**BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA**
**FACULTY OF MATHEMATICS AND COMPUTER SCIENCE**
**SPECIALIZATION COMPUTER SCIENCE IN ENGLISH**

# DIPLOMA THESIS

# AI-Powered Fake News Detection in Romanian Journalism

**Supervisor**
**Lecturer, PhD. Mihai Andrei**

*Author*
*Luca Răzvan Rătan*

2025

**UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA**
**FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ**
**SPECIALIZAREA INFORMATICĂ ÎN LIMBA ENGLEZĂ**

# LUCRARE DE LICENŢĂ

# Detectarea Știrilor False Bazată pe Inteligență Artificială în Jurnalismul Românesc

**Conducător științific**
**Dr. Mihai Andrei, Lector**

*Absolvent*
*Luca Răzvan Rătan*

2025

ABSTRACT

In the past years, along with the evolution of technology, there has been a significant change in how easily people can access news regarding any topic. While in theory, this should represent a positive impact on the general information and knowledge that people could acquire, the reality showed us that the ones who had the power to spread news chose to spread false information in order to acquire more attention and create arguments in society. Therefore, people have been frequent victims of fake news on social media and on the internet. In the field of artificial intelligence, a solution for combating this problem has been researched for many years, especially for the English language, through machine learning techniques and transformer-based models. For the Romanian language, research is still in its early stages. The aim of this paper is to conduct further research on the detection of fake news in the Romanian language to identify more accurate models for classifying news. The transformer-based models T5 and a distilled version of BERT, pre-trained on the Romanian language, were used in order to make comparisons between the two and see if they would represent good choices for this task, because despite following the same architecture, they follow different types of it. The T5 is based on the encoder-decoder architecture, and the DistilBERT model is based on the encoder-only architecture. In addition, a Convolutional Neural Network (CNN) model was created in order to offer a direct comparison between language models and a classical classification method. The models were evaluated using four main metrics: accuracy, precision, recall, and F1-score. The database used was made public upon the publication of [MMC+24]. It is a multiclass database, and it contains fake and real news from different fields. It was created by gathering the news from the FakeRom dataset and news scraped from the Veridica platform. Four experiments were done using this dataset, each one being done on a different variation of it. For two of them, the dataset was augmented through back translation, using the French and English languages. A merge of the classes representing fake news was also applied to see if it would represent a good solution for this task. The results showed that all models performed really well, with the T5 model taking over with higher accuracy. In order to show the applicability of this study, an Android mobile app was built, in which the user can enter news content in order to get it checked for fakery.

# Contents

# Chapter 1

# Introduction

Nowadays, quick access to news regarding any topic has represented a significant advantage in our society, as it provides people with an easy way to be up to date with everything that happens around them, especially on important topics such as politics, economics, professional life, or health. However, a big downside that came along with this drastic change was the emergence of fake news.

These are pieces of news that are falsely written and offered to the public, intentionally or unintentionally, or news that represents an exaggerated version of the true one, giving its reader a false impression of what happens around them.

Fake news comes in different types. One type is satire or parody. This is when false information is created just for fun, not to trick anyone. Another type is disinformation, which is false information made on purpose to change what people think. Propaganda is similar; it gives misleading information to support certain beliefs or ideas. There is also clickbait, which uses exaggerated headlines to get people to click, but it often isn't accurate. Distorted news shows true facts in the wrong way, changing how people understand things. Fabricated content is completely made-up information with no truth in it. Finally, conspiracy theories tell stories about secret plots that aren't true. Knowing these categories helps us recognize the different kinds of fake news and understand how they can affect us.

As mentioned in [MMC+24], some of the key consequences of fake news are:

- Influencing public opinion and elections
- Lowering the trust in media and institutions
- Creating panic during crises
- Impact on public health
- Affecting social cohesion
- Economic impact

In the country of Romania, there has been a significant increase in this phenomenon, affecting crucial moments like elections or the COVID pandemic, where a large number of people were tricked into believing false claims about some subjects. Therefore, like in other countries, there was a need to provide a solution to fight this problem.

Over the years, solutions like fact-checking have been provided by some publications or organizations. For example, in Romania, publications like Veridica or Factual check news for fakery and post them online, in order to inform people about them. However, this whole process of annotating news takes a long time and is tedious, consisting of spending a lot of time checking the information provided in a piece of news in order to see if it is false and checking the sources that provided that piece of information.

Considering the need for effective solutions, the purpose of this thesis is to continue the research done for automatic fake news detection for the Romanian language, using language models and a classical classification method, namely, convolutional neural networks (CNN).

The dataset that was used is a result of the research study from [MMC+24]. It is a multiclass dataset containing the following classes: "fake_news", "real_news", "propaganda", "misinformation", and "satire". Multiple experiments were tested on the models with different versions of this dataset, two of them being done on an augmented version of it, which resulted from a back translation process that was done through the English and French languages. In three of the experiments, the classification was a binary one, because of a merge that was done using the classes that represented fake news, and the removal of the "satire" class.

The original contribution consists of the usage of the T5 and DistilBERT models, for which, at the time of writing this paper, there hasn't been any research found on fake news detection in the Romanian language that used them. Also, the experiments that were tried on the used dataset, like applying back translation and merging the fake news classes into one, represent original ideas that were not tried on this dataset. The achieved results have proven to be quite good, with the highest results achieved overall by the T5 model. The DistilBERT and the CNN model also came close, offering some interesting and different results across all the experiments.

Speaking of the content of this thesis, first, the paper offers a state-of-the-art presentation, in which the research that has been done so far is presented, with different methods of implementing this task. Following this chapter, a theoretical presentation of all the used models will be offered, regarding their architectures and the differences between them. After this, the implementation processes are presented, starting with the implementation of the fake news detection process, presenting the preparation of the dataset, and the training of the models. A chapter will also be

dedicated to the results, where comparisons with the results from other papers will be made, together with all the results gathered from the different experiments that were tried during this research. Moreover, a chapter is dedicated for the app that was built in order to show the applicability of this research, talking about the requirements, the specifications, and also about the used architecture, testing and technologies, followed by the the final chapter that will consist of the final conclusion and future work that could improve this area of research.

## 1.1 Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author used ChatGPT and Gemini in order to enhance the readability and explanation process of some of the concepts presented in the paper. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the thesis.

# Chapter 2

# State of the Art

Over the years, fake news detection became a main interest of research in the field of natural language processing (NLP), exploring a variety of methods, such as language models or machine learning methods, for solving this task.

An introduction to fake news detection and its applicability to the Romanian language is made in [BRD20]. The paper emphasized the importance of research on fake news detection for the Romanian language, taking into consideration the fact that the citizens of Romania tend to trust unfiltered online content. Several research studies and projects that were centered around the English language were presented. The impact that satirical news has on how we perceive fake news was discussed, and some interesting approaches were presented, like enhancing fake news detection through news clustering based on topic, or taking a first step towards efficient fake news identification by following a different perspective, namely stance detection, that is, building machine learning models that are capable of seeing if the headline of some news matches its content.

[SSW+17] dives deeper into the subject and offers an in-depth analysis of fake news detection. It focuses more on fake news from social media, as they are more accessible to society and are disseminated more easily. Also, there is no actual validation process for them, as anyone can write something, post it, and then a large number of people read that post, some of them believing it and sharing it further. The study highlights the importance of feature extraction, taking into account not only features related to an actual post, like headline, text, or image, but also external features, such as the account from which that post was made, where the accent is put on social bots, or the reactions people are having towards the post. The paper also presents some methods and models for fake news detection.

The study presented in [UPR+22] had its main focus on the detection of fake news using Large Language Models. The researchers solely used the FakeRom dataset for the detection. To provide more background, the FakeRom dataset contains a total of 14,000 articles that are mainly focused on the subject of COVID-19.

Out of all of these, the researchers used approximately 1,200 articles, which could be used for this kind of detection. The articles were divided into six categories, and later grouped into three main ones: true news (real + plausible), fake news (propaganda + fabricated), and imaginary news (satirical + fictional). In order to have something for comparison, some machine learning models were used: Support Vector Machines (SVM), Random Forest, XGBoost, and Naive-Bayes. For the large language model, RoBERT was mainly used, a Bert version that was specifically trained on the Romanian language. In addition to the base model, convolutional neural networks (CNN) were put on top of the model in order to enhance the classification, and another experiment was done by training the models with multi-task learning (MTL) networks. Also, for the machine learning models, two methods were tested: one by using bag of words for the training, and one by using the embedding of the words. The highest accuracy was achieved by the base model of RoBERT, 75%, by using both CNNs on top and MTL networks. For the machine learning models, the highest accuracy achieved was 67%, with the SVM classifier, using word embeddings.

[MMC+24] was the main source of inspiration for this paper, also being partly a continuation for it. The FakeRom dataset was combined with news, both fake and real, that were scraped from the Veridica platform, and it was also augmented in order to solve the large class imbalance that was present, which in this case was achieved by creating different versions of the same text by using synonym replacement. After this process, the final dataset achieved a data size of approximately 5100 news articles. Once again, some machine learning models were used for the classification, such as Naive Bayes, SVM, and Logistic Regression, along with two transformer-based models: Bert and RoBERTa Large. Higher accuracies were achieved in this paper, showing the efficiency of machine learning and transformer-based models for fake news detection: 94.6% was the highest accuracy achieved by a machine learning model, namely the SVM model, while regarding the transformers, both achieved a fairly equal accuracy, 96.5% for the Bert model, and 96.2% for RoBERTa Large. In order to test different scenarios with regard to training efficiency, the models were trained and tested on different versions of the dataset, meaning taking the FakeRom and Veridica datasets separately or together for training or testing.

In [BTMR22], a much larger dataset was used, consisting of approximately 38,000 real and fake news. Here, new classification models were introduced and tested on besides classical machine learning methods and language models, namely deep learning models such as Long Short-Term Memory (LSTM) networks, a type of recurrent neural network, Convolutional Neural Networks (CNN), and Gated Recurrent Units (GRU). As opposed to the other two mentioned papers, the highest accuracy here was achieved by the deep learning models and classical models, 97.5%

being obtained by the Naive Bayes model, and all the deep learning models having obtained more than 97%, with the CNN achieving an accuracy of 98.1%. A score of 91.9% was achieved by RoBERT-small, which was the highest for the language models. Some interesting experiments that were applied on the test dataset were to check the irony and type of sentiment levels in the news. The experiments concluded that most news, both fake and real, were neutral, with the real ones presenting a quite significant percentage of positive sentiments, 15.18%, while on the irony side, most news were non-ironic, with 24.05% of the fake news being ironic.

