****

Course 61999: Capstone Project –Phase 2

**Capstone Project Phase B**

**22-2-R-4**

**Authorship Recognition in** **Classical Religious Texts**

**Students:**

Yarden Adika – 319118733, yadika14@gmail.com  
 Noor Abu Abeid – 207755695, noorabuabied@gmail.com

**Supervisors:**

Dr. Renata Avros

Prof. Zeev Volkovich

Link for the project:

[Authorship\_Recognition\_in\_Classical\_Religious\_Texts/README.md at main · yardenadika/Authorship\_Recognition\_in\_Classical\_Religious\_Texts (github.com)](https://github.com/yardenadika/Authorship_Recognition_in_Classical_Religious_Texts/blob/main/README.md)

Karmiel – January 2023

**Table of Contents**

1. [Introduction ....................................................................................................... 1](#_Toc125234300)
2. [Background and Related Work .......................................................................2](#_Toc125234301)

[**2.1 Text Analysis ............................................................................................... 2**](#_Toc125234302)

[**2.1.1 Sentiment Analysis ............................................................................2**](#_Toc125234303)

[**2.1.2 Authorship Attribution .....................................................................2**](#_Toc125234304)

[**2.2 Natural Language Processing ....................................................................3**](#_Toc125234305)

[**2.3 Word Embedding ........................................................................................3**](#_Toc125234306)

[**2.4 BERT ............................................................................................................3**](#_Toc125234307)

[**2.5 Deep Learning ..............................................................................................4**](#_Toc125234308)

[**2.5.1 Convolutional Neural Network .........................................................6**](#_Toc125234309)

1. [Expected Achievements ..................................................................................11](#_Toc125234310)
2. [Research / Engineering Process .....................................................................11](#_Toc125234311)

[**4.1 Process ........................................................................................................11**](#_Toc125234312)

[**4.1.1 Description of the research process ................................................11**](#_Toc125234313)

[**4.1.2 The solution .......................................................................................11**](#_Toc125234314)

[**4.1.3 Difficulties .........................................................................................12**](#_Toc125234315)

[**4.2 Product .......................................................................................................12**](#_Toc125234316)

[**4.2.1 Algorithm ..........................................................................................12**](#_Toc125234317)

[**4.2.2 Results ...............................................................................................14**](#_Toc125234318)

1. [Evaluation / Verification Plan ........................................................................19](#_Toc125234319)
2. [Conclusions ......................................................................................................20](#_Toc125234320)
3. [Future work .....................................................................................................20](#_Toc125234321)
4. [User Guide ........................................................................................................20](#_Toc125234322)

[**8.1 Software Environment ............................................................................20**](#_Toc125234323)

[**8.2 Running Instructions ..............................................................................20**](#_Toc125234324)

[**8.3 General Description ................................................................................21**](#_Toc125234325)

[**8.4 User Instructions .....................................................................................21**](#_Toc125234326)

1. [References ........................................................................................................22](#_Toc125234327)

**Abstract**

Research of text analysis has grown significantly in recent years. Many studies have been conducted on various aspects of the field. This area is in high demand as each new study may use a different combination of technologies which could lead to varied results - some better than others.

This project introduces a model intended to identify whether a particular text belongs to the religious texts of the Holy Koran or Bible - using deep learning techniques. It involves word embedding, BERT Transformers (araBert and hebBert), and CNN.

The project aims to improve comprehension of religious texts.

# **Introduction**

This project paper introduces a model designed to identify whether a particular text belongs to the Religious Texts of the Holy Koran or Bible - using deep learning techniques. Several researchers have previously engaged in text analysis and text source detection. However, they either used different networks for training or focused on other aspects of text classification. For example, two recent researches that engaged in text analysis were: "Fine-Grained Sentiment Analysis of Arabic COVID-19 Tweets Using BERT-Based Transformers and Dynamically Weighted Loss Function" [[1]](#_References_1) - investigating the extent to which deep learning models may assist in understanding society’s attitudes toward Coronavirus via analysis of social media posts. And the second one was "A Short-Patterning of the Texts Attributed to Al Ghazali: A “Twitter Look” at the Problem" [[2,]](#_References_1), which focused on the authenticity of manuscripts attributed to Al Ghazali.

The model proposed in this paper distinguishes between the writing styles of different authors and authenticates authorship based on that distinction - using araBert and hebBert for the embedding of the texts (books imposters), CNN (convolutional neural network) for the training process, and the Bible and the Holy Koran for testing.

The model is based on these two pieces of research and combines them to create a new approach that has yet to be applied in this research area.

In addition, it involves analyzing languages that have been less studied until now, such as Hebrew and Arabic.

The stakeholders in the development process of this model are scientists and researchers who deal with extensive texts.

The remainder of this research paper is organized as follows: Section 2 discusses Background and Related Work. Section 3 describes the Expected Achievements of the study. Section 4 explains the work process. Section 5 presents the Evaluation and Verification Plan, and finally, section 6 shows the conclusion.

# **Background and Related Work**

This paper focuses on researching text analysis. Therefore, this section discusses the term Text Analysis in general and concentrates on Text classification, a text analysis technique. In addition, there is a breakdown of the materials related to this study.

## **Text Analysis**

Text Analysis is a [machine-learning](https://monkeylearn.com/machine-learning/) technique that automatically extracts valuable insights from unstructured text data.

Text analysis tools allow businesses to structure vast quantities of information, such as emails, chats, social media, support tickets, and documents - in seconds rather than days [[2]](#_References).

One text analysis technique is text classification - which means that a classical problem in natural language processing (NLP) aims to assign labels or tags to textual units such as sentences, queries, paragraphs, and documents. It has many applications, including question answering, spam detection, sentiment analysis, news categorization, user intent classification, content moderation, etc.

