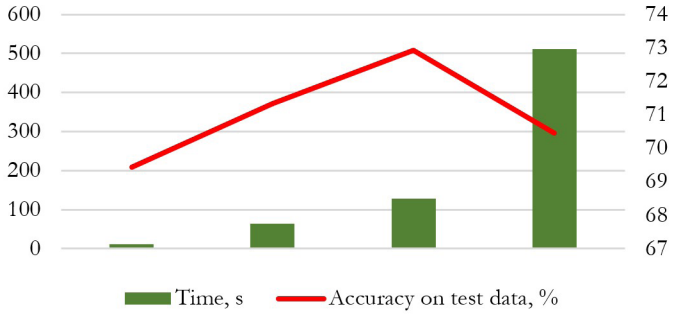
Source: compiled by the author.

The dependencies of training time and accuracy on the dimensionality of the output vector for LSTM and GRU are presented in **(Figure 13)**, respectively.

Figure 13. Dependence of training time and accuracy on the dimensionality of the output vecti=or a) for LSTM and b) GRU



a)



b)

Source: compiled by the author.

The experimental results demonstrated that when the output vectors of the recurrent layer were too large, the model tended to overfit the data, which led to a drop-in accuracy. The overfitting problem was observed when the output vector size exceeded 512. On the other hand, the best accuracy on the test dataset was obtained for all neural network architectures when the output vector size was set to 128, indicating that this was the optimal size for the problem at hand. Therefore, it was concluded that a larger output vector size does not improve the performance of the model but only increases its complexity. At the final stage, a completely different class of networks was used for the same task – a CNN **(Table 6)**.

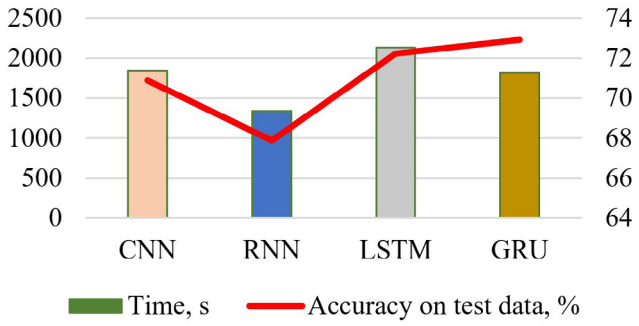
Table 6. CNN and RNN training results

Name	Time, s	Number of steps	NuAccuracy on test datamber of steps
Convolutional neural networks	1845	15000	0.70893
Recurrent neural networks	1337	15000	0.67893
Long short-term memory	2136	15000	0.722326
Gated recurrent units	1815	15000	0.729193

Source: compiled by the author.

A comparison of the 4 examined neural networks in terms of their training time and accuracy is presented in **(Figure 14)**.

Figure 14. Graph comparing the 4 examined neural networks in terms of their training time and accuracy



Source: compiled by the author.

Based on the results of training the CNN, it was found that the training period lasted slightly longer than that of the GRU. However, the accuracy obtained by the CNN was in the range of the results obtained by the RNN, indicating that CNNs can be used for this classification problem. Notably, the VRV RNN architecture performed the best among all the networks considered. Thus, it can be concluded that both CNNs and RNNs have their advantages and can be used in different scenarios based on the specific requirements of the problem at hand.

To summarise, RNNs, especially LSTM networks, have demonstrated their success in capturing the sequential nature of textual data. They can retain the memory of previous inputs, and their output depends on the current input and the previous hidden state. This makes them particularly suitable for tasks such as sentiment analysis, where the meaning of a word or phrase may depend on the context of the surrounding text. On the other hand, in recent years, CNNs have been successfully applied to natural language processing tasks. They are especially suitable for tasks that require the identification of local features in the input data, such as text classification, and can learn meaningful representations of text data by identifying important features at different levels of abstraction. CNNs have been proven to be particularly effective for short to medium texts, such as product reviews or tweets, and can outperform conventional word-of-mouth approaches. In general, the choice between RNNs and CNNs for text classification depends on the specific problem and dataset. While RNNs are better suited for capturing sequential dependencies in long texts, CNNs can perform well with short to medium texts by extracting local features and can be trained faster than RNNs. However, notably, there is frequently no clear winner and both architectures should be explored and compared for a particular task.

DISCUSSION

The field of neural network research is constantly evolving, with new architectures and methods being developed regularly. By considering a range of architectures, researchers can keep abreast of the latest advances and contribute to the development of the field. Including other architectures in future research is necessary to explore the full range of possibilities and determine the most appropriate models for specific neural networking tasks [25-29]. By including other architectures in future research, it will expand understanding of their capabilities and explore their effectiveness in different domains. In addition, including a diverse set

of architectures helps to identify potential synergies and opportunities for hybrid models that combine the strengths of several approaches.

