



Design and Optimization of Machine Learning Models

Egyetemi doktori (PhD) értekezés

Lakatos Róbert

Témavezető: Dr. Hajdu András

DEBRECENI EGYETEM

Természettudományi és Műszaki Tudományi Doktori Tanács

Informatikai Tudományok Doktori Iskola

Debrecen, 2025

Ezen értekezést a Debreceni Egyetem Természettudományi és Informatikai Doktori Tanács Informatikai Tudományok Doktori Iskola Adattudomány és vizualizáció programja keretében készítettem a Debreceni Egyetem műszaki doktori (PhD) fokozatának elnyerése céljából.

Nyilatkozom arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.

Debrecen, 2025.

.....

a jelölt aláírása

Tanúsítom, hogy Lakatos Róbert doktorjelölt 2020.02.01 és 2024.01.31 között a fent megnevezett doktori iskola Adattudomány és vizualizáció programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult.

Nyilatkozom továbbá arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.

Az értekezés elfogadását javasolom.

Debrecen, 2025.

.....

a témavezető aláírása

Design and Optimization of Machine Learning Models

Értekezés a doktori (PhD) fokozat megszerzése érdekében
az informatika tudományágban

Írta: Lakatos Róbert okleveles programtervező informatikus

Készült a Debreceni Egyetem Informatikai Tudományok Doktori Iskolája
Adattudomány és vizualizáció programja keretében

Témavezető: Dr. Hajdu András

Az értekezés bírálói:

Dr.

Dr.

A bírálóbizottság:

elnök: Dr.

tagok: Dr.

Dr.

Dr.

Dr.

Az értekezés védésének időpontja: 20...

To my wife Zsuzsa and my beloved daughters Ágnes and Kamilla

Acknowledgement

I want to express my deep gratitude to my thesis advisor, Professor Dr. András Hajdu, for his unwavering support throughout my PhD studies.

I extend my exceptional thanks to Dr. Péter Pollner and Dr. Tamás Joó for their help and support as scientific experts during my cooperative doctoral program scholarship (KDP) period.

I am grateful to all my research colleagues from whom I have learned much during our scientific work and publications.

I also want to thank the Doctoral School of Informatics at the University of Debrecen and Neumann Nonprofit Ltd for enabling me to continue my doctoral studies as a KDP scholarship holder through their cooperation.

Lastly, I would like to express my gratitude to my family, especially my wife Zsuzsa and my children Ágnes and Kamilla, whose endless patience and love enabled me to pursue scientific work actively.

My thesis was supported by the KDP-2021 and has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development, and Innovation Fund, financed under the C1774095 funding scheme.

Contents

List of Tables	v
List of Figures	vii
1 Introduction	1
2 Background	5
2.1 SIR model and Least-squares curve fitting problem . . .	5
2.2 Moving Average and Relative Strength Index	6
2.3 Types of neural networks	6
2.4 Training Methods	7
2.5 Tokenization, text vectorization, and dimension reduction techniques	8
2.6 Information-extracting	9
2.7 Metrics	10
3 Predicting the direction of the oil price trend using sentiment analysis	11
3.1 Introduction	11
3.2 Data	12
3.3 Methodology	13
3.3.1 Preprocessing	13
3.3.2 Sentiment analysis	14
3.3.3 Correlation	15

3.3.4	Time-based diversity	16
3.3.5	Noise filtering	17
3.3.6	Training and Test Dataset	18
3.3.7	Models	19
3.4	Results	20
3.5	Conclusions	22
4	A Machine Learning-based Pipeline for the Extraction of Insights from Customer Reviews	25
4.1	Introduction	25
4.2	Dataset	27
4.3	Methodology	28
4.3.1	Keyphrase extraction techniques	28
4.3.2	Pipeline	32
4.4	Results	38
4.5	Conclusions	42
5	A multimodal deep learning architecture for smoking detection with a small data approach	45
5.1	Introduction	45
5.2	Methodology	47
5.2.1	Model architecture	47
5.2.2	Format check	48
5.2.3	Processing of videos and images	49
5.2.4	Multimodal filtering	49
5.2.5	Fine-tuned image classification	52
5.2.6	Processing of text	52
5.3	Results	55
5.4	Conclusions	58
6	Investigating the domain transfer techniques for the development of LLM-driven knowledge-based systems	61
6.1	Introduction	61
6.2	Data	63

6.3	Methodology	64
6.3.1	Models	64
6.3.2	Selected metrics	64
6.3.3	Data preparation for RAG and DFT	66
6.3.4	Fine Tuning settings	70
6.3.5	RAG Settings	70
6.4	Results	70
6.5	Conclusion	74
7	Optimization of theoretical models	77
7.1	Introduction	77
7.2	Countries and data	78
7.3	Methodology	78
	Summary	83
	References	85
	List of Publications	103

List of Tables

3.1	Twitter text sources (tweets) for our analysis.	12
3.2	Correlation values between the oil products.	13
3.3	Average accuracy between sentiment models and oil price direction.	16
3.4	Comparison of models based on resource-based performance.	16
3.5	Filtered noisy records (news).	18
3.6	Average accuracy of the different hyperparameter settings for LSTM and GRU on the test set.	21
4.1	Performance of keyword extraction models on test dataset made from CPA based on expert evaluations.	32
4.2	The average accuracy of binary classification on binary mapped test dataset made from 20 newsgroups dataset.	40
4.3	Number of topics and vocabulary sizes for binary classification on binary mapped test dataset made from 20 newsgroups dataset.	40
4.4	The average accuracy of multi-classification on 20 newsgroups test dataset.	41
4.5	Average number of topics and vocabulary sizes for multiple classifications on 20 newsgroups test dataset. . . .	41
5.1	An example prompt for ChatGPT.	53

5.2	An example paragraph generated by the prompt in Table 5.1.	54
5.3	Performance of multimodal filtering on the test videos.	56
5.4	Performance of the EfficientNetB5 on the test videos. .	56
5.5	Performance of the Ensemble model (Multimodal Filtering + EfficientNetB5) on the test videos.	57
5.6	Results of a fine-tuned (LLM2) for the named entity recognition task on the test data.	57
5.7	Validation of language models on the name entity recognition task.	58
6.1	Number of paragraphs and questions of Q&A datasets.	67
6.2	Number of vectorized sentences and questions of indexed datasets.	68
6.3	Average scores of each approach.	72

List of Figures

3.1	Model decision stability.	20
3.2	Daily forecast accuracy.	21
4.1	Extracting a keyphrase candidate from a simple review using an N-gram-based approach (N=3).	29
4.2	An example for dependency parsing.	30
4.3	Flow diagram of the pipeline.	33
4.4	The BERT+R architecture modified for our system. . .	35
4.5	Recursive clustering considered in our system.	37
5.1	Schematic flow diagram of the architecture.	47
5.2	The cosine similarity of the images obtained from the video recording in chronological order.	50
5.3	The cosine similarity of the images after the ordering. .	51
6.1	Radar plot of the evaluation results of the CORD-19 and MedQuAD models.	73
7.1	A slice from the big solution space (left) and a SIR curve fitted to the German data (right).	79

Chapter 1

Introduction

Artificial intelligence research has undergone a transformative evolution, primarily driven by the imperative to enhance the efficiency and performance of machine learning models. However, the true potential of these models hinges on their optimization.

Optimization is a multifaceted concept. On the one hand, it involves algorithms designed to fine-tune models, and on the other hand, it encompasses strategies aimed at improving model effectiveness. For example, the Adam optimizer, introduced by Kingma and Ba [1], has become widely popular for training neural networks due to its adaptive learning rates and ability to handle noisy gradients efficiently. Similarly, in generative modeling tasks, specific optimization strategies used in generative adversarial networks have been crucial [2].

The optimization process inherently depends on the model type and the task's nature, with neural network-based architectures susceptible to specific optimization strategies. Therefore, in this context, the choice of optimization technique is shaped by the target task and the quality of the available data.

Critical questions that drive the optimization of machine learning models include:

- What is the most effective training method?
- How can hyperparameters be fine-tuned for peak performance?
- How can complex model architectures be optimally designed?

The nature of the problem at hand typically determines training methodologies. Machine learning tasks are commonly categorized as supervised, semi-supervised, or unsupervised learning [3], requiring distinct optimization strategies.

In supervised learning, the availability of labeled data guides the optimization process. Semi-supervised learning, in contrast, utilizes strategies that leverage labeled and unlabeled data, while unsupervised learning aims to discover hidden patterns in the data.

Additionally, advanced training strategies such as transfer learning, fine-tuning [3], and pre-training have also become essential [4], and these came into a particular massive focus with the appearance of large language models (LLMs). These models, built on transformer architectures [5], are trained on extensive datasets and have performed remarkably in machine translation, text summarization, and question-answering tasks. Research has shown that LLMs pre-trained for general tasks and fine-tuned for specific tasks outperform models trained solely on task-specific data [4]. This approach improves task-specific performance and enhances the model's generalization capabilities.

Optimizing hyperparameters is crucial for the success of machine learning models. Standard methods for hyperparameter tuning include Grid Search, Random Search [6], Bayesian optimization [7], and genetic algorithms [8]. Additionally, ensemble techniques such as Bagging, Boosting, and Stacking [9] provide robust solutions for optimizing the architecture of complex models.

In this dissertation, I investigate the ever-changing field of machine and deep learning, suggesting new methods to improve model performance and expand their abilities. I pay special attention to optimization techniques essential for efficiently tackling complex problems.

In the following chapters, I establish the theoretical foundations of our approaches and measurement techniques (Chapter 2) and present my research findings across various domains of machine learning optimization. These include pre-training neural networks, addressing multimodal challenges, and harnessing the power of generative models.

Chapter 3 explores the development of an ensemble solution based on LLMs and recurrent neural networks, enabling forecasting of oil price movements. By applying textual data, sentiment-based LLMs, and time series prediction models, I demonstrate the potential of combining diverse data sources to build a forecasting model.

In Chapter 4, I present a Pipeline based on an ensemble of machine learning and classical methods. Our measurement results prove that it provides a more sophisticated and uniform solution for extracting relevant statements from customer reviews than the currently popular methods.

Chapter 5 focuses on the performance potential of multimodal architectures, specifically in detecting smoking behaviors from text and images. I present a model that performs well, even with limited training data.

In Chapter 6, I compare the performance of RAG and domain-specific fine-tuning and show that RAG has an advantage in developing new knowledge base systems based on LLM.

Finally, in Chapter 7, I show how to trace the optimization of theoretical models such as SIR back to applying algorithms that solve least squares curve fitting problems. This allows us to produce optimal theoretical models suitable for neural network pre-training.

Chapter 2

Background

In this thesis chapter, I present the basic concepts and formulas.

2.1 SIR model and Least-squares curve fitting problem

Theoretical functions, abstract mathematical concepts, establish connections between quantities. These functions represent real-world systems, such as the SIR model [10], used to forecast virus spread. The least-squares curve fitting problem [11] involves finding the best-fitting curve that minimizes the sum of squared deviations between observed data and predicted values.

In Chapter 7, I trace the problem of SIR [10] model optimization back to the least squares curve fitting problem. The algorithms considered include gradient descent [12], Broyden, Fletcher, Goldfarb, and Shanno algorithm (BFGS) [13], Gauss-Newton [14], Levenberg-Marquardt [15] [16], and differential evolution [17].

2.2 Moving Average and Relative Strength Index

A moving average (MA) is a formula that calculates the average price of a given period. This results in a smoothed signal that helps filter out short-term noise and highlight the underlying trend. The Relative Strength Index (RSI) [18] is a momentum indicator that measures the extent of price changes of a given asset in the past period.

Stock market indicators help investors analyze market conditions and trends. In Chapter 3 of my dissertation, I apply the MA and RSI market indicators. Here, I use simple moving averages to reduce the noise of market price movements. In addition, I use the RSI indicator as one of the baselines against which I compare my solution.

