Master Of Science
"Artificial Intelligence and Visual Computing"

UNIVERSITY OF WEST ATTICA &
UNIVERSITY OF LIMOGES

FACULTY OF ENGINEERING
DEPARTMENT OF INFORMATICS AND COMPUTER ENGINEERING

Master Thesis

Transformer models for Greek and English language understanding

Student: Charalampos Mageiridis (aivc21011)
Supervisor: Tselenti Panagiota

Athens, February 27, 2024

Transformer models for Greek and English language understanding

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Παναγιώτα Τσελέντη	ΕΔΙΠ	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2024.03.20 18:01:14 +02'00'
2	Πάρις Μαστοροκώστας	Καθηγητής	Paris Mastorokostas Digitally signed by Paris Mastorokostas Date: 2024.03.21 09:56:04 +02'00'
3	Αναστάσιος Κεσίδης	Καθηγητής	Anastasios Kesidis Digitally signed by Anastasios Kesidis Date: 2024.03.21 09:47:56 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Μαγειρίδης Χαράλαμπος του Προδρόμου, με αριθμό μητρώου aivc21011 φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών Τεχνητή Νοημοσύνη και Οπτική Υπολογιστική του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

[illegible]

Χαράλαμπος Μαγειρίδης

Abstract

This thesis examines the effectiveness of Transformer models in natural language processing (NLP), focusing on text classification and sentiment analysis for both Greek and English texts. It marks a significant advancement in NLP by demonstrating how Transformers, through their self-attention mechanisms, offer a superior approach to understanding context and relationships within language compared to traditional RNN architectures.Through a detailed exploration of several Transformer models, such as BERT, RoBERTa, and GPT-2, and their comparison with RNNs, this study evaluates their performance across different languages and text types. Fine-tuning language-specific pretrained versions of these models on two distinct datasets—Greek product reviews and English tweets related to the #metoo movement—allows for an examination of the adaptability of Transformers to diverse linguistic challenges. The research ultimately illustrates that Transformer models significantly outperform traditional RNNs in sentiment analysis, highlighting their considerable potential to enhance language understanding across varied linguistic and cultural contexts.

Contents

1 Introduction

Language forms the foundation of human communication and civilization. It is our primary tool for expressing thoughts, information, sharing emotions and for passing on knowledge from one generation to another. [31].

In the age of artificial intelligence, the importance of language understanding is not just important for humans. The quest to help machines to understand, process, and generate human language is a crucial effort in the field of computer science. This effort is called Natural Language Processing (NLP) and it has become an indispensable part of our daily lives, influencing and shaping many aspects of our interaction with digital technologies. From web search engines that understand our queries to machine translation systems that help us communicate across languages, to speech recognition software that transforms spoken language into written text, NLP is everywhere. [40].

The introduction of Transformer models has marked a significant milestone in the advancement of NLP, offering groundbreaking approaches to language understanding. These models, characterized by their self-attention mechanisms, have revolutionized how machines process and generate language, setting new standards for accuracy and efficiency in tasks ranging from machine translation to sentiment analysis [57]. The transformative impact of Transformers is not just limited to enhancing existing applications but also in paving the way for novel applications that were previously unattainable, further bridging the gap between human and machine communication.

The ability of computers to understand and interact in human language brings us a step closer to seamless human-computer interaction, opening up a world of possibilities for technology to assist, enhance, and even transform human activities. However, despite the significant strides made in NLP, language understanding in machines is still an ongoing challenge. The dynamic, and contextual nature of human language presents a complex problem for machines to solve [40].

In this thesis, we will explore the complexities of language understanding, examining the methodologies, technologies, and applications that are pushing the boundaries of what computers can comprehend and achieve. By investigating the current state-of-the-art in NLP and envisioning its potential future trajectories, we aim to contribute to this exciting and rapidly evolving field of study.

1.1 Applications of Sentiment Analysis

The integration of language understanding technologies, such as sentiment analysis, into computer systems is crucial for bridging the communication divide between humans and machines. In the digital age, an immense volume of textual data is produced daily, capturing human emotions, opinions, and experiences [33]. Analyzing this data transforms unstructured text into actionable insights, enhancing user experiences by enabling more intuitive and empathetic interactions between machines and humans. Such advancements promise to revolutionize various sectors, including customer service and market research, by aligning technology more closely with human needs.

Sentiment analysis, also known as opinion mining, is the computational process of identifying and categorizing opinions expressed in a piece of text, especially to determine whether the writer's attitude towards a particular topic, product, or service is positive, negative, or neutral. This technique is invaluable in today's digital environment, where platforms like Twitter act as repositories for a vast array of textual data, including social media comments, customer reviews, and online discussions. Automated tools scan this content to discern the author's sentiments, providing businesses with objective insights that can help improve customer service and bolster brand reputation [24].

The methods for conducting sentiment analysis range from lexicon-based approaches, which rely on predefined lists of words associated with positive or negative sentiments, to machine learning techniques that train models on labeled datasets to predict the sentiment of new, unseen texts. The advent of deep learning has seen the rise of advanced models like transformers and recurrent neural networks, favored for their ability to understand the context and subtle nuances of language [24].

The significance of sentiment analysis is particularly pronounced in the context of Twitter, a leading social media platform with over 300 million accounts, where the vast and continuously growing volume of tweets provides a rich source for assessing public sentiment and viewpoints on a wide array of subjects [54], [19].

1.1.1 Sentiment analysis on Twitter data

Sentiment analysis of Twitter has gained significant traction and importance in recent years. Twitter stands as one of the most dominant social media platforms of this era, having over 300 million accounts and witnessing daily surges in user numbers [19].

Its content, reflecting people's opinions and sentiments, expands exponentially every minute, establishing it as the primary hub for assessing public sentiment and viewpoints in many subjects [54].

2 Background

2.1 Introduction

Artificial Intelligence is a vast domain that seeks to mimic human actions, enabling machines to undertake a plethora of tasks such as problem-solving, knowledge representation, and voice recognition, to name a few. At its core, the idea is to instill knowledge into machines.

With the progress in this field, we now possess advanced systems capable of adjusting their behavior without changing their code, depending solely on updates to their training data. In the midst of this wave of cutting-edge machine learning techniques driving AI's evolution, one might ponder the significance of Deep Learning [1].

Nowadays, deep learning is at the forefront of attention because its performance is much more significant than any other machine learning algorithm in such complex tasks. For instance, we note the following:

- Advanced image segmentation using multi-layered convolutional networks to identify and isolate objects within complex scenes [45].
- Implementing sequence-to-sequence models with attention mechanisms to translate sentences from one language to another while preserving the semantic meaning [57].
- Employing transformer-based models like BERT to understand the context within a text and perform tasks like question-answering, named entity recognition, sentiment analysis etc [29].
- Using generative adversarial networks (GANs) to create realistic images or text based on a given dataset, which can be used for art, design, or even generating synthetic datasets [53].

Deep learning, as explained by pioneers Yann LeCun, Yoshua Bengio, and Geoffrey Hinton in [36], is a subset of machine learning characterized by its multi-layered representation-learning methods. These methods involve multiple levels of representation, each built upon the other, starting from the raw input and progressively becoming more abstract.

By composing these simple yet non-linear modules, deep learning can learn highly complex functions. For tasks like classification, the higher layers of representation emphasize the aspects crucial for discrimination while diminishing irrelevant variations [49].

The relationship among the three concepts artificial intelligence , machine learning, and deep learning is summarized by the authors in the following figure:

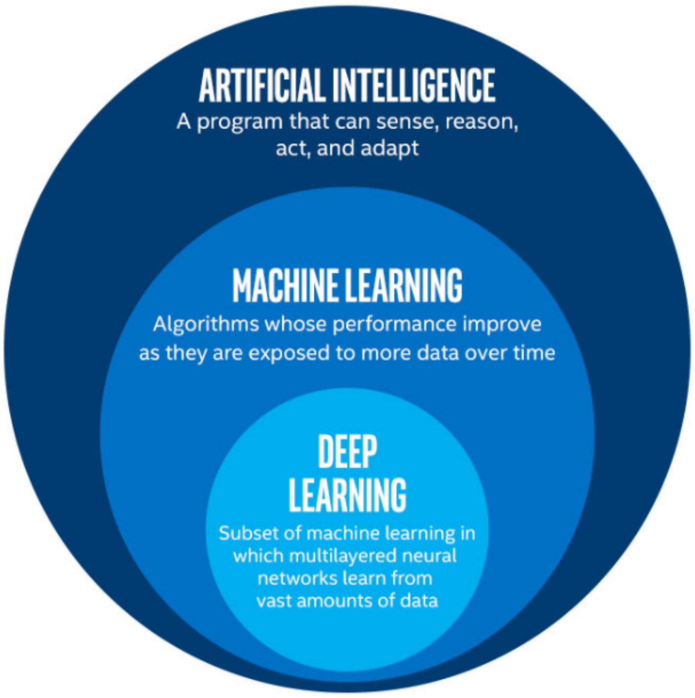

ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Figure 1: Artificial Intelligence [3]

2.2 Neural Networks

Neural networks, also referred as artificial neural networks (ANNs), belong to the machine learning domain and form the foundation of deep learning techniques. The term ”neural” is used because they try to replicate the signaling mechanism of brain neurons. In the figure 2 we can see a real neuron and in the figure 3 the artificial equivalent [23].

These networks as shown in figure 4 consist of layers of nodes: an initial input layer, several hidden layers, and a final output layer. Every node represents an artificial neuron linked to the subsequent one, possessing its own weight and threshold. If a node's output surpasses the threshold, it gets activated, transmitting its information to the succeeding layer. Otherwise, no data is forwarded [42].

Through training data, neural networks enhance their precision. Once these learning methods are optimized, they emerge as potent tools in computer science and AI. This is because they facilitate rapid data classification and grouping. With neural networks, tasks like voice and image identification can be completed in mere minutes, a process that manually could take hours. Google's search mechanism is a prominent instance of a neural network [51].

The terms ”Deep Learning” and ”neural networks” are often used synonymously, leading to potential misunderstandings. It's important to clarify that the ”deep” in deep learning denotes the number of layers in a neural network. Specifically, a neural network with more than three layers, counting both input and output layers, qualifies as a deep learning algorithm as shown in the figure 5. In contrast, a neural network with only two or three layers as in figure 4 is simply termed a basic neural network [50].

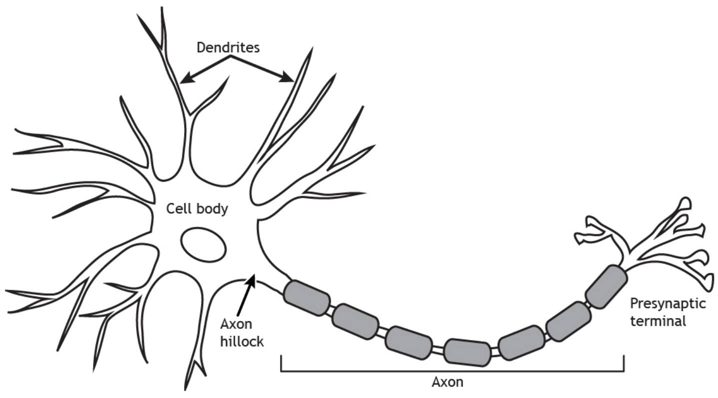


Figure 2: Neuron [15]

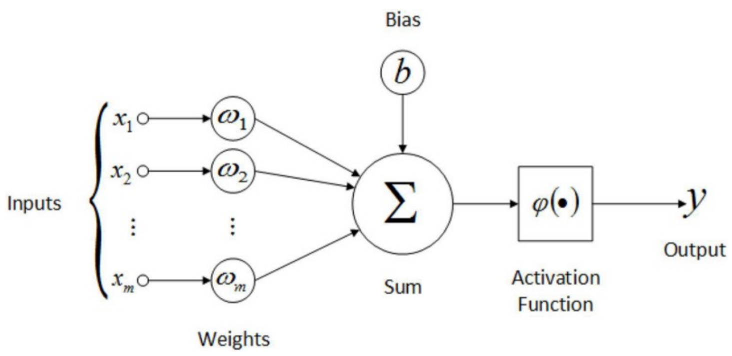


Figure 3: Artificial Neuron [5]

2.2.1 The artificial neuron

In the artificial neuron, as shown in figure 3 the χ_m are numerical values that represent either the input data or the output values of other neurons. The ω_m, the weights, are numerical values that represent either the power value of the inputs or the power value of the connections between the neurons [28].

There are operations that take place at the level of the artificial neuron. The artificial neuron will make a product between the weight (ω) and the input value (χ), then add a bias (b), the result is transmitted to an activation function (ϕ) which will add a certain non-linearity [5].

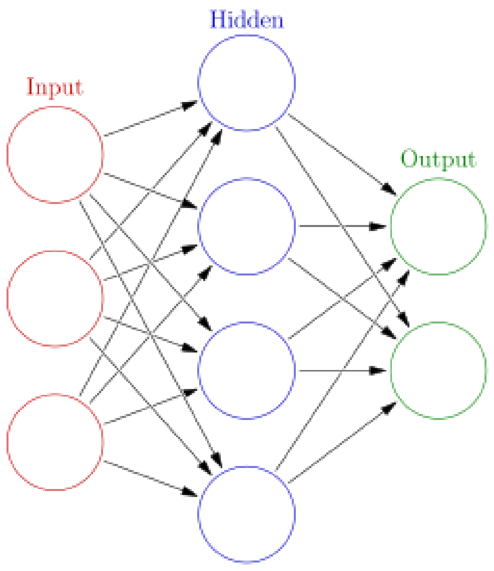


Figure 4: Artificial Neural Network [4]

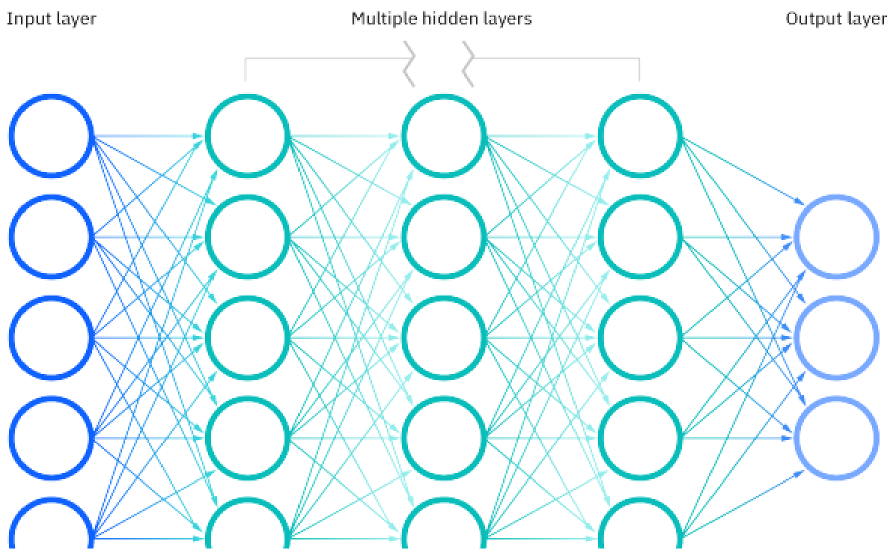


Figure 5: Deep learning [23]

2.2.2 The activation functions

After the neuron has performed the product between its inputs and its weights, it also applies a non-linearity to this result. This non-linear function is called the activation function. The activation function is an essential component of the neural network allowing it to learn and represent more complex patterns and relationships in the input data. What this function decides is whether the neuron is activated or not. After the transformation, this output is sent to the next layer. Non-linearity is important in neural networks without the activation function, a neural network simply becomes a linear model [9].

There are many activation functions utilized in neural networks, but the most common ones include the Rectified Linear Unit (ReLU), Sigmoid, Hyperbolic Tangent (Tanh), and Softmax. Each of these functions has its own advantages and is suited for specific types of tasks and data distributions:

- **ReLU**: This activation function is commonly employed, particularly in Convolutional Neural Networks. It's computationally efficient and avoids issues like the Vanishing Gradient Problem. However, it's not zero-centered, leading to the "dying ReLU" issue where certain nodes become inactive for all negative inputs, rendering them unresponsive during learning [30].
- **Sigmoid**: This activation function remains largely for historical reference and isn't typically used in contemporary models. It's computationally intensive, leads to the vanishing gradient issue, and isn't zero-centered. However, it's occasionally employed for binary classification challenges [43].
- **Softmax**: The softmax can be viewed as an extended version of the sigmoid function. It's applied in multi-class classification scenarios. Like the sigmoid, it yields values between 0 and 1, making it suitable for the concluding layer in classification models [47].
- **Tanh**: The tanh is similar to the sigmoid function, but it is centered around zero. The advantage of being zero-centered is that it allows the gradients to be more balanced and not biased towards positive values. This means that during the optimization process, the gradients can adjust in any direction, facilitating more efficient learning and convergence [9].