P. Zhou et al. [30] in their research consider one of these networks and solve the problem of efficient representation and classification of text data. They propose a new architecture that combines a bidirectional LSTM, a type of RNN capable of extracting contextual information from past and future sequences, with a two-dimensional maximum pooling, a technique commonly used in computer vision tasks. The key idea behind their approach is to exploit the bidirectional nature of LSTM to capture both sequential dependencies in text and contextual information from both areas. By integrating the bidirectional LSTM with a two-dimensional maximal pooling, the model can extract important features from the text representation while preserving the spatial relations between words [31-34]. The study by P. Zhou et al. [30] can complement the main study by providing valuable information on various aspects of text classification using neural networks. A comparative analysis of different RNN and CNN architectures allows us to understand the relative strengths and weaknesses of these models in terms of training time and accuracy.

C. Zhou et al. [35] solve the problem of efficiently capturing local and global contextual information from text sequences. Conventional methods, such as RNNs and CNNs, have limitations in capturing long-term dependencies or maintaining spatial relations between words. The Z-LSTM architecture is designed to overcome these limitations. The key idea of the Z-LSTM model is to combine the strengths of CNN and LSTM. The research proposes a hierarchical structure where convolutional layers are followed by LSTM layers. Convolutional layers act at the character level, highlighting local features and preserving spatial relations. These features are then passed on to the LSTM layer, which captures global dependencies and the context of the text. Conversely, study by C. Zhou et al. [35] can benefit from the findings of the authors study by benchmarking its Z-LSTM model against established RNN and CNN architectures, providing a more robust assessment of its effectiveness and highlighting its unique advantages in capturing both local and global contextual information in text sequences. Together, these studies can contribute to a richer and more nuanced understanding of neural network approaches for text processing.

Thus, in the work of L. Huang et al. [36] propose a new approach to text classification using graph neural networks (GNN). The authors argue that conventional approaches to text classification, which treat text as an unordered collection of words, are unable to capture the rich semantic relationships between words and phrases in natural language. To address this problem, they propose a framework that creates a graphical representation of text where nodes correspond to words or phrases and edges represent semantic relationships between them. They then apply a GNN to represent this graph to learn a high-level representation of the text that is fed into the classifier. The research evaluates the proposed approach on several benchmark text classification datasets and demonstrates that it outperforms several state-of-the-art approaches. By combining the author's and L. Huang et al. [36] findings, researchers can potentially create hybrid models that leverage the strengths of both RNNs/CNNs and GNNs, achieving even higher accuracy in text classification tasks while efficiently capturing semantic information within the text data.

The research of R. Wang et al. [37] proposes a hybrid deep learning architecture for text classification that combines the advantages of RNNs and CNNs. The authors argue that while CNNs are effective in capturing local patterns in text, they have difficulty capturing long-term dependencies, while RNNs are excellent at modelling sequential data but may miss important local patterns. To overcome these limitations, the authors propose a convolutional recurrent neural network (CRNN) architecture that combines both RNNs and CNNs. The model first applies a series of convolutional layers to the input text, removing local features. The resulting object maps are then fed into an RNN that captures long-term relationships between local objects.

Finally, the RNN output is fed to a fully connected layer for classification. The authors evaluate their approach on several benchmark text classification datasets and demonstrate that it outperforms several state-of-the-art approaches, including traditional RNNs and CNNs. The main and R. Wang et al. [37] studies provide a more comprehensive toolkit for researchers and practitioners in the field of text classification, offering a range of model choices and innovative hybrid solutions to tackle different text classification challenges.

In addition, an approach using RNNs and CNNs is presented in G. Chen et al. [38]. The research considers the problem of categorising text with multiple labels when a document can belong to several categories simultaneously. The authors propose an integrated approach that combines two types of neural networks. Here, CNNs are used to extract features from the input text through a series of convolution and pooling operations, and RNNs are used to model temporal dependencies between words in the input text using long-term short-term memory units. As a result, the proposed approach combines the strengths of both types of architecture. In particular, CNNs are used to capture local features of the input text, while RNNs are used to capture global features and temporal dependencies between words. The research of the authors of this article and the results of G. Chen et al. [38], combined, offer a broader perspective on the application of RNNs and CNNs in text analysis, allowing researchers to make an informed choice based on the specific requirements of their text classification tasks.