2.3 Types of neural networks

Recurrent neural networks are models that consist of feedback units. Feedback units imply that information from previous steps produces the current output. Their disadvantage is that they need to handle long-term dependency relationships better due to the vanishing gradient problem. However, some of its types can eliminate the vanishing gradient problem, so these can be used to predict time series. Such recurrent networks are Long Short-Term Memory (LSTM) [19] and Gated Recurrent Unit (GRU) [20]. CNNs [21] are mainly used for handling two-dimensional structures such as images. With the help of convolutional layers, the network can extract essential features from the data. Transformer-based networks [5], initially developed for text sequences, outperform CNNs in text processing and handle long-term dependencies better than RNNs. Also, some versions can be used to analyze images (ViT [22]) or time series (LTSF [23]).

I focused on recurrent, convolutional, and transformer-based networks during my dissertation research. In Chapter 3, I optimized recurrent networks, used convolutional networks in Chapter 3 to build a multi-modal architecture, and dealt with transformers in Chapters 3, 4, 5 and 6.

2.4 Training Methods

Pre-training [4] means training a model on a big data set to learn the dataset’s general patterns. Transfer learning [3] applies knowledge from a pre-trained model to solve a new, specific task by adjusting only particular layers, often used in tasks like image recognition. Fine-tuning [3] refines a pre-trained model to a target task by updating layers with new data, allowing for domain-specific adjustments, though excessive fine-tuning may erode the model. Retrieval-augmented Generation (RAG) [24] enhances generative models by incorporating external information, improving accuracy, and reducing hallucinations without direct fine-tuning.

The differences between these training methods lie in the details of the processes and how they are applied to the tasks. In pre-training, we teach general knowledge; in transfer learning, we transfer this knowledge to another task, while in fine-tuning, we retrain the transferred knowledge to the target problem. However, it is not trivial when to stop the training process. Early stopping [25] is a possible solution. In the case of early stopping, for example, to stagnate the further improvement of a specific measure, we assign when the teaching should stop to the loss function.

Fine-tuning and pre-training are essential in chapters 3, 5, and 6 of the dissertation. Furthermore, we apply fine-tuning in chapters 5 and 6 to achieve better model performance. Finally, in Chapter 6, I compare the efficiency of the RAG and domain-specific fine-tuning in the case of generative LLMs.

2.5 Tokenization, text vectorization, and dimension reduction techniques

Tokenization is breaking down the text into smaller, more manageable pieces. The most popular tokenization techniques are Byte-Pair Encoding [26], WordPiece [27], Unigram [28], and SentencePiece [29] subword-based techniques.

Text vectorization is the process of converting text data into numeric vectors. The basic vectorization techniques are one-hot encoding, counter vectorizing, and the TF-IDF [30–32]. Machine- and deep-learning-based methods can convert words or tokens into numerical vectors. Such methods are GloVe [33], which is a statistical-based approach, and Word2Vec [33], and FastText [34], which use neural networks.

The problem with working with embedded vector spaces is that textual data is converted into high-dimensional vectors, which are resource-intensive to create if we want to achieve good semantic quality. The resource demand can be reduced by various dimension reduction techniques such as Principal Component Analysis (PCA) [35–37], Uniform Manifold Approximation and Projection (UMAP) [38, 39], or t-distributed Stochastic Neighbor Embedding (t-SNE) [40]. However, dimension reduction has a price, which appears as a loss of information during the processing process.

In the thesis's 3, 4, 5 and 6 chapters, I deal with the models in which tokenization is strictly connected. Furthermore, text vectorization and size reduction techniques are essential in my thesis's 4 and 6 chapters.

2.6 Information-extracting

Information extraction methods generally involve frequency-based extraction, dependency analysis, keyphrase extraction, semantic searching, and topic modeling.

Methods focusing on word frequency [41] can effectively analyze essential texts. Dependency parsing [42] seeks to identify particular elements within a text by applying the grammatical rules of the language. Additionally, semantic search relies on text embedding techniques [33, 34, 43].

Although frequency-based techniques, dependency parsing, or semantic embedding offer the possibility of finding the essential elements of a text, they do not provide a clear answer to the question of higher-level relationships. Therefore, if we need to find the main terms of the text but also group them according to their content to answer higher-level questions, we need to use topic modeling. Latent Dirichlet Allocation (LDA) [44], and clustering are popular approaches to extracting individual topics. However, vector spaces created from texts using different vectorization techniques are typically challenging to cluster, but some clustering methods can also be applied to these unique vector spaces. For example, k-means [45], agglomerative [46], DBSCAN [47], and HDBSCAN [48] clustering. In addition to all this, there are more sophisticated solutions that categorize texts by combining neural networks and other methods. Examples include Top2Vec [49] or BERTopic [50, 51].

The information-extraction techniques are examined in Chapter 4 to optimize the model's performance.

2.7 Metrics

To optimize machine learning models, we must determine the measures we use for the observation error. I validated the improvement in the performance of the solutions I developed using the following measures.

During the models ' training and evaluation, I used Accuracy, Precision, Recall, F1-score, Mean Squared Error, and Categorical Cross-Entropy as general metrics [3, 52]. For the vectors generated from the texts, I used cosine similarity [32].

In addition, I used Bilingual Evaluation Understudy (BLEU) [53], Metric for Evaluation of Translation with Explicit (METEOR) [54], and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [55] metrics so that I can measure the performance of the domain-specific fine-tuning and RAG. The BLEU is an algorithm devised to assess the fidelity of text that has undergone machine translation from one natural language to another. The METEOR is an automated metric for evaluating machine translation, built upon a broadened notion of unigram correspondence between machine-generated and human-crafted reference translations. ROUGE is a set of metrics to assess automatic summarization and text translation models in NLP.

To optimize machine learning models, it is important to determine the measures that we use for the observation error. I validated the improvement in the performance of the solutions I developed using the following measures.

Chapter 3

Predicting the direction of the oil price trend using sentiment analysis

3.1 Introduction

In this chapter, we introduce a natural language processing model that uses machine learning and general statistics to predict the price of exchange-traded products. Our model focuses on forecasting the daily trend of oil prices using Twitter data. The model is built on transformer-based techniques within a recurrent neural network [56] framework, with optimization of corresponding hyperparameters. Our approach involves leveraging sentiment characteristics and vocabulary extracted from Twitter (referred to as X) [57] news. We discovered that specific news sources have a stronger correlation with changes in oil prices, leading us to refine the training data accordingly. Additionally, we implemented noise filtering to eliminate insignificant words from the textual information. As a result, we generated a data source demonstrating an 84.08% high-precision correlation with the direction of oil prices.

The remainder of the chapter is structured as follows: In Section 3.2, I will discuss our approach to collecting relevant news concerning the trend in oil prices from the Twitter community platform, along with the rationale behind the choice of this source. Subsequently, in Section 3.3, after the presentation of the data, I will delineate our methodology and the specific components utilized in our research. I will comprehensively present the experimental results in Section 3.4. Finally, Section 3.5 will offer conclusions, emphasizing the usability and the potential avenues for future development of our findings and model.

3.2 Data

The price data for Crude [58] and Brent [59] oil was obtained from Yahoo Finance [59] on July 30, 2007. We could access 3,591 daily trading data points for *Brent* and 3,648 for *Crude* which was a record level. Text data was collected using the Twitter developer platform. The researcher registration function of the developer service allows users to download 10 million tweets per month. Each tweet is limited to 280 characters, greatly simplifying our text processing and analysis tasks. Table 3.1 includes the exact names of the news sources and the number of downloaded tweets.

Table 3.1: Twitter text sources (tweets) for our analysis.

Time interval	Twitter sources	Number of tweets
2009-2023	<i>Bloomberg Markets</i>	116,293
2011-2023	<i>Bloomberg Energy</i>	23,585
2007-2023	<i>MarketWatch</i>	338,990
2010-2023	<i>Oil & Gas IQ News</i>	30,918
2009-2023	<i>Yahoo Finance</i>	241,726
2008-2023	<i>Reuters Business</i>	266,250

3.3 Methodology

3.3.1 Preprocessing

We applied a preprocessing step to both price and textual data to perform sentiment analysis on the extracted texts and compare these values to price movements.

We employed a 10-day moving average to smooth closing prices to mitigate the impact of price volatility. This method involves calculating the average of the previous ten days to define the price for the next day. Doing so effectively captures two weeks, as no trading occurs on weekends.

The current significant stock prices in the oil market are Brent and Crude oil products. We aimed to assess the hypothesis that these two products' average value can be considered a single oil price movement. To do so, we analyzed by calculating the Kendall [60], Pearson [61], and Spearman [62] correlation coefficients (refer to Table 3.2) between the prices of Crude and Brent oil products. The computed mean correlation value was 0.94, indicating a strong positive correlation between the two.

Table 3.2: Correlation values between the oil products.

Correlation coefficient	Result
Kendall	0.88
Pearson	0.97
Spearman	0.98
Mean of Kendall, Pearson, and Spearman	0.94

With the text cleaning, our goal was to support sentiment models as much as possible in making the right decision. We removed the multiple-typed space characters to do this and filtered out the special

characters. We also removed the HTML links from the text. Finally, we converted all text to lowercase.

Since we received news from users that was not only oil-related, we introduced a simple filter step to the workflow. We only considered tweets containing the word *oil* relevant. With this simple solution, we could significantly reduce the number of records to be scanned and accelerate further processing. We want to note that we have considered the risk that news related to other energy carriers or market events could also affect the price of oil.

As a final step, in the case of text data, we looked at how much oil news a Twitter user had published each year. Except for *Oil & Gas IQ News*, the amount of usable news is over 500 per year, which has been available to us since 2017 from the sources. For the *Oil & Gas IQ News* source, this was more of a 2014-2017 period. For this reason, for the sake of uniform measurement, we have removed the user *Oil & Gas IQ News* as a potential news source.

3.3.2 Sentiment analysis

We chose three types of models for the sentiment analysis of the texts. These models are Vader, the Sentiment Roberta Large English (SRLE) [63], and Twitter Roberta Base Sentiment (TRBS) [64]. The developers of the NLTK [65] software library implemented the Vader model. The SRLE and TRBS models are BERT-based architectures published in the huggingface [66] software library. The Vader and TRBS models have been optimized specifically for text data created by social networking sites, especially TRBS, explicitly designed for tweets.

The Vader model has four outputs: positive, negative, neutral, and compound. Each output is between 0 and 1, depending on how strong the particular sentiment is. The compound value brings the positive, negative, and neutral values to a uniform scale of 0 to 1, so we used this output of the Vader model by rescaling to an interval between -1 and 1. Values closer to 1 are more optimistic, while values closer to -1 mean

more negative sentiment. TRBS, like Vader, used positive, negative, and neutral values to characterize sentiment. Therefore, for this model, the compound value was generated using the ratio of negative and positive values fixed on the scale of -1 to 1. SRLE uses two outputs: the probabilities of positive and negative sentiments. In the case of SRL, we mapped the sentiment values from the two outputs, similar to those applied in the case of TRBS.

We applied each model to the texts of all Twitter sources. To determine which ones best characterize the oil price, we aggregated them by day. The sentiment values for the given day were collected and averaged during the aggregation. That means an estimate of a model's sentiment for a given day (which is nothing more than the model's assumed direction of oil price movements) was equal to the average sentiment values for all the news.

3.3.3 Correlation

We calculated the correlation between the oil price movements and the models' predictions to examine the characteristics of a given source for the Vader, TRBS, and SRLE models. We mapped the oil price for simple movement directions to investigate the correlation. That means if the difference between the closing value and the opening value on that day was positive, in other words, the price moved upwards, we assigned a value of +1; otherwise, if the difference was negative, we assigned a value of -1. In other words, we used only two labels: -1 and +1 (without any values in between). With this transformation, we mapped the price movement to correspond to the sentiment values moving between the -1 and +1 values.

This transformed the relationship between the calculated sentiment values and movement directions into a binary classification task, and we then measured its accuracy. Accordingly, we created an accuracy matrix where cells are accuracy values calculated between predictions of a sentiment analysis model (-1 or +1) and the actual price move-

ment directions (-1 or +1). The investigation found that the average performance of the Vader, TRBS, and SRLE models is similar if we measure this uniformly for all sources (see Table 3.3).

Table 3.3: Average accuracy between sentiment models and oil price direction.