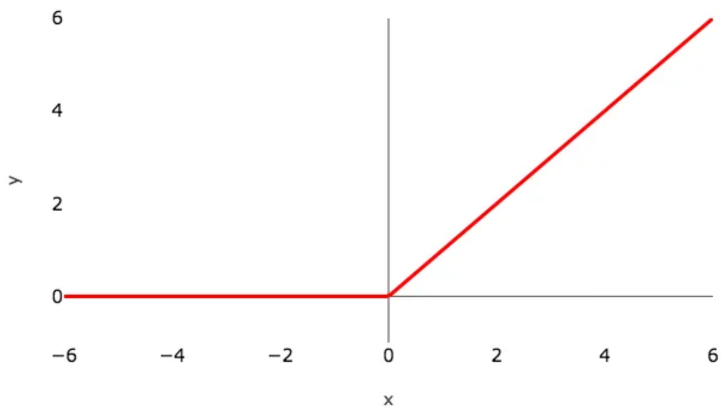


Figure 6: ReLU [9]

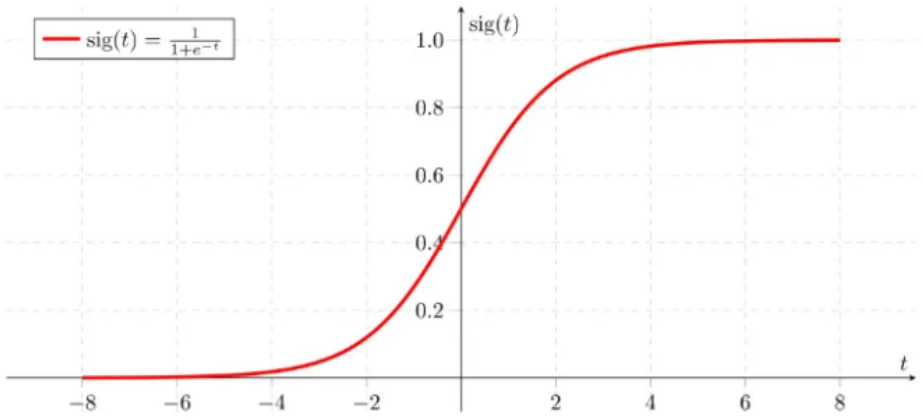


Figure 7: Sigmoid [9]

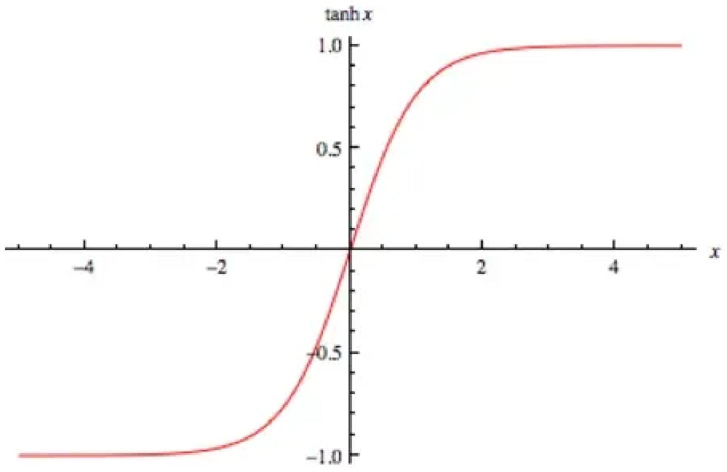


Figure 8: Tanh [9]

2.3 Shallow learning vs Deep learning

Traditional (shallow) neural networks refer to neural networks with a limited number (typically 0-2) of hidden layers, which distinguishes them from deep neural networks. Examples of such networks include support vector machines, decision trees etc. [35].

On the other hand, most deep neural architectures are constructed by integrating and reconfiguring a select set of architectural building blocks, or layers [59]. As suggested in [35], a neural network is considered ’deep’ when it has more than one hidden layer.

2.4 Deep learning Training

2.4.1 Introduction - gradient descent

The deep learning learning process is centered around training the neural network using iterative optimizers. These optimizers mainly aim to reduce the cost function represented as f(x) (often referred to as the objective function, loss function, or the error function) to a very low value [32].
Although a variety of algorithms can support this training process [39], the iterative optimization method using gradient descent is predominantly preferred for most neural network models [34].
Throughout the training phase, the algorithm tries to find the global minimum without getting trapped by local minima. A diagram provided by Goodfellow et al. in 9 illustrates this concept.

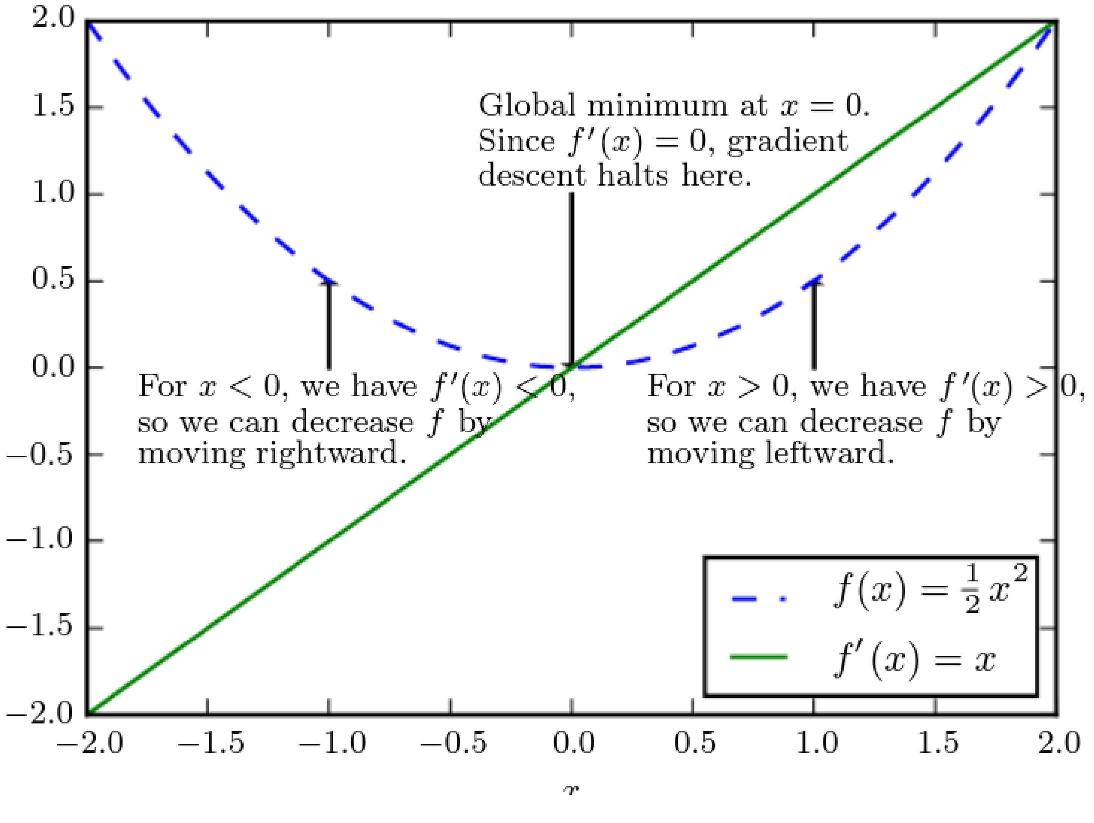


Figure 9: An illustration of the search for the optimum process [32]

Optimization algorithms can struggle to locate a global minimum when faced with numerous local minima or flat regions. In deep learning, we typically settle for these solutions as long as they achieve notably low values of the cost function. [27] This approximation is visualized at figure 10 .

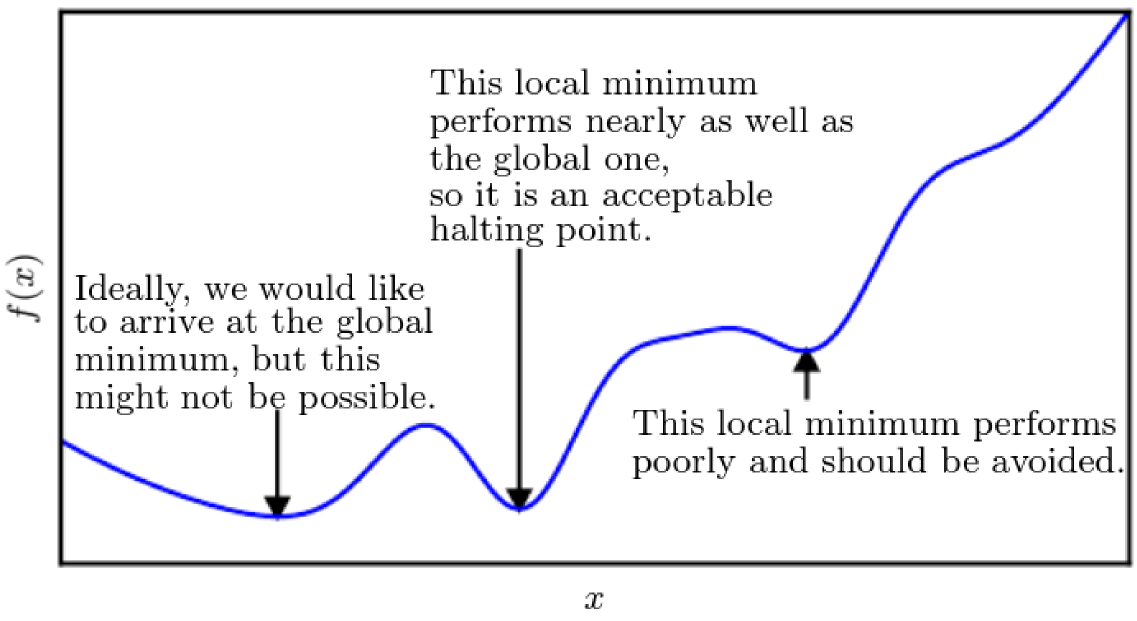


Figure 10: Approximate minimization [32]

2.4.2 Variants of gradient descent

There are three main types of gradient descent algorithm variants. The primary distinction between them is the amount of data we use when calculating the gradient at each learning step [16].

- **Batch gradient descent:** Also known as vanilla gradient descent, this optimization method uses the entire training set. It's termed as deterministic gradient methods or batch gradient descent because all training examples are processed in a single large batch. The term "batch" in this context refers to a collection of examples [55].
- **Stochastic gradient descent:** This optimization method processes one example at a time, hence it's sometimes referred to as stochastic methods or online gradient descent. It performs updates one example at a time, making it faster and suitable for online learning [25].
- **Mini-batch gradient descent:** This method combines the concepts of batch and stochastic gradient descent. It updates the model for every mini-batch of n training examples. This approach reduces the variance of parameter updates, leading to more stable convergence, and benefits from optimized matrix operations in modern deep learning libraries, enhancing the efficiency of computing the gradient with respect to a mini-batch. Common mini-batch sizes typically range from 50 to 256, though they can vary based on the specific application [55].

2.4.3 Gradient descent optimizers

While there are several optimizers for gradient descent, such as Adagrad and RMSProp, the most well-known is the Adaptive Moment Estimation (Adam). Utilizing an optimizer like Adam in gradient descent provides numerous benefits that enhance both the training process and the performance of a neural network. [16].

- **Adaptive Learning Rates:** Traditional gradient descent uses a fixed learning rate for all parameters. Adam, on the other hand, computes individual adaptive learning rates for each parameter. This means that each parameter is updated based on its own historical gradient information, allowing for more tailored updates.
- **Momentum:** Adam incorporates the concept of momentum by using moving averages of the parameters. This helps in smoothing out the updates and can lead to faster convergence. Momentum essentially helps the optimizer to navigate through the loss landscape by "remembering" its previous direction, which can be especially useful in areas of the loss surface that are flat or have noisy gradients [34].
- **Avoiding Vanishing and Exploding Gradients:**The adaptive learning rates in Adam can help mitigate issues related to vanishing and exploding gradients, which can be problematic in deep networks.
- **Bias Correction:**Adam includes a mechanism to correct biases in its estimates of the first and second moments of the gradients. This ensures that the estimates are more accurate, especially during the initial stages of training.
- **Efficiency and Stability:**Adam often requires less tuning of the learning rate and converges faster than standard stochastic gradient descent. Its adaptive nature makes it more robust to different types of data, architectures, and hyperparameters.
- **Suits Complex Models:** For deep learning models with a large number of parameters and complex architectures, advanced optimizers like Adam can provide significant benefits in terms of training speed and model performance [55].

2.5 Feedforward Neural Networks

A feedforward neural network (FNN) is a primary category among artificial neural networks. It's distinguished by how information travels between its layers: the data moves solely in a forward direction, starting from the input nodes, passing through any hidden nodes, and ending at the output nodes, without looping back. This linear flow contrasts with recurrent neural networks, where information can circulate in two directions. Contemporary feedforward networks utilize the backpropagation technique for training and are often informally called "vanilla" neural networks [58]. Most common types of FNN are:

- Convolutional Neural Networks (CNN): A Convolutional Neural Network (CNN) is designed to automatically and adaptively learn spatial hierarchies of features from input images. It assumes that spatially closer entries in the input are more semantically related than distant ones [47].

  A CNN is composed of a sequence of layers, with each layer transforming one set of activations to another using a differentiable function. The primary layers used to construct a CNN are:
  - **The convolutional layer:** A filter moves across the input, searching for similarities between the input and the filter's pattern. This produces a new matrix, called a feature map, which highlights regions where the desired feature was recognized.
  - **The pooling layer:** down-samples the spatial dimensions while retaining the most important information.
  - **The fully connected layer:** The Fully Connected (FC) layer is composed of neurons, weights, and biases, and it serves to link neurons across distinct layers. Typically positioned just before the output layer, these layers make up the concluding sections of a CNN architecture.

  According to Li et al. (2022) these networks are especially effective for tasks related to image recognition due to their ability to capture spatial hierarchies [44].

- Multi-layer perceptron: The simplest (non-binary) FFNN is the perceptron, in which the inputs are directly linked to the outputs; it performs a linear combination of the inputs followed by a thresholding operation. As a result, perceptrons can only depict straight-line functions and are suitable only for linearly separable classification or regression problems, exemplified by their inability to solve the XOR problem.
  On the other hand, an FFNN with additional layers between the input and output, often termed a multi-layer perceptron (MLP), can represent not just linear functions but also more intricate ones by incorporating non-linear functions like the sigmoid. The intermediate layers in an MLP are commonly referred to as hidden layers [20].

2.6 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed to recognize patterns in sequences of data, such as time series or natural language. Unlike traditional neural networks, which process input data in a feed-forward manner, RNNs maintain a memory of previous inputs, allowing them to exhibit dynamic temporal behavior [17].

The architecture of an RNN involves loops that allow information to be passed from one step in the network to the next, enabling the network to maintain a 'memory' of previous inputs in its internal state. This memory enables RNNs to process sequences of varying lengths, making them especially suited for tasks like language modeling and time series forecasting [48].

Most common RNN variants:

- Fully recurrent or Vanilla RNN: This is the most basic form of an RNN that maintains just one hidden state ($a^{<t-1>}$) and uses it in order to compute the hidden state ($a^{<t-1>}$)for the next input ($x^{<t>}$) [18].

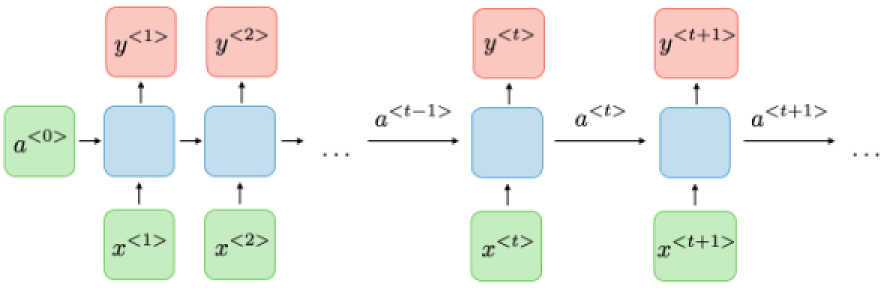


Figure 11: RNN [18]

For each time step t, the activation ($a^{<t>}$) and the output ($y^{<t>}$)are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad (1)$$

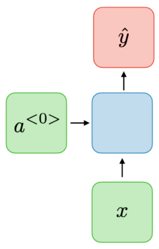
$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \quad (2)$$

Where W_{ax}, W_{aa}, W_{ya}, b_a, b_y are coefficients that are shared temporally and g_1, g_2 are the activation functions [18].

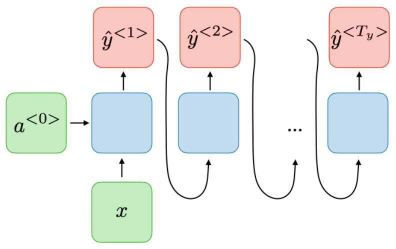
According to Kaur and Mohta (2019) there are four main types of RNNs, as depicted in the figure 12, with each one being suitable for a specific application:

- **One to one:** This kind of RNN functions similarly to a basic neural network and is often referred to as a Vanilla Neural Network. In this network structure, there's a single input and a single output.

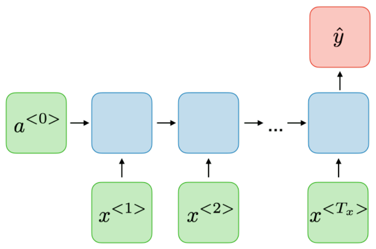
- **One to many:**In this RNN variant, a single input corresponds to multiple outputs. A prime example of this network structure is image captioning, where an image is provided and a sentence with several words is predicted in response.
- **Many to one:** In this network configuration, multiple inputs are introduced at various stages, culminating in a single output. Such a network structure is commonly employed in tasks like sentiment analysis, where numerous words are inputted to ultimately predict the overall sentiment of the sentence.
- **Many to many:**In this neural network configuration, both the inputs and outputs are multiple, tailored to specific problems. A prime example of this setup is language translation, where words from one language are inputted and translated into multiple words in another language as the output [41].