There are many text classification tasks, for example:

### **Sentiment Analysis**

This is the task of analyzing people’s opinions in textual data (e.g., product reviews, movie reviews, or tweets) and extracting their polarity and viewpoint. The task can be shaped as either a binary or a multi-class problem. Binary sentiment analysis classifies texts into positive and negative classes, while multi-class sentiment analysis classifies texts into fine-grained labels or multi-level intensities.

One of the researches conducted on this task is the sentiment analysis of Arabic-speaking people during COVID-19. The paper "Fine-Grained Sentiment Analysis of Arabic COVID-19 Tweets Using BERT-Based Transformers and Dynamically Weighted Loss Function" [[1]](#_References) chose the BERT model based on transformers. To solve the imbalanced data issue, they used a dynamical loss function.

### **Authorship Attribution**

Authorship attribution is the task of identifying the author of a given document. Various style markers have been proposed in the literature to deal with the authorship attribution task. Frequencies of function words and Part-Of-Speech n-grams are very reliable and effective for this task. However, despite being state of the art, they partly rely on the invalid bag-of-words assumption, which stipulates that text is a set of independent words or segments of words [[3]](#_References_1).

## **Natural Language Processing**

NLP is the ability of a computer program to understand human language as it is spoken and written - referred to as natural language. It has existed for over 50 years and has roots in linguistics. It has a variety of real-world applications in several fields, including medical research, search engines, and business intelligence.

## **Word Embedding**

Word embeddings are word representation that allows words with similar meanings to have an equal representation. They are a distributed representation of text that significantly improved the performance of deep learning models on challenging NLP (Natural Language Processing) problems.

**Contextual embeddings** are generally obtained from the transformer-based models and define the meaning/representation of each word according to the sentence it's placed in and the relation between a comment and its neighbors. In this kind of word embedding, there may be more than one meaning for each word.

**Transformer-based models** are neural networks that absorb context and meaning by tracking relationships in sequential data like a word in its sentence. These models apply an evolving set of mathematical techniques, called attention or self-attention, to detect subtle ways even distant data elements in a series influence and depend on each other [[4]](#_References).

## **BERT**

BERT stands for Bidirectional Encoder Representations from Transformers. It is an open-source machine learning framework for NLP designed to pre-train unlabeled text by conditioning both left and right contexts in all layers.

Because BERT is based on a transformer, every output element is connected to every input element, and the weightings between them are dynamically calculated based on their connection.

Bidirectional means BERT can read simultaneously in both directions (left-to-right and right-to-left).

This capability makes BERT pre-training focus on two different NLP tasks: Masked Language Modeling and Next Sentence Prediction.