Models	Average accuracy
Vader	0.547
TRBS	0.553
SRLE	0.575

3.3.4 Time-based diversity

Each news source’s content is decided by humans. Therefore, it is very likely that different considerations may decide what news is posted. This means that we can intuitively say that the accuracy of each news source can vary over time. To examine this issue, we also performed our accuracy-based correlation analysis every month. Table 3.4 shows how often a particular model outperformed all other models for different sources in a given month.

Table 3.4: Comparison of models based on resource-based performance.

Twitter source	Vader	TRBS	SRLE
<i>Bloomberg Markets</i>	3	3	4
<i>Bloomberg Energy</i>	1	2	3
<i>MarketWatch</i>	4	2	10
<i>Reuters Business</i>	4	6	4
<i>Yahoo Finance</i>	5	2	5

As a result, we have found clear intermittent trends in which certain news sources are more representative of oil price movements.

Depending on this – by using the monthly aggregated data from all news sources – we can produce a more accurate, diverse, but uniform dataset. We are, therefore, creating a new dataset, into which we have selected and mixed data from different news sources as monthly breakdowns. Namely, we can measure which source was the most accurate each month. Therefore, we can check which month and source to include in the dataset. The accuracy of our time-based diverse dataset generated in this way reached 75.6%.

3.3.5 Noise filtering

Based on other human factors, we hypothesized that there may be news that typically distorts the sentiment index. If we consider such news textual noise, we can filter it out. We tried to find such news by building a dictionary.

To build the dictionary, we first had to clean up the text. Therefore, stop words [32] and special characters have been removed from the news. Furthermore, the remaining words were lemmatized [32]. We searched for all the news on the day where the daily sentiment value and the oil price direction differed. We separated the news that pushed the daily value in the wrong direction from those that pointed in the right direction. We looked at these as significantly noisy news. Then, we built a dictionary from the bands of significantly noisy news by counting the frequency of each word. We then dropped all words from the dictionary that did not have a frequency greater than 1. We also threw away all the words in the news whose sentiment was not contrary to the oil price that day.

The dictionary was then used as a filter, and we threw out all the news that contained a word that was also part of our dictionary. With this addition, the method affected an average of 7% of the news as sources (see Table 3.5). However, it significantly increased their

accuracy: the diverse dataset generated after filtering increased from 75.6% to 84.8%.

Table 3.5: Filtered noisy records (news).

Twitter source	All records	Dropped (noisy) records	Ratio (%)
<i>Bloomberg Markets</i>	6,386	496	8
<i>Bloomberg NRG</i>	6,517	399	6
<i>MarketWatch</i>	7,067	286	4
<i>Yahoo Finance</i>	4,961	303	6
<i>Reuters Business</i>	8,187	652	8

3.3.6 Training and Test Dataset

For the training – due to the time series prediction – we created the train and test dataset using the out-of-time sampling, separated in a ratio of 80:20. In other words, we used the first 80% of the entire dataset without mixing for the training dataset and 20% for testing. In this splitting method, we also created a baseline and sentiment-boosted dataset.

The baseline dataset included open, close, high, low, and volume values related to oil prices. The value to be predicted was selected as the value of the price direction smoothed by the 10-day moving average. We used the opening and closing values to calculate whether the price would increase or decrease on a given day. The values were mapped to -1 and +1. The assigned value was +1 if the price increased and -1 if it decreased. We then smoothed the direction values using a moving average of 10 to filter the speculative noises and highlight the trend direction. With the dataset obtained in this way, we trained the models so that they would be able to predict the direction of tomorrow’s trend

from the price characteristics of the past days. The baseline dataset was supplemented with a sentiment feature we engineered following the method discussed in the previous chapters (see sections 3.3.2, 3.3.3, 3.3.4, and 3.3.5). That formed our sentiment-boosted dataset. Sentiment information, as already mentioned, fitted 84.8% with the price movement direction. However, since we now focused primarily on the trend and smoothed the direction of movement, the fitting between sentiment values and the direction of movement trend was reduced to 80%.

Because out-of-time sampling was used, the distribution of the training and test datasets needed to be uniform. We had four times more samples in the positive direction than the negative one. Namely, The test dataset was overloaded positively because the oil price trend was strongly positive during that period.

3.3.7 Models

Our testing models were designed to be optimized for time series prediction with outputs between +1 and -1. We used GRU and LSTM recurrent units, which we supplemented with a dense layer of one neuron with the hyperbolic tangent activation function. The advantage of the hyperbolic tangent activation function is that it pushes the output to be precisely between the -1 and +1 intervals we prefer. Several hyperparameters were tested in terms of the number of units. We chose these hyperparameters experimentally. The chosen number of units in the recurrent layer was 5, 15, 20, 25, 30, 35, 40, 45, or 50. We built a save and stop callback into the learning mechanism to find the best weights during the optimization process. This ensured we had a backup of the best model if the stop callback occurred. We used an early stopping of 10 as the stop callback. In addition, Adam was used as the optimization algorithm. We used mean squared error for loss and metrics throughout the learning process. However, the evaluation

was also performed by rounding the predicted output to -1 or 1 to get a deep insight into the prediction accuracy.

Furthermore, we used the RSI indicator as a baseline for a more accurate picture of our model’s performance. We calculated the given RSI index with 7 and 14 parameters daily for this. Since the RSI value ranges from 0 to 100, we mapped the RSI values to the interval between -1 and +1 with a simple bisection, and then we smoothed it using ten moving averages in the price direction to forecast the trend.

We calculated the accuracy, recall, precision, and F1 score for all the models.

3.4 Results

Our results show that, for the baseline data, the models could not learn to predict the correct direction (see Figure 3.1).

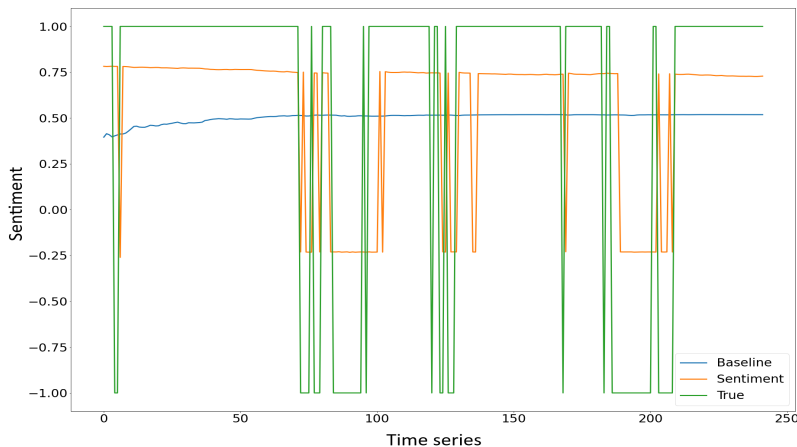


Figure 3.1: Model decision stability.

The recall value and decision characteristics clearly show that it predicted only in the positive direction and achieved 79% (see Table 3.6) average accuracy only because the test dataset had the same ratio of positive changes.

Table 3.6: Average accuracy of the different hyperparameter settings for LSTM and GRU on the test set.

Model	Accuracy	Precision	Recall	F1-score
RSI 7	0.8223	0.9271	0.8599	0.8922
RSI 14	0.8182	0.9323	0.8524	0.8905
Baseline	0.7925	0.7925	1.0000	0.8800
Sentiment	0.8714	0.9211	0.9162	0.9186

The RSI models could predict both directions, but overall, it gave an average of 82% (see Table 3.6) performance, which is worse than our sentiment-based solution. The sentiment-boosted model was more uncertain in the negative direction, but it could predict both movement directions, as shown in Figures 3.1 and 3.2.

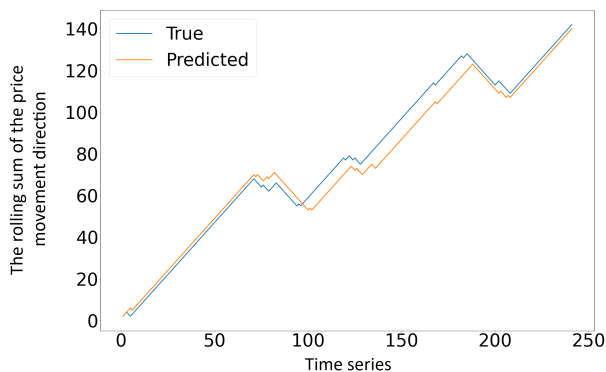


Figure 3.2: Daily forecast accuracy.

The accuracy of the models that learned from the sentiment data reached 87% (see Table 3.6) even though sentiment values estimated the direction of movement with 80% accuracy.

That means that the price data boosted with the sentiment values achieved an additional 7% increase in accuracy. Furthermore, this model achieved the best Recall and F1-score scores, surpassing all baseline and RSI models. The only exception was the baseline model, which achieved 1.0 Recall, but that was due to the model only predicting a single direction +1 every time.

3.5 Conclusions

Although the sentiments of tweets posted by different news sources capture the trend of price movements measurably, we need to consider human factors to apply them to predict price trends. Namely, the quality of news posts is not uniform but varies from period to period. However, considering these factors, we have developed a statistical-based NLP method to generate a better-quality dataset from the tweets for machine learning models. The sentiment analysis can support the forecast and increase accuracy as other indicators. However, the disadvantage is that the performance of this method is greatly influenced by the quality of each news source and our ability to remove noisy items.

Summary

This chapter introduces a natural language processing model to predict exchange-traded product prices using machine learning and general statistical methods. Our model is tailored to forecast daily trends in one of the most pivotal energy commodities, oil prices, utilizing data derived from Twitter activity. Our approach is underpinned by transformer-based techniques within a recurrent neural network framework coupled with meticulous hyperparameter optimization. Central to our methodology is extracting sentiment characteristics and vocabulary from Twitter content. Through our research, we have identified specific news sources that exhibit stronger correlations with fluctuations in oil prices. This insight has been instrumental in refining our training dataset.

Additionally, we have implemented noise filtering to eliminate insignificant words from the textual data, consequently creating a refined data source. As a result, sentiment values derived from this refined dataset demonstrated a remarkable 84.08% precision in correlating with the directional movement of oil prices.

The findings discussed in this chapter were presented at the 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS) in a paper titled 3Predicting the direction of the oil price trend using sentiment analysis. [67] [P12]

Chapter 4

A Machine Learning-based Pipeline for the Extraction of Insights from Customer Reviews

4.1 Introduction

Our primary goal was to develop a comprehensive, end-to-end system that can reliably extract critical information from customer reviews in the e-commerce domain. We wanted to design a pipeline that identifies topics and key phrases in customer feedback. The effectiveness of our approach can be measured by how well the extracted vital phrases support classification tasks. This classification-based evaluation aligns with existing methodologies, such as LDA's case, where topic modeling is evaluated by converting the results into a binary classification task [68, 69].

We investigated several techniques, including the N-gram model [70], dependency parsing [42], and embedded vector space-based keyword extraction methods to achieve this. We also explored clustering

approaches like distance- or density-based clustering and hierarchical clustering [46,71]. Additionally, we tested more advanced topic modeling methods, including LDA [68,69], Top2Vec [49], and BERTopic [50]. These methods offer a diverse range of approaches to text analysis; however, they all face limitations when applied to our specific problem. For example, LDA needs help with the dynamic nature of customer reviews, where the number of topics and their definitions can change as new data is introduced. Top2Vec and BERTopic improve LDA by providing semantically richer topic representations but suffer from implementation limitations such as the maximum number of possible topics or runtime instability.

In our experiments, the LDA, Top2Vec, and BERTopic models were useful starting points but ultimately fell short of providing a complete solution. By building on the strengths of these approaches, we developed a more adaptable solution that meets the specific demands of e-commerce text analysis.

Our pipeline integrates various methods to create a flexible pipeline for e-commerce applications. It uses a semantic embedding approach to extract keywords and key phrases from customer reviews and applies recursive hierarchical clustering to group these phrases into coherent topics. Our pipeline involves contextual factors like sentiment, relevance, and noise analysis.