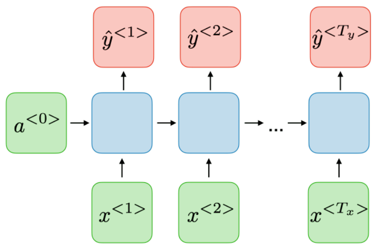
(a) One-to-one [18]
$T_x = T_y = 1$



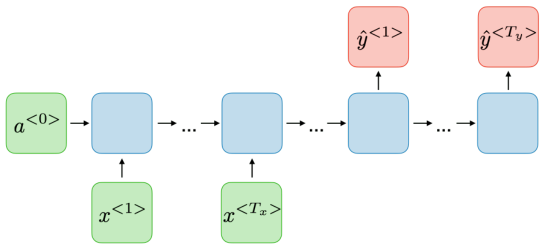
(b) One-to-many [18]
$T_x = 1, T_y > 1$



(c) Many-to-one [18]
$T_x > 1, T_y = 1$



(d) Many-to-many [18]
$T_x = T_y$



(e) Many-to-many [18]
$T_x \neq T_y$

Figure 12: Different types of RNN

As stated in [56], RNNs can approximate any continuous function to a desired level of accuracy with an adequate number of neurons. However, they can encounter challenges such as vanishing and exploding gradients during training, which may impede their capacity to learn long-term dependencies.

- **Vanishing Gradients:** During backpropagation, we compute gradients of the loss with respect to the model's parameters. The gradients can become very small due to the repeated multiplication of values less than one (especially with certain activation functions like sigmoid or tanh). This means that the weights of the earlier layers in the network will receive very tiny updates, leading to slow or stalled learning.We end up hitting the maximum number of steps we are allowed to take before we find the optimal value.

  - **Exploding Gradients:** if the gradients become too large due to repeated multiplication of large values, they can cause numerical instability. This is because large weight updates can lead the model to overshoot the optimal values during training.

- **Long short-term memory:** Long Short-Term Memory networks, commonly referred to simply as LSTM, are a special type of RNN, introduced by Hochreiter and Schmidhuber in 1997 [38].
  Although the recurrent neural networks presented in the previous section are theoretically capable of learning arbitrary sequence-update rules, In practical scenarios, these models struggle to learn the impact of distant past events. Recognizing such distant relationships is essential for effective language modeling, as the interpretation of a complex sentence can hinge on the connection between words spaced far apart [59].

  The phenomenon described is known as the vanishing gradient problem [37].To address this issue, the Long Short-Term Memory (LSTM) cell was introduced [59].

  The LSTM is a modification of the standard RNN layer, specifically designed to propagate signals from distant past events to the present state. This architecture incorporates a memory cell and is characterized by three distinct gates: input, forget, and output.

  - **The memory cell** acts as a reservoir, retaining information over extended durations. In visual representations of LSTMs, this cell is typically depicted as a horizontal line, and its content is refreshed during every time step.
  - **The input gate** determines the volume of new information flowing into the memory cell. Utilizing a sigmoid activation function, it evaluates the current input and the preceding hidden state to ascertain the data to be stored in the memory cell.
  - **The forget gate** decides which portions of the information in the memory cell should be discarded. Also employing a sigmoid function, it assesses the present input and the last hidden state to determine the data to be omitted.
  - **The output gate** regulates the data read from the memory cell, influencing the hidden state. Using a sigmoid function, it evaluates the current input and the previous hidden state to decide the information that will refresh the hidden state.

  During each iteration, the LSTM calculates the values for the three gates, which in turn influence the memory cell and hidden state updates. This updated hidden state is then forwarded to the subsequent LSTM unit in the sequence [21].

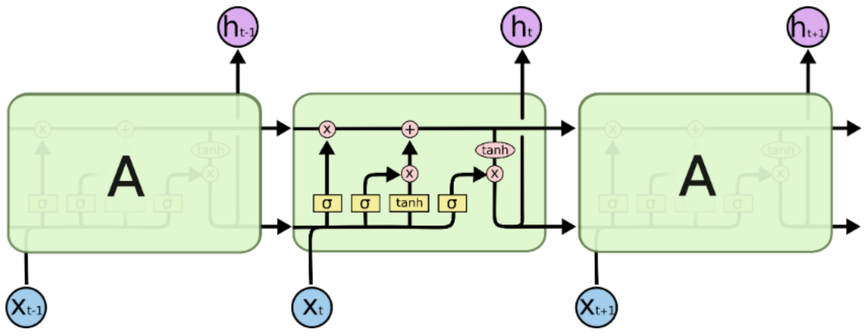


Figure 13: LSTM [21]

2.7 Encoder- decoder Architecture

The Encoder-Decoder structure is a key part of deep learning. It's built on the Sequence-to-Sequence method. Tools like the Recurrent Neural Network (RNN) and its special type called LSTM are really helpful for tasks where we transform one sequence into another. This Encoder-Decoder setup is a top choice for translating languages using computers. It often does as well as, or even better than, older translation methods. The goal of these models is to get the best possible translation based on the input they're given.

There are three main blocks in the encoder-decoder model:

- **Encoder:**The Encoder will convert the input sequence into a single-dimensional vector (hidden vector).
- **Hidden vector:** This vector is designed to capture the information from all input elements, serving as the starting hidden state for the decoder, to aid in making precise predictions.
- **Decoder:**The decoder will convert the hidden vector into the output sequence [8].

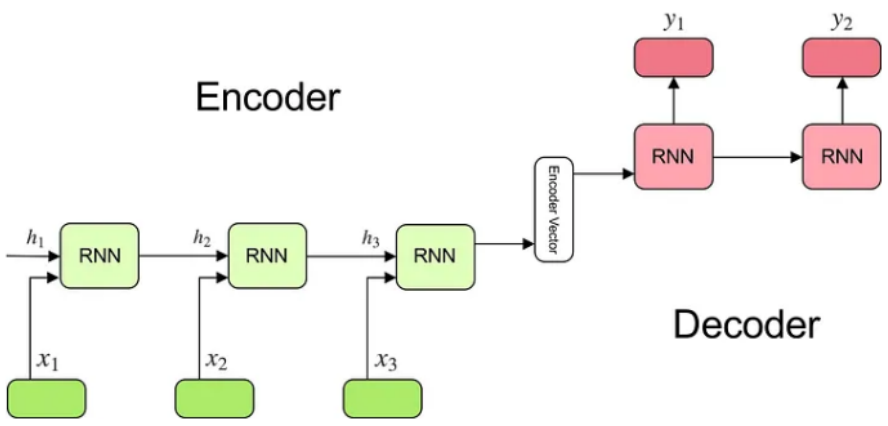
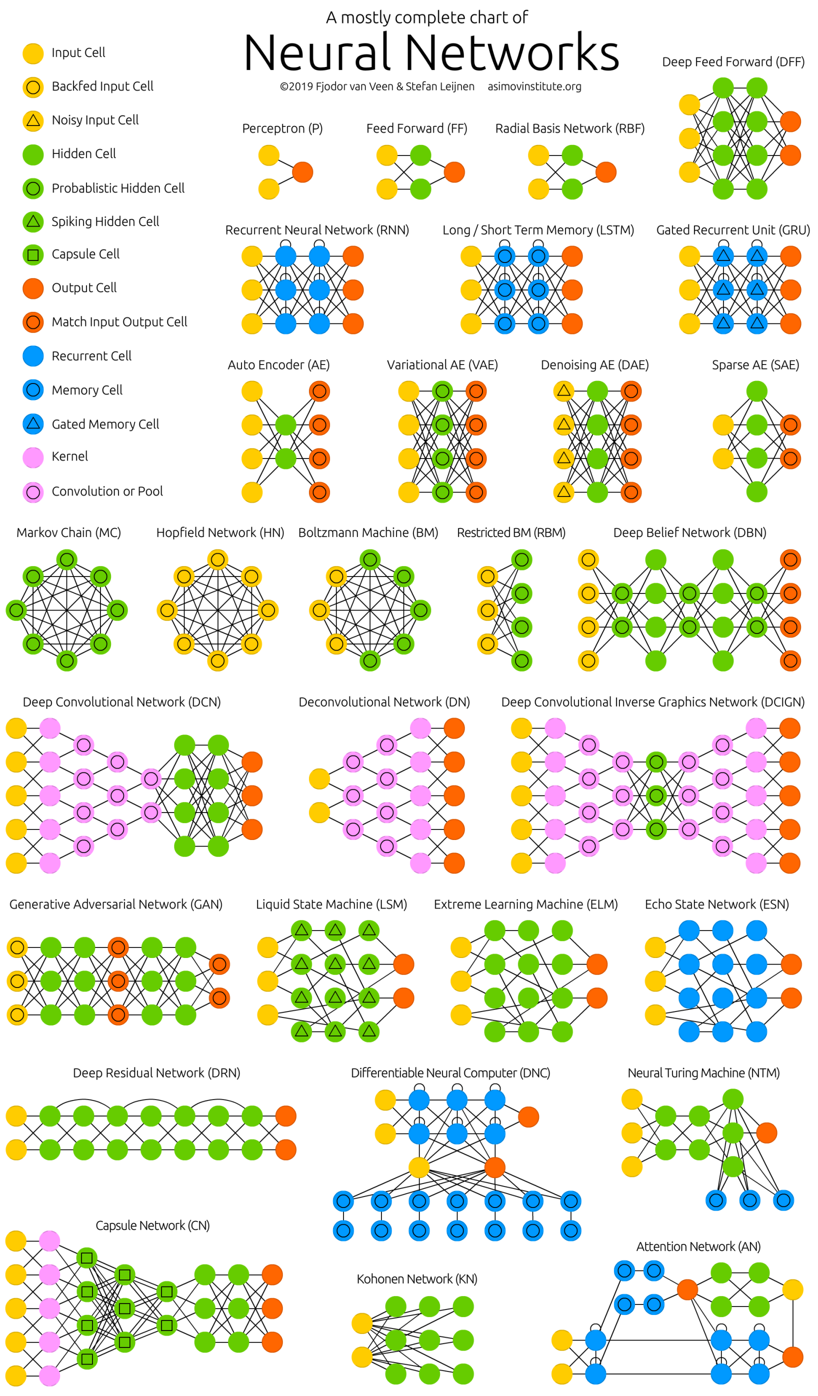


Figure 14: Encoder-decoder sequence to sequence model [8]

2.8 Neural Network Architectures

Neural networks have changed the world of machine learning and artificial intelligence introducing a myriad of architectures that continue to emerge at a rapid pace A comprehensive chart showcasing a wide range of these architectures can be found in figure 15.



A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)

Figure 15: Neural Networks zoo [14]

2.9 Limitations of RNNs and Encoder-Decoder Models

While RNNs, LSTMs, and encoder-decoder models have demonstrated impressive performance across various tasks, they face challenges related to long-term dependencies, especially in longer sequences. Their design, which is based on sequential processing, isn't optimized for computational efficiency. These models can't operate in parallel and lack a mechanism allowing one data piece (or "token") to directly interact with or reference another [10].

2.9.1 Attention technique

The attention mechanism in neural networks is akin to how humans focus on specific details when performing a task. Instead of treating all information equally, attention allows the network to prioritize parts based on their context.

For RNNs, rather than compressing an entire sentence into a single hidden state, each word has its associated hidden state, which is then used in the decoding phase. These hidden states are utilized at every RNN step during decoding, allowing the network to focus as needed. However, even with attention, RNNs still face challenges. A significant limitation is their inability to process inputs (like words) concurrently, leading to extended translation times, especially for large text datasets [26].

2.9.2 Convolutional Neural Networks as a Solution

To address the limitations of RNNs, even those with attention mechanisms, Convolutional Neural Networks (CNNs) are introduced. CNNs provide the following advantages:

- Easily parallelize operations on a per-layer basis.
- Take advantage of local dependencies.
- Achieve a logarithmic relationship between positions.

However, while CNNs excel at parallel data processing and detecting local patterns, they might not always capture the intricate relationships between words in sentences. In translation tasks, the meaning of a sentence often hinges on the interactions and dependencies among its words. The position of a word can change the entire meaning of a sentence, and an earlier word can influence the translation of a subsequent one [10].

2.10 Transformers

In late 2017, a groundbreaking architecture emerged in the NLP realm, bypassing the use of RNNs or CNNs and addressing the challenges associated with them. The transformer model, introduced by Vaswani et al. (2017) [57], brought about notable enhancements in model performance. Moreover, these models reduced training time due to their superior parallelization capabilities compared to RNNs or CNNs. This advancement was largely attributed to the a new variant of the attention mechanism, called self attention [10].

The model is designed based on the encoder-decoder architecture, with 6 layers each for the encoder and decoder [57].
In the encoder, each layer is composed of a self-attention mechanism and a feed-forward segment.
Every encoder layer also includes a residual connection, which is then followed by normalization [57], see figure 16.

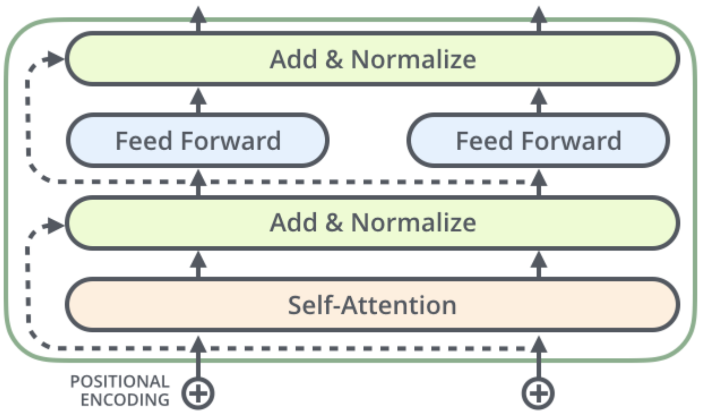


Figure 16: Encoder layer [11]

The input sequence of the encoder is continuous representations $(X_1, ..., X_n)$ (embeddings) [1] which are mapped to $Z = (Z_1, ..., Z_n)$ (with the contextual information of the input sequence [2]). At the end decoder takes the Z as input and provides the output $(Y_1, ..., Y_n)$.

The decoder has the encoder's structure, consisting of 6 layers, each with a self-attention and feed-forward segment. Additionally, positioned between these two components in the decoder is an "encoder-decoder attention" layer. Once the encoder processes the input sequence, the resulting output is converted into attention vectors, K and V. These vectors are passed to the "encoder-decoder attention" layers. The decoder sequentially produces tokens based on the preceding output token.

[1]Embeddings or features of the input tokens/words.
[2]Relations to other words or tokens in its surrounding.

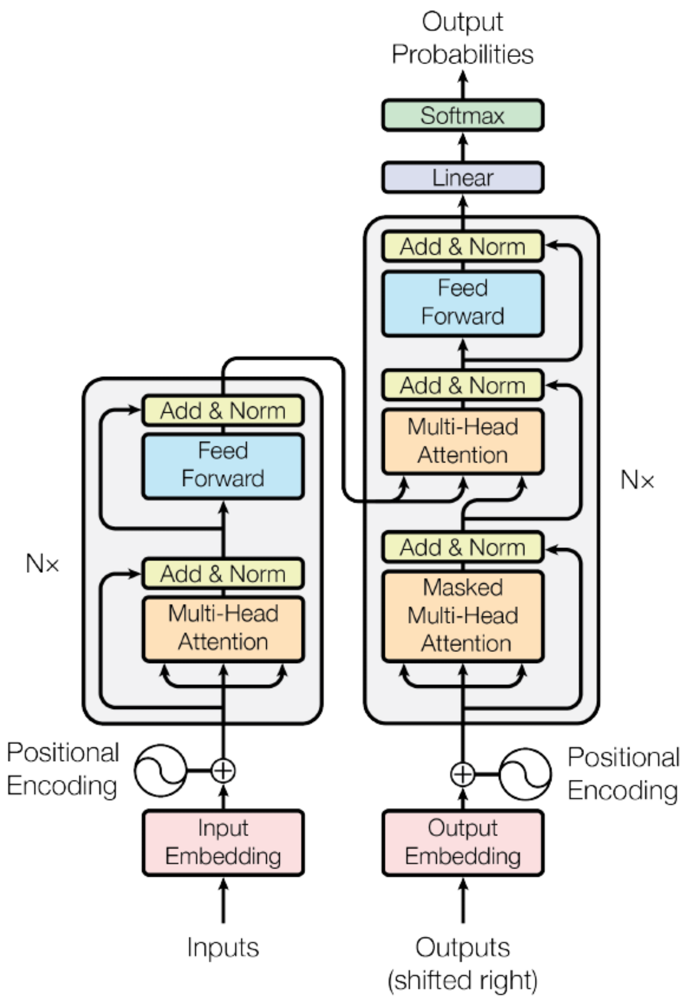


Figure 17: The Transformer - model architecture. [57]

2.10.1 Self-Attention

The self-attention mechanism (also known as Scaled Dot-Product) ensures that the representation of a token is influenced not just by itself, but also by the context from other tokens in the sequence. For every token, the attention process uses the encoder's input to generate three distinct vectors with dimension d. These vectors are derived by multiplying the input token's embedding with three separate matrices, the weights of which are refined during training. For the entire sequence, these vectors are stored in matrices Q, K, and V. The results are then computed as described [10].

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}) \cdot V \tag{3}$$

Softmax results in a probability distribution that dictates the amount of attention each token should receive [57].