Another interesting approach is shown in the work of J.S. Manoharan [39]. The researcher proposes using a capsule network (CapsNet) for text classification. CapsNets are a type of neural network that uses "capsules" instead of traditional neurons, which can encode various spatial relationships between objects. The author argues that CapsNets are better suited for text classification than conventional neural networks as they can capture hierarchical relationships between words and phrases. The proposed architecture consists of several layers of capsules that gradually aggregate information from the input text. The capsules in each layer compute a "pose matrix" that encodes the position, orientation, and scale of the input item. The pose matrices are then transformed by a series of "routing" layers that compute a weight matrix indicating the degree to which each capsule should be assigned to a higher-level capsule. The authors of the present study consider text classification from a different perspective, focusing on existing neural networks, while J.S. Manoharan [39] proposes a completely new architecture. The choice between these approaches will depend on the specific requirements and characteristics of the text classification task.

J. Faouzi and O. Colliot [40] explored the main classic machine learning methods. According to the results, RNNs excel at handling sequential data, as they maintain memory of previous inputs and are adept at understanding context and relationships in a sequence. They can capture long-term dependencies, crucial for tasks involving longer texts. However, they struggle with very long sequences, are computationally intensive, and can be memory-intensive. On the other hand, CNNs are efficient for parallel processing, excellent at extracting local features, and effective for shorter texts where local context matters. Nevertheless, they may not capture the sequential nature of text as effectively as RNNs and are limited by fixed input sizes, making them less suitable for very long texts. In general, CNNs perform well on short texts like sentences or short paragraphs, while RNNs, particularly LSTMs and GRUs, shine in handling longer texts for tasks like machine translation or document summarization. Together with the author's study, J. Faouzi and O. Colliot [40] work offer a nuanced perspective on the versatility of neural network architectures in text processing, enabling practitioners to make informed decisions when selecting models for various text classification tasks.

In summary, comparing the performance of RNNs and CNNs for semantic text colouring analysis is crucial. Currently, RNNs outperform traditional machine learning algorithms, achieving high accuracy and F1 scores, while CNNs are continually improving for text classification. However, it's important to acknowledge that

neither RNNs nor CNNs are flawless, and there are limitations to their performance. Additionally, given the ever-evolving nature of natural language processing, exploring alternative models like GNN, CRNN, or CapsNet is equally important.

CONCLUSIONS

In this research, a comparative analysis of RNN architectures and CNNs was performed using the example of the binary text classification task. Comparing the data presented in Section 3, it can be concluded that a simple RNB is the fastest to train (1337 seconds), but its accuracy is inferior to LSTM and GRU. Therewith, the accuracy rates of LSTM and GRU are almost the same (LSTM – 72.2%, GRU – 72.9%), but GRU is trained faster (GRU – 1815 seconds, LSTM – 2136 seconds). The training period for CNN took a little longer than for GRU, and the accuracy is within the range of the results of a conventional RNN, thus, it can be concluded that CNN can be used for this classification problem. However, RNNs proved to be the best. The obtained results provide valuable information about the effectiveness of RNNs and CNNs for classifying semantic colouring in text. By analysing various metrics, the performance of each model architecture was evaluated. In addition, metrics such as training time and accuracy were considered to provide a comprehensive evaluation of the models. The results of the study add to the existing knowledge in the field of text classification. They demonstrate a practical implementation of RNNs and CNNs for sentiment analysis and provide valuable insights into the performance of these models in terms of their ability to accurately classify movie viewing sentiment.

In summary, analysis highlights the importance of carefully selecting parameters to match a dataset's underlying patterns. In sentiment analysis task, RNNs outperformed other classifiers, such as CNNs, due to RNNs' aptitude for sequential data processing, a key feature of textual data. Hence, for similar tasks, we recommend considering RNN-based architectures. However, it's important to note that this conclusion is dataset-specific, and different datasets or architectures may yield different results.

The practical significance of this study lies in its demonstration of the operational process of RNNs using Python and TensorFlow, with a specific application in sentiment analysis of film reviews. By leveraging the power of Python's simplicity and TensorFlow's flexibility, the study showcases the entire pipeline of developing a neural network model, including data pre-processing, model architecture design, training, evaluation, and deployment. Future research should focus on testing various models and architectures to enhance the accuracy and efficiency of text classification systems in this evolving field.

ACKNOWLEDGEMENTS

The study was created within the framework of the project financed by the National Research Fund of Ukraine, registered No. 30/0103 from 01.05.2023 "Methods and means of researching markers of ageing and their influence on post-ageing effects for prolonging the working period", which is carried out at the Department of Artificial Intelligence Systems of the Institute of Computer Sciences and Information of technologies of the Lviv Polytechnic National University.