The chapter is organized as follows: Section 4.2 explains data preparation. Section 4.3 outlines our methodology and keyphrase extraction techniques. Section 4.4 compares our model’s performance to state-of-the-art solutions. Finally, section 4.5 presents our conclusions.

4.2 Dataset

We used data from the Amazon Review (2018) [72], and 20newsgroups [73] dataset to evaluate our system.

The Amazon Review dataset contains 233.1 million Amazon product reviews (review text, ratings, and auxiliary votes), along with corresponding product metadata and links. For our purposes, we selected the *electronic* subset of this dataset, which contains 7,824,482 reviews, and created a dataset (hereafter Electronics_50K) as follows: We tokenized the text of the review using the vocabulary and tokenization method of the BERT neural network architecture. We kept reviews only with lengths between 16 and 128 tokens. We fixed the input size to 128 for efficiency reasons. We performed a uniform sampling to obtain 10,000 reviews for each rating (1 to 5, without fractions). This resulted in a dataset of 50,000 reviews.

The 20newsgroups is a widely used benchmarking dataset that provides training and test subsets. The dataset contains 20 classes in 6 categories. Because several classes belong to a given category.

We have created multiple different classification tasks from the Amazon Reviews dataset. Due to its relatively large size compared to the available resources, the Cell Phones and Accessories (CPA) subset was used, similar to the evaluation of the keyword extraction models. From the dataset, we selected products with a uniform set of 100 reviews for each class (ratings from 1 to 5). Thus, we created a classification task with 6000 training and 1500 testing records, which we used for a five-label multi-classification. In addition, from the CPA dataset, we took a random sample of 100 sentences.

Furthermore, using the 20newsgroups dataset, we defined binary classification tasks as follows: We chose a single class from each category to obtain sufficiently different elements for binary classification. From the 6 classes selected, we created all possible binary combinations as classification tasks, i.e., differentiating between each pair of classes.

4.3 Methodology

In this section, we present our evaluation strategy and describe how we implemented the most popular keyword extraction techniques and measured their performance to choose the method we incorporated into our system. Finally, we present our solution, which we included in a unified pipeline.

4.3.1 Keyphrase extraction techniques

We implemented solutions based on N-gram, Dependency parsing, and embedded vectors to determine which keyword extraction technique to implement in our system. We then assessed the capabilities of each technique. In subsections, we describe the details of each implementation and we describe the method and results of the evaluation of each technique, which determined the type of solution we chose.

N-gram-based approach

N-grams refer to sequences of N adjacent words within a given text or speech, where N is the number of words in the sequence (e.g., 2-grams or bigrams, 3-grams or trigrams). N-grams are widely used in NLP tasks because they help to capture local context and relationships between words. N-gram-based approaches aim to extract keywords and keyphrases by assessing the significance of N-grams that occur in a given text.

To extract keyphrases from a text using N-grams and traditional language-processing tools, we can follow the steps below:

1. Pre-process and clean the text by removing stop words [32] using an appropriate dictionary.
2. Extract nouns and their frequency from the text to create a dictionary of key terms, as they often encapsulate the main concepts.

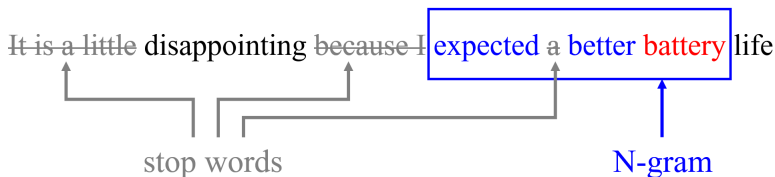


Figure 4.1: Extracting a keyphrase candidate from a simple review using an N-gram-based approach ($N=3$).

3. Extract N-grams around the identified key terms. This contextual information can provide a more nuanced understanding of the role of the noun in the text; see also Figure 4.1.
4. Select the top N-grams based on their term frequency [30] and inverse document frequency-based [31] (TF-IDF) using a threshold for the TF-IDF score.

Although it is a straightforward method, it has limitations. For instance, it cannot eliminate synonyms and variations. Furthermore, the co-occurrence of phrases should also be considered, for example, by using techniques like collocation extraction to find proper keyphrases for a given text.

Dependency parsing

In the case of this method, we break down the reviews into sentences and identify their respective keywords. Then, we look for the context of the keywords, which we achieve by applying dependency analysis. An example of dependency analysis is shown in Figure 4.2. During this process, we parse the whole sentence to obtain its grammatical structure and identify "head" words and words that are grammatically connected to them. After this, we identify the keyword kw and its "head" word h ; then, we look for words that were connected to h while keeping the original order of the words in the sentence. We

look for words connected to h instead of kw because kw was usually an adjective, adverb, or some other word that expresses emotions or qualities and has no particular meaning by itself. So instead of looking for words connected to this term, we focused on the word that is grammatically connected to kw (e.g., a noun). During the search procedure, we exclude common words, such as prepositional modifiers and words representing coordinating conjunctions. This way, the resulting phrase contained the most important parts (nouns, adjectives, etc.).

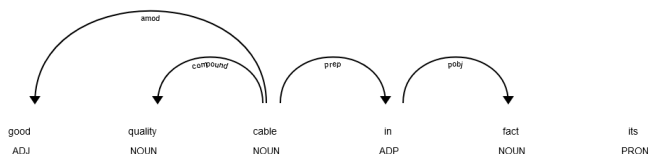


Figure 4.2: An example for dependency parsing.

This procedure results in very long phrases that are sometimes not much shorter than the original sentence. To decrease their lengths, we integrated a thresholding mechanism into the search procedure to decide which keywords to keep and which to discard. This thresholding was based solely on the sentiment of the keyword: positive keywords were kept if their sentiment score [74] was above 0.89, while negative and neutral ones were kept if their score was above 0.79. The exact thresholding levels were calculated based on our training set and were optimized based on the average length of the resulting contexts and their interpretability. Optimization was a step-by-step method with expert reinforcement. During this, starting from a still meaningful lower value of 0.49 with a step size of 0.1, we produced all possible outputs of a 100-item randomly sampled validation set, based on which the optimal parameters were defined, relying on evaluation by human experts. We performed this step-by-step optimization for both positive and negative keywords.

Embedding-based keyword extraction

One commonly used technique to measure the similarity between two words, phrases, or even sentences is to transform them into embedded vectors. The advantage of converting to a vector is that the semantic relationship between individual vectors can be measured using cosine similarity. By measuring the similarity, we can find the expressions in a given sentence whose meaning is closest to the meaning of the entire sentence.

Its main advantage is that it is less sensitive to noise thanks to the embedding. The disadvantage is that the performance depends to a large extent on the method of the valorization. So choosing the wrong embedding method can lead to bad results. One implementation of embedded-based keyword extraction is KeyBERT [75].

Evaluation of Keyword Extraction Methods

We compared the embedding, N-gram, and dependency parsing solutions to find the best keyword extraction method. For independent experts to be able to compare and evaluate each model, we produced a test dataset. For our experiments, from the Cell Phones and Accessories (CPA) dataset, we took a random sample of 100 sentences. Seven independent human experts evaluated their results.

We randomly selected 100,000 products from this subset and chose the four products with the most reviews. We introduced this step to ensure that there would be a sufficient number of reviews for the same product, including a sufficient number of explicitly positive and negative samples. The four products with the most reviews already provided a sufficient sample for further narrowing. In the next step, we removed the neutral reviews with a rating of 1 and 5. This step was introduced because our experience had shown that extracting information from extreme emotional expressions is more difficult. Words with a strong emotional charge and negative sentences can obscure the information, making it difficult to extract.

The remaining reviews were split into sentences and sub-sentences, and we corrected the grammatical abbreviations. We then removed sentences containing fewer than 6 or more than 14 words. In our experience, sentences shorter than 6 words generally do not contain meaningful information, and the 14-word upper limit (by using the splitting into sub-sentences) fits well with the length of an average English sentence. From the remaining sentences, we randomly selected 100.

Finally, N-gram, dependency parsing, and semantic embedding-based keyword extraction methods were applied to each of the 100 sentences. The keywords generated using the tested models were evaluated by independent experts, who voted for each sentence which method best extracted the meaning of that sentence. The experts could vote for multiple models or choose the "none of them" category.

The evaluation's result (see Table 4.1) showed that the embedding-based technique dominated in 61% of the cases. Therefore we implemented this method into our pipeline.

Table 4.1: Performance of keyword extraction models on test dataset made from CPA based on expert evaluations.

Model	Expert votes
None Of Them	15
Embedded	61
N-gram	15
Dependency Parsing	9

4.3.2 Pipeline

We implemented the information extraction process as a single pipeline. We designed the pipeline to ensure that the model applied can evolve with the growth and changes in the dataset. In addition, it needs to

be stable and easy to maintain so that it can be made available as a service to other applications. Figure 4.3 shows the entire pipeline.

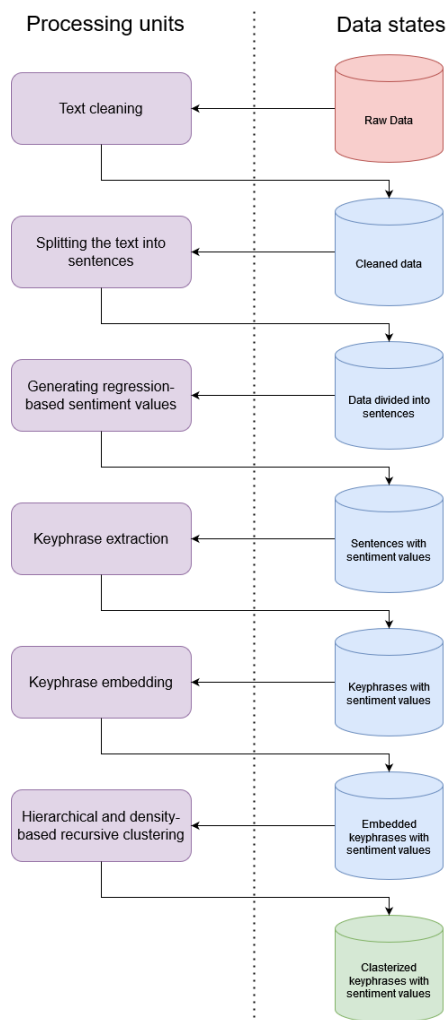


Figure 4.3: Flow diagram of the pipeline.

Text cleaning

The purpose of text cleaning is to remove characters and text elements from the raw data that do not contain relevant information and may distort the results obtained by the model. This is the first level of our pipeline, where we performed these text transformations to ensure that the text meets the tokenization requirements of the model.

As described in more detail in section 4.2, our requirement of tokenization is to work properly with the specialized BERT+R neural network. Text cleaning removes punctuation and unnecessary spaces that we think might cause noise. In addition, the entire text has been lowercase, and language-specific abbreviations have been removed to meet the tokenization requirements of the BERT+R model used.

Splitting the text into sentences

A single review may contain multiple relevant topics that express the opinion of the customer. In addition, one sentence or sub-sentence usually contains one or more statements about the same topic, and users make several statements within a sentence without using punctuation.

Therefore, in our implementation, we divided the text by sentence- and sub-sentence-ending punctuation used in English. Thus, the information carried by each statement is extracted sentence by sentence.

Generating regression-based sentiment values

Dataset labels are formed by user ratings. This means that users rated its elements on a scale of 1 to 5. This means a five-class classification. However, our goal was to create a more sophisticated solution. We have, therefore, created a model that can predict the sentiment of a sentence containing a statement on a scale of 1 to 5 with high accuracy. For this purpose, we used a pre-trained BERT network and modified its architecture by adding a regression layer to the output instead of a classification layer. We then fine-tuned the specific BERT+R model

on our dataset. The resulting model, depicted in Figure 4.4, allows us to predict sentiment values with finer granularity.