2.10.2 Multi-Head Attention

The Transformer is more effective at identifying dependencies in the input sequence when using multiple attention mechanisms rather than just a single one.

In their foundational research, they calculated 8 attention matrices using distinct Q, K, and V values. These matrices were then combined and multiplied by a weight matrix W [57]:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O \tag{4}$$

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{5}$$

where weight matrices:

$W_i^Q, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$
$W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$
$W^O \in \mathbb{R}^{8d_v \times d_{\text{model}}}$
$d_{\text{model}}/8 = 64$

The Transformer network's self-attention layer is equipped with 8 attention heads.

2.10.3 Applications of attention

- In "encoder-decoder attention" layers, queries originate from the prior decoder layer, while memory keys and values are sourced from the encoder's output. This setup enables each decoder position to focus on every input sequence position, reflecting common attention mechanisms in sequence-to-sequence models.
- The encoder comprises self-attention layers. In these layers, keys, values, and queries all originate from the output of the encoder's preceding layer. This allows every position in the encoder to focus on all positions from its previous layer.
- Self-attention layers in the decoder let each position attend to all prior and current positions in the decoder. To maintain the auto-regressive nature, leftward information flow is prevented. This is achieved in the scaled dot-product attention by masking illegal connections, setting them to negative infinity before applying the softmax [2].

2.10.4 BERT

In this segment, we introduce BERT, a model proposed by Devlin et al. in 2018. Unlike many contemporary language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by considering both the left and right context in all layers.

According to Devlin et al. (2018) BERT is pre-trained using two unsupervised tasks: Masked Language Modeling and Next Sentence Prediction:

- **Masked Language Modeling:** The main goal of this task is to predict the hidden tokens in an input sequence. For instance, given the sequence ”In Autumn, the .... fall from the trees,” the most likely object to fall from a tree in Autumn is leaves. This prediction task is executed by substituting the input token with a [MASK] token, as in ”In Autumn, the [MASK] fall from the trees.” BERT then attempts to determine the masked word. BERT's ability to make such predictions stems from its extensive exposure to various texts, allowing it to learn linguistic patterns exceptionally well during its pretraining [29]. Although BERT may not inherently understand the nuances of ”Autumn,” ”trees,” or ”leaves,” it can identify the linguistic patterns and the context in which these words appear, leading it to conclude that ”leaves” is the most likely answer [13].
- **Next Sentence Prediction:** The model receives two sentences and must determine if they are related. This process aids the model in recognizing long-term relationships and understanding the essence of entire sequences.

The creators introduced two versions: a base and a large variant. We utilize the base version, which consists of 12 encoder layers. Each of these layers employs 12 attention heads to compute the Multi-Head Attention, producing embeddings with a dimension of dmodel = 768, amounting to roughly 110 million parameters.

2.10.5 RoBERTa

The RoBERTa model, (Robustly Optimized BERT Pretraining Approach), was developed and introduced by Y. Liu and colleagues in 2019 [46] as a refined version of the original BERT model. The authors maintained the basic architecture of BERT, known for its stacked encoder mechanism and introduced three concepts specifically designed to enhance the pre-training phase of the model.

Their first major alteration involved the elimination of the Next Sentence Prediction (NSP), the initial creators of BERT had posited that including NSP in pretraining was beneficial to the model's overall efficacy but after thorough testing, it is conclude that NSP could actually be detrimental, leading to suboptimal performance in various applied tasks. As a result, the RoBERTa model abandoned NSP, opting instead to focus solely on the task of Masked Language Modelling (MLM) during its pre-training.

The second significant enhancement proposed by the RoBERTa team was the implementation of a dynamic masking strategy for the MLM task in contrast with static masking technique employed in BERT. Dynamic masking, introduced variability by randomly masking words in real-time as new batches of data were processed. This change was demonstrated to improve the model's performance, particularly in relation to two specific downstream tasks.

Lastly, regarding the Masked-Language Modeling input, the authors chose the FULL-SENTENCES approach for its straightforward implementation. This method involves filling a batch with as many complete sentences as will fit from a single document. If a sentence doesn't fit within the batch, they add padding to reach the maximum length. When the end of a document is reached before filling the batch, a separator token is inserted, and the process continues with sentences from the next document. This approach balances efficiency with effectiveness, providing a practical solution for pre-training the RoBERTa model [46].

2.10.6 GPT-1

Prior to GPTs, the best-performing NLP models were tailored for specific tasks, such as determining sentiment or text implications, using supervised learning. The downside of these models was that they required vast amounts of specialized, pre-labeled data, which is hard to come by, and they struggled to adapt to tasks they weren't specifically trained for [12].

The method proposed begins by pretraining a language model using a broad spectrum of general, unlabeled data. This initial stage employs the transformer model's architecture, which is foundational to the GPT-1 framework. Through this unsupervised learning phase, the model assimilates language patterns and structures autonomously, without any task-specific objectives.

Following the pretraining, the model transitions into the fine-tuning phase. This supervised learning process utilizes labeled datasets, enabling the model to develop proficiency in specialized tasks such as categorization or sentiment analysis. The sequential methodology of unsupervised pretraining succeeded by supervised fine-tuning significantly bolsters the capabilities of GPT-1, equipping it to adeptly handle diverse language-based tasks.

GPT models, including GPT-1, are characterized by their decoder-only architecture, which stands in contrast to other models that may use both encoders and decoders. In GPT's architecture, the encoder is absent, and the input is processed directly by the decoder stack. Nonetheless, the system incorporates input embedding and positional encoding to prepare the input text. This preprocessed information is relayed to the bottom layer of the decoder stack. To generate the final text output, a linear layer followed by a softmax layer processes the output from the topmost level of decoders.

The architecture of the decoder is a variant of the encoder, distinguished by its implementation of masked self-attention. This mechanism ensures that during prediction, any given token can only be influenced by preceding tokens, not by those that follow. This is achieved by setting the attention scores to negative infinity for future tokens, thereby preventing the model from accessing them during the training phase [52].

2.10.7 GPT-2

GPT-2 is the second iteration in the GPT series and an enhanced version of GPT-1. It's crafted using a larger set of training data and a greater number of adjustable parameters, enabling it to deliver more sophisticated language outputs.

GPT-2 is built with the ambition of excelling at numerous language tasks without being specifically trained for each one. It achieves this through an approach called 'task conditioning.' Essentially, this means the model is designed to produce different results based on the task it's asked to perform, even if it starts with the same information. This capability demonstrates GPT-2's flexibility and its advanced understanding of language tasks.

To compile a comprehensive and high-quality dataset, the researchers harvested content from Reddit, specifically targeting outbound links from highly upvoted posts. They created a dataset known as WebText, which contained 40GB of text from over 8 million documents, a sizeable increase from the Book Corpus dataset utilized for the original GPT-1 model. Wikipedia articles were intentionally excluded from WebText to avoid overlap with many test sets, which commonly include Wikipedia content.

GPT-2 was developed with 1.5 billion parameters, a significant expansion from the 117 million parameters in GPT-1.

2.10.8 Transformers comparison

The vanilla transformer model, foundational to modern NLP architectures, consists of both encoders and decoders. The encoders process the input text, while the decoders generate the output, making it suitable for tasks like translation [57].

The table 1 presents a comparison of the previously mentioned models that are based on or adapt the original transformer structure, detailing their specific configurations, parameter counts, and the sizes of the datasets used for training.

BERT for instance, utilizes only the encoder part of the transformer. It's designed to understand context in both directions (bidirectional) and is suitable for tasks that require understanding the meaning of text segments [7].

RoBERTa follows BERT's lead, also using only encoders and refining the pre-training process.

Conversely, GPT models, including GPT, GPT-2, and GPT-3, diverge by using only the decoder part of the transformer [12].

The bidirectional nature of BERT implies that it considers both preceding and succeeding text to understand context.

For example, given a sentence with a blank:

Model	Architecture	Parameter count	Training data
Bert	Transformer encoder	110 milion(Base)	16GB
RoBERTa	Transformer encoder	355 milion	160GB
GPT-2	Transformer decoder	1.5 bilion	40GB

Table 1: Comparison of Model Architectures and Parameters

"I really enjoyed this ____! The plot was great and the acting was fantastic"

BERT (and RoBERTa) examines the following to infer the missing word:

- "I really enjoyed this" (**left context**)
- "The plot was great and the acting was fantastic" (**right context**)

GPT-2, in contrast, is an autoregressive model predicting the next word solely from previous text, thus it would only use "I really enjoyed this" (**left context**) to guess the blank [6].

3 Methodology - Applications

3.1 Datasets

Two datasets were used to gather information. The first dataset focused on English tweets about the #MeToo movement and was obtained from the website https://data.world [3].

This dataset came in a SQLite format with four main tables of information: associations, hashtags, tweets, and user details as shown in the Figure 18.

[3] https://data.world/from81/390k-metoo-tweets-cleaned

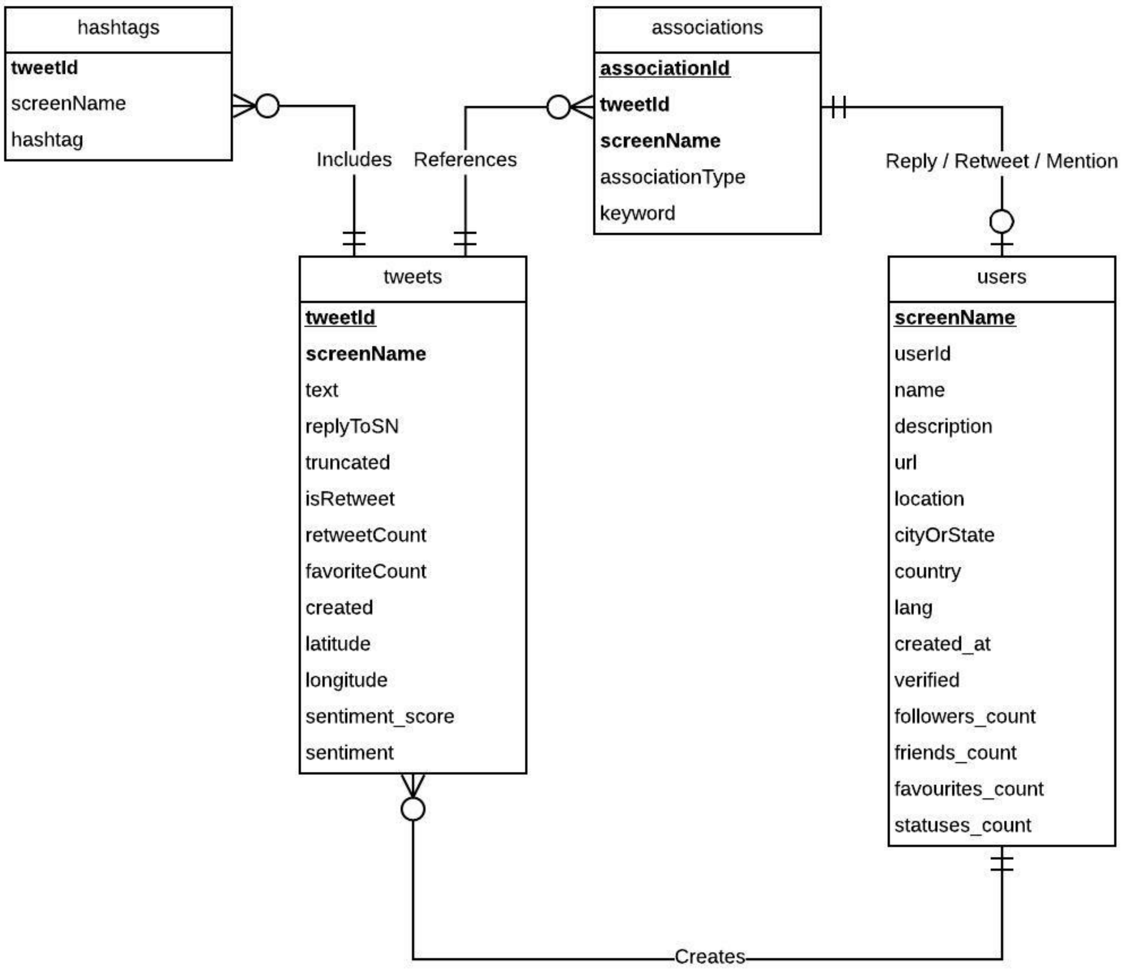


Figure 18: #MeToo Database schema[4]

From this, all the retweets were excluded from the dataset to avoid duplication and ensure that only original tweets were analyzed.

After this, by linking the information about tweets with the hashtags information, the dataset was narrowed down to only include tweets with the hashtag #MeToo using the following SQL query.

```
select text, sentiment  from tweets t
join hashtags h on t.tweetId = h.tweetId
where h.hashtag = '#MeToo' and t.isRetweet = 0
```

This process resulted in two key pieces of data for each tweet: the tweet's text and its associated sentiment, both of which were exported to a CSV format using the SQLite command line.

[4]Retrieved from https://data.world/from81/390k-metoo-tweets-cleaned/file/schema.jpeg

The #MeToo dataset contains text data from social media, which has been cleaned for analysis. It includes two main features: the text of the posts and their sentiment labels. In this dataset, sentiments are categorized as ’negative’, ’neutral’, or ’positive’, offering a broader range of emotional expressions compared to binary categorization. For ease of analysis, these sentiments are converted into numbers: 0 for ’negative’, 1 for ’neutral’, and 2 for ’positive’. Additionally, a new column called ’content_length’ has been added. This column measures how long each post is by counting its characters, providing insight into the length of social media communications.

As demonstrated in the sentiment distribution chart 19, this dataset encompasses three distinct sentiment classes: positive, negative, and neutral. The positive class consists of 34,326 observations, the negative class comprises 37,171, and the neutral class includes 40,694 observations.

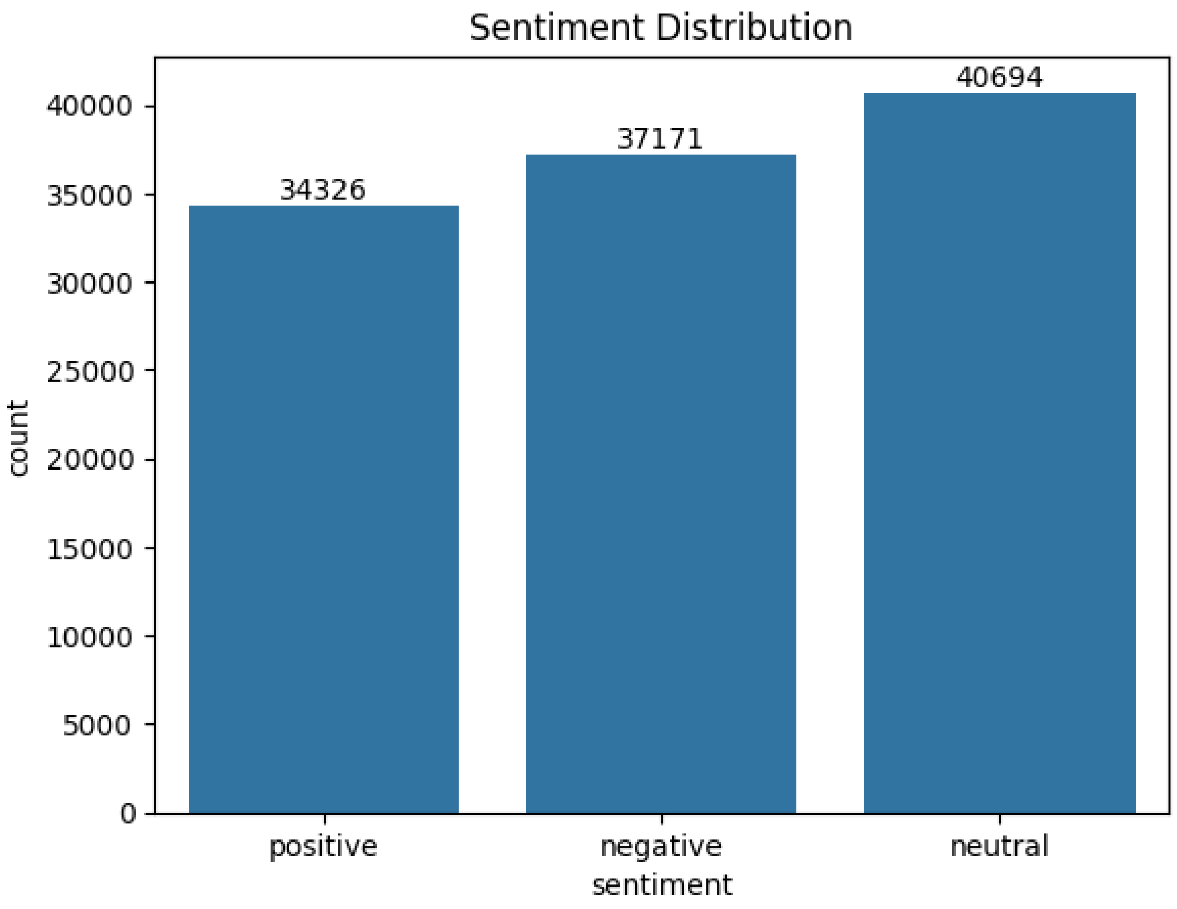


Figure 19: Length distribution - #MeToo

The analysis, as visualized in image 20 , indicates a wide range of content lengths. The most common tweet length is 110 characters, suggesting a tendency for concise communication within the platform’s character limit. The maximum tweet length observed is 255 characters, and the minimum is as short as 6 characters. The average tweet length is approximately 108 characters, with a median of 89 characters.