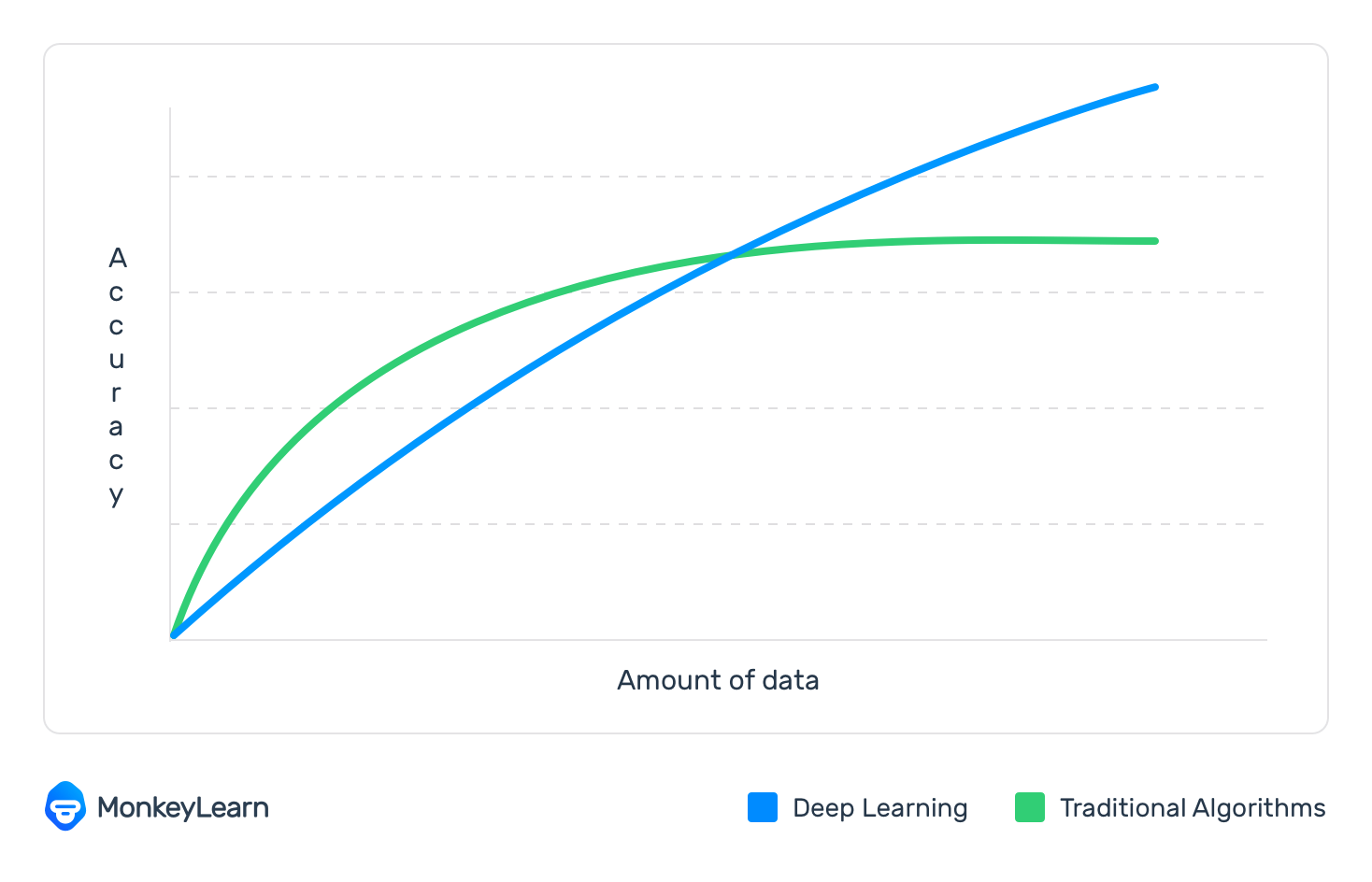
## **Deep Learning**

Deep learning is a set of algorithms and techniques inspired by the human brain called neural networks. Deep learning architectures offer enormous benefits for text classification because they function at super high accuracy with lower-level engineering and computation [[5]](#_References).

The two main deep learning architectures for text classification are [Convolutional Neural Networks](https://machinelearningmastery.com/crash-course-convolutional-neural-networks/) (CNN) and Recurrent Neural Networks (RNN).

Deep learning is hierarchical machine learning, using multiple algorithms in a progressive chain of events. It’s similar to how the human brain makes decisions, using different techniques simultaneously to process vast amounts of data.

Deep learning algorithms require more training data than traditional machine learning algorithms (at least millions of tagged examples). However, they do not have a threshold for learning from training data, like traditional machine learning algorithms, such as SVM and NBeep learning classifiers, and continue to improve the more data you feed them.



**Figure 1: a comparison between deep learning and traditional machine learning**

Deep learning algorithms, such as Word2Vec or GloVe, are also used to obtain better vector representations for words and improve the accuracy of classifiers trained with traditional machine learning algorithms.

**2.5.1 Neural Network**

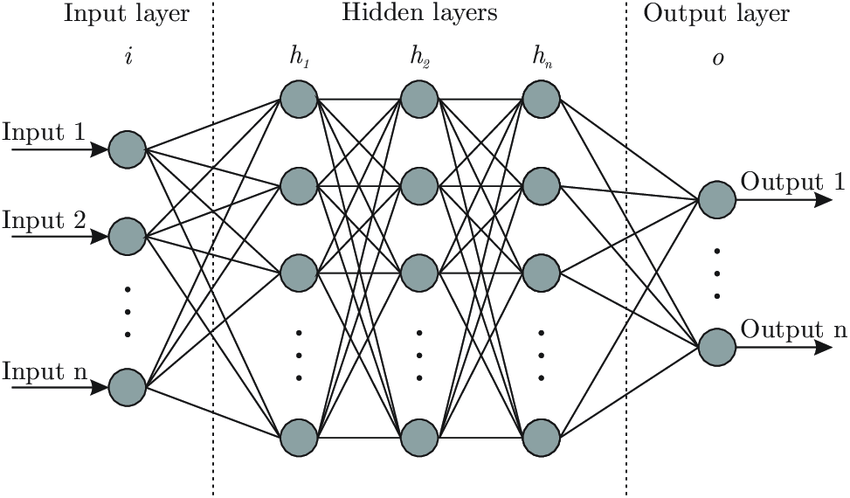
A neural network is a computational mathematical model inspired by brain processes in a natural neural network and used as part of machine learning. These networks usually consist of an input layer, hidden layers, and an output layer.

The **input layer** of a neural network is composed of artificial input neurons and introduces initial data into the system for further processing by subsequent layers of artificial neurons. The input layer begins the workflow for the artificial neural network. The idea is to calculate output using weights [[6]](#_References).

The **hidden layer** is isolated from the external world. Hidden layers are mathematical functions designed to produce an output specific to an intended result. Each hidden layer function is specialized to create a defined output. The hidden layers process all the information between layers.

The **output layer** is responsible for producing the result. The output layer carries the input from the previous layer, performs the calculations using its neurons, and then computes the output.

The neural network model consists of layers, and each layer has its responsibility. Layers process raw data and find patterns and objects hidden from human eyes. Each layer works as an input to the next layer.



**Figure 2: The data flows from the input layer to the Output layer through the Hidden layers**

To train a neural network, there are three sets of data:

**Training data set** Allows the network to understand and recognize the different weights between the nodes.

**Validation data set** Used to tune the data sets.

**Test data set** Used to evaluate the accuracy and margin error.

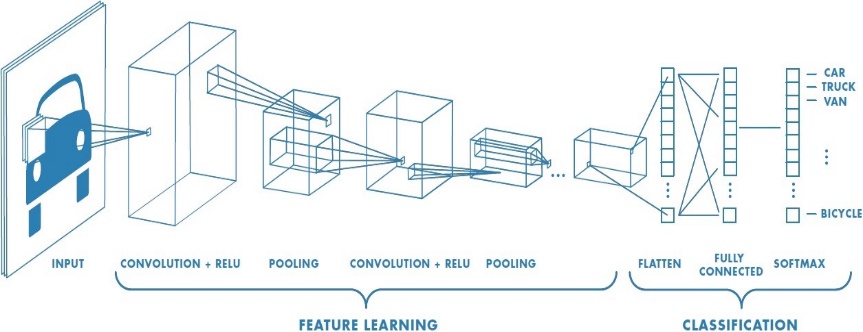
A neural network also contains an **activation function** that defines how the weighted sum of the input is transformed into an output from a node or nodes in a network layer.