REFERENCES

1. **Martins RM, Gresse Von Wangenheim C.** Findings on teaching machine learning in high school:A ten-year systematic literature review. Informatics in Education. 2022. https://doi.org/10.15388/infedu.2023.18

2. **Hopkins E.** Machine learning tools, algorithms, and techniques. J Self-Gov Manage Econ. 2022. 1: 43-55

3. **Murphy KP.** Probabilistic machine learning: An introduction. Cambridge: MIT Press; 2022.

4. **Zhou ZH.** Open-environment machine learning. National Science Review, 2022. 9(8): nwac123.

5. **Zhu J, Jiang Q, Shen Y, Qian C, Xu F, Zhu Q.** Application of recurrent neural network to mechanical fault diagnosis: A review. J Mech Sci Technol. 2022. 36: 527-542.

6. **Ghimire D, Kil D, Kim SH.** A survey on efficient convolutional neural networks and hardware acceleration. Electronics, 2022. 11(6): 945.

7. **Lukashchuk H, Onufriv Ia, Tupis S.** Green space and planning structure optimisation ways in parks and monuments of landscape architecture. Archit Stud. 2023. 9(1): 23-35.

8. **Hasanli R, Aliyev I, Poladov N, Azimova L, Tagiyev T.** Isothermal transformations in high-strength cast iron. Sci Herald of Uzhhorod Univ Series "Physics" 2022. (51): 48-58.

9. **Nazari M, Alidadi M.** Measuring credit risk of bank customers using artificial neural network. J Manage Res. 2013. 5(2): 17-27.

10. **Pluzhnyk O, Drok P.** Electronic document management in the system of modern libraries: efficiency, security and innovation. Soc Doc Commun. 2023. 19: 198-212.

11. **Wüthrich MV**. Bias regularization in neural network models for general insurance pricing. Eur Actuar J. 2020. 10: 179-202.

12. **Sedliačik J, Pinchevska O, Lopatko K, Lopatko L.** Effect of magnesium nanoparticles on formaldehyde emissions from wood composite materials. Ukr J For Wood Sci. 2023. 14(3): 78-90.

13. **Aviv I, Barger A, Pyatigorsky S.** Novel Machine Learning Approach for Automatic Employees' Soft Skills Assessment: Group Collaboration Analysis Case Study. 5th International Conference on Intelligent Computing in Data Sciences, ICDS 2021. Virtual, Online: Institute of Electrical and Electronics Engineers. 2021. https://doi.org/10.1109/ICDS53782.2021.9626760

14. **Kulgildinova TA, Uaissova GI.** Realization of frame-based technologies in the context of education informatization. J Theor Appl Inf Technol. 2016. 89(1): 236-242.

15. **Tabriz N, Nurtazina ZB, Kozhamuratov MT, Skak K, Mutaikhan Z.** Effects of secondary infections on the multidrug-resistant Tuberculosis: A cohort study. Med J Islamic Repub Iran 2021. 35(1):1-7.

16. **Shynkariuk IuM.** Alternative representation of space and time: Geometric solution of problems of relativity theory. Sci Herald of Uzhhorod Univ Series "Physics" 2022. (51): 74-82.

17. **Panov V.** The scientific process of two interferometers (optical) development and the mitigation of external influence. Sci Herald of Uzhhorod Univ Series "Physics" 2023. (53): 19-30.

18. **Kirimbayeva Z, Abutalip A, Mussayeva A, Kuzembekova G, Yegorova N.** Epizootological monitoring of some bacterial infectious diseases of animals on the territory of the Republic of Kazakhstan. Comp Immunol Microbiol Infect Dis. 2023. 102: 102061.

19. **Oliynyk O, Barg W, Oliynyk Y, Dubrov S, Gurianov V, Rorat M.** Lack of Difference in Tocilizumab Efficacy in the Treatment of Severe COVID-19 Caused by Different SARS-CoV-2 Variants. J Pers Med. 2022. 12(7): 1103.

20. **Ho NH, Yang, HJ, Kim SH, Lee G.** Multimodal approach of speech emotion recognition using multi-level multi-head fusion attention-based recurrent neural network. IEEE Access. 2020. 8: 61672-6168.

21. **Chamishka S, Madhavi I, Nawaratne R, Alahakoon D, De Silva D, Chilamkurti N, Nanayakkara V.** A voice-based real-time emotion detection technique using recurrent neural network empowered feature modelling. Multimedia Tools Appl. 2022. 81: 35173-35194.