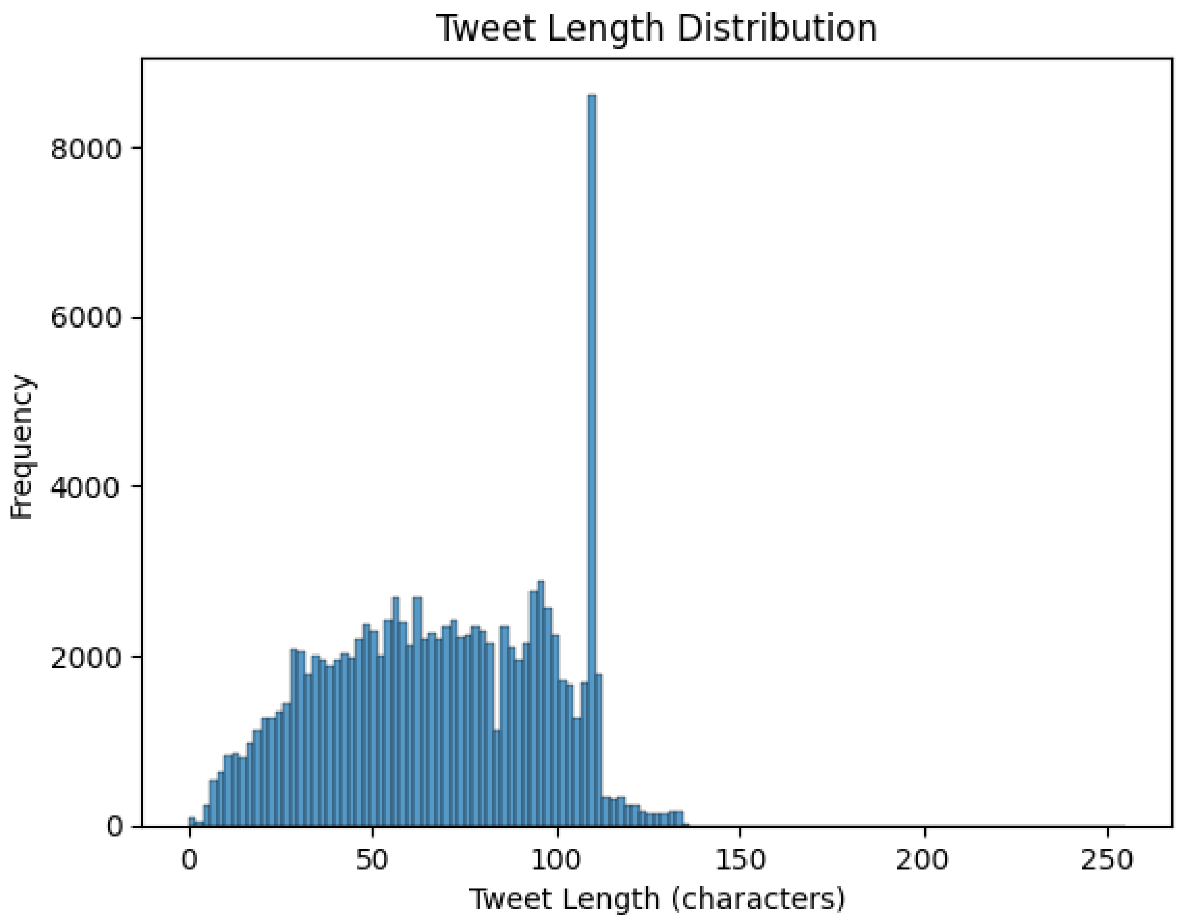


Figure 20: Length distribution - #MeToo

The secondary dataset, sourced from Kaggle, includes Greek store reviews from the website Skroutz for sentiment analysis[5]. This dataset comprises 6,552 reviews, evenly divided into two categories: 3,276 positive and 3,276 negative reviews. The dataset is structured into three main components: an ID number for each review, the review text, and a sentiment label indicating whether the review is positive or negative. This data is provided in an Excel (.xlsx) file format.

To parse the data, the `openpyxl` library, in conjunction with `pandas`, was utilized. `Openpyxl` is necessary for reading from and writing to Excel files in Python, especially when dealing with the .xlsx file format, while `pandas` facilitates data manipulation and analysis, offering data structures and operations for manipulating numerical tables and time series.

The product reviews dataset comprises cleaned text data extracted from product reviews. It encompasses two primary attributes: the content of the review and a corresponding sentiment label. The sentiment labels are categorized as either 'negative' or 'positive'. To facilitate computational analysis, these labels are numerically encoded, with

[5]`https://www.kaggle.com/datasets/nikosfragkis/skroutz-shop-reviews-sentiment-analysis/download?datasetVersionNumber=1`

'negative' mapped to 0 and 'positive' to 1. Additionally, a new column, 'content_length', is introduced to represent the length of each review in terms of the number of characters.

An analysis of the sentiment distribution in the dataset, as shown in image 21, indicates an equal representation of both negative and positive sentiments, with each category comprising 3276 observations. This balanced distribution is crucial for ensuring unbiased model training and evaluation.

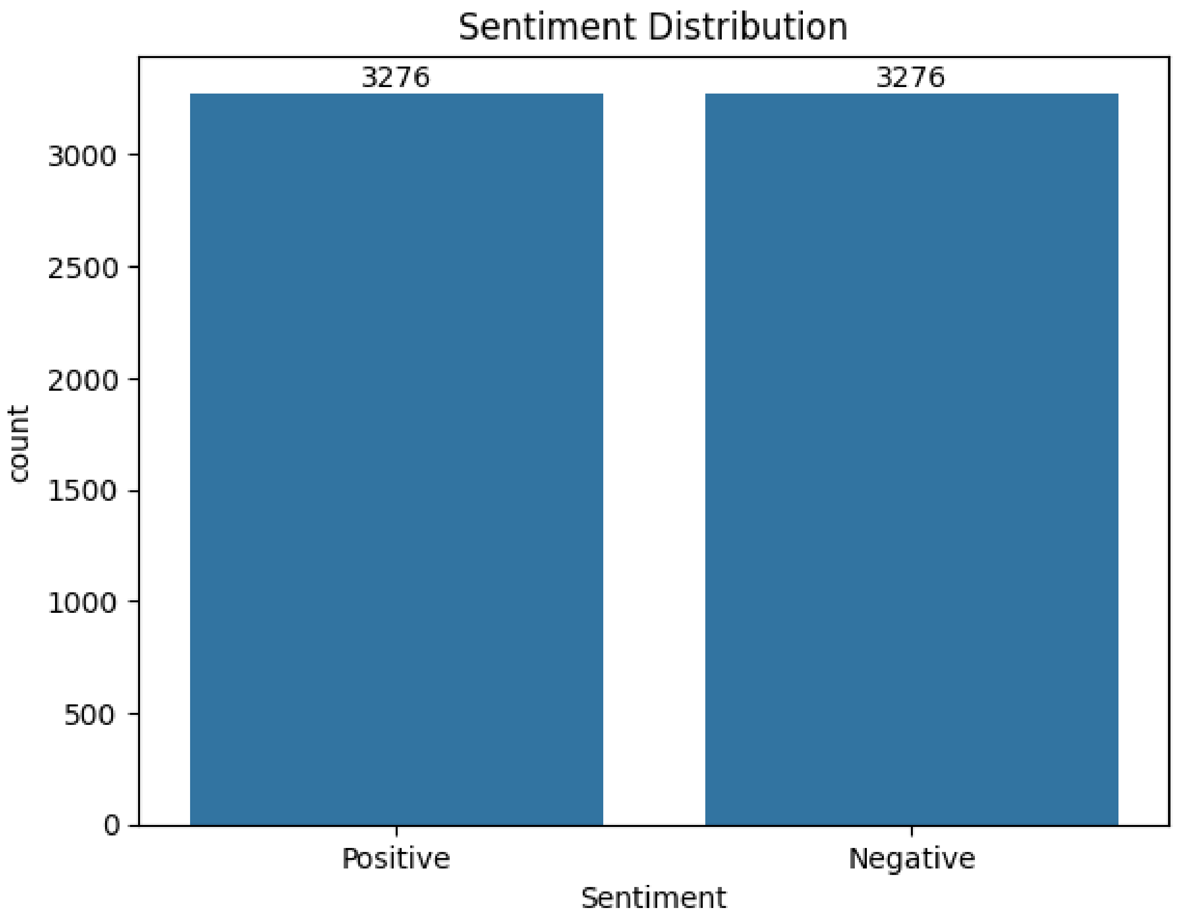


Figure 21: Sentiment distribution - product reviews

Furthermore, an exploration into the content length of the reviews, as illustrated in image 22, yields insightful details. The length of reviews varies significantly, with the maximum length being 7285 characters and the minimum length 26 characters. On average, the reviews consist of approximately 495 characters, with a median length of 314 characters. This variation in length highlights the diverse nature of the review content, ranging from concise to more elaborate expressions of opinions. The content length distribution, depicted in the referenced image, is an essential factor to consider, particularly when designing models for text analysis. It impacts both the feature extraction process and the choice of model architecture.

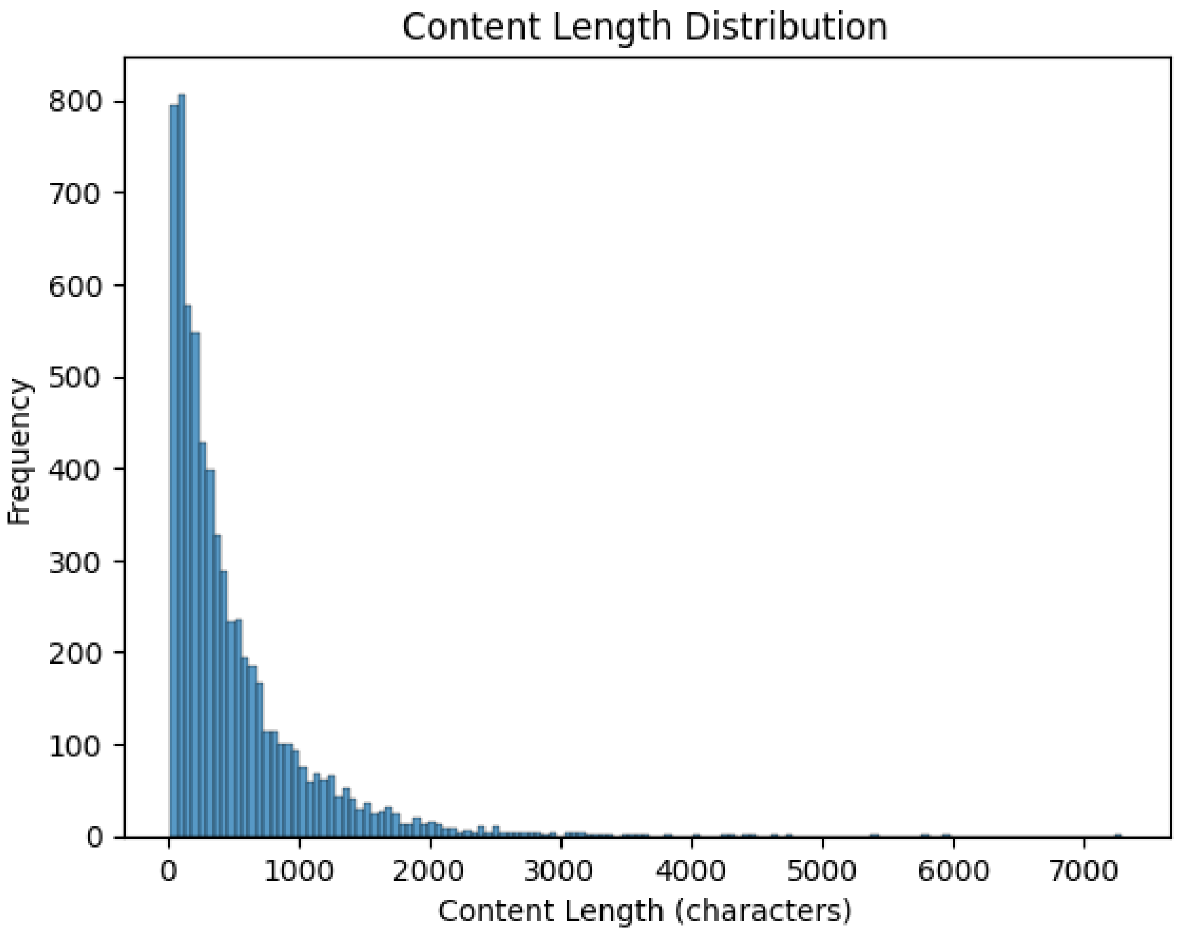


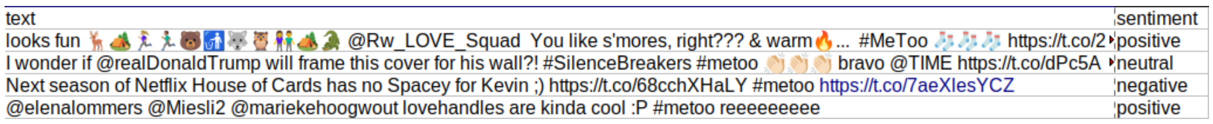
Figure 22: Length distribution - product reviews

3.2 Preprocessing

Before diving into sentiment analysis, it was imperative to prepare the datasets through a series of preprocessing steps to ensure data quality and relevance. Preprocessing, particularly data cleansing, is essential in text analysis to remove noise and standardize the data, which in turn improves the accuracy of the analysis.

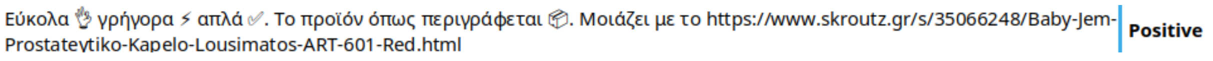
The datasets, both contained various forms of noise typical in social media text. These forms of noise include, but are not limited to, slang, typos, URLs, hashtags, user mentions, as well as emoticons and emojis. To illustrate the necessity of data cleansing, below are five randomly selected texts from each of the initial datasets:

As illustrated in Figure 23 and 24, the snapshots present a complex blend of words, emoticons, emojis, slang, user mentions, hashtags, URLs, and various other elements commonly found in social media text.



text	sentiment
looks fun 🦌🏕🏃‍♀️🏃🐻🚮🐺🦉👫🏕🐊 @Rw_LOVE_Squad You like s'mores, right??? & warm🔥... #MeToo 🧦🧦🧦 https://t.co/2	positive
I wonder if @realDonaldTrump will frame this cover for his wall?! #SilenceBreakers #metoo 👏👏👏 bravo @TIME https://t.co/dPc5A	neutral
Next season of Netflix House of Cards has no Spacey for Kevin ;) https://t.co/68cchXHaLY #metoo https://t.co/7aeXlesYCZ	negative
@elenalommers @Miesli2 @mariekehoogwout lovehandles are kinda cool :P #metoo reeeeeeeee	positive

Figure 23: Random tweets from raw dataset



Εύκολα 👌 γρήγορα ⚡ απλά ✅. Το προϊόν όπως περιγράφεται 📦. Μοιάζει με το https://www.skroutz.gr/s/35066248/Baby-Jem-Prostatevtiko-Kapelo-Lousimatos-ART-601-Red.html	**Positive**

Figure 24: Random review from raw dataset

3.2.1 General text preprocessing

Several preprocessing steps were undertaken to clean the dataset:

- URLs and user mentions were removed to eliminate non-relevant text and to focus on the content of the messages.
- Hashtags were also removed to maintain consistency across the dataset, as their inclusion could introduce bias toward certain topics.
- The text was converted to lowercase to ensure uniformity and to prevent the same words in different cases from being counted as distinct tokens.
- Any remaining HTML entities were removed to clean the data further, ensuring that the analysis would be performed on pure textual content without any markup or web formatting elements.

The preprocessing steps outlined above were primarily executed using regular expressions and the HTML Python module to detect and clean out any remaining HTML entities from the dataset.

3.2.2 Emojis and Emoticons Removal

Emojis and emoticons, while expressive and widely used in digital communication, can pose several challenges in text analysis.

The terms "emoticon" and "emoji" are frequently used as if they mean the same thing, but they actually refer to two different concepts.

Emoticons are textual representations of a writer's mood or facial expressions, created using characters from a standard keyboard. For example, :-) represents a smiley face.

On the other hand, emojis are graphical images that represent emotions, objects, or symbols, and are not constructed from keyboard characters. Instead, they are chosen from a set of available images and are standardized across different platforms through Unicode encoding.

Emoticons and emojis require distinct handling: emoticons, being text-based, can be parsed and interpreted directly from the text. In contrast, emojis, as image-based symbols, may necessitate additional encoding or decoding steps because they are not standard text characters. Both of these processes—interpreting emoticons and encoding/decoding emojis—are beyond the scope of this thesis.

The challenges posed by emojis and emoticons in text analysis were addressed by removing them from the dataset to maintain focus on textual content. Emojis were efficiently stripped out using a third-party Python library called demoji, which is specifically designed for detecting and removing emoji characters from text.

For emoticons, a comprehensive list was sourced from the "List of emoticons" page on Wikipedia [6]. This list was serialized into a file using Python's pickle module. The serialized list was then used to systematically search through the dataset, replacing any found emoticons with an empty string, effectively removing them from the analysis. This approach ensured that both types of visual sentiment indicators were excluded, allowing the sentiment analysis to focus solely on the text.