### **2.5.1 Convolutional Neural Network**

A Convolutional Neural Network (CNN) is a Deep Learning algorithm that can take in input such as text, audio, or picture and detect a specific feature in the input [[7]](#_References). Each input will pass convolution layers with filters, pooling layers, and fully connected layers, using an activation function to classify an object using probabilistic values (between 0 to 1).

CNN is made up of two main layers:

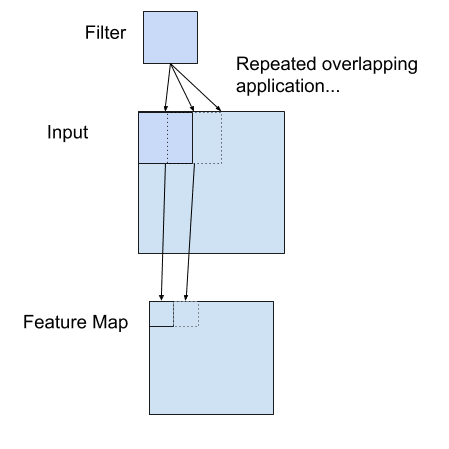
* A convolution layer for obtaining features from the data
* A pooling layer for reducing the size of the feature map



**Figure 3: Neural network with many convolutional layers**

#### **2.5.1.1 The Convolution Layer**

The first layer extracts features from the input. This layer preserves the relationship between pixels and learning features using small input data squares. This mathematical operation convolves the input matrix with a filter (known as the kernel). The result of this operation is called a **Feature map.** The kernel is applied systematically to each input data's overlapping part or filter-sized patch. It computes a new pixel as a weighted sum of the pixels it passed, the new pixels stored in the output matrix.



**Figure 4: Convolve input with kernel**

#### **2.5.1.2 Pooling Layer**

They are mainly responsible for reducing the feature map dimensions and retaining only the critical information. There are different pooling types: max pooling and average pooling.

**Max Pooling** returns the maximum value from the input portion convolved with the kernel.

**Average Pooling**returns the average of all the values from the convolved input.

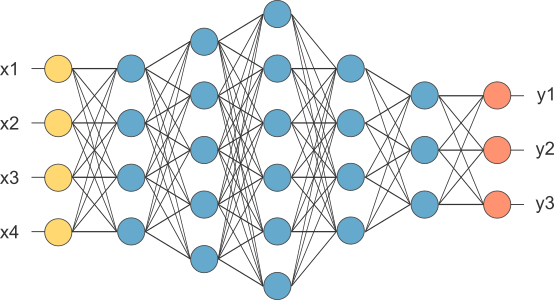


**Figure 5: Max pooling and Average pooling**

#### **2.5.1.3 Fully Connected Layer (FC)**

Responsible for flattening the feature map to a 1-dimensional array. This layer combines features to create a model.

And the last step after the flattening is the **activation function** used to classify the outputs.



**Figure 6: After pooling and flattening, the feature map matrix will be converted to a vector (x1, x2, x3, …)**

#### **2.5.1.4 Activation Function**

A simple function that transforms inputs into outputs with a specific range. This function is added to an artificial neural network to help the network learn complex patterns in the data. These functions introduce nonlinear real-world properties to artificial neural networks.

In a neural network, x is defined as the inputs, is the weights, and passes f(x) as the network's output.

A neural network without activation function will act as a linear regression with limited learning power.

The function:

The function returns 0 if it receives negative input and x for a positive one.

##### **2.5.1.4.1 Softmax Function**

The Softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1.

The input values for the function can be positive, negative, zero, or greater than one. The Softmax function transforms them into values between 0 and 1. So the outputs from the Softmax function can be interrupted as a probability. If the Softmax function receives negative or small input, the output will be a small probability. And if the Softmax function receives positive or large input, the output will be a large probability.

The Softmax can be used for multi-class classification.

Many multi-layer neural networks end in a layer that outputs real-valued scores that are challenging to work with. The Softmax is very useful in these networks because it converts the scores to a normalized probability distribution, which can be displayed to a user or input to other systems. For this reason, it is customary to append a Softmax function as the final layer of the neural network.

The function is:

The values are the elements of the input vector. They are divided by the sum of all the input vector elements. It ensures that all the output values of the function will sum to 1 and each be in the range (0, 1)

The Softmax function is essential in the training of a neural network.

In CNN, the output scores of the last layers (fully connected) are not probabilities.

Therefore after the CNN ends, there is a need to add the Softmax layer to convert the output into a probability.

When the classification is done for a particular class, the desired output values are 0 or 1. In this case, another layer is added to the network for the **loss function** that quantifies how far the network’s output probabilities are from the desired values. The smaller the loss function, the closer the output vector is to the correct class.

Because Softmax is a continuously differentiable function, it is possible to calculate the derivative of the loss function for every weight in the network for every input in the training set.

This property allows us to adjust the network’s weights to reduce the loss function, make the network output closer to the desired values, and improve the network’s accuracy.

##### **2.5.1.4.2 ReLU and Sigmoid** **Function**

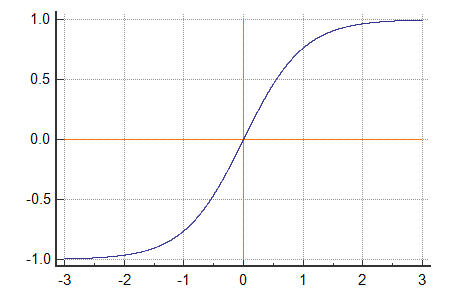
ReLU (Rectified Linear Unit) is the most used activation function in deep learning models.