22. **Barzegar V, Laflamme S, Hu C, Dodson J.** Ensemble of recurrent neural networks with long short-term memory cells for high-rate structural health monitoring. Mech Syst Signal Process. 2022. 164: 108201.

23. **Yao L, Mao C, Luo Y.** Graph convolutional networks for text classification. Proceedings of the AAAI Conf Artif Intell, 2019. 33(1): 7370-7377.

24. **Yig W, Kann K, Yu M, Schütze H.** Comparative study of CNN and RNN for natural language processing. 2017. https://doi.org/10.48550/arXiv.1702.01923

25. **Tanchak A, Katovsky K, Haysak I, Adam J, Holomb R.** Research of spallation reaction on plutonium target irradiated by protons with energy of 660 MeV. Sci Herald of Uzhhorod Univ Series "Physics" 2022. (52): 36-45.

26. **Petryshyn H, Kryvorychko O, Lukashchuk H, Danylko N, Klishch O.** Changing the qualities of urban space by means of landscape architecture. Archit Stud. 2022. 8(1): 22-33.

27. **Danilenko I, Gorban O, da Costa Zaragoza de Oliveira Pedro PM, Viegas J, Shapovalova O, Akhkozov L, Konstantinova T, Lyubchyk S.** Photocatalytic Composite Nanomaterial and Engineering Solution for Inactivation of Airborne Bacteria. Top Catal. 2021. 64(13-16): 772-779.

28. **Suleymanov TA, Shukurova AS.** Antioxidant activity of metanolic extracts from the leaves of Rubus idaeus L., Juglans regia L. and aerial part of Viscum album L. From the flora of Azerbaijan. Azerb. Pharm. Pharmacother J. 2022. 22(1). https://www.azpharmjournal.org/en/jurnal/azerbaycan-florasinda-rubus-idaeus

29. **Shaprynskyi V, Nazarchuk O, Faustova M, Mitiuk B, Dmytriiev D, Dobrovanov O, Kralinsky K, Babina Y.** Some aspects of infectious complications in patients with surgical diseases mult icentr trials. Lek Obz. 2020. 69(7-8): 257-260.

30. **Zhou P, Qi Z, Zheng S, Xu J, Bao H, Xu B.** Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. 2016. https://doi.org/10.48550/arXiv.1611.06639

31. **Rakhimov G, Shevnikov M, Plahtiy D, Nedilska U, Krachan T.** Life forms of plants of natural and anthropogenic landscapes. Sci Horiz. 2023. 26(1): 62-72.

32. **Jaskiewicz F, Kowalewski D, Kaniecka E, Kozlowski R, Marczak M, Timler D.** Factors Influencing Self-Confidence and Willingness to Perform Cardiopulmonary Resuscitation among Working Adults-A Quasi-Experimental Study in a Training Environment. Inter J Envir Res Publ Heal. 2022. 19(14): 8334.

33. **Parisi GF, Brindisi G, Indolfi C, Diaferio L, Marchese G, Ghiglioni DG, Zicari AM, Miraglia del Giudice M.** Upper airway involvement in pediatric COVID-19. Pediatr. Allergy Immun. 2020. 31(S26): 85-88.

34. **Jalilova I, Mansurova LN, Zulfugarova NS.** Development of a competitive map of antibiotics for the pharmaceutical market of Azerbaijan. Azerb Pharm Pharmacother J. 2022. 22(2): 75-81.

35. **Zhou C, Sun C, Liu Z, Lau F.** A C-LSTM neural network for text classification. 2015. https://doi.org/10.48550/arXiv.1511.08630

36. **Huang L, Ma D, Li S, Zhang X, Wang H.** Text level graph neural network for text classification. 2019. https://doi.org/10.48550/arXiv.1910.02356

37. **Wang R, Li Z, Cao J, Chen T, Wang L.** Convolutional recurrent neural networks for text classification. In: 2019 International Joint Conference on Neural Networks (IJCNN). Budapest: IEEE; 2019. p. 1-6.

38. **Chen G, Ye D, Xing Z, Chen J, Cambria E.** Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In: 2017 International joint conference on neural networks (IJCNN). Anchorage: IEEE; 2017. p. 2377-2383.

39. **Manoharan JS. 2021.** Capsule network algorithm for performance optimization of text classification. J Soft Comput Paradigm. 2021. 3(1): 1-9.

40. **Faouzi J, Colliot O.** Classic Machine Learning Methods. In: O. Colliot (Eds), Machine Learning for Brain Disorders. Neuromethods, vol 197. New York: Humana; 2023. p. 25-75. https://doi.org/10.1007/978-1-0716-3195-9_2