A very interesting approach on this subject was explored in [BD23], where the main problem that was discussed and tried to be solved was the relatively small amount of quality data that is available for fake news detection in the Romanian language. More specifically, finding officially and correctly annotated news in Romanian, and in large amounts, is difficult. Therefore, based on past research, the paper presented a method of data augmentation through back translation (BT) for Romanian news. This involved using more approaches in order to find a method that was as efficient as possible. The BT process was done using mBART, a multilingual language model that effectively processes different languages and is particularly useful for machine translation tasks, and the well-known Google Translate. In addition, the BT was tried on two different languages: French and English. The results showed the effectiveness of BT. On four different machine learning models that were trained and tested on the augmented datasets and the original dataset that was collected from a fact-checking website for Romanian news, the accuracy was better on all four models when trained on the augmented datasets, with the highest one being achieved with the Random Forest classifier, 80.75% on the augmented data compared to 73.71% on the original one, when applying BT using mBART, using French as the target language.

Another paper that researched this topic, [DFG23], compared the detection accuracy on datasets that had significantly different sizes. As it was expected, the accuracies of the models trained on the datasets differed significantly. The common models that were trained on both of them were some of the classical machine learning models: SVM, Linear Regression (LR), and Random Forest, and one other interesting choice: Stochastic Gradient Descent (SGD) classifier. All these models scored an accuracy of over 80% on the larger dataset while having under 65% on the smaller one. It is worth mentioning, however, that the smaller dataset was preprocessed such that the only classes were fake and real news, as the raw form of it contained items that had three other labels: partially true, partially false, and truncated articles. A deep learning model was used for the larger dataset and also had the best accuracy overall, 93.50%. An interesting experiment for the classification was attempted by using Latent Dirichlet Allocation (LDA) models. This implied

representing each article by its topic distribution and using it as feature vectors that were afterward fed to the classification models. It was noticed that increasing the number of topics also increased the accuracy, but only until a certain point, with 65 topics representing the highest value, 69.70%, accuracy getting lower after getting higher in numbers.

[SK23] presented an innovative method of detection for the English language. The authors created a multimodal method of detecting fake news, by also taking into consideration images that are posted or published along with the text of the article itself, as research shows that news articles that have images are shared 11 times compared to those that only contain text. The DeBERTa model, which is mainly based on Bert and RoBERTa models, was used for the textual features, while ConvNeXT was used for the image analysis, which is a convolutional model that is trained on the large dataset of images, ImageNet. After this, the outputs were concatenated in the last layer in order to produce the classification result. After being trained and tested on multiple datasets with news in the English language, three in number, the model provided a high accuracy, the highest compared to other models tested on the same datasets, 91.2%, 91.3%, and 90.2%.

Even though important research has been done on this kind of detection in the Romanian language, it is still in its early stages. The purpose of this paper is to create a continuation and offer more information regarding methods for detecting fake news. As mentioned, the T5 model and a distilled version of BERT are used in order to see how they compare with models used in other research and if they can represent a better choice for this NLP task. In addition, a classical classification model, namely a CNN, is used in order to see how it performs compared to the language models, as research has shown that it performs really well on this task.

The T5 model follows an encoder-decoder transformer architecture. Because of its structure, the T5 model is considered a sequence-to-sequence model, making it best for tasks like language translation or question answering, but not necessarily for classification tasks like the one targeted in this paper. However, it still is a good choice for research, as mentioned in [MMC+24], where it was mentioned as a choice for future research.

On the other hand, the distilled version of BERT [ACC+21] represents from the start a good solution, as its teacher, BERT, was already shown to be efficient in some of the mentioned research, achieving high and better levels of accuracy compared to some of the machine learning models. However, BERT is a very large language model, with tens of millions of parameters more than DistilBERT, and in this situation, its distilled version might prove to be very useful, as it is known to be more efficient.

# Chapter 3

# Models Characterization

Providing a theoretical background on the three models is necessary in order to offer a better idea about how they work, considering the fact that each of them is different from one another, having their unique particularities.

## 3.1 About Large Language Models

In the past years, Large Language Models (LLMs) have been a highly relevant subject in the field of computer science and even outside the field, with people from different backgrounds and of different ages using them for day-to-day activities or for important professional tasks, and with researchers trying to make more and more advancements in this particular branch, achieving incredible results as time passed by. The breakthrough of their research, which made them so popular and so efficient, was the introduction of the Transformer architecture (Figure 3.1), which has as its main attraction the presence of a self-attention layer. This new architecture was presented in [VSP+17], and unlike previous models that relied on recurrent neural networks (RNNs) or convolutional networks, the Transformer exclusively uses attention mechanisms, specifically multi-head self-attention, to model relationships across sequences.

The Transformer architecture consists of stacked encoder-decoder structures. Each encoder and decoder includes multiple layers with two primary components: multi-head self-attention and feed-forward neural networks. This architecture effectively addresses the inherent limitations of recurrent models, which are sequential and thus difficult to parallelize, and convolutional models, which struggle with capturing long-range dependencies.

The innovative "scaled dot-product attention" mechanism calculates attention weights, allowing the model to focus dynamically on relevant parts of the input sequences. Multi-head attention further enhances this by enabling the model to si-

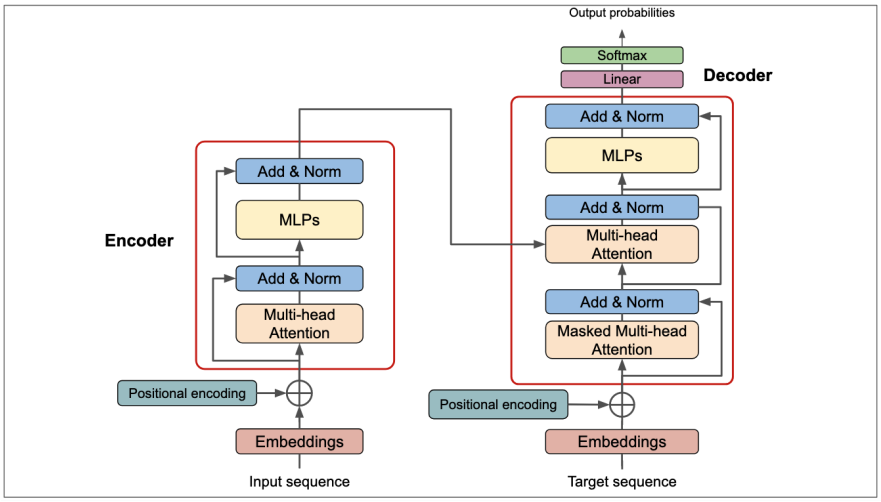



Figure 3.1: The Transformer Architecture (Source: [Nya]

multaneously consider information from different representation subspaces, greatly improving performance and computational efficiency.

Experimental results demonstrate the Transformer's superiority in translation tasks, setting new benchmarks in performance. For instance, on a well-known task in the research field for translation, the WMT 2014, for English to German translations, the Transformer achieved significantly better results than previous state-of-the-art methods, achieving better BLEU scores with substantially lower training costs.

The Transformer architecture has since profoundly influenced natural language processing, becoming the foundation for subsequent major developments such as BERT and GPT models.

## 3.2 The DistilBERT model

The DistilBERT model follows the encoder-only transformer architecture, just like the base, larger model, BERT. What this means is that through its encoder, the model "encodes" the input text that it receives into contextualized vector representations, frequently referred to as embeddings, which can be fed to a classification "head", represented by some neural network layers, in order to get the task-specific result.

The model was acquired through the method of Knowledge Distillation (KD). A comprehensive explanation is provided in [XLT+24]. Knowledge Distillation (KD) is a technique in artificial intelligence (AI) and deep learning (DL) that focuses on transferring knowledge from a large, complex model (teacher) to a smaller, more efficient model (student). This method is essential for reducing the computational needs and resource limitations associated with deploying large-scale models in real-world scenarios.

Traditionally, prior to the emergence of Large Language Models (LLMs), KD methods aimed at transferring insights from intricate and often unwieldy neural networks to more streamlined and effective models. This initiative was primarily driven by the need to deploy machine learning models in settings with restricted resources, like mobile devices or edge computing systems, where computational power and memory capacity are both limited. The earlier approaches primarily concentrated on improvised neural architecture selection and training goals designed for individual tasks.

The rise of LLMs has transformed the field of KD. The emphasis has moved from simple architecture compression to the extraction and transfer of knowledge. This shift is primarily attributed to the vast and profound knowledge contained within LLMs such as GPT-4 and Gemini, as well as the challenges of compressing them using conventional techniques like pruning or quantization due to their non-transparent parameters. The current goal of KD utilizing LLMs is to draw out the specific knowledge these models hold.

Distillation provides improved capabilities for smaller models, enabling open-source models to significantly enhance their performance by learning from advanced proprietary models. This process is comparable to a highly skilled teacher transferring knowledge to their student. It also serves an essential function in compressing large language models, increasing their efficiency without a substantial reduction in performance. This approach reduces computational demands and promotes the environmental sustainability of AI operations. However, there are some drawbacks that must be considered, such as reliance on the quality of the teacher model and the initial seed, catastrophic forgetting, and ethical issues, including data privacy and security concerns, as utilizing proprietary large language models often involves transmitting sensitive information to external servers, which raises questions about data privacy and security, particularly regarding confidential data.

The DistilBERT model for the Romanian language was presented in [ACC+21], a paper that addresses the challenge of deploying large-scale pre-trained language models in computationally limited environments, specifically for low-resource languages such as Romanian. The authors introduce three distilled models: DistilBERT-base-ro, Distil-RoBERT-base, and DistilMulti-BERT-base-ro. These models are derived from Romanian BERT variants through KD.

The models are trained using Romanian language corpora, applying careful preprocessing to remove noisy and irrelevant data, ensuring high-quality training sets. The distilled versions significantly reduce model size by approximately 35% compared to their teacher models, achieving twice the inference speed on GPUs.

Extensive evaluation across five Romanian NLP tasks, such as named entity recognition, part-of-speech tagging, dialect identification, sentiment analysis, and

semantic textual similarity, demonstrates that the distilled models maintain performance comparable to their larger counterparts. Notably, in certain tasks, the distilled models even outperform the original, larger models.

To quantify the fidelity of the knowledge transferred, the authors introduce novel evaluation metrics such as regression loyalty, alongside traditional metrics like label and probability loyalty. These analyses confirm that the distilled models' predictions closely align with their respective teachers.

The distilled Romanian BERT models, along with the training and evaluation scripts, have been made publicly available, significantly enhancing the resources for Romanian NLP tasks and providing a valuable foundation for further optimization and deployment in resource-constrained environments. The model used in this paper can be found on HuggingFace (source: `https://huggingface.co/racai/distilbert-base-romanian-cased`).

For fake news detection in the Romanian language, at the moment of writing this thesis, no usages of this model were found. However, the larger models from the same "family" were used, like the BERT or RoBERTa models, achieving in some articles accuracies of over 90%.

## 3.3 The T5 model

Unlike encoder-only architectures such as BERT, which focus solely on producing contextual embeddings for the input tokens, the T5 follows the full encoder-decoder architecture, utilizing its encoder to generate contextual representations and then leverage its decoder to autoregressively produce target sequences, meaning that it generates each token in the output sequence one at a time, in order, where each new token is predicted based on all previously generated tokens.

The T5 model was first introduced in [RSR+19]. The authors propose an innovative approach that reformulates various NLP tasks—including translation, question answering, summarization, and classification—as text-to-text tasks, allowing a single Transformer-based model to handle diverse language processing challenges seamlessly.