This function looks and acts like a linear activation function. It is easier to optimize a neural network when its behavior is linear or close to linear.

When using sigmoid functions, whose derivatives have only values from -2 to 2 and are flat elsewhere, the gradient decreases with the increasing number of layers. The gradient shows where the loss function approaches 0. This is called the vanishing gradient problem [[8]](#_References).

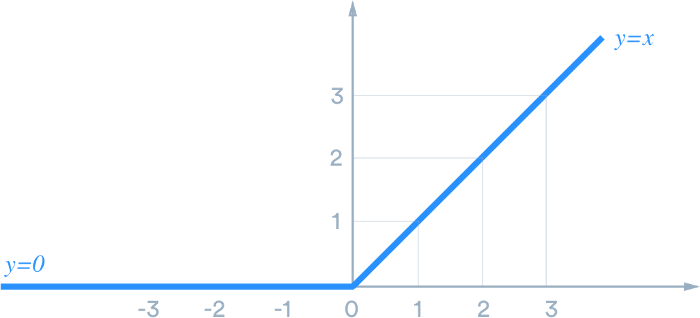


**Figure 7: Sigmoid Function**



**Figure 8: Tanh Function**

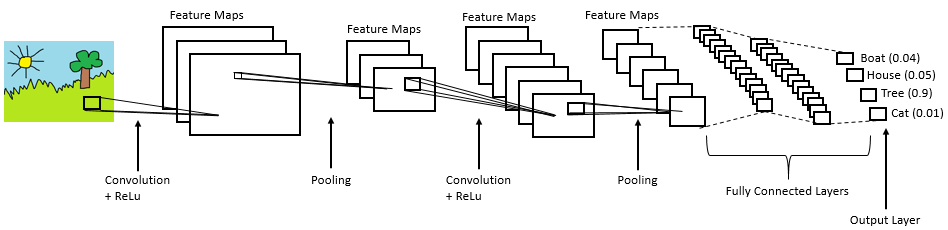
The ReLU function does not have the problem of vanishing gradients because the slope is not saturated when the input becomes large. For this reason, models that use the ReLU activation function converge faster.



**Figure 9: ReLU Graph**

ReLU is a simple function and does not consist of heavy calculations. Therefore, the model requires less time to train or run.

Another property of ReLU is sparsity. This is a desirable property in learning because it can accelerate learning and simplify the model. The ReLU function can generate an actual zero value. Unlike the sigmoid functions that learn to approximate a zero output (learn to create a value close to zero) but not an actual zero value [[9]](#_References).



**Figure 10:** **Complete CNN architecture**

# **Expected Achievements**

This project aims to create a new methodology using short texts in the form of patterns combined with advanced models to achieve better classification results.

The final goal is to comprehend religious text structure more precisely. The idea is to create a model that identifies whether a particular text belongs to the Religious Texts of the Holy Koran or Bible.

The unique feature of this project is that it conducts a new examination of texts in Hebrew and Arabic. Two languages have been less studied until now than English so far.

Therefore, the anticipated answer is a Boolean: either belongs to the religious texts or not.

# **Research / Engineering Process**

## **Process**

### **Description of the research process** In phase A the first step was comprehensively understanding all the related materials and papers/articles.

Then, based on that research, the most appropriate model for this project and its relevant components was selected.

After that, the project phase's order and flow were determined.

Finally, the construction of the algorithm was complete. According to the study performed and the recommendation of the supervisors, the proposed approach most appropriate for analyzing and finding a solution to the problem is described in the article.

In phase B, after choosing the most appropriate model for our purpose, the following steps have done:

* Finding data on the two languages
* Choosing the programming language (python)
* Choosing the related libraries
* Choosing the most appropriate programming environment (Jupyter)
* Determining the algorithm stages
* Understanding how to implement each stage in the algorithm, for example, the hebBert model.

### **The solution**

The solution that this project offers is a new model that can classify text and conducts Authorship Recognition and analysis.

### **Difficulties**

One of the difficulties during the project was finding articles and materials that could be used in the study. Because text classification is often applied for sentiment analysis, working with less-studied languages such as Hebrew and Arabic was another difficulty.

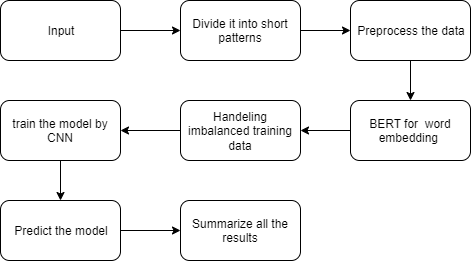
Another was finding books that could be used with the correct permissions for the imposters.

The main difficulty of this project was figuring out how to work with the hebBert model and combine it with the rest of the program.

A technical problem during the implementation stage was finding the right software to run the model. Spyder by anaconda was the first software examination, and it was not found suitable as it took a couple of days to run the model, which slowed the programming progress. Then Google Colab was tested, and it required changing the structure of the code. After modification was done, it didn't work due to continually changing libraries. Finally, Jupyter was selected, the code compiled in a reasonable time (a couple of minutes), there was no need to change libraries constantly, and the overall use was straightforward and user-friendly.