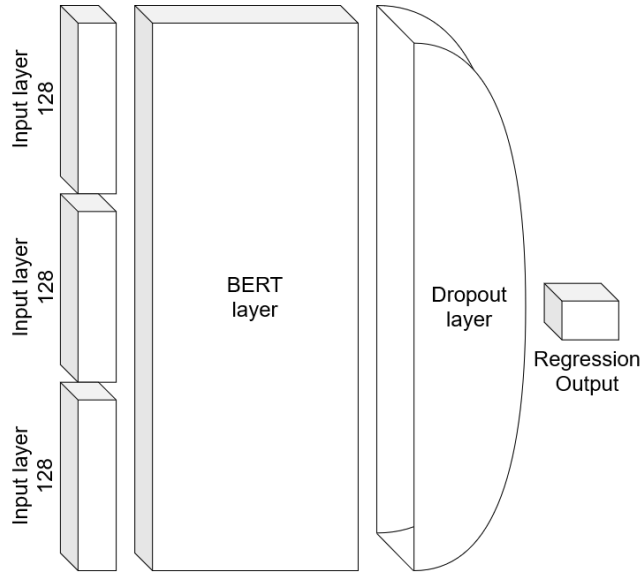


Figure 4.4: The BERT+R architecture modified for our system.

Furthermore, our system has become more sensitive to the differences between negation and assertion sentences, and this function of our system allows us to filter the texts based on sentiment.

After decomposing user reviews into sentences we predicted sentiment scores with our BERT+R model for each sentence.

Keyphrase extraction

The Sentence-BERT (SBERT) [76] is a special transformer-based architecture designed for sentence embedding. The network is designed to assign vectors to each sentence such that their similarity is measured by cosine similarity. SBERT is a modification of the pre-trained

BERT network that uses Siamese and triplet network structures to derive semantically meaningful sentence embeddings.

We applied semantic embedding to extract the keywords. For this purpose, the text was split into sentences, and the words belonging to each sentence and their bigram and trigram combinations were embedded separately in a 768-dimensional vector space using SBERT. Then, we measured the cosine distance of the given words and their bigrams and trigrams from the sentence that contained them. In the next step, we selected from each sentence the top three expressions whose semantic meaning is closest to the sentence containing it.

This gave us an item triplet for each sentence, which could contain words and phrases or their negation auxiliaries. In addition, we used combinations of bigrams and trigrams with elementary words since these are the N-gram combinations that still make sense in natural languages. In fact, by maximizing the value of N, we can eliminate redundant combinations.

Hierarchical and density-based recursive clustering

After cleaning, classifying, and extracting the key terms from the text, we highlight key terms and their embedded vectors. Finally, we applied our clustering approach to topic search in the nested vector space created from key terms.

A special property of embedded vector spaces generated from text data is that they are difficult to cluster, as the distances between adjacent vectors are usually similar. However, slightly dense clusters appear in the vector space for similar contents. To extract these slightly dense clusters, we need to remove outliers. To achieve this, we used our special hierarchical solution. This consists of using hierarchical clustering with recursive execution, as shown in Figure 4.5.

Based on our investigations, the elements of slightly dense clusters describe the same topics. Our experience shows that the cosine similarity between these clusters is above 0.7. These slightly denser

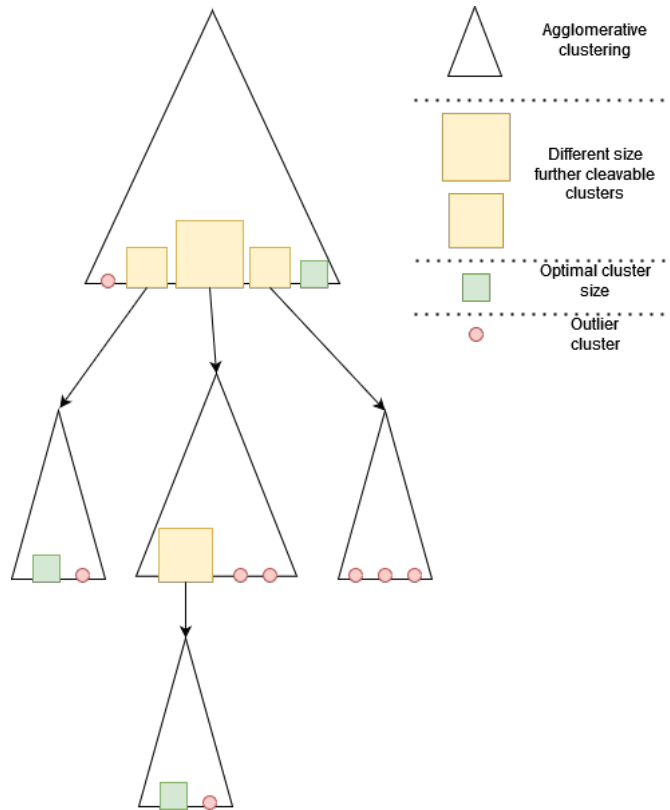


Figure 4.5: Recursive clustering considered in our system.

clusters could be extracted by recursively applying hierarchical clustering. This method re-clusters the clusters with densities below 0.7 in the next clustering cycle. This is repeated until the minimum number of elements (in our case, 5) or a density value of 0.7 is reached. Naturally, these hyperparameters of the pipeline can be freely adjusted to improve the generalization ability of our system depending on the nature of the text.

4.4 Results

To assess our own and the referenced models’ information-retention capabilities, we brought unsupervised keyword extraction back to a supervised text classification task. Evaluating unsupervised learning models often involves relating them to a supervised classification task, e.g., as seen in the case of LDA, where authors use binary classification to measure its effectiveness.

If we hypothesize that the models can extract meaningful keywords, the accuracy of an independent classification model using only those extracted keywords should be correlated with the information value of the words. This assumption is supported by the observation that text cleaning, which removes noise, improves classification accuracy. However, this improvement is only seen when elements considered noise are removed. Consequently, if words with information value for classification are eliminated, the accuracy of classification models will decrease.

It is plausible that the quality of the extracted words can impact classification outcomes. If the classification accuracy significantly drops compared to when all text elements are used, it suggests that the topics and their associated words might not have been correctly extracted. Conversely, when relevant words are extracted, classification accuracy should ideally increase. Nevertheless, this evaluation method allows for comparing the relevance of words extracted by different models for information retention.

Because of the different approaches of the different models, each model has been set up to maximize their performance as follows:

- BERTopic has limited configuration options, so we optimized the keyword extraction of the model. The BERTopic version we used allowed us to define a maximum of 30 words for a topic.
- For Top2Vec, the *mincount* parameter of the model is responsible for the number of words to extract. Therefore, we systematically

tested values from 10 to 600 and found that 500 was the maximum value where the model remained stable.

- The automatic determination of the number of topics is not built into LDA by default, so a more complex optimization was performed for this model. In LDA, a separate parameter can be used to set the number of topics searched for. This parameter was set to be between 2 and 50. Within this range, we tried both online and batch optimization techniques to find the best number of topics. The optimal topic number for LDA has the best average perplexity among the selected topics.
- In the case of our model, we left the parameters at the values we had found during the development. This means that each of the topics (sets) that our model selected had more than five elements, and the cosine similarity value between the elements of each topic was greater than 0.7. Sets outside this range were treated as outliers, i.e., irrelevant.

The topic words extracted from the models were used as a dictionary, and only these words were retained from the datasets for each evaluation of the model for the classification tasks. The texts filtered by the model dictionaries were vectorized using one-hot, count vectorization, and TF-IDF techniques. We then used a regression model for each model evaluation. The advantage of the regression model is that it is simple, and the operation of the model can be clearly explained using weights. In each case, we trained the classification models on a training dataset filtered by the dictionaries generated by the keyword extractor models, and their accuracy was measured on the test set.

In addition, since we tested several parameters for the LDA and Top2Vec models and tried to reach the maximum performance of every model, we considered their average performances. In the binary classification tasks, we measured an average accuracy of 89.82% for our model, 82.75% for LDA with batch optimization, 82.38% for LDA with

online optimization, 76.21% for Top2Vec, and 79.46% for BERTopic (see Table 4.2).

Table 4.2: The average accuracy of binary classification on binary mapped test dataset made from 20 newsgroups dataset.

Model	TF-IDF	Count vect	One-hot
BERTopic	80.21%	79.57%	78.61%
LDA (batch)	83.99%	82.09%	82.25%
LDA (online)	83.58%	81.69%	81.86%
Top2Vec	77.11%	76.04%	75.48%
Our pipeline	90.92%	89.18%	89.35%

In terms of the number of topics found, our model found 58 topics on average, LDA with batch optimization found 7 topics, LDA with on-line optimization found 7 topics, To2Vec found 3 topics, and BERTopic found 7 topics, as also shown in Table 4.3.

Table 4.3: Number of topics and vocabulary sizes for binary classification on binary mapped test dataset made from 20 newsgroups dataset.

Model	Topic number	Vocabulary size
BERTopic	7	127
LDA (batch)	7	5859
LDA (online)	7	6061
Top2Vec	3	100
Our pipeline	58	3867

We also used the 20 Newsgroups multi-task dataset without binary mapping and measured the models’ performance on it. For the multi-classification tasks, our model provided an average accuracy of 45.24%, we got 35.73% for LDA with batch optimization, 35.48% for LDA with

online optimization, 39.41% for Top2Vec and 42.8% for BERTopic; see Table 4.4.

Table 4.4: The average accuracy of multi-classification on 20 news-groups test dataset.

Model	TF-IDF	Count vect.	One-hot
BERTopic	46.13%	40.27%	42.00%
LDA (batch)	37.29%	34.87%	35.03%
LDA (online)	36.98%	34.41%	35.03%
Top2Vec	41.88%	37.89%	38.47%
Our pipeline	48.47%	43.33%	43.93%

In terms of the number of topics found, our model found 64 topics on average, LDA with batch optimization 2 topics, LDA with online optimization 2 topics, Top2Vec 17 topics, and BERTopic 94 topics; see Table 4.5.

Table 4.5: Average number of topics and vocabulary sizes for multiple classifications on 20 newsgroups test dataset.

Models	Topic number	Vocabulary size
BERTopic	92	1357
LDA (batch)	2	2242
LDA (online)	2	2275
Top2Vec	17	371
Our pipeline	64	2860

4.5 Conclusions

This study investigated the inherent challenges of extracting relevant information from unstructured free texts. We demonstrated that the widely used methods are not suitable to comprehensively address more complex problems. Our experiments confirmed the current lack of a single, universally applicable solution for keyword/keyphrase extraction and topic modeling.

In response, we developed a novel pipeline by iteratively reviewing, combining, and refining various potential solutions. This consolidated pipeline streamlines the information extraction process through efficient data cleaning, keyphrase identification, and topic organization. Notably, this framework demonstrates increased efficiency compared to existing literature models.

Furthermore, our pipeline exhibits robust information-retention capabilities. By leveraging semantic embedding, the system excels at identifying key phrases and their associated topics, delivering higher quality results than alternative solutions within the given text processing domain. Additionally, the modular architecture facilitates the seamless integration of future components, adapting the pipeline to address diverse tasks.

Summary

The efficiency of NLP has improved dramatically with the advent of machine learning models, particularly neural network-based solutions. However, some tasks are still challenging, especially when considering specific domains. This paper presents a model that can extract insights from customer reviews using machine learning methods integrated into a pipeline. For topic modeling, our composite model uses transformer-based neural networks designed for NLP, vector embedding-based keyword extraction, and clustering. The elements of our model have been integrated and tailored to better meet the requirements of efficient information extraction and topic modeling of the extracted information for opinion mining.

Our approach was validated and compared with other state-of-the-art methods using publicly available benchmark datasets. The results show that our system performs better than existing topic modeling and keyword extraction methods in this task.

The findings discussed in this chapter were presented in a paper titled "A Machine Learning-Based Pipeline for the Extraction of Insights from Customer Reviews" [77] [P3] in the Big Data and Cognitive Computing journal.

Chapter 5

A multimodal deep learning architecture for smoking detection with a small data approach

5.1 Introduction

The WHO currently estimates that smoking causes around 8 million deaths a day. 15% of people aged 15 years and over smoke in the OECD countries and 17% in the European Union [78]. Moreover, of the 8 million daily deaths, 15% result from passive smoking [79]. The studies [80,81] below highlight the influence of smoking portrayal in movies and the effectiveness of health communication models. However, quantifying media influence is complex. For internet media like social sites, precise ad statistics are unavailable. Furthermore, calculating incited and unmarked ads poses a significant difficulty as well. Methods for identifying content include continuous monitoring of advertising intensity [82], structured data generated by questionnaires [83], and AI-based solutions that can effectively support these goals. The authors

of [84] point out in their review that artificial intelligence is a powerful tool that can advance tobacco control research and policy-making.