The model was trained on a new, large-scale dataset named the "Colossal Clean Crawled Corpus" (C4). This dataset is constructed by filtering and cleaning text from Common Crawl to ensure quality and relevance. The training involves a self-supervised denoising objective, where the model learns to reconstruct the original text from deliberately corrupted inputs.

A comprehensive experimental analysis explores several aspects of transfer learning, such as pre-training objectives, architectural variations, the scale of data and model size, and different fine-tuning approaches. By systematically scaling up both

model size (up to 11 billion parameters) and the amount of training data, the authors achieve state-of-the-art performance across numerous NLP benchmarks, including GLUE, SuperGLUE, CNN/Daily Mail summarization, SQuAD question answering, and WMT machine translation tasks.

The paper significantly contributed to NLP research by demonstrating the effectiveness of a unified text-to-text paradigm, establishing benchmarks for future studies, and providing publicly accessible resources like the C4 dataset, pre-trained models, and experimental code.

Regarding the task of fake news detection, no research was found that used this model to perform this task in the Romanian language. The model is usually known to perform best on text-to-text specific tasks, like language translation of text summarization. For this paper, the base version of the T5 was used, from the HuggingFace website (source: `https://huggingface.co/google-t5/t5-base`). There are three versions in total, small and large being the others. The base version was chosen as a midpoint between efficiency, accuracy, and memory.

The Table 3.1 below shows the summarized key differences of the model, regarding the architecture type, origin model, model size, and input/output format

| **Feature** | **DistilBERT** | **T5** |
|---|---|---|
| Architecture Type | Encoder-only Transformer | Encoder-decoder Transformer |
| Origin Model | Distilled from BERT | Original model architecture |
| Model Size | 81 million parameters | 220 million parameters |
| Input/Output Format | Input text → embeddings/classification | Text input → Text output (text-to-text format) |

Table 3.1: Comparison between DistilBERT and T5 architectures.

## 3.4 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) are a type of deep neural network that revolutionized areas such as computer vision and have also been applied in text-oriented tasks. As presented in [ON15], they are mainly used for solving difficult pattern recognition problems based on images. By architecture, convolutional layers are capable of automatically learning hierarchical data features, which makes them very useful for classification purposes, such as text classification. Unlike regular ANNs, they are suitable for applying to data with a known topology, such as two-dimensional images or, if talking about text, one-dimensional sequences.

A usual CNN architecture is formed out of various layers, each having a specific function in processing data. These layers include convolutional layers, pooling layers, and fully connected layers. The network's capacity to learn and generalize is determined by how is the setup of these layers and how they are arranged.

Convolutional layers are important components of CNN architecture. They utilize a collection of small but significant filters (also known as kernels) on the input data. A filter consists of a matrix of weights that is applied to the input, executing convolution operations. The outcome of this operation is an activation map, which emphasizes specific characteristics of the input data. For instance, when it comes to images, a filter may identify edges, textures, or corners. When using text applications that involve representations, like word embeddings, a filter has the ability to recognize n-grams, certain word patterns, and even syntactic structures that indicate a unique writing style.

After a convolution layer is applied in deep learning models or neural networks, it is common to include a pooling layer. This specific layer aims to reduce the complexity of the data representation step by step, thereby decreasing both the number of parameters and the computational load of the model. In pooling layers, two popular methods used are max pooling and general pooling (which includes average pooling). Max pooling selects the highest value within a specified area of the feature map. This method of downsampling aids in achieving translation invariance by making the network less sensitive to the positioning of features in the input data.

Following layers of convolution and pooling operations, the feature maps undergo conversion into a one-dimensional vector, which is then connected to one or more fully connected layers resembling those found in conventional artificial neural networks (ANNs). These layers are designed to produce class scores based on the activations for classification tasks. The ultimate connected layer typically comprises neurons for the number of classes and employs an activation function, like softmax, to determine the probabilities associated with each class membership. The combination of extracting features (using convolution and pooling techniques) together with grasping connections (via the fully connected layers) enhances the effectiveness of CNN models in classification tasks.

When it comes to fake news detection, CNNs have been used frequently in this type of research, achieving high results and representing a good choice for this type of task.

# Chapter 4

# Implementation

## 4.1 Training Requirements

The language models T5 and DistilBERT, together with a CNN model, have been fine-tuned and trained, respectively, for the specific task of fake news detection. That means that for a specific piece of text from some news given to them, their output will represent a ”fake” or ”real” label regarding the veracity of the piece of news. An experiment was also done on a variation of the original dataset, which is a multilabel one, containing five classes: ”real_news”, ”fake_news”, ”propaganda”, ”satire”, and ”misinformation”. The models were trained on a dataset created from real (non-artificial) news (fake and real) and back-translated versions of it, from English and French; this dataset version represents an original contribution. There were also other variations of it that the models were trained on. More details will be provided in Chapter 5. The requirements of all the training processes of the models include:

- Ensure the balance of the fake and real classes, and preprocess the data before training in order to remove any noise that would harm the training process
- As an addition for the DistilBERT model, find the best configuration and structure for a classification head
- For the T5 model, insert a prefix for each text from the dataset
- Create a structure for the CNN model
- Split the data for training, validation, and testing
- Ensure the best options for the training parameters in order to achieve good results: number of epochs, batch size, learning rate, etc

- In order to provide a good evaluation, use evaluation metrics like accuracy, recall, precision, and F1 score
- Save the final models with the best results for the possibility of using them in the app

## 4.2 Dataset

### 4.2.1 Dataset Collection

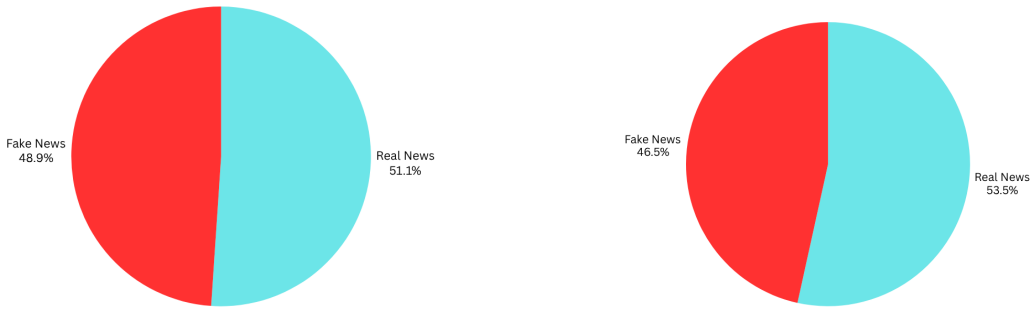
The dataset that was used to conduct this research was created and introduced in [MMC+24], and was also posted publicly later (source: `https://huggingface.co/datasets/mihalca/Fakerom_updated_original`). It contains data from the FakeRom dataset, which is another public dataset containing news in the Romanian language, having subjects mainly from the COVID pandemic, combined with scraped data from the Veridica website, which contains news also written originally in the Romanian language. Veridica is a news publication that also performs fact-checking. In total, the dataset consists of approximately 2200 rows of news, each having two fields: "content" and "tag". The "content" field contains the actual text content of the pieces of news, and the "tag" field contains their label, which can have a value from five total ones: "real_news", "fake_news", "propaganda", "misinformation", and "satire". Most of the texts have up to 550 words, while a couple of hundred have over 550 words, approximately 200 out of the 2200.

### 4.2.2 Dataset Preparation

Having a lot of class imbalance, with almost 50% of the news being real news, 20% having the tag "fake_news", and the other 30% being divided upon the other types, with satires consisting of only 8%, a concatenation was applied for the news having the following types: "fake_news", "propaganda", and "misinformation", all of them being labeled with the "fake_news" tag. This was done in order to have a more balanced dataset, taking into consideration the fact that the pieces of news consisting of propaganda and misinformation are actually subclasses of fake news. The pieces of news classified as satire were completely removed, as they were in an insignificant number, and concatenating them with the fake news could have been a bad choice, considering that satires are not really fake news, as they represent intentionally written fake news with in a sarcastic manner, with the sole purpose of creating amusement for the reader.

The final dataset consisted of a total of approximately 1800 pieces of news, labeled with the "fake_news" and "real_news" tags. The texts that had over 550 words

were removed. This was done with the purpose of having texts as short as possible, without shrinking the dataset too much, in order to have enough for training and testing. The maximum number of tokens the language models were pre-trained on, which is 512, was taken into consideration for this. Even though the token limit for the language models is 512 tokens, a maximum of 550 words still represents a valid length for the texts, taking into consideration the preprocessing step for the dataset that shortens the texts through stop word removal. Even though there were some texts that still had a larger number of tokens than 512, most of the information could have been kept through the truncation option that the language models have, which simply discards the excess tokens that go over the limit. The resulting dataset consisted of 974 real news and 848 fake news, compared to 1032 and 989 in the same order at the beginning, before removing the texts over 550 words. Even though the resulting dataset presented a bit more imbalance in the classes, it was still a fairly equal division. The balances of the datasets before and after the removal of news are presented in Figure 4.1. At that point, the main problem was represented by the rather small amount of data, so data augmentation was the next process that followed. The final dataset consisted of 5463 news, out of which 2541 were fake, and 2922 were real. When it comes to data consisting of text, augmentation can be done in multiple ways. As it showed excellent results when used on natural language processing tasks, like in [BD23], on fake news detection, back translation was used in this research to obtain a larger dataset.




(a) Balance before removal (b) Balance after removal

Figure 4.1: Data balance before and after news removal

Back translation consists of obtaining a new version of a piece of text by translating it into a target language, and then back to the original language. Through this mechanism, some words or expressions from the original text will be replaced with new ones. To implement this process, the mBART and Helsinki language models were used to translate the texts to French and English, and then back to Romanian.