3.3 Evaluation

This chapter is an important step to language understanding using artificial intelligence. In this practical part, the evaluation is conducted using two distinct datasets: a collection of tweets and a compilation of products reviews. These diverse datasets provide a robust platform for assessing the models' ability to understand and interpret sentiment in varied contexts.

We start with a basic vanilla Recurrent Neural Network (RNN). It's a simple model and good at reading sequences of words. Next, we use a better model known as Long Short-Term Memory (LSTM), which fixes some of the problems the first model has, especially with understanding sentences in which the concepts are spread out across longer sections of text.

The highlight of our work is tweaking models like BERT, RoBERTa, and GPT-2. These models are really good at paying attention to the details in language, which helps them get a deeper sense of emotions in text.By fine-tuning these models, we aim to enhance their proficiency in sentiment analysis, a critical aspect of language understanding.

3.3.1 Evaluation metrics

Our evaluation methodology employs metrics like accuracy, precision, recall, and the F1 score. These metrics serve as objective standards, allowing us to compare the models' performance.

- Accuracy: measures the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined. It is a general indicator

[6] `https://en.wikipedia.org/wiki/List_of_emoticons`

of a model's performance.

$$accuracy_c = \frac{TP_c + TN_c}{TP_c + TN_c + FP_c + FN_c} \tag{6}$$

- Precision: refers to the proportion of true positive predictions in relation to the total number of positive predictions made. It essentially shows how many of the positively predicted cases were actually positive, highlighting the model's ability to avoid false positives.

$$precision_c = \frac{TP_c}{TP_c + FP_c} \tag{7}$$

- Recall: (sensitivity), measures the proportion of actual positive cases that the model correctly identified. This metric is crucial for understanding how well the model is at detecting positive instances.

$$recall_c = \frac{TP_c}{TP_c + FN_c} \tag{8}$$

- F1 Score: is a harmonic mean of precision and recall, providing a single score that balances both metrics. It is particularly useful when the balance between precision and recall is important in the context of the model's application.

$$f1\text{-}score_c = 2 \times \frac{precision_c \times recall_c}{precision_c + recall_c} \tag{9}$$

TP_c True Positives for class c: The number of instances correctly identified as belonging to class c.

TN_c True Negatives for class c: The number of instances correctly identified as not belonging to class c.

FP_c False Positives for class c: The number of instances incorrectly identified as belonging to class c.

FN_c False Negatives for class c: The number of instances incorrectly identified as not belonging to class c.

Through this analysis, we aim to identify the model that demonstrates the highest efficacy in sentiment analysis.

3.4 Tools and libraries

In the practical implementation of the study, different tools and libraries were employed to develop and evaluate the models.

For the recurrent neural network (RNN) and long short-term memory (LSTM) models, development was carried out locally. Python was chosen as the primary programming language and PyCharm served as the integrated development environment (IDE), offering an efficient and user-friendly platform for coding and testing.

The deep learning library Keras, with CUDA support enabled, was the main tool for constructing and training the neural network models, allowing for the use of GPU acceleration to significantly speed up the training process.

For dataset preprocessing and analysis, particularly with the Greek datasets, popular Python libraries such as NumPy and pandas were utilized. These libraries provided efficient data manipulation and analysis capabilities. Scikit-learn was used for computing various metrics, enabling a rigorous evaluation of the model performances.

In contrast, for the implementation of advanced models like BERT, RoBERTa, and GPT-2, Google Colab was utilized. The Transformers library and PyTorch were integral to the implementation of these models. Additionally, for managing the #MeToo dataset, the sqlite3 command-line tool was used to extract specific tables from the database.

The thesis document was composed using LaTeX, with TeXstudio as the chosen editing platform. LaTeX was selected for its high level of control over document formatting, especially adept at handling technical and mathematical expressions. TeXstudio, with its comprehensive suite of features tailored for LaTeX, facilitated a more efficient and streamlined writing process.

4 Experiments

This section is dedicated to the experimental process and the results obtained from the models RNN, LSTM, BERT, RoBERTa, and GPT-2. Each model's setup, including the number of epochs, hyperparameters, and other critical settings, will be clearly outlined, ensuring a comprehensive understanding of the testing methodology during the experiments. The analysis of the results aims to provide a clear perspective on how each model performs in understanding both Greek and English languages.

4.1 Experimental setup

The sentiment analysis with the *Product Reviews* dataset is approached as a binary classification problem. In this dataset, the sentiment labels are categorized as 'negative' and 'positive,' which are mapped to numerical values of 0 and 1, respectively. On the other hand, the *MeToo* dataset represents a multi-class classification problem. It includes three types of sentiment labels: 'negative,' 'neutral,' and 'positive,' each correspondingly mapped to numerical values of 0, 1, and 2. In order to evaluate the model's performance in new unseen data, the data is divided allocating 80% for training and 20% for testing.

4.1.1 RNNs

Regarding the compilation of Recurrent Neural Networks (RNNs) models, there are distinct configurations for two different datasets. For the *Product Reviews* dataset, which is a binary classification, the models are compiled with the `adam` optimizer and the `binary_crossentropy` loss function. Conversely, for the *MeToo* dataset, which requires multi-class classification, the compilation utilizes the `adam` optimizer paired with the `categorical_crossentropy` loss function. The training process for these models spans over 5 epochs, processing 32 samples in each batch. This batch size is a crucial factor in determining the efficiency and effectiveness of the training.

4.1.2 Transformers

In the experimental setup for evaluating Transformer models, the study employed three distinct models: BERT, RoBERTa, and GPT-2, each adapted for specific language datasets. Despite the variations in models, the hyperparameters remained consistent across experiments, ensuring uniformity in the evaluation criteria.

- Model Variations: For the English dataset, the utilized models are 'bert-base-uncased' for BERT, 'roberta-base' for RoBERTa, and GPT-2. For the Greek dataset, the utilized models are 'bert-base-greek-uncased-v1' for BERT, 'roberta-small-greek' for RoBERTa, and 'gpt2-small-greek-v2' for GPT-2.
- Data Handling and Encoding: The datasets comprised text data with sentiment labels. For the product reviews dataset, the sentiment classification was binary, with labels 'negative' and 'positive' numerically encoded as 0 and 1, respectively. In contract, for the #MeToo dataset the sentiment classification was multiclass with labels 'negative', 'neutral', 'positive' encoded as 0,1,2 respectively.
- Common Hyperparameters: Across all experiments, the models shared identical hyperparameters. This included a consistent learning rate 2e-5, batch size 16, number of training epochs 5 , and weight decay 0.01 to regularize and prevent overfitting.
- Training and Evaluation: Each model was fine-tuned on its respective dataset, undergoing the same number of epochs and employing early stopping mechanisms to prevent overfitting. The primary evaluation metric was the F1 score, offering a balanced measure of precision and recall, especially crucial in the binary classification of the Greek dataset.

4.2 Vanilla RNN

The model consists of an Embedding layer, a SimpleRNN layer, and a Dense output layer.

- Embedding layer: The layer used to convert integer representations of words (typically indices) into dense vectors of fixed size
- Simple RNN layer: The core RNN layer that uses simple feedback loops to process sequences of data.
- Dense layer: This is a fully connected neural network layer that outputs the final predictions. The activation function typically follows this layer to output probabilities for each class.

4.2.1 Vanilla RNN on product reviews dataset

For the product reviews dataset we obtained the following results:

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 32)	320000
simple_rnn (SimpleRNN)	(None, 32)	2080
dense (Dense)	(None, 1)	33

Table 2: Vanilla RNN architecture

	precision	recall	f1-score
negative	0.59	0.60	0.60
positive	0.59	0.57	0.58
accuracy			0.59

Table 3: RNN - Product reviews results

The vanilla RNN model has given mediocre performance across its metrics, lying around at 0.59 for negatively and positively classes. The performance across both classes implies that the model does not have a strong bias in any direction against either negative or positive sentiments. On the other hand, the low performance of the model can be attributed to the inherent problems of the Vanilla RNN architecture, most specifically the problem of capturing long-range dependencies within the data.

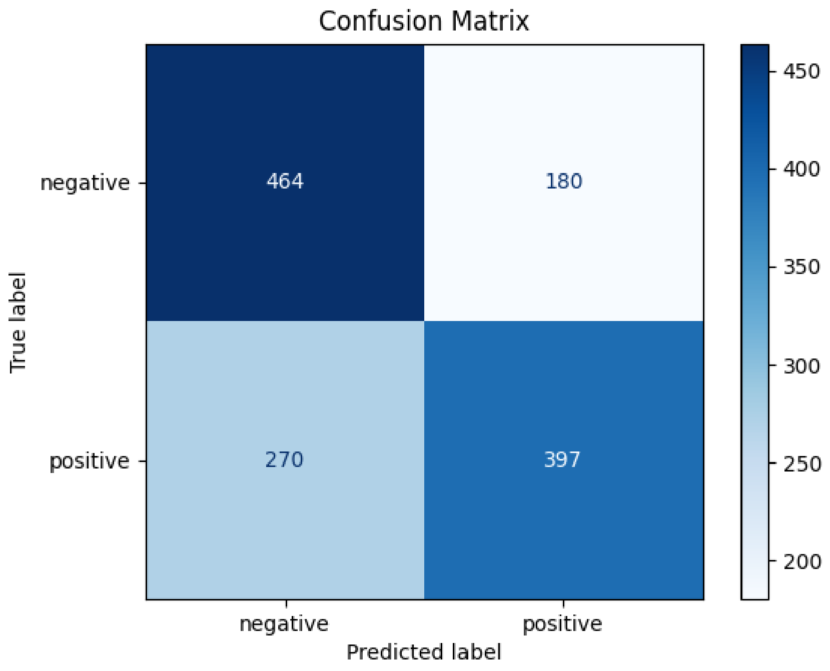


Figure 25: Confusion matrix - Vanilla RNN on product reviews dataset

4.2.2 Vanilla RNN on metoo dataset

For the #MeToo dataset, we obtained the following results:

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 32)	320000
simple_rnn (SimpleRNN)	(None, 32)	2080
dense (Dense)	(None, 3)	66

Table 4: Vanilla RNN architecture

	precision	recall	f1-score
negative	0.52	0.48	0.50
neutral	0.63	0.81	0.71
positive	0.67	0.51	0.58
accuracy			0.61

Table 5: RNN - metoo results

The evaluation results of the model present a varied level of performance across its classes, however, the weakest performance is in the negative class, where precision is at 0.52 and recall at 0.48. On the other hand, the neutral class, works considerably better with a precision of 0.63 and much higher recall of 0.81, which appears to speak of a much stronger ability of the model to classify correctly the instances belonging to that class.

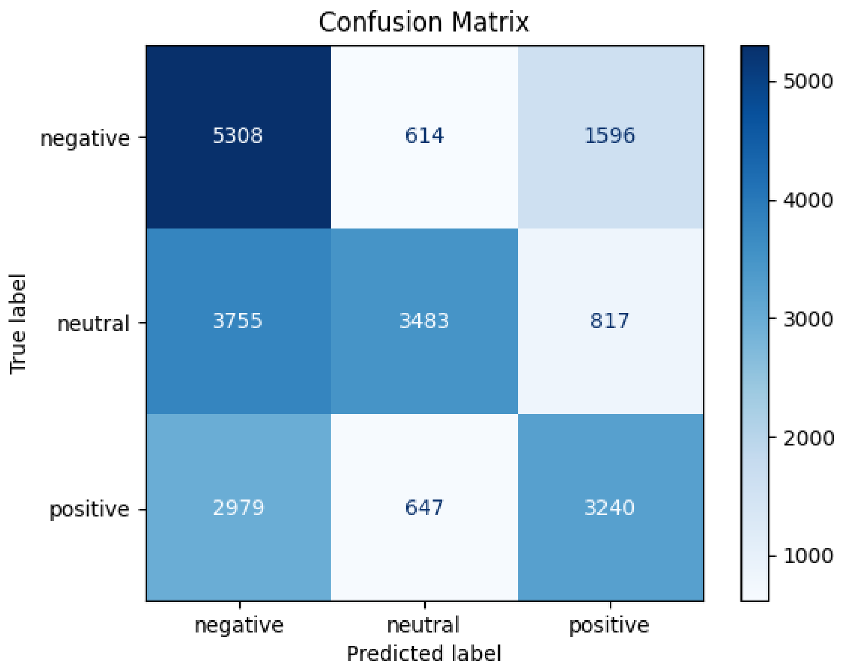


Figure 26: Confusion matrix - Vanilla RNN on #metoo dataset

Given the well-documented difficulties of Vanilla RNNs with processing long-term dependencies and intricate data patterns, it could be worthwhile to consider more sophisticated RNN architectures like LSTMs to potentially improve the model's performance.

4.3 LSTM

In the coming section, we delve into the Long Short-Term Memory (LSTM) model. LSTM is a subtype of Recurrent Neural Networks (RNNs) and it is specifically designed to overcome previous limitations, particularly in capturing long-term dependencies within sequential data. The unique architecture of the network enables it to retain information over long periods, making it suitable for sophisticated tasks that require understanding sequences of data with long intervals.

The following layers constitute our LSTM network:

- Embedding layer: The layer used to convert integer representations of words (typically indices) into dense vectors of fixed size.
- LSTM layer: This layer is a Long Short-Term Memory layer, which is a type of RNN (Recurrent Neural Network) that can capture long-term dependencies and relationships in sequence data. The LSTM layer has 64 units, which means it will maintain 64-dimensional hidden state vectors.
- Dense layer: This is a fully connected neural network layer that outputs the final predictions. The activation function typically follows this layer to output probabilities for each class.

4.3.1 LSTM on product reviews dataset

For the product reviews dataset, we obtained the following results:

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 32)	320000
lstm (LSTM)	(None, 1002)	4148280
dense (Dense)	(None, 1)	1003

Table 6: LSTM architecture

	precision	recall	f1-score
negative	0.67	0.84	0.74
positive	0.79	0.60	0.68
accuracy			0.72

Table 7: LSTM - Product reviews results

The LSTM model presents a balanced performance in binary classification. In the negative class, it achieves effective identification with a precision of 0.67 and a notably high recall of 0.84, leading to an F1-score of 0.74. This demonstrates a good capability in correctly identifying most negative instances. Conversely, for the positive class, it shows a higher precision of 0.79 but a lower recall of 0.60.

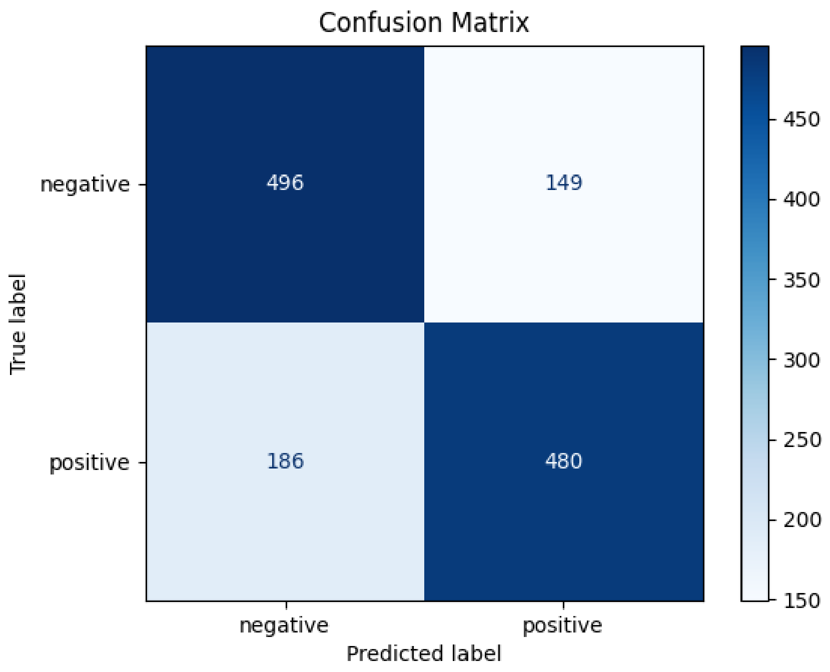


Figure 27: Confusion matrix - LSTM on product reviews dataset

The results show that the model is more accurate when predicting positive cases but tends to miss a significant proportion of actual positive samples.

4.3.2 LSTM on metoo dataset

For the #metoo dataset, we obtained the following results:

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 32)	320000
lstm (LSTM)	(None, 64)	24832
dense (Dense)	(None, 3)	195

Table 8: LSTM architecture

	precision	recall	f1-score
negative	0.75	0.75	0.75
neutral	0.78	0.80	0.79
positive	0.75	0.73	0.74
accuracy			0.76

Table 9: LSTM - metoo results

The LSTM model presents a relatively balanced performance in a multi-class classification task. The negative class exhibits good performance with both precision and recall at 0.75, achieving an F1-score of 0.75 as well, which reveals the model's capability to effectively identify negative samples while minimizing false negatives.

The neutral class is slightly better in classification, with precision at 0.78, recall at 0.80, and an F1-score of 0.79, indicating a more robust capability in accurately classifying neutral instances.