## **Product**

### **Algorithm**



**Figure 11: The flow of the project**

Preprocessing:

* Divide the texts into tokens
* Convert to lowercase
* Remove punctuation marks/Tashkeel, special characters, and digits
* Remove remaining tokens that are not alphabetic
* Remove stop words
* Remove Nan Values

Handling of imbalanced training data:

1. Under-sampling of the majority class
2. Over-sampling of the minority class

Word Embeddings:

BERT (AraBERT, HebBERT)

CNN:

* 3 layers
* Kernels of size 3, 4, 5
* 300 embedding dimension
* Nonlinear ReLU activation function
* Maximum pooling layers of size 1 – down-sample the obtained features of each convolution layer
* Dropping out 30% – overcome the overfitting problem
* Fully-connected layer by Dense with linear and Softmax activation function

The final process is as follows:

1. Get imposters and divide them into short pieces like "tweets" - these texts will be our training data
2. Preprocess the data
3. Word embedding by Bert
4. Balance the data
5. Create the CNN model
6. Train the model
7. Predict the authorship of a new text by the model
8. Analyze the result and effectiveness of the algorithm
9. Summarize the results

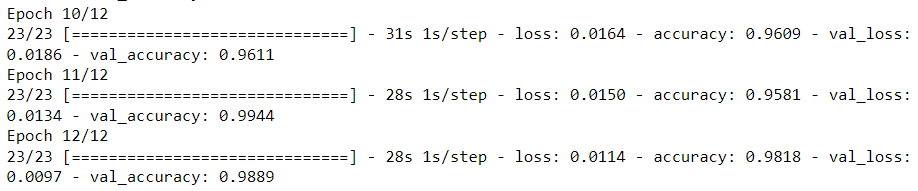
### **Results**

The results for the Bible:  
Our dataset was the Bible book and imposters data, consisting of books written by various known authors in the Hebrew language.

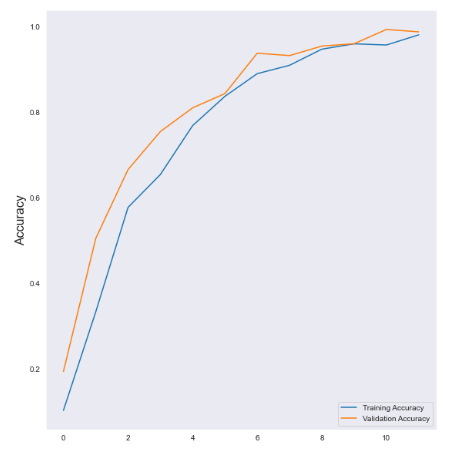
We trained our model with 12 epochs.

Accuracy (a number that generally describes how the model performs across all classes):

As shown in figure 12, you can see that our model had reached training accuracy of 0.986 and 0.944 for validation accuracy, which means that our model performs well because of the high accuracy value.



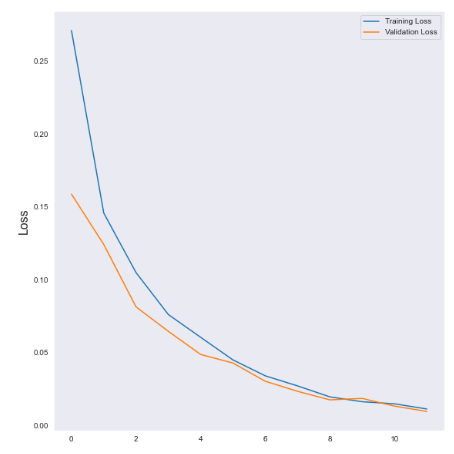
**Figure 12: The 3 final** **epochs of the train**



**Figure 13: Accuracy** **graph**

Loss - a number indicating how poor the model's prediction was on a single example. (If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater):

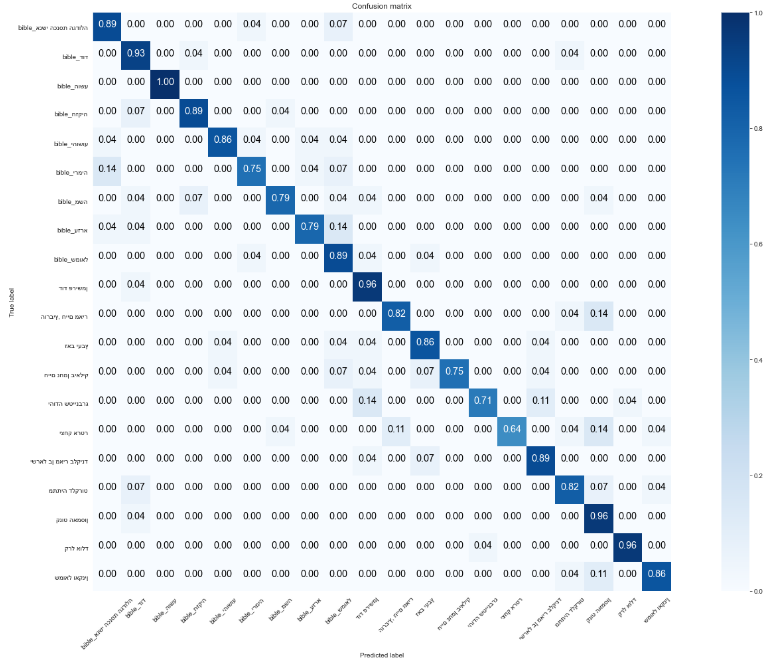
As shown in figure 14, you can see that our model training loss is 0.009 and 0.019 for validation loss, which is a low value, and it means that the model prediction is good.



**Figure 14: Loss graph**

Prediction data results - Confusion matrix (a summary of prediction results on a classification problem):

As shown in figure 15, the values in the matrix are diagonal, which means that the model evaluation succeeded.



**Figure 15: The confusion matrix**

Result:

We took for our prediction a text from the Bible book that wasn't included in the training data. Then we let the model predict this text's label (author).

As we can see, the result of classifying the text is that the text belongs to the Bible. Moreover, our model can recognize which author wrote this Bible part. In this case, the author is "דוד".