Initially, only mBART was chosen for both languages, but for the French back

translation, some issues were encountered. The translations to French and back to Romanian were faulty and contained English words. An example can be seen below in Table 4.1. As can be seen, even though the translation seems to start well, at some point the words are written in English, and they still remain in English even after the Romanian translation.

| **Original text** | **French translation** | **Romanian translation** |
|---|---|---|
| Atacul iudeilor asupra libertății de exprimare continuă, deoarece legislatorii au elaborat un proiect de lege care să facă ilegală distribuirea de pliante în California. AB 3024, propus de către deputatul Chris Ward, D-San Diego, ca proiect de lege pentru „a reduce gunoiul de ură", ar extinde definiția statului de „intimidare prin amenințare cu violența" din codul civil al statului. | L'attaque des voeux sur la liberté d'expression se poursuit, puisque les législateurs ont élaboré un projet de loi qui ferait de la distribution de pliants en Californie illicite. AB 3024, proposed by M. Chris Ward, D-San Diego, as a projet de loi pour "reduire le fardeau de l'hostilité", would extend the state's definition of "intimidation by threat of violence" from the state's civil code. | Atacul voilor asupra libertatii de exprimare continua, in vreme ce legislatorii au elaborat un proiect de lege care ar face distribuirea de pliante in California ilegala. AB 3024, proposed by M. Chris Ward, D-San Diego, as a bill to "reduce the burden of hostility", would extend the state's definition of "intimidation by threat of violence" from the state's civil code. |

Table 4.1: mBART faulty back translation through French

For this reason, mBART was kept for the English Back Translation, and the Helsinki-NLP model was used for the French Back Translation. This way, we could also ensure some variety in the translations, being done through different models. Below, in Tables 4.2 and 4.3, examples with the results of the Back Translation processes can be seen.

| **Original text** | **Romanian translation from English** |
|---|---|
| Atacul iudeilor asupra libertății de exprimare continuă, deoarece legislatorii au elaborat un proiect de lege care să facă ilegală distribuirea de pliante în California. AB 3024, propus de către deputatul Chris Ward, D-San Diego, ca proiect de lege pentru „a reduce gunoiul de ură", ar extinde definiția statului de „intimidare prin amenințare cu violența" din codul civil al statului. | Atacul evreilor asupra libertăţii de exprimare continuă, întrucât legiuitorii au elaborat un proiect de lege care ar face distribuirea de blană în California ilegală. AB 3024, propus de deputatul european Chris Ward, D-San Diego, ca proiect de lege pentru "reducerea gunoiului de ură", ar extinde definiţia statului de "ameninţare prin ameninţare cu violenţă" din codul civil al statului. |

Table 4.2: mBART English Back Translation

| Original text | Romanian translation from French |
|---|---|
| Atacul iudeilor asupra libertății de exprimare continuă, deoarece legislatorii au elaborat un proiect de lege care să facă ilegală distribuirea de pliante în California. AB 3024, propus de către deputatul Chris Ward, D-San Diego, ca proiect de lege pentru „a reduce gunoiul de ură", ar extinde definiția statului de „intimidare prin amenințare cu violența" din codul civil al statului. | Atacul evreilor împotriva libertăţii de exprimare continuă, deoarece legislatorii au elaborat un proiect de lege pentru a face ilegală distribuirea fluturaşilor în California. AB 3024, propus de deputatul Chris Ward, D-San Diego, ca proiect de lege de reducere a deșeurilor de ură, ar extinde definiția statului de intimidare prin amenințarea de violență în Codul civil al statului. |

Table 4.3: Helsinki-NLP French Back Translation

Following this dataset augmentation, a preprocessing step was applied to all the texts. The following changes were applied:

- Stop words removal
- Digits removal
- URLs removal
- Punctuation removal
- Conversion to lower case
- Lemmatization

Some rows that presented errors in the texts, like empty texts, were removed in order to preserve the correctness of the dataset.

In Table 4.4 it can be seen how does the final preprocessed version of a piece of text looks like.

| Original text | Preprocessed text |
|---|---|
| Atacul iudeilor asupra libertății de exprimare continuă, deoarece legislatorii au elaborat un proiect de lege care să facă ilegală distribuirea de pliante în California. AB 3024, propus de către deputatul Chris Ward, D-San Diego, ca proiect de lege pentru „a reduce gunoiul de ură", ar extinde definiția statului de „intimidare prin amenințare cu violența" din codul civil al statului. | atac iudeilor libertate exprimare continuu legislator elabora proiect lege face ilegal distribuire pliant california ab propune deputat chris ward d-san diego proiect lege reduce gunoi ură extinde definiție stat intimidare amenințare violență cod civil stat |

Table 4.4: Comparison between original and preprocessed piece of text

## 4.3 Detection Models

The main detection models that were used were the Large Language Models T5 and DistilBERT, for the Romanian language. To provide another form of comparison, a CNN model was also trained on the dataset to see how the performances of LLMs compare to the performance of a simpler model. In total, four experiments were done on the models, the first three of them being done on different variations of the original dataset, with the final experiment being done on a multiclass dataset, having as source the same paper that provided the original dataset, [MMC+24]. The final experiment had the sole purpose of making a direct comparison of the results, as it will be mentioned in Chapter 5.

### 4.3.1 The Distilled Version of BERT

For the DistilBERT model, two methods of fine-tuning were tried.

The first one involved the Parameter-Efficient Fine-Tuning (PEFT) technique, using Low-Rank Adaption (LoRA). PEFT focuses on modifying a minimal portion of a pre-trained model's parameters while keeping the majority unchanged. This selective fine-tuning allows the model to specialize in new tasks without the need for extensive computational resources. For instance, instead of retraining an entire model that might be very large in size, PEFT methods can achieve comparable performance by updating only a few megabytes worth of parameters. LoRA implies the introduction of low-rank matrices into the model's architecture, enabling efficient fine-tuning with reduced parameter updates.

This means that only some specifically targeted linear layers were used in the process of training, rather than going through the whole model. However, this method resulted in significantly lower results compared to the second method, so it was dropped completely in order to efficienctize the fine-tuning processes for DistilBERT and T5.

The second method implied a full fine-tuning of the model. Out of the total dataset, 80% was run through the model, with 10% used for validation, and 10% for testing. As DistilBERT is an encoder-only model, a classification head was put on top of it, formed out of two linear layers, with a ReLU activation function between them. A dropout of 0.3 was also applied to "deactivate" some of the neurons in order to prevent overfitting. A reduction to 256 dimensions was applied through the layers from the dimensions of the model, which were in a number of 768.

As it is an encoder-only model, the labels "fake_news" and "real_news" were converted to the 0 and 1 digits, and for the experiment consisting of a multiclass variation of the dataset, the digits 2, 3, and 4 represented the "propaganda", "satire",

and ”misinformation” classes, in this order.

### 4.3.2 The T5 model

For the fine-tuning of the T5 model, full fine-tuning was applied. This time, the dataset had to be prefixed with an expression in order to let the model know what task it had to perform. The expression ”clasifică știre: ” was prefixed before each text of the dataset, meaning ”classify news: ” in English, and the tags were changed to ”fals” and ”real”, meaning ”fake” and ”real” in Romanian, in order for them to be as simple as possibile, because tags like ”fake_news” or ”real_news” could impose problems to that model, as it is a generative model that gives its responses in a text form, with actual words.

After the training process, testing the model involved running all the test data through it and comparing the result of the model with the actual label of the data. As it is a text-to-text model, a conversion to lower case was necessary for the generated responses of T5, to make sure that the case in which a correct response would be ruled out because it did not match the label exactly would not be present (for example ”Real” would not match ”real”).

### 4.3.3 The CNN model

The starting layer for the CNN model is represented by the embedding layer, which is crucial for handling text data. Its job is to transform discrete integer-encoded words (the token IDs) into dense, continuous vector representations. The tokenization process was done using a Romanian BERT model, namely ”bert-base-romanian-cased-v1” from HuggingFace (source: `https://huggingface.co/dumitrescustefan/bert-base-romanian-cased-v1`). The dimension of the representation vector was set to 128. Next, a one-dimensional convolutional layer follows, responsible for extracting patterns from the sequence of word embeddings, with 128 filters, a kernel size of 5, and a ReLU activation function. For down-sampling, a max pooling layer was also used following the convolutional layer. A dense layer with 64 neurons and a ReLU activation function was then added, followed by a dropout layer with a rate of 0.5, meaning that 50% of neurons in the previous layer’s output would be randomly set to zero at each update, in order to prevent overfitting. Lastly, the final output layer was added, responsible for giving the result of the detection, with one output neuron and a sigmoid activation function for the first three experiments, where a threshold of 0.5 was used to obtain the result, with values smaller than 0.5 meaning a false article, and values over 0.5 meaning a reals article, and 5 output neurons and a softmax activation function for the fourth experiment.

## 4.4 Used Technologies

For the augmentation and preprocessing steps for the dataset, the "transformers" library from HuggingFace was used in order to get access to the mBART and Helsinki-NLP models and use them. For mBART, "MBart50TokenizerFast" was used to access the tokenizer, and "MBartForConditionalGeneration" was used to get the model, more specifically "mbart-large-50-many-to-many-mmt" from HuggingFace (source: `https://huggingface.co/facebook/mbart-large-50-many-to-many-mmt`), introduced in [TTL+20]. For the Helsinki-NLP model, two separate models were actually used, namely "Helsinki-NLP/opus-mt-ro-fr" and "Helsinki-NLP/opus-mt-fr-ro" from HuggingFace (sources: `https://huggingface.co/Helsinki-NLP/opus-mt-ro-fr` and `https://huggingface.co/Helsinki-NLP/opus-mt-fr-ro`), using "MarianTokenizer" and "MarianMTModels" to get access to the models and their tokenizers. What this means is that there was one model from the Helsinki-NLP family that was used for each step of the French Back Translation. The names are descriptive enough, as it can be seen that one model was doing translation from Romanian to French only, and the other was doing translation from French to Romanian.

For the preprocessing part, the Spacy library held an important role, as it provided access to a Spacy model for the Romanian language that allowed the removal of stop words and the application of lemmatization. In addition, it was used in the Back Translation process with the French language, as there were some problems found with the Helsinki-NLP model, which seemed to stop translating after a number of sentences and only output an incomplete result, with a translation that represented only a part of the original text. Through Spacy, the texts of the news were split into sentences, fed to the model, and then concatenated in the end in order to get the new full version of the original text. Regular expressions were also used to preprocess the data. Through the "re" library, URLs and digits were easily extracted.

The "sklearn" library was used for preparing the training, validation, and testing data, and also for calculating the accuracies of the models, through the actual accuracy, precision, recall, f1-score, and also through a confusion matrix. The confusion matrix helps provide a visual and more understandable representation of the performance of the models.

In addition, the transformers library was used again for the fine-tuning process of the language models by accessing the classical "TrainingArguments" and "Trainer" for the DistilBERT model, and the specific "Seq2SeqTrainingArguments" and "Seq2SeqTrainer" for the T5 model in order to make the training process possible. With this library, access to the models and their tokenizers was also provided

through "AutoModelForSequenceClassification" and "AutoTokenizer" for DistilBERT, and "T5Tokenizer" and "T5ForConditionalGeneration" for T5, respectively.

Everything regarding the training process of the CNN model was done using the "tensorflow" library through keras, and of course, for accessing the tokenizer of the BERT model for the Romanian language, the "transformers" library was used one more time.

The environment in which all the work was done was Google Colab, in order to achieve faster development through the access of Graphics Processing Units (GPUs), and Google Drive was used for saving the final versions of the models.

# Chapter 5

# Results

In order to offer a perspective that is as representative and clear as possible with regards to the accuracy of the models, this chapter will be split into 4 important subchapters, representing the four experiments that were done using the dataset that was used for this paper.