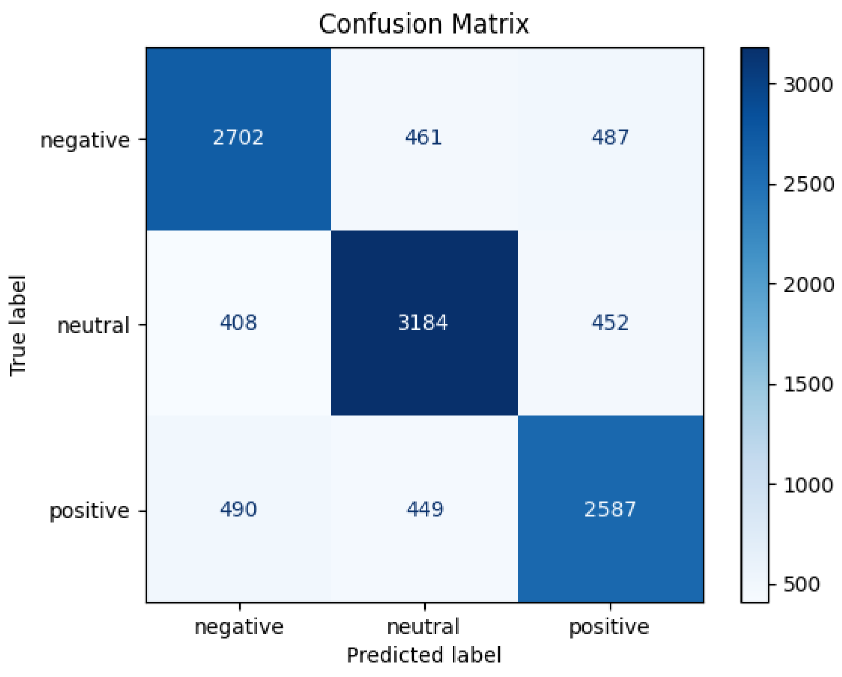


Figure 28: Confusion matrix - LSTM on #metoo dataset

4.4 BERT

In contrast to earlier models that processed text in a strictly unidirectional manner, BERT introduces a bidirectional approach to text analysis. This innovative method means that for every word in a sentence, BERT evaluates both the preceding and following context, offering a comprehensive view of the surrounding text. This capability to consider the entire context allows BERT to gain a deeper and more nuanced understanding of language. Consequently, BERT excels in tasks such as sentiment analysis, question answering, and language inference, thanks to its enriched language comprehension.

In this study's BERT model evaluation, different versions of the BERT model were employed, each optimized for the specific language dataset to ensure maximum performance. For the English dataset, the 'bert-base-uncased' model, pretrained on the English language, was selected [7] . This choice leveraged the model's capability to understand and process English texts effectively, particularly beneficial for the dataset of #metoo tweets.

[7]https://huggingface.co/bert-base-uncased

For the Greek dataset, which included product reviews, a BERT model pretrained on the Greek language was utilized [8]. This approach was critical to capture the nuances and contextual elements unique to the Greek language.

Both models were fine-tuned using the same set of parameters to ensure methodological consistency and facilitate a straightforward comparison of results across languages. The parameters included a learning rate of 2e-5, a batch size of 16 for both training and evaluation, a total of 5 training epochs, and a weight decay of 0.01. The evaluation strategy was implemented at each epoch, saving the best model every time. An early stopping mechanism with a patience of 3 epochs was introduced to prevent overfitting. The primary optimization metric was 'eval_loss', and a custom 'compute_metrics' function was employed to calculate the f1 macro score, providing a thorough evaluation of the model's performance. These parameters were selected to balance efficient training with effective learning, ensuring the models were optimally tuned for their respective tasks.

4.4.1 Bert on product reviews dataset

For the product reviews dataset, we obtained the following results:

	precision	recall	f1-score
negative	0.94	0.95	0.95
positive	0.95	0.95	0.95
accuracy			0.95

Table 10: BERT results

The evaluation results of the model illustrate its superior efficacy in text classification tasks. It has exhibited high precision and recall, achieving scores of 0.94 and 0.95 for the negative and positive classes, respectively. These figures underscore the model's strong capacity to accurately classify instances into their appropriate categories while effectively minimizing both false positives and false negatives.

Additionally, the model attained an outstanding F1-score of 0.95 across both classes. This score signifies the model's well-balanced proficiency in terms of precision and recall.

[8] https://github.com/nlpaueb/greek-bert

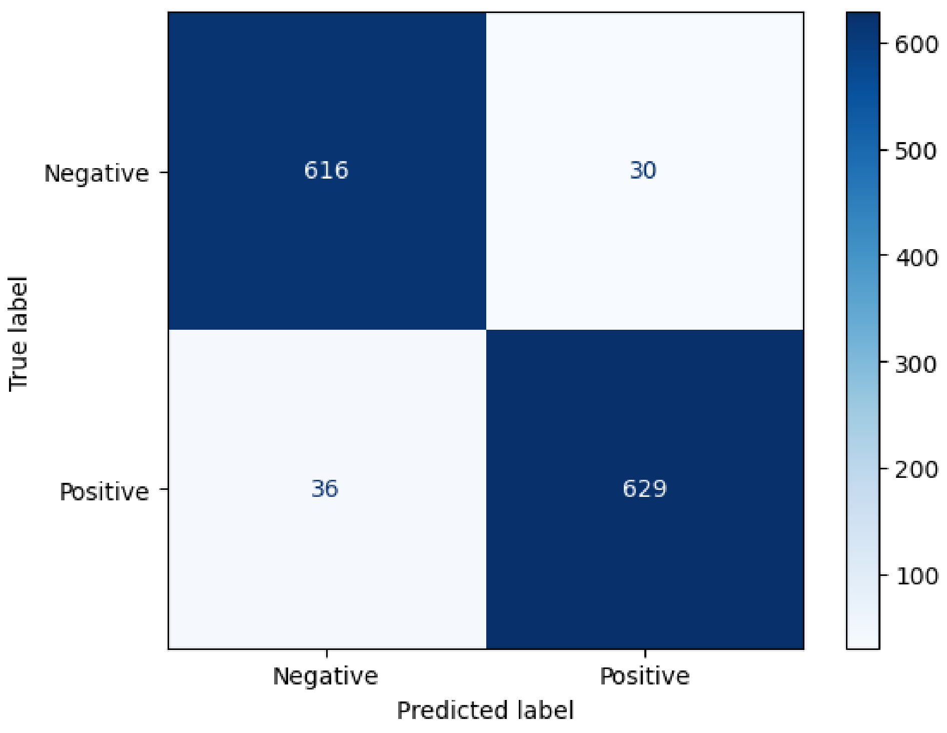


Figure 29: Confusion matrix - Bert on product reviews dataset

4.4.2 Bert on metoo dataset

For the #metoo dataset, we obtained the following results:

	precision	recall	f1-score
negative	0.96	0.96	0.96
neutral	0.98	0.98	0.98
positive	0.96	0.95	0.95
accuracy			0.96

Table 11: Bert - metoo results

The model achieved notable performance, high precision and recall across the classes, with scores of 0.96 for negative, 0.98 for neutral, and 0.96 for positive sentiments, respectively.

These results show the model's high capability in accurately classifying texts into their respective sentiment categories. The overall model accuracy was quite high at 0.96, indicating the model's effectiveness in correctly classifying a significant proportion of the instances in the dataset.

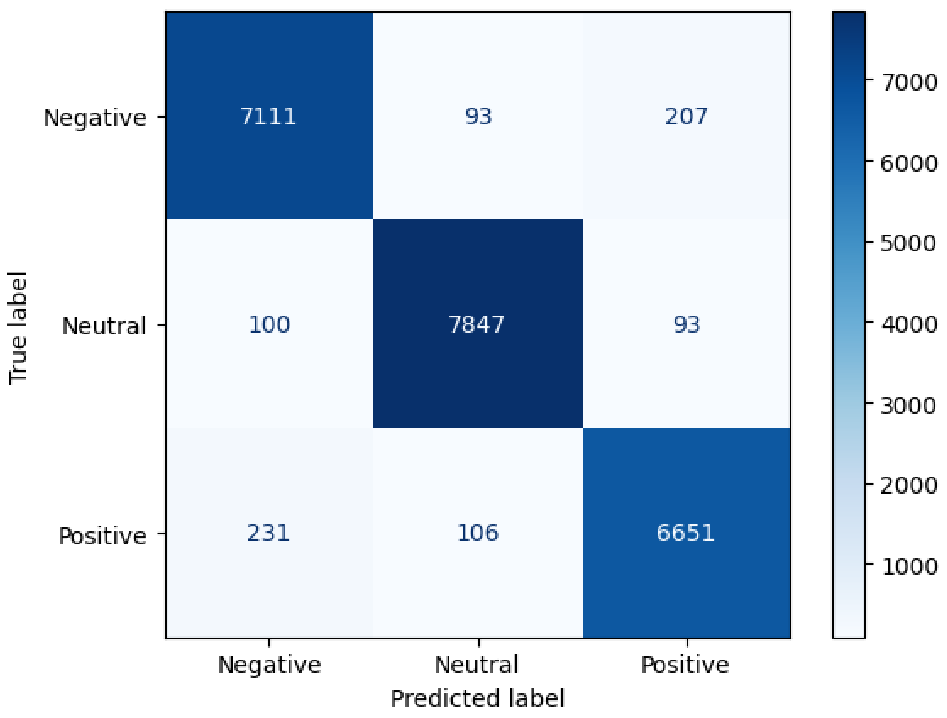


Figure 30: Confusion matrix - BERT on #metoo dataset

4.5 RoBERTa

RoBERTa differentiates itself from Bert through training on a larger corpus along with the elimination of the Next Sentence Prediction task. These refinements enable RoBERTa to capture complexities in language with greater precision.

In order to evaluate the RoBERTa model, language-specific versions were employed to ensure the most effective analysis of each dataset. For the English dataset, the 'roberta-base' model [9], pretrained on the English language, was utilized for the accurate analysis of the English tweets dataset.

In contrast, for the product reviews dataset, it was used a version of RoBERTa, pretrained on the Greek language [10].

Both these language-specific models were fine-tuned using the same set of parameters, as outlined in the provided code. These parameters included a learning rate of 2e-5, a per-device training and evaluation batch size of 16, and a total of 5 training epochs. The weight decay was set at 0.01. The training and evaluation processes were structured to occur at the end of each epoch, with the best model being saved at each epoch. Additionally, to ensure optimal model performance and prevent overfitting, an EarlyStoppingCallback with a patience of 3 epochs was implemented.

[9]https://huggingface.co/docs/transformers/model_doc/roberta
[10]https://huggingface.co/ClassCat/roberta-small-greek

The metric for determining the best model was 'eval_loss', and the 'compute_metrics' function was utilized to calculate the f1 macro score, providing a comprehensive measure of the models' performance in sentiment analysis.

4.5.1 Roberta on product reviews dataset

For the product reviews dataset, we obtained the following results:

	precision	recall	f1-score
negative	0.92	0.94	0.93
positive	0.94	0.92	0.93
accuracy			0.93

Table 12: roBERTa - Product reviews results

The performance of the model on the product reviews dataset, shows a high level of accuracy and an impressive balance in sentiment analysis. Notably, both the precision and recall scores exceed 0.92 for negative and positive classes, indicating the model's accuracy and effectiveness in identifying relevant instances. This is further emphasized by the balanced performance across both sentiment classes, with no significant bias toward either, and consistent f1-scores of 0.93 for both classes.

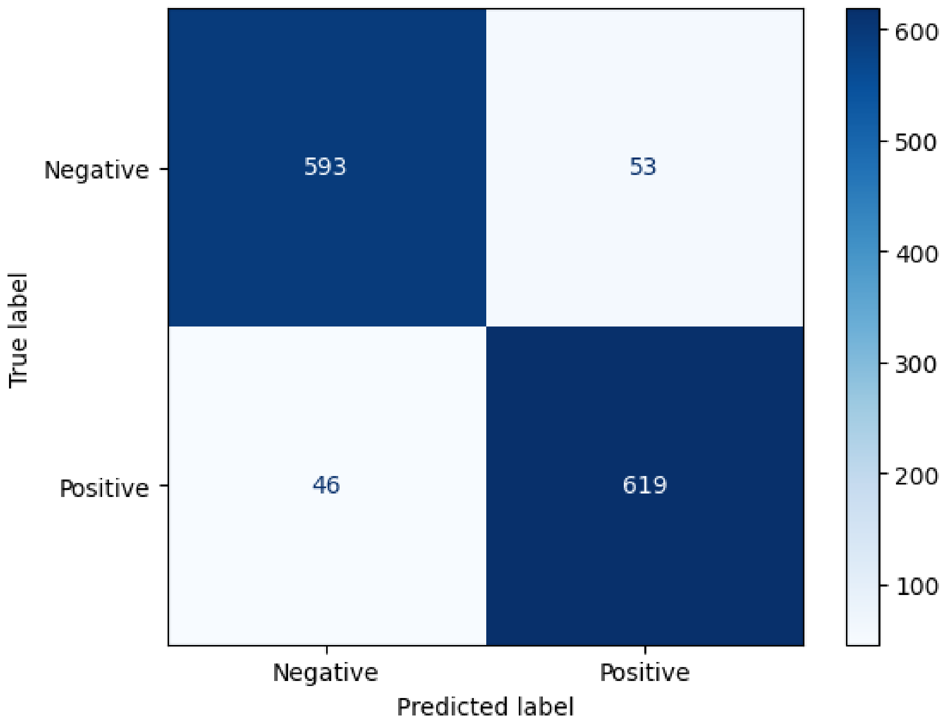


Figure 31: Confusion matrix - roBERTa on product reviews dataset

4.5.2 Roberta on Metoo dataset

For the #metoo dataset, we obtained the following results: The performance of the model

	precision	recall	f1-score
negative	0.95	0.96	0.95
neutral	0.97	0.98	0.97
positive	0.96	0.94	0.95
accuracy			0.96

Table 13: roBERTa - metoo results

on the #metoo dataset is exceptionally impressive, demonstrating high precision across negative, neutral, and positive sentiment classes, with scores of 0.95, 0.97, and 0.96, respectively. This accuracy, coupled with equally high recall scores (0.96 for negative, 0.98 for neutral, and 0.94 for positive), indicates the model's effectiveness in correctly identifying and classifying sentiments.

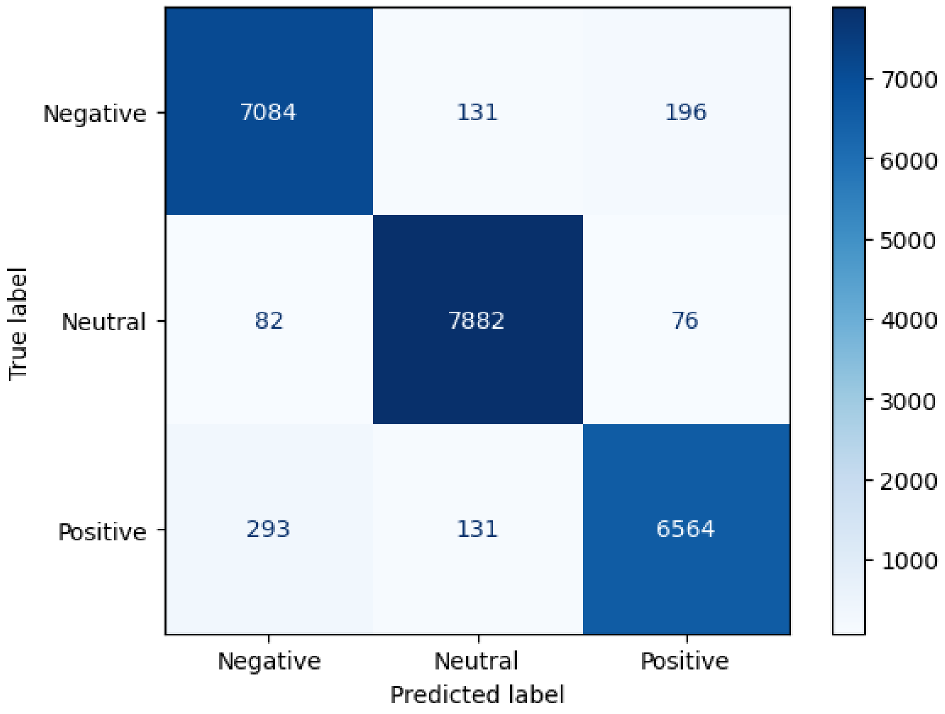


Figure 32: Confusion matrix - roBERTa on #metoo dataset

4.6 GPT-2

GPT-2, developed by OpenAI, represents a significant leap in the field of natural language processing. GPT-2's proficiency in comprehending and producing human-like text relies in its deep learning model, which contains a substantial number of layers and parameters, enabling it to capture a wide array of linguistic nuances and styles.

In order to evaluate the, distinct versions of the GPT-2 model were employed for each language dataset, ensuring optimal language-specific performance. For the English dataset, the standard GPT2Model [11], which is pretrained on the English language, was used.

Conversely, for the Greek dataset, a specialized version of GPT-2, pretrained on the Greek language [12], was utilized.

For both language versions of the GPT-2 model, a uniform set of training parameters was applied to maintain methodological consistency across both datasets and facilitate direct comparative analysis. These parameters, as defined in the provided Python code, included a learning rate of 2e-5, a training batch size of 16, an evaluation batch size of 8, and gradient accumulation steps set to 2. The models were trained over a total of 5 epochs with a weight decay of 0.01. The evaluation strategy was implemented at the end of each epoch, with the best model being saved correspondingly. Additionally, an EarlyStoppingCallback with a patience of 3 epochs was incorporated to prevent overfitting.

The 'eval_loss' was used as the primary metric for model optimization. The 'compute_metrics' function, designed to calculate the f1 macro score, provided a comprehensive evaluation of the model's performance.