Nonetheless, these methods are highly data-intensive. An excellent example is the popular ResNet [85] image processing network, trained on the ImageNet dataset [86] containing over 14 million images. Regarding text processing, we can mention the famous and pioneering BERT network [87] trained on the Toronto BookCorpus [88]. However, generative text processing models like GPT [4] were trained in larger datasets than BERT. For instance, the training set of GPT 3.0 was the CommonCrawl [89] dataset, which has a size of 570 GB.

We can highlight classification and object detection in image processing to detect smoking. The most famous image processing models are VGG [90], ResNet [85], Xception [91], EfficientNet [92], Inception [93], and YOLO [94]. Moreover, there are architectures like CAMFFNet [95], which are specifically recommended for smoking detection. For a more precise identification of the content elements of the texts, we can use the named-entity recognition [96] models. In addition, the development of multimodal models also gain increasing focus [97, 98], which can use texts and images to solve tasks at the same time.

We propose a multimodal architecture using pre-trained image and language models to detect smoking-related content in text and images. Combining image processing networks with these models creates a data-efficient and robust solution that can be enhanced with expert input.

The chapter is organized as follows: Section 5.2 explains our methodology. Section 5.3 we present our result. Finally, section 5.4 describes our conclusions.

5.2 Methodology

5.2.1 Model architecture

As illustrated in Figure 5.1 by a schematic flow diagram, our solution relies on pre-trained language and image processing models and can handle both textual and image data.

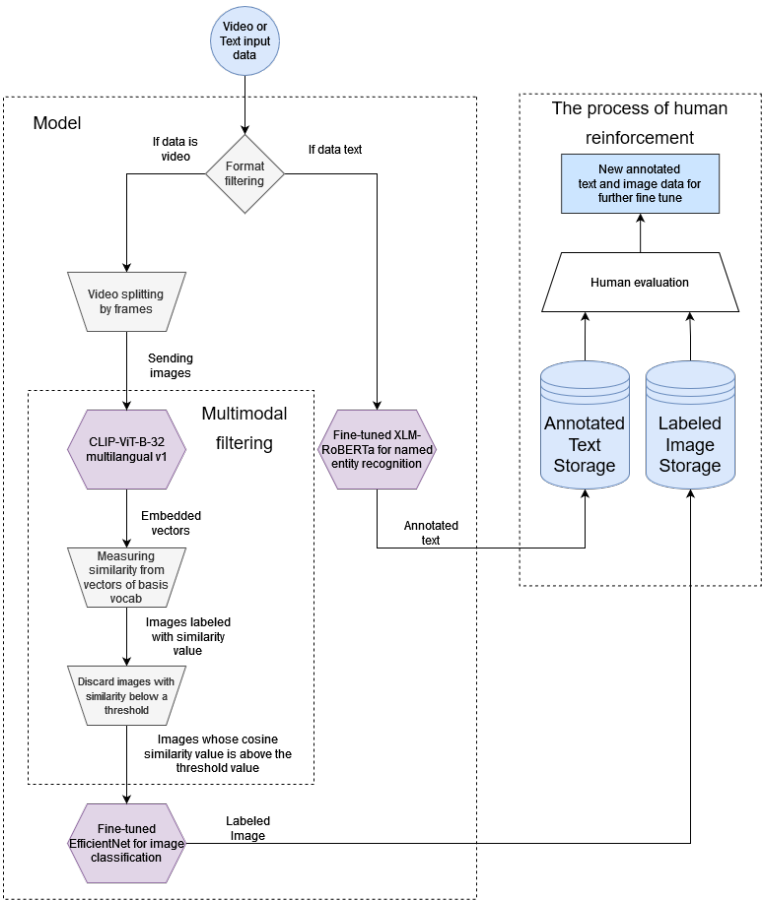


Figure 5.1: Schematic flow diagram of the architecture.

The first step of our pipeline is to define the input data format because we need to direct the data to the appropriate model regarding its format. The video recordings are analyzed with multimodal and image processing models, while the texts are analyzed with a large language model. In the case of video recordings, we applied the CLIP-ViT-B-32 multilingual [99,100] model. The model has been developed for over 50 languages with a special training technique [99]. The model supports Hungarian, which was our target language. We use the CLIP-ViT-B-32 model as a filter. After filtering, to achieve more accurate results, we recommend using the pre-trained EfficientNet B5 model, which we fine-tuned with smoking images for the classification task.

To process texts, we use named entity recognition to identify smoking related terms. For this purpose, we have integrated into our architecture an XLM-RoBERTa model [101] that is pre-trained, multilingual, and also supports the Hungarian language, which is important for us.

5.2.2 Format check

The first step in processing is deciding whether the model has to process video recordings or text data. Since there are many formats for videos and texts, we chose the simple solution of only supporting mp4 and txt file formats. The mp4 is a popular video format, and practically all other video recording formats can be converted to mp4. We consider txt files utf8-encoded raw text files that are ideally free of various metadata. It is important to emphasize that here we ignore the text cleaning processes required to prepare raw text files. The reason is that we did not deal with faulty or txt files requiring further cleaning during the trial.

5.2.3 Processing of videos and images

The next step in the processing of video footage is to break it down into frames by sampling every second. The ViT image encoder of the CLIP-ViT-B-32 model was trained by its creators for various image sizes. For this, they used the ImageNet [86] dataset in which the images have an average size of 469×387 pixels. The developers of CLIP-ViT-B-32 do not recommend an exact size for the image encoder. The model only specifies a minimum size of 224×224 . In the case of EfficientNetB5, the developers have optimized to a size of 224×224 . For these reasons, we have taken this image size as a reference and transformed the images sampled from the video recordings to this size.

5.2.4 Multimodal filtering

The images sampled from the video recordings were filtered using the CLIP-ViT-B-32 multilingual v1 model. The pre-trained CLIP-ViT-B-32 multilingual v1 model consists of two main components from a ViT [22] image processing model and a DistilBERT-based [102] multilingual language model. We convert into 512 dimensions embedded vector [103] the images and texts with CLIP-ViT-B-32. The embedded vectors for texts and images can be compared based on their content meaning if we measure cosine similarities between the vectors. Since we aimed to find smoking-related images, we defined a smoking-related term. We converted it to a vector and measured it against the embedded vectors generated from the video images. The term we chose was the word "smoking". We can use more complex expressions, which could complicate the measurement results interpretation.

The cosine similarity of the vectors produced by embedding the images always results in a scalar value compared to the vector created from our expression related to "smoking". However, the decision limit between the distances measured between the vectors produced by the CLIP-ViT-B-32 model is not always clear. Namely, even in the case of images with meanings other than "smoking", we get a value that is not

too distant. We had to understand the distribution of the smoking images to eliminate this kind of blurring of the decision boundary. To this end, we examined the characteristics of the distribution of the images. It is clear from Figure 5.2 that because the images with a semantic meaning closer to smoking appear randomly in a video recording, it is difficult to grasp the series of images that can be useful for us. Figure 5.2 has the cosine similarity values belonging to the individual images. At the same time, the horizontal axis shows the position of the images in the video. To solve this problem, we introduced the following procedure. If we put the cosine similarity values in ascending order, we get a function that describes the ordered evolution of the cosine similarity values.

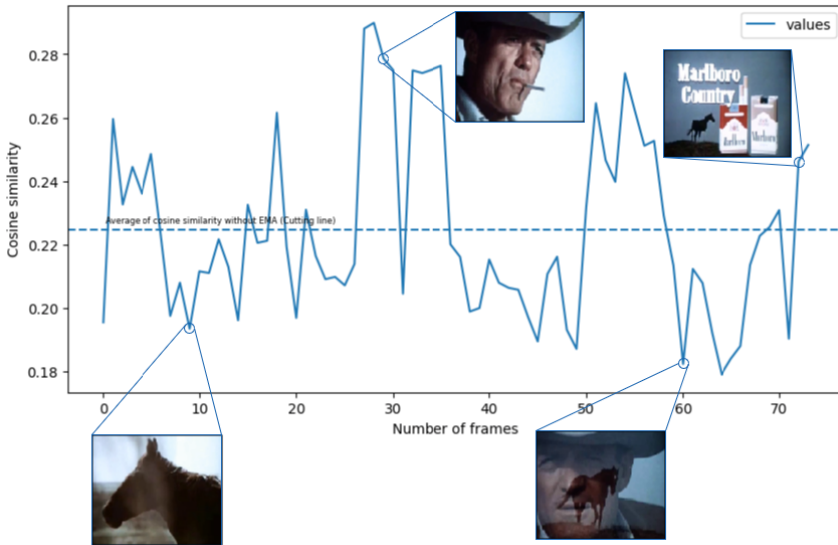


Figure 5.2: The cosine similarity of the images obtained from the video recording in chronological order.

The ordered function generated from Figure 5.2 can be seen in Figure 5.3. As shown in Figures 5.2 and 5.3, we found that if we take the similarity value of the images sampled from the given sample to the word "smoking", their average results in a cutting line, and we can use it as a filter.

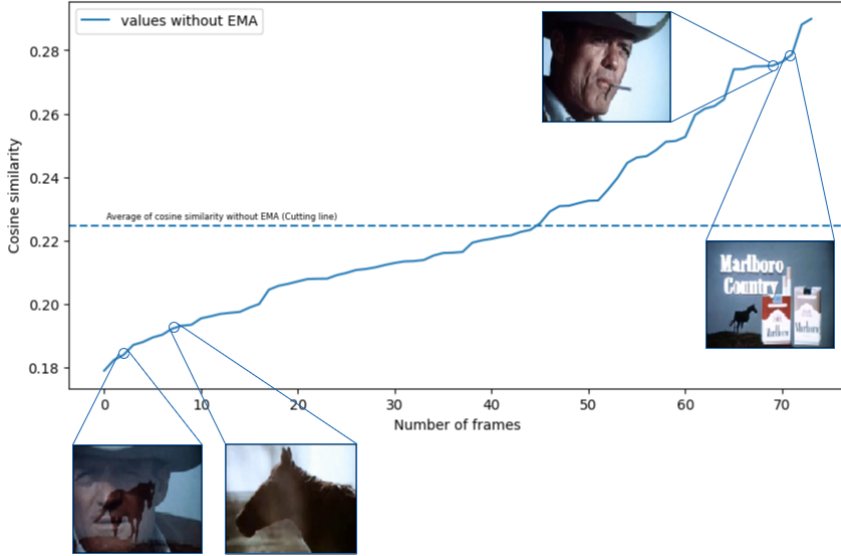


Figure 5.3: The cosine similarity of the images after the ordering.

Furthermore, considering the specifics of the video recordings, we consider that the average can be corrected with a constant value. In this mean, the constant value can thus also be defined as the hyperparameter of the model. We chose the 0 default value for the correction constant for more apparent measurements. The choice of the best constant value may differ depending on the recording type and may distort the exact measurement results.

5.2.5 Fine-tuned image classification

After filtering the image set with a multimodal model, we applied an image processing model to classify the remaining images further to improve accuracy. Among the publicly available datasets on smoking, we have used the "smoker and non-smoker" [104] for augmented [105] fine-tuning. We selected the following models for the task. EfficientNet, Inception, ResNet, VGG, and Xception. The EfficientNet B5 version performed the best with an accuracy of 93.75%.

5.2.6 Processing of text

In the case of detecting smoking terms in texts, we approached the problem as a NER task and focused on the Hungarian language, since we could not find a dataset containing annotated smoking phrases available in Hungarian. Therefore, to generate the annotated data, we used the generational capabilities of ChatGPT [106], the smoking-related words of the Hungarian synonyms and antonyms dictionary [107], and prompt engineering.