## 5.1 Experiment One

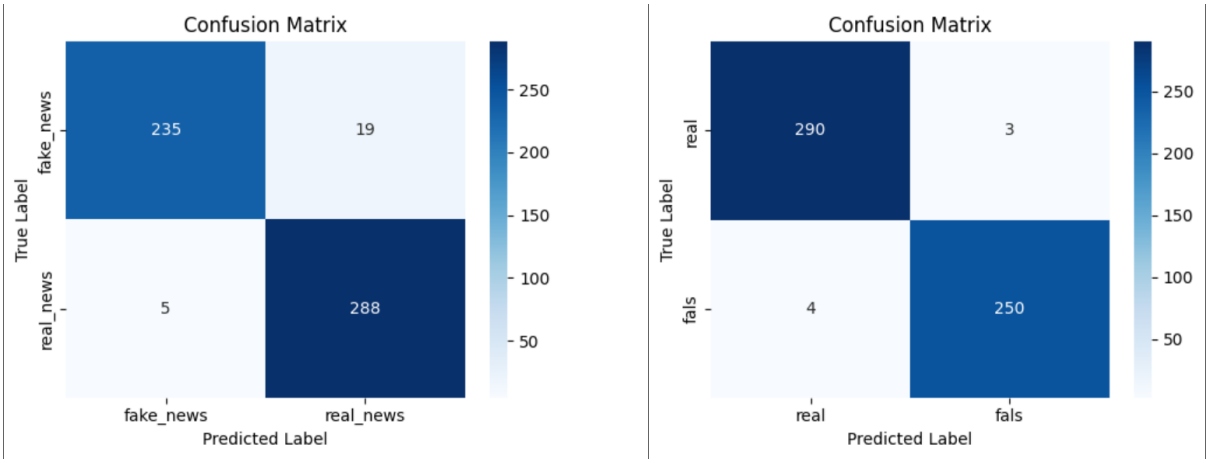
The first experiment consisted of training the models on the augmented dataset through the English and French back-translations. Using a random split, 10% of the dataset was retained for testing, while another 10% for validation. In this experiment, the models performed really well, achieving top results that can be seen in Table 5.1.

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1 score** |
|---|---|---|---|---|
| DistilBERT | 0.956 | 0.938 | 0.982 | 0.96 |
| T5 | 0.987 | 0.987 | 0.987 | 0.987 |
| CNN | 0.959 | 0.960 | 0.959 | 0.959 |

Table 5.1: Model results on experiment one

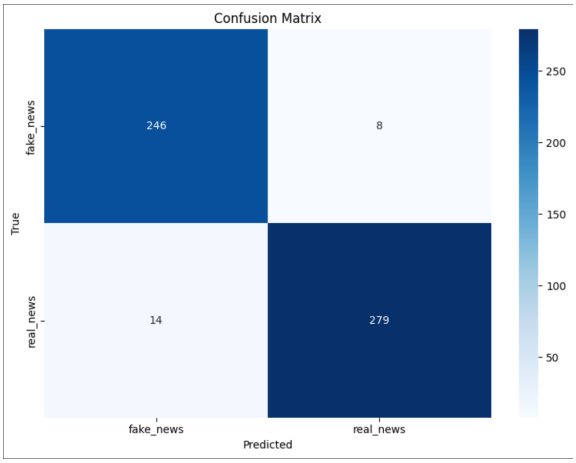
Looking at the results, we can see that the T5 model had the best results, by a pretty significant distance, with the CNN and DistilBERT having roughly the same results. What is really interesting in this situation is the high recall score of the DistilBERT, almost reaching that of the T5 model. However, besides this data, we also have to look at the scenario in which these results were achieved. Taking into consideration the fact that out of approximately 5400 news articles, 3600 were artificial, obtained through back translation, the evaluation of the models is biased, having no certainty that the models can generalize well on real (not artificial) news articles.

The confusion matrices from Figure 5.1 show some interesting differences between DistilBERT and the CNN model. Even though they have roughly the same results, it can be seen that the most inaccuracies are opposite with regard to these two models. DistilBERT falsely detected 19 fake news articles to be real, while the CNN model falsely detected 14 real news articles to be fake. For the T5 model, there is not a large difference between the falsely detected articles. As it is a sequence-to-sequence model, the labels of the articles were turned into the Romanian words for their type, namely "real" for real news, and "fals" for fake news.




(a) DistilBERT confusion matrix

(b) T5 confusion matrix)




(c) CNN confusion matrix

Figure 5.1: Confusion matrices experiment one

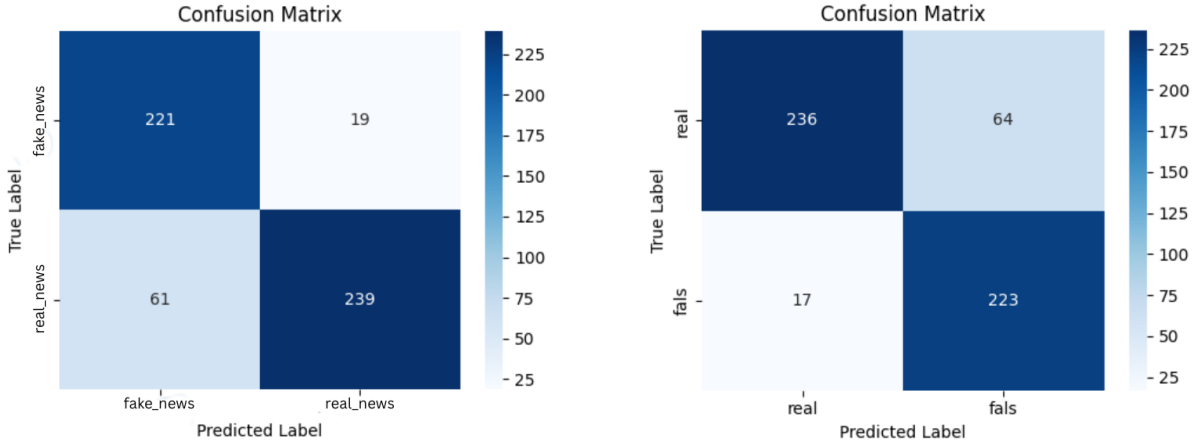
During the training of all the models, some different versions of hyperparameters were tried in order to see how the results varied. For the T5 model, the hyperparameters that provided the best results include a learning rate of 3e-4, and a batch size of 8, achieving the best training results after 10 epochs. For the DistilBERT model, a learning rate of 5e-5 was used, and a weight decay of 0.1 was added. The batch size was 16, with 8 epochs needed to reach the best results during training.

The CNN model was trained over 10 epochs, with a batch size of 32. The Adam optimizer and binary cross-entropy loss function were used for training.

## 5.2 Experiment Two

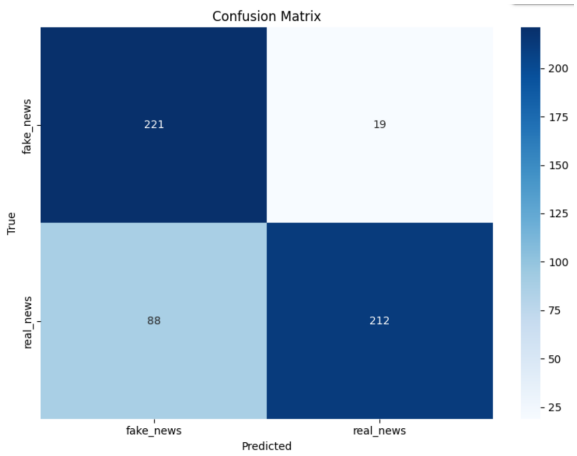
Considering that in the first experiment, the models were not tested solely on real news articles, the second experiment consisted of testing the models on only real ones. Therefore, out of the original database, 540 real news articles were taken, and the others were left out in order to have a training dataset together with their augmented version (from the English and French back translations).

In Figure 5.2, the detections of the models can be seen. It can be noticed that the major problem all the models have is with regard to the real news articles, which have been falsely identified by them as being false in a pretty big number.




(a) DistilBERT confusion matrix

(b) T5 confusion matrix




(c) CNN confusion matrix

Figure 5.2: Confusion matrices experiment two

In this experiment, the models presented a large overfitting, achieving over 95%

accuracy on the training data, and the following results when they were evaluated on the testing data:

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| DistilBERT | 0.851 | 0.926 | 0.796 | 0.865 |
| T5 | 0.850 | 0.854 | 0.857 | 0.849 |
| CNN | 0.801 | 0.827 | 0.801 | 0.801 |

Table 5.2: Model results on experiment two

Here, a bigger difference between the language models and the CNN can be seen, with both achieving a significantly better score than the CNN. A detail worth noticing here is the rather good precision score of the DistilBERT model, with a value of 0.926.

For training the models, the best hyperparameters were 10 epochs and a batch size of 16, with a 5e-5 learning rate and 0.1 weight decay for the T5 model. During the DistilBERT training, the epochs and learning rate remained the same, with the weight decay and batch size changed to 0.3 and 8, respectively. For the CNN, the batch size was 64, compared to 32 in the first experiment.

## 5.3 Experiment three

Seeing the overfitting that occurred in the second experiment, the fact that maybe the overfitting was produced by the back-translated artificial versions of the dataset was taken into consideration. Therefore, another experiment was tried to see what happens if only the original dataset, in the preprocessed version, is used. The obtained results were significantly better compared to the results from the second experiment, and even though the testing data was significantly smaller, the training dataset was also smaller. The full results can be seen in Table 5.3

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| DistilBERT | 0.896 | 0.898 | 0.908 | 0.903 |
| T5 | 0.912 | 0.915 | 0.909 | 0.911 |
| CNN | 0.901 | 0.902 | 0.901 | 0.901 |

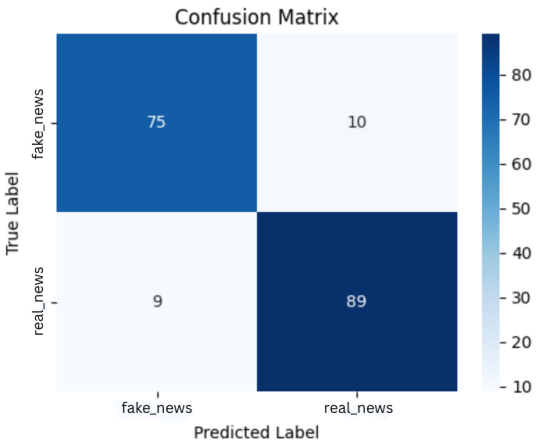
Table 5.3: Model results on experiment three

Still, the T5 model holds its position as the model with the best results, while the CNN and DistilBERT models appear to be on the same level, with the CNN having higher accuracy and precision, but with the DistilBERT taking over with the recall and F1 score results. It is worth mentioning that in this experiment, the models also presented some overfitting, but not as much. Compared to the previous experiment,

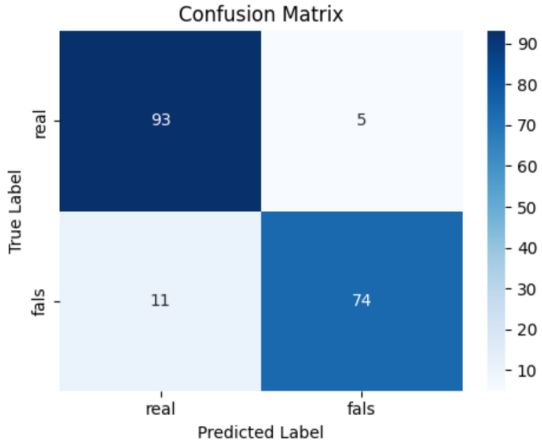
the models achieved during training a maximum accuracy of 93%, so the difference between the results from training and the ones from testing is not that large.