4.6.1 GPT-2 on product reviews dataset

For the product reviews dataset, we obtained the following results:

	precision	recall	f1-score
negative	0.93	0.91	0.92
positive	0.91	0.94	0.92
accuracy			0.92

Table 14: GPT-2 - Product reviews results

The metrics on both the negative and positive sentiment classes show high performance, with precision scores of 0.93 for negative and 0.91 for positive sentiments. This indicates a high degree of accuracy in the model's classifications. In terms of recall, the model successfully identified 91% of the negative and 94% of the positive sentiments, showcasing its effectiveness in capturing the majority of relevant instances.

[11]https://huggingface.co/docs/transformers/model_doc/gpt2
[12]https://huggingface.co/ClassCat/gpt2-small-greek-v2

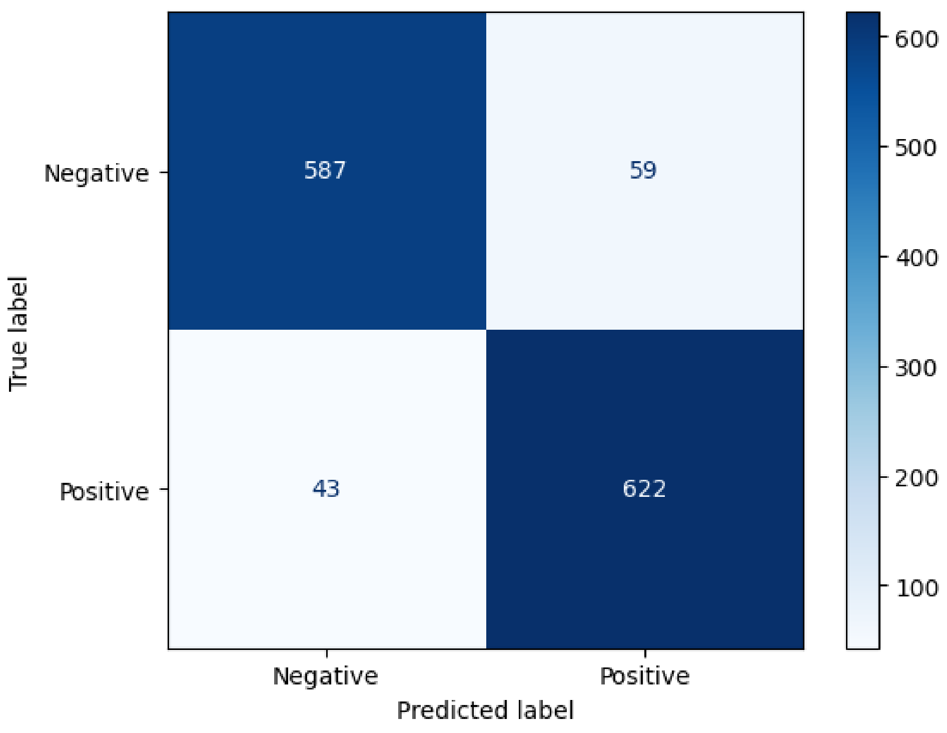


Figure 33: Confusion matrix - GPT-2 on product reviews dataset

4.6.2 GPT-2 on metoo dataset

For the #metoo dataset, we obtained the following results:

	precision	recall	f1-score
negative	0.92	0.95	0.93
neutral	0.97	0.94	0.96
positive	0.93	0.93	0.93
accuracy			0.94

Table 15: GPT-2 - metoo results

The model achieves precision scores of 0.92, 0.97, and 0.93 for negative, neutral, and positive sentiments respectively, indicating its high accuracy in identifying correctly each class. The recall scores are similarly impressive, especially for the negative and neutral classes.

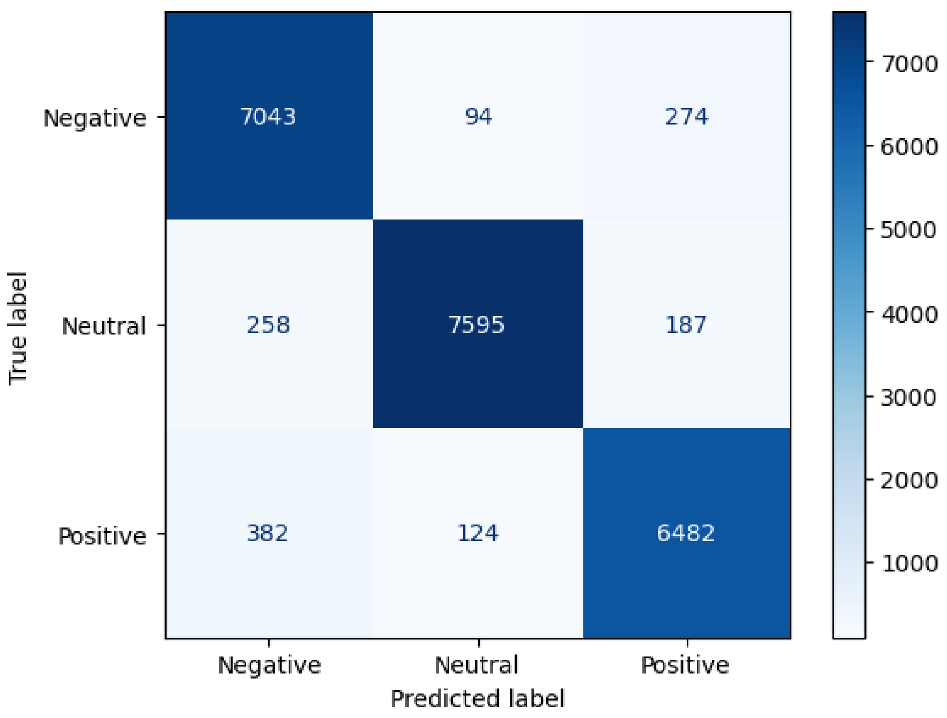


Figure 34: Confusion matrix - GPT-2 on #metoo dataset

5 Comparative analysis

The detailed performance analysis of RNN and LSTM models, as shown in tables 16 and 17, exposes their limitations in handling complex tasks across both Greek and English datasets. The LSTM, as an advanced architecture over the vanilla RNN, shows improvement in both F1-scores and accuracy compared to its predecessor. However, these improvements are not sufficient to compete with the advanced Transformer models. This discrepancy highlights the critical limitations of recurrent architectures in processing long sequences and managing the text context effectively, which is crucial for text understanding.

The Vanilla RNN's performance presents lower accuracy and F1-scores. This underscores the fundamental challenges RNNs face with gradient vanishing and exploding problems, which severely limit their ability to learn from long text sequences.

LSTMs, while designed to overcome the short-term memory limitations of RNNs, still fall short of the benchmark set by Transformer models. The LSTM's performance improvement over RNNs is evident, indicating its enhanced capability in handling sequence dependencies through its memory cell and gate mechanisms. Nevertheless, the LSTM's structure, which linearly processes data, restricts its ability to capture the full scope of contextual information available in parallel processing models like Transformers. This limitation is reflected in its relatively better, yet insufficient, performance metrics when tackling the complex sentiments expressed in product reviews and #metoo tweets.

This is contrasted sharply with Transformer models as shown in tables 16 and 17, which excel in these areas due to their innovative self-attention mechanism, allowing them to weigh the importance of different parts of the text simultaneously.

Model	Class	F1-Score	Accuracy
Vanilla RNN	Negative	0.60	0.59
	Positive	0.58	
LSTM	Negative	0.74	0.72
	Positive	0.68	
BERT	Negative	0.95	0.95
	Positive	0.95	
RoBERTa	Negative	0.93	0.93
	Positive	0.93	
GPT-2	Negative	0.92	0.92
	Positive	0.92	

Table 16: Comparison of BERT, GPT-2, and RoBERTa on product reviews Dataset

Model	Class	F1-Score	Accuracy
	Negative	0.50	0.61
Vanilla RNN	Neutral	0.71	
	Positive	0.58	
	Negative	0.75	0.76
LSTM	Neutral	0.79	
	Positive	0.74	
	Negative	0.96	0.96
BERT	Neutral	0.98	
	Positive	0.95	
	Negative	0.95	0.96
RoBERTa	Neutral	0.97	
	Positive	0.95	
	Negative	0.93	0.94
GPT-2	Neutral	0.96	
	Positive	0.93	

Table 17: Comparison of BERT, GPT-2, and RoBERTa on #metoo Dataset

The performance analysis shows that BERT works better for text classification in with both languages. It get the highest accuracy and f1-scores on all datasets, meaning that BERT is a great model at finding important patterns in text. RoBERTa , although it has lower performance than BERT, it is still very reliable . On the contrary GPT-2 had the lowest performance metrics, compared to BERT and roBERTa. This fact can be explained due to the differences in their architectures and how they process and understand the text. The bidirectional nature of BERT and RoBERTa allows them to better understand the full context of a sentence in contrast to GPT-2 which processes text sequentially and makes decisions based on the preceding text.

6 Conclusion

In this thesis, we highlight how the Transformer models are much more efficient than traditional RNNs in the specific area of language understanding on two different datasets belonging to the English and Greek languages. The first dataset is binomial since it is composed of positive and negative examples from product reviews in Greek, while the second one is polynomial and is composed of positive, negative, and neutral examples from Twitter in English.

The performance of vanilla RNNs was observed to be the poorest among all the models examined. That is due to their natural constraints to capture long-term dependencies in sequential data is critical for accurate sentiment analysis. Using the LSTM model, the accuracy was improved around 22.03% in the Greek dataset as well as 24.59% for the English dataset. This shows the general superiority of LSTMs in handling sequence data in comparison to RNNs.

However, the most remarkable results were observed with Transformer models end especially BERT which achieved the highest accuracy and f1-scores for each class in both Greek and English dataset. Although not quite reaching the performance levels of BERT, both RoBERTa and GPT-2 still showcased remarkable results, demonstrating the efficacy of these advanced models in handling complex classification tasks.

This study had a few key limitations that need to be taken into account for future research. Firstly, the Greek language dataset was quite small, with just 6,552 observations. This means our model may not generalize well to new data limiting its usefulness. Additionally, the English language dataset was unevenly balanced, which we didn't address in this study and this could have biased the model's results towards categories with more data. There were also some limitations in exploring model settings. A more comprehensive search or experimentation for optimal hyperparameters could potentially improve the model's performance. Furthermore, time and budget constraints limited the scope of experimentation, particularly in techniques like pretraining newer models from scratch.

Future research should expand on this work by analyzing larger and more varied datasets, investigating the potential of pretraining and finetuning open-source large language models such as Falcon or LLaMA 2. Additionally, a comparative analysis with state-of-the-art models like GPT-4, using the OpenAI API, could provide valuable benchmarks and insights.

This research has shown that Transformer models, such as BERT, RoBERTa, and GPT-2, have better results in sentiment analysis than traditional RNN models. The study showed that RNNs have limitations, while Transformer models can understand complex language better. Despite challenges, this research provides a strong basis for future work. Future research should use larger datasets and use open-source language models in new ways.

References

[1] Ai vs. machine learning vs. deep learning vs. neural networks: What's the difference? https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/. Accessed: 2023-08-10.

[2] The annotated transformer. https://nlp.seas.harvard.edu/2018/04/03/attention.html. Accessed: 2023-11-12.

[3] Artificial intelligence, machine learning, and deep learning. what's the real difference? https://medium.com/swlh/artificial-intelligence-machine-learning-and-deep-learning-whats-the-real-differe
Accessed: 2023-06-28.

[4] Artificial neural network. https://en.wikipedia.org/wiki/Artificial_neural_network. Accessed: 2023-06-28.

[5] Basic introduction to feed-forward network in deep learning. https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-feed-forward-network-in-deep-learning/. Accessed: 2023-06-28.

[6] Bert — bidirectional encoder representations. https://samanemami.medium.com/bert-bidirectional-encoder-representations-e98833f9dfcd. Accessed: 2023-09-10.

[7] Dissecting bert part 1: The encoder. https://medium.com/dissecting-bert/dissecting-bert-part-1-d3c3d495cdb3. Accessed: 2023-09-10.

[8] Encoders-decoders, sequence to sequence architecture. https://medium.com/analytics-vidhya/encoders-decoders-sequence-to-sequence-architecture-5644efbb3392. Accessed: 2023-08-10.

[9] Everything you need to know about "activation functions" in deep learning models. https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-8
Accessed: 2023-07-28.

[10] How transformers work. https://towardsdatascience.com/transformers-141e32e69591. Accessed: 2023-08-10.

[11] The illustrated transformer. https://jalammar.github.io/illustrated-transformer/. Accessed: 2023-08-10.

[12] The journey of open ai gpt models. https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2. Accessed: 2023-09-10.

[13] Masked-language modeling with bert. https://towardsdatascience.com/masked-language-modelling-with-bert-7d49793e5d2c. Accessed: 2023-08-10.

[14] The neural network zoo. https://www.asimovinstitute.org/neural-network-zoo/. Accessed: 2023-07-30.

[15] The neuron. https://med.libretexts.org/Bookshelves/Pharmacology_and_Neuroscience/Foundations_of_Neuroscience_%28Henley%29/01%3A_Neuron_Structure_and_Function/1.01%3A_The_Neuron. Accessed: 2023-06-28.

[16] An overview of gradient descent optimization algorithms. https://www.ruder.io/optimizing-gradient-descent/. Accessed: 2023-08-10.

[17] Recurrent neural network. https://en.wikipedia.org/wiki/Recurrent_neural_network. Accessed: 2023-08-02.

[18] Recurrent neural networks cheatsheet. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. Accessed: 2023-08-02.

[19] Twitter users worldwide. https://www.statista.com/statistics/303681/twitter-users-worldwide/. Accessed: 2023-06-28.

[20] Types of neural networks and definition of neural network. https://www.mygreatlearning.com/blog/types-of-neural-networks/. Accessed: 2023-07-30.

[21] Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2023-08-10.

[22] What exactly happens when we fine-tune bert? https://towardsdatascience.com/what-exactly-happens-when-we-fine-tune-bert-f5dc32885d76. Accessed: 2023-08-10.

[23] What is a neural network? https://www.ibm.com/topics/neural-networks. Accessed: 2023-06-28.

[24] What is sentiment analysis? https://aws.amazon.com/what-is/sentiment-analysis/. Accessed: 2023-06-25.

[25] S.-i. Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[26] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[27] Y. Bengio. Gradient-Based Optimization of Hyperparameters. *Neural Computation*, 12(8):1889–1900, 08 2000.

[28] J. Dean. Large-scale deep learning for building intelligent computer systems. 2016.

[29] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[30] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022.

[31] D. Favareau. The symbolic species: The co-evolution of language and the brain. *Issues in Applied Linguistics*, 9, 12 1998.

[32] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[33] A. L. Guzman. Ontological boundaries between humans and computers and the implications for human-machine communication. *Human-Machine Communication*, 1:37–54, 2020.

[34] S. H. Haji and A. M. Abdulazeez. Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 18(4):2715–2743, 2021.

[35] S. R. Herrera, M. Ceberio, and V. Kreinovich. When is deep learning better and when is shallow learning better: Qualitative analysis. (1691), 2022.

[36] G. Hinton, Y. LeCun, and Y. Bengio. Deep learning. *Nature*, 521(7553):436–444, 2015.

[37] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06:107–116, 4 1998.

[38] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[39] G. S. P. R. V. Z. Ivan Vasilev, Daniel Slater. *Python Deep Learning: Exploring deep learning techniques, neural network architectures and GANs with PyTorch, Keras and TensorFlow*. Packt Publishing, 2 edition, 2019.

[40] D. Jurafsky and J. H. Martin. *Speech and language processing*. 2023.

[41] M. Kaur and A. Mohta. A review of deep learning with recurrent neural network. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 460–465. IEEE, 2019.

[42] J. Lawrence. *Introduction to Neural Networks*. California Scientific Software, USA, 1993.

[43] Y. Lexuan. Improvements on activation functions in ann: An overview. *Management Science and Engineering*, 14(1):53–58, 2020.

[44] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[45] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius. Deep learning for procedural content generation. *Neural Computing and Applications*, 33:19–37, 1 2021.

[46] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[47] A. Ma'arif, W. Rahmaniar, H. I. K. Fathurrahman, A. Z. K. Frisky, et al. Understanding of convolutional neural network (cnn): A review. *International Journal of Robotics & Control Systems*, 2(4), 2022.

[48] L. R. Medsker and L. Jain. Recurrent neural networks. *Design and Applications*, 5(64-67):2, 2001.

[49] V. Mnih, K. Kavukcuoglu, D. Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[50] M. A. Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[51] R. Parloff. Why deep learning is suddenly changing your life. *Fortune. New York: Time Inc*, 2016.

[52] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[53] S. E. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016.

[54] M. Rodríguez-Ibánez, A. Casánez-Ventura, F. Castejón-Mateos, and P.-M. Cuenca-Jiménez. A review on sentiment analysis from social media platforms. *Expert Systems with Applications*, 223:119862, 2023.

[55] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[56] A. M. Schäfer and H. G. Zimmermann. Recurrent neural networks are universal approximators. In S. D. Kollias, A. Stafylopatis, W. Duch, and E. Oja, editors, *Artificial Neural Networks – ICANN 2006*, pages 632–640, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[58] Wikipedia. Feedforward neural network, 2023. Accessed: 2023-07-28.

[59] R. B. Zadeh and B. Ramsundar. *TensorFlow for Deep Learning.* O'Reilly Media, 2017.

List of Figures