AI-based text generation is an effective tool for creating learning data, but it is important to consider possible disadvantages as well. The advantage of text generator services (such as ChatGPT or Gemini [108]) using large language models is that they can quickly and efficiently create large amounts of text and can be applied to any size dataset. This makes them ideal for creating large learning sets. Generators can create text in a variety of styles and formats, including news, articles, blog posts, and even creative writing. Their disadvantage is that they may be prone to bias and the text generated by less advanced models may contain errors, repetitions, and grammatical errors. This can lead to distorted results in the trained AI model. Successful application requires careful selection of the AI model and quality control of the training dataset.

Accordingly, we selected words related to smoking from the synonyms and antonyms dictionary, such as cigarette, king deck, smoke,

cigar, pipe, coffin nail, spangles like the factory chimney, tobacco, and cigarette. After that, we asked ChatGPT to suggest further smoking-related terms besides words from the Hungarian dictionary.

We created blocks of a maximum of 5 elements from the words in our dictionary. Each block contained a random combination of a maximum of 5 words. The blocks are disjoint, so they do not contain the same words. This mixing step was done 10 times. This means that, in one iteration, we could form 8 blocks of 5-element disjunct random blocks from our 43-word dictionary. By doing all these 10 times, we produced 80 blocks. However, due to the 10 repetitions, the 80 blocks were no longer disjoint. In other words, if we string all the blocks together, we get a dictionary in which every synonym for smoking appears a maximum of 10 times.

We made a prompt template to which, by attaching each block, we instructed ChatGPT to generate texts containing the specified expressions. Since ChatGPT uses the Hungarian language well, the generated texts contained our selected words by the rules of the Hungarian language, with the correct conjugation. An example of our prompts is illustrated in Table 5.1.

Table 5.1: An example prompt for ChatGPT.

Generate a short text about smoking. The text strictly contains the following words in the different sentences: smoking, tobacco, cigar.

We did not specify how long texts should be generated by ChatGPT or that every word of a 5-element block should be included in the generated text. When we experimented with ChatGPT generating fixed-length texts, it failed. Therefore, we have removed the requirement for this. Using this method, we created a smoking-related corpus

consisting of 80 paragraphs, 49000 characters, and 7160 words. An English example of a generated text is presented in Table 5.2.

Table 5.2: An example paragraph generated by the prompt in Table 5.1.

Smoking is a widespread and addictive habit that involves inhaling and exhaling the smoke produced by burning tobacco. Whether it's a hand-rolled cigar or a manufactured cigarette, the act of smoking revolves around the consumption of tobacco. Despite the well-known health risks, many individuals continue to engage in smoking due to its addictive nature. The allure of a cigar or a cigarette can be strong, making it challenging for people to quit smoking even when they are aware of its detrimental effects. Education and support are crucial in helping individuals break free from the cycle of smoking and its associated harms.

To find the best model according to the possibilities of our computing environment and the support of the Hungarian language, we tested the following models: XLM RoBERTa base (LLM1) and large (LLM2), DistilBERT base cased (LLM3), huBERT base [109] (LLM4), BERT base multilingual [110] cased (LLM5) and uncased (LLM6), Sentence BERT [111] (LLM7). We trained the models on a NER task. For the token classification task. That is, models must learn to identify smoking-related terms (tokens) in the text and distinguish them from non-smoking terms (tokens). The best model was the (LLM2), which achieved 98% accuracy and 96% F1-score on the validation dataset and an F1-score of 91% with an accuracy of 98% on the test dataset.

5.3 Results

We collected video materials to test the image processing part of our architecture. The source of the video materials was the video-sharing site YouTube. Taking into account the legal rules regarding the usability of YouTube videos, we have collected 5 pieces short advertising films from the Malboro and Philip Morris companies. We ensured not to download videos longer than 2 minutes because longer videos, such as movies, would have required a special approach and additional pre-processing. Furthermore, we downloaded the videos at 240p resolution and divided them into frames by sampling every second. Each frame was transformed to a resolution of 224×224 pixels. We manually annotated all videos. The downloaded videos least 64 seconds and contained 13 seconds of smoking an average.

With the multimodal filtering technique, we discarded the images that did not contain smoking. Multimodal filtering found 25 seconds of smoking on average in the recording. The accuracy of the identified images was 62%. The multimodal filtering could filter out more than half of the 64-second, on average, videos. We also measured the performance of the fine-tuned EfficientNet B5 model by itself. The model detected an average of 28 seconds of smoking with 60% accuracy. We found that the predictions of the two constructions were sufficiently diverse to connect them using the boosting ensemble [112] solution. By connecting the two models, the average duration of perceived smoking became 12 seconds with 4 seconds on average error and 74% accuracy. The ensemble solution was the best approach since the original videos contained an average of 13 seconds of smoking. Tables 5.3, 5.4, and 5.5 contain the exact quantitative results broken down into test videos.

Table 5.3: Performance of multimodal filtering on the test videos.

Video	Full length in seconds	Length of the smoking scene in seconds	Accuracy
Malboro 1	75	21	0.8919
Malboro 2	50	5	0.5200
Malboro 3	87	12	0.6781
Philip Moris 1	35	15	0.4286
Philip Moris 2	75	13	0.5730

Table 5.4: Performance of the EfficientNetB5 on the test videos.

Video	Full length in seconds	Length of the smoking scene in seconds	Accuracy
Malboro 1	75	21	0.6892
Malboro 2	50	5	0.4600
Malboro 3	87	12	0.6437
Philip Moris 1	35	15	0.5433
Philip Moris 2	75	13	0.6933

We created training and validation datasets from Hungarian synonyms for smoking using ChatGPT. We trained our chosen large language models until their accuracy on the validation dataset did not increase for at least 10 epochs. The (LLM1) model achieved the best performance on the validation dataset with an F1-score of 96% and 98% accuracy. For the final measurement, we created test data from an online text related to smoking by manual annotation [113].

Table 5.5: Performance of the Ensemble model (Multimodal Filtering + EfficientNetB5) on the test videos.

Video	Full length in seconds	Length of the smoking scene in seconds	Accuracy
Malboro 1	75	21	0.9054
Malboro 2	50	5	0.6600
Malboro 3	87	12	0.7931
Philip Moris 1	35	15	0.5710
Philip Moris 2	75	13	0.7600

The fine-tuned (LLM1) model achieved 98% accuracy and 0.91 F1 score on the test dataset. The measurement results of the chosen models can be viewed in more detail in Tables 5.6 and 5.7, where, in addition to the accuracy and F_1 -score values, the recall, precision, and cross-entropy loss values can also be found.

Table 5.6: Results of a fine-tuned (LLM2) for the named entity recognition task on the test data.

Metrics	Results
Cross-entropy loss	0.0736
Precision	0.9115
Recall	0.9035
F1-score	0.9075
Accuracy	0.9825

Table 5.7: Validation of language models on the name entity recognition task.

Models	Cross entropy loss	Precision	Recall	F1- score	Accuracy
LLM1	0.176	0.9580	0.9650	0.9615	0.9840
LLM2	0.140	0.9511	0.9669	0.9590	0.9829
LLM4	0.143	0.9326	0.9499	0.9412	0.9787
LLM3	0.101	0.9348	0.9597	0.9467	0.9784
LLM5	0.096	0.9450	0.9666	0.9557	0.9821
LLM6	0.186	0.9170	0.9442	0.9304	0.9746
LLM7	0.152	0.9315	0.9301	0.9308	0.9724

5.4 Conclusions

Multimodal and image classification models are decisive for classification tasks. In return, however, they are complex and require substantial training data, which can reduce their explainability and usability. Our solution showed that pre-trained multimodal and image classification models exist that allow smoking detection even with limited data and in low-resource languages if we use the potential of human reinforcement, generative, and ensemble methods. In addition, we see further development opportunities if our approach is supplemented with an object detector, which can determine the time of occurrence of objects and their position. Moreover, with the expected optimization of the automatic generation of images in the future and the growth of the available computing power, our method used for texts can work in the case of images.

Summary

Covert tobacco advertisements often raise regulatory measures. This chapter presents that artificial intelligence has great potential for detecting hidden advertising and allows unbiased, reproducible, and fair quantification of tobacco-related media content. We proposed an integrated text and image processing model based on deep learning and generative methods, which can detect smoking cases in textual and visual formats, even with little available training data. Our model can achieve 74% accuracy for images and 98% for text. We thereby confirmed that using pre-trained multimodal, image, and text processing models available through deep learning enables the detection of smoking in different media, even with a small training dataset.

The findings discussed in this chapter were presented in a paper titled "A multimodal deep learning architecture for smoking detection with a small data approach" [114] [P2] in the Frontiers in Artificial Intelligence journal.

Chapter 6

Investigating the domain transfer techniques for the development of LLM-driven knowledge-based systems

6.1 Introduction

Transformer-based large language models (LLMs) have significantly advanced the field of natural language processing (NLP) by achieving unprecedented performance across various linguistic tasks. Notable models such as BERT [87], GPT [87], and XLNet [115], along with their respective derivatives, have set new benchmarks in areas like text classification, machine translation, and question-answering. Among these, generative models, particularly GPT, have demonstrated outstanding capabilities in comprehending and generating human-like text [116]. OpenAI's GPT architecture, as seen in widely adopted services like ChatGPT [106] and Microsoft's Copilot [117], has become a benchmark for generative performance while competing models such as

Google’s PaLM [118], Gemini [108], and Meta’s open-source LLaMA model [119] are emerging as potent alternatives.

This surge in LLM capabilities has profoundly influenced the evolution of knowledge-based systems. Historically, such systems depended on structured databases and traditional querying methods to extract information [120]. However, the contextually aware text generation capabilities of LLMs present a new paradigm, extending the potential of these systems beyond mere data retrieval to performing reasoning and knowledge synthesis. This paradigm shift enables more advanced applications, particularly in question-answering and information-retrieval tasks [121].

Two primary approaches have emerged to tailor LLMs for domain-specific applications: domain-specific fine-tuning (DFT) and retrieval-augmented generation (RAG). DFT allows models to specialize in particular areas by training on domain-specific datasets, improving performance in tasks like machine translation [122] and domain-relevant applications [123]. RAG complements this by integrating external information from relevant datasets into the prompt, enhancing the model’s accuracy and reducing the need for frequent retraining [124].

However, integrating these techniques into knowledge-based systems remains a complex task. Standardized methodologies are needed for designing systems that leverage LLMs, particularly in domain-specific contexts. This gap underscores the need for further research and clear guidelines. In response to these challenges, this study aims to propose a structured methodology for constructing knowledge-based systems utilizing LLMs.

The chapter is organized as follows: Section 6.2 covers databases for optimization and method selection, Section 6.3 discusses the methodology, Section 6.4 shows empirical results, and Section 6.5 summarizes findings and conclusions.

6.2 Data

We applied two approaches to preparing the data. First, we examined how to create datasets for PDF and Microsoft Word-based scientific publications. The primary motivation for creating our datasets was to build our system, the effectiveness of which we can support with measurement results. Second, we composed another dataset for the measurements besides the own-created data. This second dataset was collected from publicly available data.

For the dataset we created, we curated a collection of specific publications on urban monitoring and corn cultivation with the help of the National Library of the University of Debrecen and the Faculty of Agriculture, Food Science, and Environmental Management. This corpus, comprising 69 pieces of literature on corn cultivation (CORN) and 83 on urban monitoring (UB), provided a rich source of domain-specific terminology and concepts. Each article or book was available to us in PDF or Word format.

As an independent and open-access dataset, we utilized the CORD-19 dataset [125] and MedQuAD [126] [127]. A CORD-19 is a freely available repository of thousands of scientific articles on COVID-19, SARS-CoV-2, and related coronaviruses. This dataset encompasses thousands of scholarly articles. It is in JSON format and represents about 80 GB of text data. MedQuAD includes 47457 medical question-answer pairs created from 12 NIH websites (e.g., cancer.gov, GARD, MedlinePlus Health Topics). The collection covers 37 question types (e.g., Treatment, Diagnosis, Side Effects) associated with diseases, drugs, and other medical entities such as tests.

Subsection 6.3.3 describes in more detail the data preparation processes for the model’s FN and the RAG application.