When it comes to the detection data, Figure 5.3 presents the confusion matrices of the models.

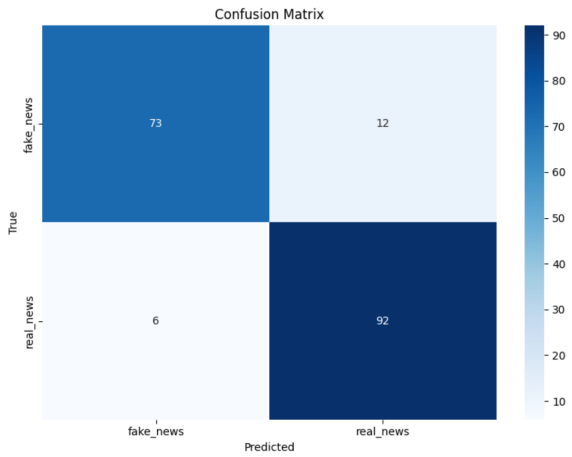
The hyperparameters with the best results in this experiment were a batch size of 8 and a weight decay of 0, with the epochs and learning rate being the same as in the previous experiment for the T5 language model. For DistilBERT, the weight decay this time was 0.1, with 8 training epochs and a batch size of 16, and the learning rate remained unchanged. During the CNN training, the batch size was switched back to 32, as the size of 64 resulted in too much overfitting.




(a) DistilBERT confusion matrix




(b) T5 confusion matrix




(c) CNN confusion matrix

Figure 5.3: Confusion matrices experiment three

## 5.4 Experiment four

The fourth experiment consisted of performing a multilabel classification task on the models. This was done by using the dataset that was made publicly available upon the publication of [MMC+24] (source: `https://huggingface.co/datasets/mihalca/FakeRO_updated`), in order to make a direct comparison with the results from it, as it was also the main source of inspiration for this thesis. This dataset version was a result of an augmentation process from the original dataset, containing the five classes mentioned earlier in the paper: "real_news", "fake_news", "propaganda", "misinformation", and "satire". It has a total of 5160 rows, each class being distributed equally.

The results obtained were very good compared to the ones in the paper, with the models tested on the same dataset (called the "NEW" dataset in the paper). They can be seen below.

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| DistilBERT | 0.984 | 0.984 | 0.984 | 0.984 |
| T5 | 0.951 | 0.953 | 0.951 | 0.951 |
| CNN | 0.986 | 0.986 | 0.986 | 0.986 |

Table 5.4: Model results on experiment four

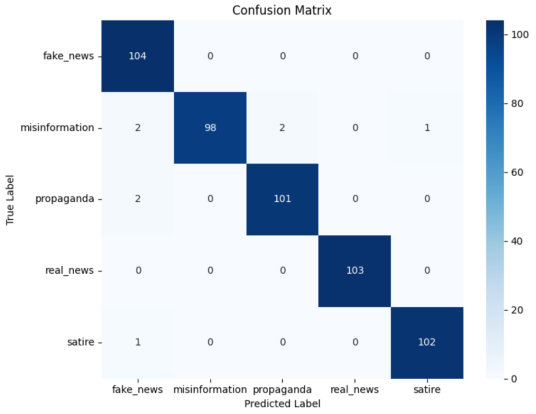
This experiment represented the first time the T5 model had worse results than the other models, as can be seen. The highest results obtained in [MMC+24] were obtained using the language models, specifically BERT and RoBERTaLarge, which obtained 95.5% and 96.2%, respectively, in accuracy. In a direct comparison, this shows that the DistilBERT model might represent an overall better solution for this task, achieving a higher score than BERT, while also being smaller. The CNN also represents a very efficient method, considering that the created model also obtained a higher accuracy than the language models from the paper. The full results from [MMC+24] can be seen in Table 5.5

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| BERT | 0.955 | 0.958 | 0.955 | 0.955 |
| RoBERTa Large | 0.962 | 0.962 | 0.962 | 0.962 |
| Support Vector Machine | 0.944 | 0.945 | 0.943 | 0.943 |
| Logistic Regression | 0.904 | 0.905 | 0.904 | 0.903 |
| Multinomial Naive Bayes | 0.833 | 0.839 | 0.833 | 0.830 |

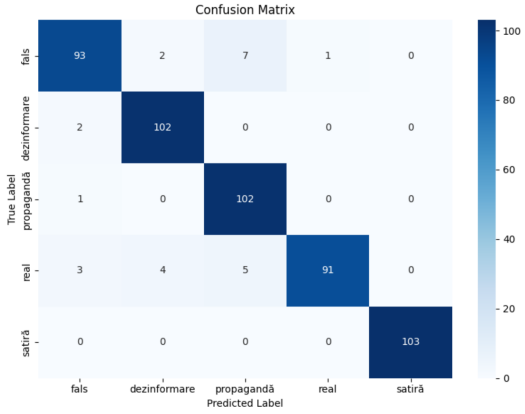
Table 5.5: All Results from [MMC+24]

Talking about the confusion matrices, we have the following results, presented in Figure 5.4. As in the case of the binary classification, the labels were changed for the training of the T5 model to their representative Romanian words: ”fals” for fake news, ”real” for real news, ”satiră” for satires, ”propagandă” for propaganda, and ”dezinformare” for misinformation. It can be observed that the T5 model had the most problems with the fake news, classifying them wrongly to be propaganda. Despite this, we have to take into account the fact that propaganda is also a type a fake news, so the classification is not that far from the truth.

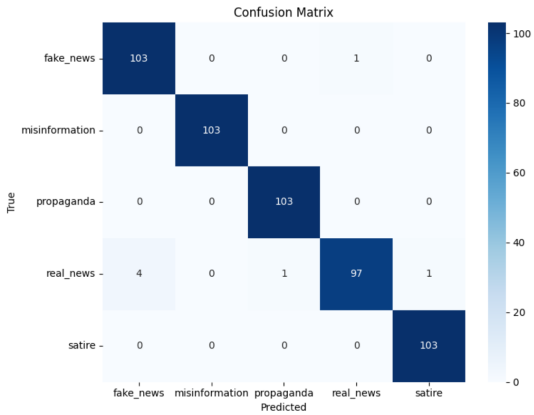
For this multilabel training process, the T5 model only had the weight decay changed to 0.1. For DistilBERT, the learning rate was still the same, but this time the batch size was 8, over the training of 10 epochs, with a 0 weight decay. Being a multilabel classification, the loss function for the CNN was changed to sparse categorical crossentropy. The batch size remained the same.




(a) DistilBERT confusion matrix




(b) T5 confusion matrix




(c) CNN confusion matrix

Figure 5.4: Confusion matrices experiment four

## 5.5 Discussion

When talking about the binary classification, it can be said overall that the merge of the initial classes: ”misinformation” and ”propaganda” into the ”fake_news” class provided to work and was quite efficient. If we take into consideration both good generalization in the classification on real data, and high results, we can say that the models performed best on the base dataset, with approximately 1800 news, this scenario being applied in the third experiment. Even though the results from the first experiment have been shown to be very good and within a big difference from the other two experiments, we cannot say with confidence that they would ensure a good performance of the models on unseen, completely new data. Looking over all three experiments, we can extract the fact that the T5 model performed the best, as in two of them, it overcame the other models by at least 2% in terms of accuracy.

With regards to the back translation method for data augmentation, the general conclusion is that, at least for this dataset, the method does not represent a very good solution for achieving higher results and providing a better training process for models. Multiple experiments were tried, such as taking less news for testing, and therefore having more data for the training process, or using original data and back-translated data with French only for training, but no significant changes occurred. The purpose of this augmentation was to provide models with more data for better training, but fake news detection depends a lot on specific words used. Fake news articles often contain words that seek to impress, to shock people and to keep them reading, and to make them believe that something is real. If we get, however, a different version of the article, with similar, but not the exact initial words (like it is in the case of back-translation augmentation), we might not get a very good representation of the article we started with. The new words that resulted might not be used in real (not artificial) fake news, or worse, they might be words that are actually used in real, true news. This could also be valid vice-versa, in obtaining a new version of true news, but with words that might not be representative of actual true news. This whole scenario is something that might confuse the models while training, and it might be a reason for the overfitting that the T5, DistilBERT, and CNN models had during the second experiment.

Continuing with the fourth experiment, the situation is similar to the one from the first experiment, because the published dataset is an augmented version of the original one, by using a different method, namely, synonym replacement, but in this case, the augmentation had a different purpose. In the case of the binary classification, the augmentation was done to simply enlarge the dataset in order to provide more data to the models. In the case of [MMC+24], the augmentation was done to solve a problem regarding the severe imbalance of the dataset, so the situations

cannot be directly compared. However, all the models performed really well on the task, even compared to the results from the original paper.

If we want to create an overall image of the performance of all the models, we can affirm that the CNN model had an impressive performance, considering the fact that it is not a language model. Language models are mentioned most of the time when talking about tasks involving texts, and are usually considered the best solutions for these kinds of tasks. Making comparisons to other papers, the results were also better, mostly. In [BD23], the best result obtained overall was 80.75% in accuracy, which is lower than any other accuracy obtained by all three models, besides the accuracy from the CNN from the second experiment. In [UPR+22], the best results were obtained using language models combined with CNN, but they did not come close compared to the DistilBERT and T5 models, achieving at best a 75% accuracy. However, the paper [BTMR22] obtained better results overall, considering the fact that the used dataset only contained real, non-artificial news, with the highest score being 98.1%, using CNN. For all these papers, it is important to note that the datasets used were different.

# Chapter 6

# Application Development and Usage

The purpose of the application is to show the applicability of this research and to provide a tangible way for possible future users to make use of this automatic fake news detection. To make it easier to use and more accessible, the app was designed for mobile devices, specifically for the Android platform, using the Kotlin programming language.

## 6.1 Specifications and Requirements

A local database was used in order to provide the user with the possibility of seeing their local history of checked news. For this, an object containing the following fields was used: the ID of the piece of news, the content of the news, the date when it was checked, and the detection result.

The technologies used in the app are Jetpack Compose, used for more efficient and easier creation of the UI screens, Room, a library used to integrate a local database for the app, and the ViewModel library, for the implementation of the MVVM architecture. In addition, in order to establish a connection with the API, the Retrofit library was used. The development process of the application was done in the Android Studio development environment.

Regarding the detection process, the model that provided the best results in terms of accuracy, efficiency, and other metrics like recall, precision, and F1 score was used for the detection. In order for the app to have access to the model, the FastAPI web framework for the Python programming language was used in order to build a service API to make calls to the model from the app. The process consists of sending a POST request through the "/predict" endpoint containing the input text from the user to the API, after which the text goes through a preprocess step, then the model processes it, and the detection response is sent back to the client. In case an error occurs, a 500 status code is sent.