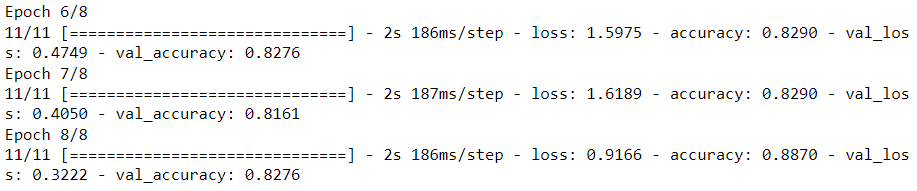
**Figure 16: The final classification**

The results for the Holy Koran:

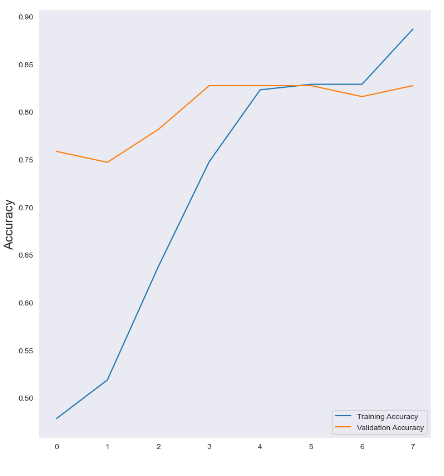
Our dataset was the Holy Koran book and imposters data, consisting of books in the Arabic language written by various known authors.

We trained our model with 8 epochs.

Accuracy (a number that generally describes how the model performs across all classes):  
As shown in figure 16, you can see that our model reached a training accuracy of 0.887 and 0.827 for validation, which means that our model performs well because of the high accuracy value.



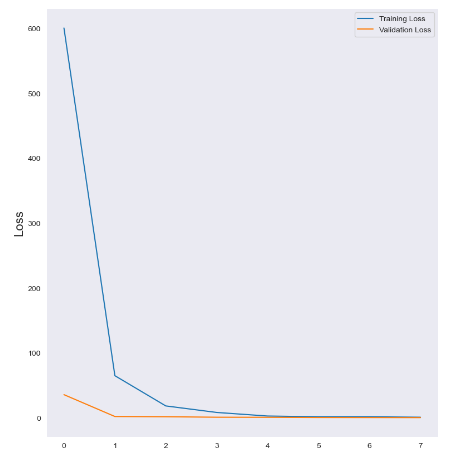
**Figure 17: The 3 final** **epochs of the train**



**Figure 18: Accuracy** **graph**

Loss - a number indicating how poor the model's prediction was on a single example. (If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater):

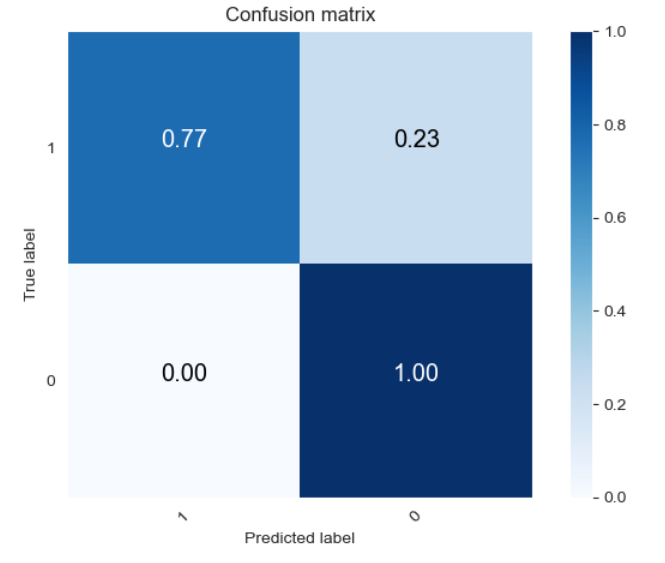
As shown in figure 18, you can see that our model training loss is 0.916 and 0.322 for validation, which is a low value and means that the model prediction is good.



**Figure 19: Loss graph**

Prediction data results - Confusion matrix (a summary of prediction results on a classification problem):

As shown in figure 19, the values in the matrix are diagonal, which means that the model evaluation succeeded.



**Figure 20: The confusion matrix**

Result:

We took for our prediction a text from the Holy Koran that wasn't included in the training data. Then we let the model predict the labels: 0 for the imposters and 1 for the Holy Koran.

As shown in figure 20, the result of classifying the text is that the text belongs to the Holy Koran.

* It is important to note that the final result of the model on the Arabic language includes the prediction of whether the text belongs to the Holy Koran or not, and that's because Muslim people believe that there are no authors of the Holy Koran, so we built the model by considering this.



**Figure 21: The final classification**

# **Evaluation / Verification Plan**

During our programming stage, we tested ways to evaluate our product. For example, the accuracy value helps us check whether the model performs well.

Therefore, we ensured that the accuracy value was as high as possible.

In addition, by plotting the accuracy and loss graphs, we evaluated our model.

Another factor was constructing and plotting the confusion matrix and getting all the values in the diagonal, which indicates the excellent performance.

Finally, we evaluated our model using the "model.evaluate" function.

The following tests were used to evaluate the system's performance

|  |  |
| --- | --- |
| Test Description | Expected Results |
| Check the path of the data folder | Show the labels of the data if correct and error if not |
| Check the preprocess | Remove the unnecessary characters (numbers, symbols, Tashkeel, punctuation) |
| Predict the author of the test file | Predict the correct author of the test file |
| Predict the author of the same test file a couple of times | The same authorship prediction |
| Predict whether the test file belongs to the holy books (Bible or Holy Koran) | The test file belongs to the holy books (Bible or Holy Koran) |
| Check the effect of changing the number of epochs on the training process | If it is too small, then the training won't be good, but if it is too high, then the training will be overfitted |
| Check the effect of changing the threshold size on the training data | If it is too high, then the training data will be too small, but if it is too small, then the gap between the labels will be too big |

# **Conclusions**

In the project's second phase, we experienced building a new model from the ground up. We found problems in our design and corrected them during the programming process. We learned how important time management is and that we always have to consider space for bugs and unexpected challenges.