The requirements of using the application from the point of view of the user are the following:

- The user enters the app and is directed to the main screen of the application.
- The user has the possibility to enter, by pasting or writing, a piece of news, in order to check it for fakery. The text needs to have a maximum of 550 words.
- After clicking on the button for detection, a dialog will show to the user containing the result of the detection, with a follow-up question with regards to saving the piece of news locally.
- The user can choose if they want to save the news or not.
- The other screen present in the app will contain the local history of the checked news of the user that he chose to save. For every piece of news, the user will be able to see the whole content of it and also copy it to the clipboard.
- In the history screen, the user will also be able to search for his saved news.

A Use Case Diagram is shown in Figure 6.1 below to offer a clearer image of the use of the application.

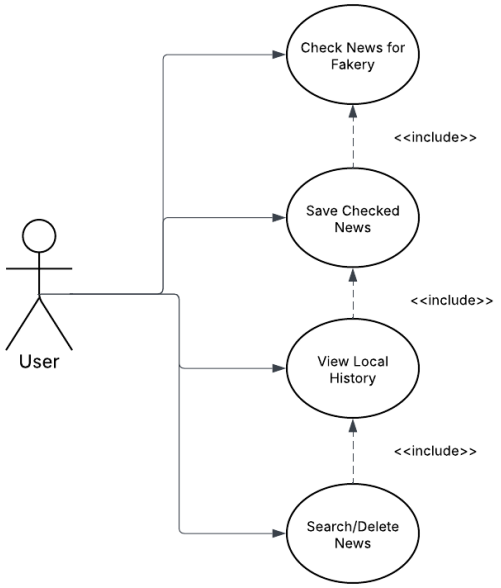



Figure 6.1: App Use Case Diagram

## 6.2 Architecture

The structure of the app follows the Model-View-ViewModel (MVVM) architecture, shown in Figure 6.2, in order to provide a clear separation of concerns.

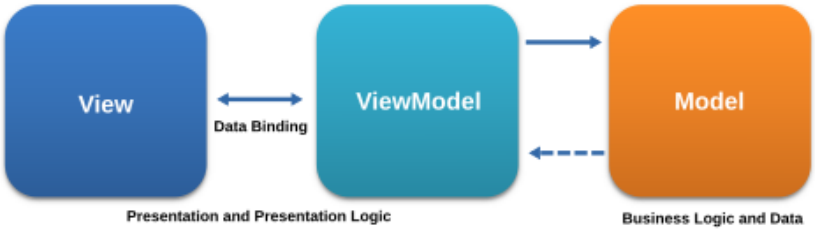



Figure 6.2: The MVVM Architecture (source: [EGG]

This way, the View layer contains everything UI-related, the ViewModel layer handles the logic of what is shown in the user interface (UI), and the Model layer contains the business logic and data.

Figure 6.4 provides a clearer view of the architecture. The app starts from the ”ThesisApplication” class, more specifically from the ”onCreate” function. From there, the ”AppDataContainer” creates everything the app needs in the backend size: the database, the repositories, and the Retrofit instance for accessing the API. In addition, the call of the function will also trigger the creation of the main activity, represented by the ”MainActivity” class. Theoretically, as can also be seen in the diagram, the ”MainActivity” is not connected to anything in the application. That is because the ”onCreate” function from the main activity calls the main function responsible for the creation of the UI of the app. Nothing related to the UI is an actual class, but it is instead represented by functions, which are specifically called Composables. Composables represent components from the UI. They can represent anything from a simple input field to an entire screen of an application.

Moving forward with the structure of the app, we are talking about the application of the MVVM architecture. The ”ApplicationViewModel” represents the view model of the application, inheriting the ”ViewModel” class available through the library with the same name. The UI accesses the view model through the Composables. Every Composable that can change (an input field where a user can enter text, a list of items) has access to the view model by having it as a parameter in the function. Therefore, in this way, we have the View-ViewModel relation. Whenever the user performs an action that requires a change of state in the UI, the UI notifies the view model by calling it. Upon the call, the ViewModel-Model relation follows. The view model calls the model in turn, which performs the business logic required, then the view model updates the state of the UI, and sends the state to it in order to update it. These states are represented in the app by ”NewsUiState” or ”DetectionUiState”, for example. The most frequently used methods for creating these states are by using the ”StateFlow” and ”MutableState” classes. ”MutableState” is usually used for small pieces of the UI, like it is used, for example, to track the changes in the dialog that will appear to the user upon the receipt of the detection


Figure 6.3: Example of MVVM usage in the app

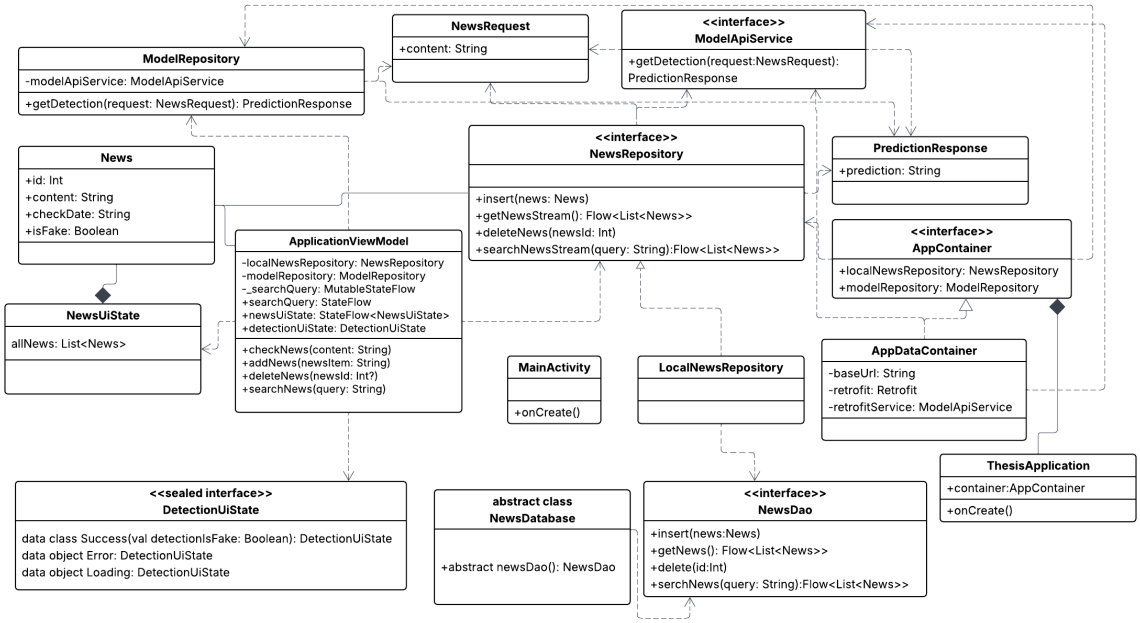



Figure 6.4: UML Class Diagram

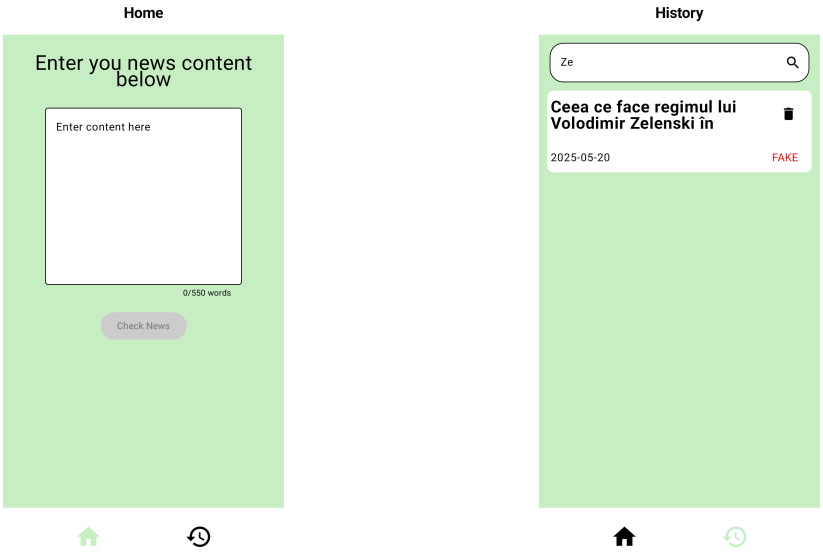
response, which is represented by a simple text, ”real” or ”fake”, process that is done through ”DetectionUiState”. The user will be notified visually of whether the app is awaiting a response (the Loading data object), or the app did not successfully communicate with the API (the Error data object), or the app received the detection result (the Success data class containing the detection response). ”StateFlow” in this case is used for the state of the list of news, which represents a larger part of the app. This state flow is observed by the view model and its latest value is automatically taken from it, every time changes occur. A concrete example is offered in Figure 6.3. Suppose a user clicks the delete icon on a news item. The ”onClick()” function from the icon is triggered, which contains a call to the ”deleteNews” function from the ”ApplicationViewModel”. The function receives as a parameter the id of the news item to be deleted, and calls the delete function from ”NewsRepository” (or more specifically, the ”LocalNewsRepository”, which implements the interface), which in turn will access the ”NewsDao” and calls the ”delete” function from the class. As a result, the selected news will be deleted, and the new state of the list will be automatically updated in the view model and updated visually in the UI.

## 6.3 User interface

The UI was designed in such a way that the user would find it intuitive to navigate through it. In order to provide seamless navigation through the screens, a bottom navigation bar was used, where a button for each screen was placed. A top bar is also present, showing the title of the current screen. In the main screen, a simple, large input field is in the center of the screen, with a button for checking the news below it. The input field cannot be empty. If it is, the checking button will not be enabled. The same will happen if the text passes the 550-word limit.

For the local history screen, list items with news containing a delete icon, the date when the piece of news was checked, and the result of the check will be present. Upon clicking the delete icon, a confirmation dialog regarding the deletion will appear to make sure the user wants to continue with this action. Above the list, a search bar is available for easily finding the saved news texts.

Below, in Figure 6.5, a clearer view is presented regarding the user interface of the app.




(a) Main Screen (b) History Screen

Figure 6.5: App Main and History Screens

## 6.4 Testing

Like in any other application, testing was a necessary process. Unit tests and integration tests were made for this app.

Firstly, some unit tests were done on the view model in order to ensure that the state of the detection is updated correctly. For this, the ”ModelRepository” was

mocked in order for the view model to receive responses and see if it correctly handles them. Both successful and error cases were tried to see if the state of the detection is updated to ”Success” state, with the correct detection response, and to ”Error” respectively, in case some exception was thrown upon the call of the ”getDetection” function. In addition, in both cases, the presence of the ”Loading” state was asserted.

Some integration tests were followed for the ViewModel, particularly for its interaction with the ”LocalNewsRepository”. For this, the ”NewsDao” was mocked. The states of the list of news and search query were tested to see if they also update correctly when performing the insert, delete, and search operations through the view model. Moreover, some other integration tests were made, this time using the relations between all three important components of the app: the view model, the news repository, and the news data access object. This implied performing instrumentation tests, which ran on an emulated Android device, accessible through Android Studio. Once again, the insert, delete, and search actions were tested.

Unit tests were also performed on the ”NewsDao”, ”LocalNewsRepository”, and ”ModelRepository” components. As the ”NewsDao” is theoretically just an interface, because its actions are made through the connection with a database, instrumentation tests were done on this component as well. The delete, insert, and read operations were tested, along with the search operation. The same operations were also tested for the ”LocalNewsRepository”, for which the ”NewsDao” was mocked once again. For ”ModelRepository”, the tests were rudimentary, represented by just testing to see if it correctly gets the response from an API, which was mocked.

In order to make the testing process efficient, some important libraries were used. JUnit was the most notable library, being a foundational testing framework for Java and Kotlin. It provides the basic annotations like ”@Test”, ”@Before”, ”@After”, and crucial assertion methods (”assertEquals”, ”assertTrue”, etc.). The ”AndroidX Test Extensions” library provides JUnit4 rules and APIs specifically for Android instrumentation tests. It extends JUnit’s capabilities to the Android environment. ”AndroidX Test Runner” is the fundamental test runner for Android instrumentation tests. It orchestrates the execution of tests on an Android device or emulator. Considering that some of the operations between the components were asynchronous, some libraries had to be used to facilitate testing them. The ”Kotlin Coroutines Test” library represented a very important usage, being essential for writing deterministic and efficient tests for Kotlin Coroutines. Through this library, the ”runTest” function was used, which provides a special coroutine scope that automatically manages coroutine execution. Turbine also had a significant role, dealing with testing Kotlin flows and simplifying the process. MockK was the library used to create the mock components that replaced real components like ”NewsDao”, ”ModelApiService”,

or ”ModelRepository’, through functions like ”mockk”, for mocking a class, or ”coEvery”, for defining what a mocked function should return or what exception it should throw.

## 6.5 Complete Application Presentation

Figure 6.6 below provides a clearer image of the entire application that was built. Upon accessing the dataset of news, the classes were merged such that two main classes remained: real and fake news. An augmentation process was followed using back translation through the English and French languages. Then the final version of the dataset resulted from a preprocessing step, including URL removal, stopwords removal, digits removal, punctuation removal, conversion to lowercase, and lemmatization. Using this dataset and other versions of it, presented in Chapter 5, including the multiclass dataset from experiment four, the model with the best results was chosen for inference, which was the T5 model from the third experiment, with an accuracy of 91%, considering the fact that it was solely tested on real data. The model was integrated into an API, using the FastAPI framework, which preprocesses the input text and gives it to the model, which sends its detection response when a POST request is sent by the application that was built, when the user enters a news article in the input field of the app.

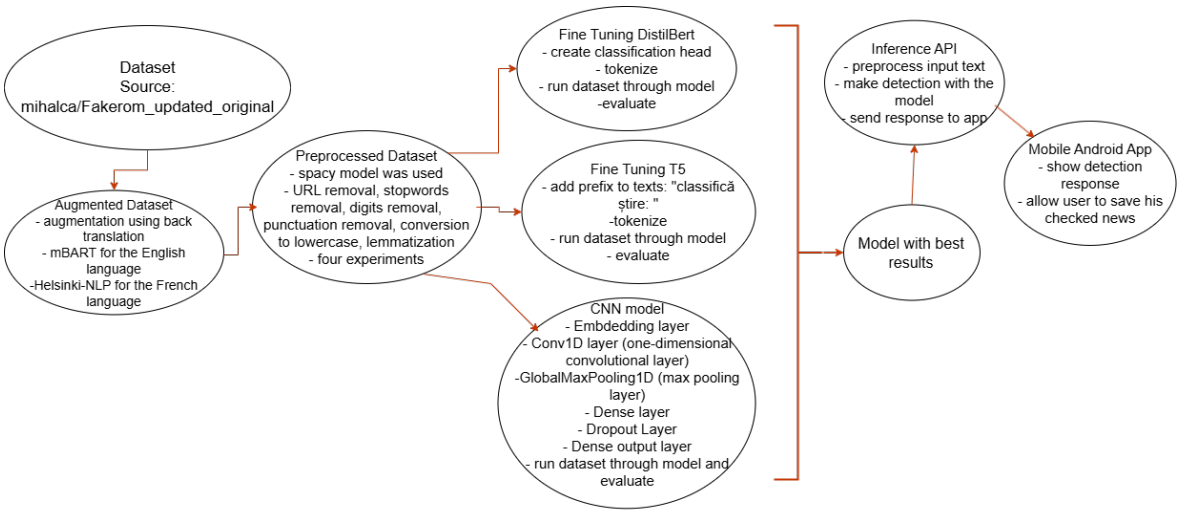



Figure 6.6: Full App Diagram

# Chapter 7

# Conclusions

In this paper, a continuation of the research on fake news detection in the Romanian language was provided, in order to offer a clearer perspective on new possible models that might represent a solution for this kind of task, and on what can be done further.

The main surprise was represented by the T5 model, which is not usually considered when it comes to classification tasks, having an encoder-decoder architecture. It performed incredibly well, being the model with the overall best results across all experiments. However, all the models that were used in this study can be taken into consideration when performing this type of task, as none of them had any major flaws. DistilBERT could prove to be a very good solution, as it is a significantly smaller and therefore faster model, which performed very similarly to other solutions provided in other studies, which were usually represented by models from the BERT family, when it came to language models.

In addition to the performance of the models, addressing class imbalance by merging classes that represent roughly the same type of news might represent a solution for future studies as well. Propaganda, misinformation, and disinformation all have the same negative effect on people and are similar, with disinformation being the most frequent type of fake news people come into contact with, because this type is represented by articles that are deliberately written to deceive people. This method may be helpful when trying to collect more resources for this task, having larger datasets that are equally balanced. Larger datasets lead to better and more accurate results, which can actually tell us if the implemented solutions can be used in real-world applications. For now, testing models on just hundreds of samples does not give us a clear image of how well they would perform over the large variety of news that we have access to every day.

With regards to the work done in this paper, future work and improvements could include training the models on datasets containing news from a broader range of subjects and domains. Web scraping might represent an efficient method for gath-

ering news, as it was used in other research studies on this topic as well. Most publications have their news articles posted online, publicly, and they represent a large source for gathering data, as these types of websites that contain news are also structured in a way that makes it easy to select specific domains for news while navigating through them, thus making it also easier for scraping these websites categorically, in order to have a large variety of articles. However, a challenge is represented by officially annotating the articles as real or false, as a thorough check would be required. On the app side, the UI could be modernized in terms of look, and more features could be put into the app, such as a language detector for the input the user enters, in order to make sure that the text is in the Romanian language, or an option for the user to input the text that he has through an image (screenshot, for example) that contains the text, which would be detected through an AI model. In addition, this type of app might prove to be useful if available publicly for installation, so a deployment on the platforms for mobile apps (Play Store, App Store, for example) would be a valuable next step, greatly expanding its reach and accessibility to a wider user base.

As a conclusion on the general topic, the large scale at which fake news articles are distributed across the world will remain a concern as years pass by, and detecting them through artificial intelligence still remains the best solution for combating them, in my opinion. Further work and research are still required, specifically for the Romanian language and a lot of other languages that are not popular in society when it comes to learning them. Talking specifically about the Romanian language, the major problem is the lack of resources. More work needs to be done in gathering as much data as possible, containing non-artificial news, both fake and real. No larger dataset than the one that was used in this paper was found at the time of writing. There are a lot of important subjects on which news is written, with politics, medicine, economics, and education being just a few that can be named. In order to provide a good generalization over as many types of news as possible, larger datasets need to be created. For example, the dataset used in this paper has a high percentage of medicine-related news, mostly from the COVID period, and it might not be enough to ensure good results with future news that will be written on different subjects.

The manual checking that some publications perform is still present nowadays, and it will probably be present in the future, but compared to the possibilities that artificial intelligence offers, it is a very slow and tedious process, that cannot be done by a simple person that is reading something on the internet or social media. By perfecting detection models for this task, people will also be able to find out on their own if what they are reading is true or not.

# Bibliography


[ACC+21] Andrei-Marius Avram, Darius Catrina, Dumitru-Clementin Cercel, Mihai Dascălu, Traian Rebedea, Vasile Păiş, and Dan Tufiş. Distilling the Knowledge of Romanian BERTs Using Multiple Teachers, 2021.

[BD23] Marian Bucos and Bogdan Drăgulescu. Enhancing Fake News Detection in Romanian Using Transformer-Based Back Translation Augmentation. *Applied Sciences*, page 13207, 2023.

[BRD20] Costin Busioc, Stefan Ruseti, and Mihai Dascalu. A Literature Review of NLP Approaches to Fake News Detection and Their Applicability to Romanian-Language News Analysis. *Transilvania*, pages 65–71, 2020.

[BTMR22] Marius Cristian Buzea, Stefan Trausan-Matu, and Traian Rebedea. Automatic Fake News Detection for Romanian Online News. *Information*, page 151, 2022.

[DFG23] Liviu Dinu, Elena Casiana Fusu, and Daniela Gifu. Veracity Analysis of Romanian Fake News. *Procedia Computer Science*, pages 3303–3312, 2023.

[EGG] Builtin Estefanía García Gallardo. What Is MVVM (Model-View-ViewModel)? `https://builtin.com/software-engineering-perspectives/mvvm-architecture`.

[MMC+24] Elisa Valentina Moisi, Bogdan Cornel Mihalca, Simina Maria Coman, Alexandrina Mirela Pater, and Daniela Elena Popescu. Romanian Fake News Detection Using Machine Learning and Transformer-Based Approaches. *Applied Sciences*, page 11825, 2024.

[Nya] Jean Nyandwi. The Transformer Blueprint: A Holistic Guide to the Transformer Neural Network Architecture. `https://deeprevision.github.io/posts/001-transformer/`.

[ON15] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, 2015.

[RSR+19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, 2019.

[SK23] Kamonashish Saha and Ziad Kobti. DeBERTNeXT: A Multimodal Fake News Detection Framework. In Jiří Mikyška, Clélia De Mulatier, Maciej Paszynski, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M.A. Sloot, editors, *Computational Science – ICCS 2023*, pages 348–356. Springer Nature Switzerland, 2023.

[SSW+17] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake News Detection on Social Media: A Data Mining Perspective, 2017.

[TTL+20] Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. Multilingual Translation with Extensible Multilingual Pretraining and Finetuning, 2020.

[UPR+22] University Politehnica of Bucharest, Andrei Preda, Stefan Ruseti, University Politehnica of Bucharest, Simina-Maria Terian, University Lucian Blaga" of Sibiu", Mihai Dascălu, and University Politehnica of Bucharest. Romanian Fake News Identification using Language Models. In *RoCHI - International Conference on Human-Computer Interaction*, pages 73–79. MATRIX ROM, 2022.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, 2017.

[XLT+24] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A Survey on Knowledge Distillation of Large Language Models, 2024.