We have expanded our knowledge of NLP and all the different ways it can be implemented. Also, we learned how to work as a team in the best way possible despite the difficulties along the way.

Lastly, we achieved our goal, and the model meets the expectations.

# **Future work**

The next step in improving this project will be to create a UI so that any user can use this model to classify authorship, not only engineers and programmers.

# **User Guide**

## **8.1 Software Environment**

Jupyter notebook/V code

## **8.2 Running Instructions**

The model requires the following libraries:

* arabert
* gc
* random
* keras
* matplotlib
* nltk
* numpy
* os
* seaborn
* sklearn
* string
* tensorflow
* torch
* transformers

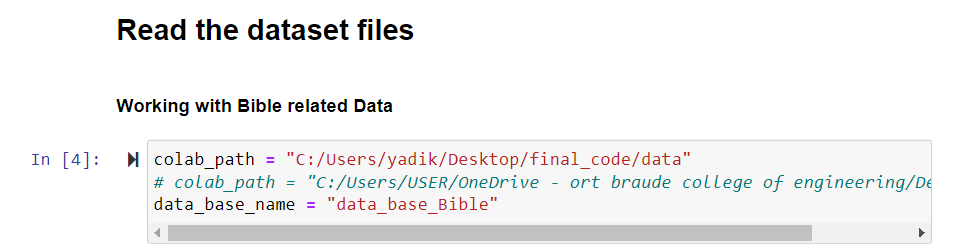
## **8.3 General Description**

The model is based on deep learning and is used to classify text and conduct Authorship Recognition and analysis.

It is flexible and can classify different data sets based on the input in the data folder.

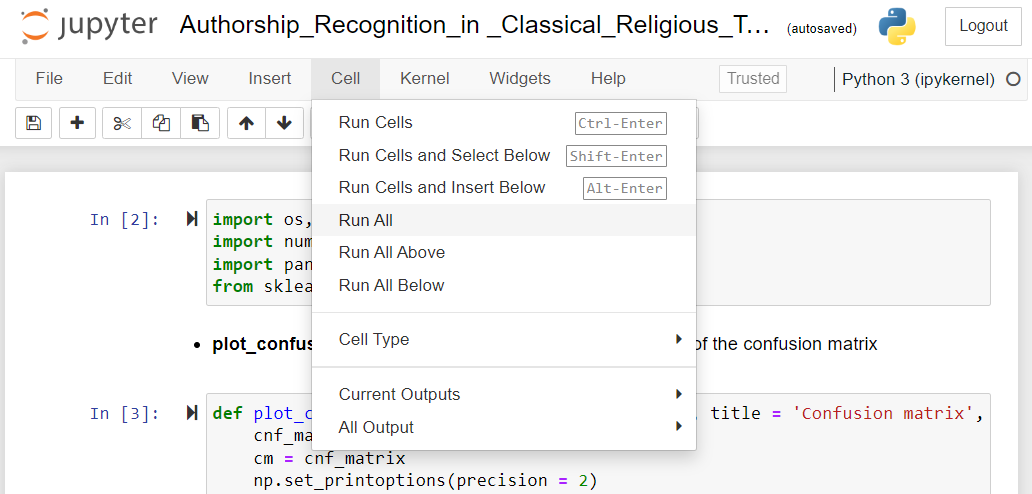
## **User Instructions**

* + 1. Save the data folder in a directory you can easily access on your computer
    2. Open the code on Jupyter
    3. Change the path to the data



**Figure 22: The data location**

* + 1. Run all the cells



**Figure 23: The running button**

* + 1. After the model is finished, the result will be at the end of the page

In conclusion, all you need to run the model is Jupyter and a path to the data folder, then press the "Run All" button.

# **References**

1. [Mathematics | Free Full-Text | A Short-Patterning of the Texts Attributed to Al Ghazali: A “Twitter Look” at the Problem (mdpi.com)](https://www.mdpi.com/2227-7390/8/11/1937)
2. [A Survey of Popular Image and Text analysis Techniques | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/abstract/document/9031023)
3. <https://www.sciencedirect.com/topics/computer-science/authorship-attribution>
4. [Detecting Emergent Intersectional Biases: Contextualized Word Embeddings Contain a Distribution of Human-like Biases | Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society](https://dl.acm.org/doi/abs/10.1145/3461702.3462536)
5. [Deep Learning--based Text Classification: A Comprehensive Review: ACM Computing Surveys: Vol 54, No 3](https://dl.acm.org/doi/abs/10.1145/3439726)
6. [State-of-the-art in artificial neural network applications: A survey - ScienceDirect](https://www.sciencedirect.com/science/article/pii/S2405844018332067)<https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/>
7. [Understanding of a convolutional neural network | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/abstract/document/8308186?casa_token=e3lN3_4reoQAAAAA:hhyzBX21j__1tSSp6rHoXxF-AhtG3mBV6197bL0E3tdwfcBn3gG-uosfHrPyyZZ8R5Q1ivBsyd1jz8M)
8. [A flexible sigmoid function of determinate growth. - Abstract - Europe PMC](https://europepmc.org/article/MED/12547689)
9. [[1803.08375] Deep Learning using Rectified Linear Units (ReLU) (arxiv.org)](https://arxiv.org/abs/1803.08375)