6.3 Methodology

To decide whether RAG or DFT is the better approach to creating an LLM-based system, we used the models presented in section 6.3.1. The performance of these models was measured using the metrics given in section 6.3.2 for both RAG and DFT applications. For these measurements, we describe in section 6.3.3 how we prepared the data, and in sections 6.3.4 and 6.3.5, we show which settings were applied for DFT and RAG.

6.3.1 Models

We have selected the LLM models according to the following requirements:

- The models must be LLM, well-documented, and have freely downloadable pre-trained versions.
- The models have been well implemented. That means they should be part of the model repository of the HuggingFace [66] and PyTorch [128] development libraries.
- A resource provided by the NVIDIA DGX A100 System should be sufficient to train and test the models.

Based on these criteria, we selected the GPT-J-6B [129], OPT-6.7B [130], LLaMA-7B [119], and LLaMA2-7B [119] models.

6.3.2 Selected metrics

The following metrics were used to determine the performance of the different language models: BLEU, ROUGE, METEOR, and cosine similarity.

BLEU is used to measure machine translation performance. BLEU measures n-gram accuracy, which means it counts how many n-grams

of the generated text are found in the reference translation. ROUGE is used to measure the performance of machine translation and text summarization tasks and measures recall, which means that it counts how many n-grams of the reference translation are found in the generated text. ROUGE is designed to work around some of BLEU’s limitations. Namely, ROUGE places more emphasis on recall than BLEU and better takes into account the meaning of the text. The METEOR score is a metric used to evaluate the quality of machine translation, text summaries, and creative text formats. It considers the Recall, Precision, and word order of factors to determine how efficient a model is.

Cosine similarity is also used to measure the similarity of texts. To use it, the text must be converted into sentence or word vectors, then the cosine similarity between the vectors must be calculated. A higher cosine similarity means that the texts are more similar semantically to each other. We applied this approach by dividing the generated and reference text into sentences with our system and then converting the individual sentences into embedded vectors using the MiniLM L6 v2 [131] sentence transformer.

For a formal description of our method applying cosine similarity, let $R = \{r_1, \dots, r_M\}$ and $G = \{g_1, \dots, g_M\}$ denote the reference and generated text, respectively, broken down to the same number of M corresponding paragraphs r_i and g_i with $i = 1, \dots, M$. All the reference and generated paragraphs consist of sentences represented by their embedded vectors $r_i = \{rs_{i,1}, rs_{i,2}, \dots\}$ and $g_i = \{gs_{i,1}, gs_{i,2}, \dots\}$ ($i = 1, \dots, M$). For each reference sentence $rs_{i,j}$ we find the best matching generated sentence in the corresponding paragraph g_i and calculate their similarity. Finally, as an overall similarity score (CS) between the reference text R and the generated one G , we calculate the average similarity regarding all the composing paragraphs as

$$CS = \sum_{i=1}^M \frac{\max_k (Cosine(rs_{i,j}, gs_{i,k}))}{M}. \quad (6.1)$$

6.3.3 Data preparation for RAG and DFT

We used different data preparation approaches for the RAG and DFT. For DFT, we considered the Q&A dataset-based training method (as it was used for the Stanford Alpaca [132]) as a guiding principle. In addition, we created easily searchable datasets for RAG to identify contexts that support effective responses.

Q&A datasets for DFT

To prepare Q&A datasets, we split the collected CORN and UB datasets into paragraphs. In the next step, we converted them to raw text and cleaned them with the help of human experts.

We extracted a subset of CORD-19 data. We selected the articles based on the following filter criteria: articles must have abstracts, be included in the PubMed Central repository, and have open access to medical biology and life science. Articles must also have an arXiv ID, which strengthens open access. Articles must not contain latex elements so they can be readable easily and validated by human experts.

In the MedQuAD dataset, there are incomplete question-answer pairs. Therefore, we kept only those records that the questions and their corresponding answers, types, and focus values were fully available.

We divided our datasets CORN, UB, and CORD-19 into paragraphs, taking into account each model’s tokenizer. When dividing the paragraphs, we worked so that the individual text parts could not be longer than 256 tokens according to any model’s tokenizer.

To create the questions of the Q&A dataset, we used the BERT-based generator [133]. We generated five questions for each paragraph. Duplicates were filtered and removed to ensure the two questions differed for the same paragraph. Thus, we created a maximum of five but at least one question in the database for each paragraph. With this, we applied an oversampling to the dataset. We also created a context-extended version for each dataset by adding context to the questions.

We created the contexts so that they contain answers to the questions. We fine-tuned the models on this dataset, and the models had to learn how to extract the answer from the context as efficiently as possible when it was available.

Since the MedQuAD data had expert-validated question-answer pairs, therefore, to ensure that they could not be semantically damaged, but in terms of their length, they are the same as the length of the answers included in the other experimental data sets, we discarded those pairs where the answers were longer than 256 tokens.

Table 6.1 lists the number of paragraphs and questions in the created Q&A ($Q\&A_{CORN}$, $Q\&A_{UB}$, $Q\&A_{CORD-19}$, $Q\&A_{MedQuAD}$) datasets.

Table 6.1: Number of paragraphs and questions of Q&A datasets.

Dataset	Paragraphs	Questions
$Q\&A_{CORN}$	7058	28790
$Q\&A_{UB}$	8553	27974
$Q\&A_{COVID}$	18004	58290
$Q\&A_{MedQuAD}$	7871	7871

RAG datasets

The performance of RAG is highly contingent on the precision of the contextual information used to generate answers. To rigorously assess RAG’s capabilities, we employed two distinct methodologies.

First, we utilized the $Q\&A_{CORN}$, $Q\&A_{UB}$, $Q\&A_{CORD-19}$, and $Q\&A_{MedQuAD}$ datasets, initially developed for DFT. We ensured that each paragraph was accompanied by at least one corresponding question in these datasets. Each generated question was transformed into a vector representation using the MiniLM L6 v2 sentence transformer [111]. This allowed us to measure semantic similarity using our CS metric. In this setup, answers were drawn from paragraphs

containing questions most similar to the reference, forming our first indexed dataset type (ID_q).

In the second approach, we segmented each answer into individual sentences and embedded these sentences using the same MiniLM L6 v2 sentence transformer. We excluded sentences shorter than 10 words or longer than 30 to optimize embedding efficiency. This enabled us to manage each sentence as a vectorized index, forming our second type of indexed dataset (ID_s).

For all the datasets CORN, UB, CORD-19, and MedQuAD, both types of indexed datasets (ID_q and ID_s) were created. Table 6.2 summarizes their properties.

Table 6.2: Number of vectorized sentences and questions of indexed datasets.

Dataset	Sentences (ID_s)	Questions (ID_q)
ID^{CORN}	37874	28790
ID^{UB}	40002	27974
ID^{COVID}	56861	58290
$ID^{MedQuAD}$	32310	7871

Training, validation, and test datasets

We divided the $Q\&A_{CORN}$, $Q\&A_{UB}$, $Q\&A_{CORD-19}$, and $Q\&A_{MedQuAD}$ datasets into training and validation sets in an 80/20 ratio. Due to our question-and-answer generation methodology, a single answer may correspond to multiple similar questions. Therefore, we ensured that each question and its corresponding answer appeared only once in the validation dataset.

To evaluate the models’ inference abilities, we created distinct test datasets. For this, we used questions and corresponding answers unseen by the models during fine-tuning. We employed topic modeling

based on nested vectors to achieve this, generating three test datasets from the $Q\&A_{CORN}$, $Q\&A_{UB}$, and $Q\&A_{CORD-19}$ datasets.

In the first step, we embedded all sentences of the ID_s datasets with the Sentence Transformer. We then reduced the dimensionality of these embeddings from 386 to 2 using UMAP [38] and subsequently clustered them using HDBSCAN [48]. The clustering parameters were set to a maximum of 15 clusters and a minimum of 6 because the optimal results were obtained when clusters contained approximately 256 tokens from sentences of 10 to 30 words. Texts outside this word range were excluded.

For FN, we split the datasets ($Q\&A_{CORN}$, $Q\&A_{UB}$, $Q\&A_{COVID}$) into training and validation datasets in an 80/20 ratio. We used a special approach for resolution. We did not simply split the datasets, but from those question-answer pairs where we had more than 1 reference question, we selected as many corresponding 20% of the entire dataset. With this, we achieved that the validation accuracy of the models measures the ability of association of the models. The inference ability of the models was measured on the test dataset. After the mapping and clustering, the outlier clusters were removed. We then measured the exact number of tokens contained in each cluster with the tokenizer of each model, and we removed clusters with more than 256 tokens. Finally, we generated questions for the created clusters. The test data created contained 279 (CORN), 311 (CORD-19), and 249 (UB) question-answer pairs.

We utilized the predefined question types for the $Q\&A_{MedQuAD}$ dataset, which consists of expert-curated question-answer pairs, to generate the training, validation, and test datasets. Specifically, 80% of each question type was allocated to the training set, while 20% was reserved for validation and testing. From this 20%, we selected 300 entries for the test dataset, ensuring its size was comparable to the other test datasets.

6.3.4 Fine Tuning settings

For training, we worked on an NVIDIA DGX computing system. We fine-tuned our models using standard Hugging Face training code with the following belief hyperparameters in the case of all models: loss function to categorical cross-entropy, batch size to 4, learning rate $2e-4$, epochs 5, and max length 256.

6.3.5 RAG Settings

We use the LLAMA-2-7b model for RAG, which has been fine-tuned by instruction and human reinforcement. Therefore, it is well suited for RAG. Since this does not hold for the models GPT-J-6b, OPT-6.7b, and LLaMA-7 b, we excluded them from measuring RAG performance. LLaMA-2-7b has a maximum input size of 4096 tokens. The size of the input determines the size of the attachable context. For this reason, we introduced a limit filter to the size of the context.

The filter’s task is to truncate search results that result in an input to the model longer than 4096 tokens. The input’s length depends on the question’s length and context. Therefore, the filter dynamically manages the length of the allowed context as a function of the size of the question.

Furthermore, a threshold parameter is introduced to control the number of context items found. The possible values of the threshold parameter range between 0 and 1 and context items with a cosine similarity below the threshold are automatically dropped. The threshold was applied with different values during the evaluation (see section 6.4).

6.4 Results

Our evaluation strategy was to measure the models’ ROUGE, BLEU, and METEOR scores. Moreover, we calculated the CS of the gener-

ated responses compared to the reference responses according to cosine similarity, and (6.1). In addition, during the evaluation, we followed the following strategies for measuring fine-tuned models and RAG.

We fine-tuned the GPT-J-6b, OPT-6.7b, LLaMA-7b, and LLaMA-2-7b models with the datasets $Q\&A_{CORN}$, $Q\&A_{UB}$, $Q\&A_{CORD-19}$, $Q\&A_{MedQuAD}$ to make them domain-specific. For the DFT of the models, we measured the validation accuracy at the end of each epoch and saved the models. We only evaluated the best-performing models on the test datasets. To do this, we passed all the questions from the test datasets to the most accurate models. Finally, we calculated BLEU, ROUGE, and METEOR scores and CS values between the responses generated by the models and the reference responses.

The DFT process was conducted as outlined in section 6.3.4. The models consistently achieved their lowest validation loss between epochs 2 and 4 during the fine-tuning. The LLaMA-7b model was the best on the $Q\&A_{CORN}$ and $Q\&A_{CORD-19}$ datasets. On the $Q\&A_{UD}$ and $Q\&A_{MedQuAD}$ datasets, LLaMA-2-7b performed the best. The GPT-J-6b and OPT-6.7b models learned with a higher validation loss than the LLaMA models.

In the evaluation of RAG, the context injected and the content of the context are critical. We defined the filtering by a threshold value based on cosine similarity. The threshold value specified what was considered relevant information during the search in the dataset. As described in the section dealing with data preparation, we worked with two types of datasets (ID_q , ID_s). The measurements were made for all datasets. The threshold values were defined on a scale from 0 to 1 with a step of 0.1. This meant that in the case of any question, we discarded matches worse than the threshold value. Precisely, for example, in the case of a threshold value of 0.5 and a given question taken from the test dataset, only those paragraphs (ID_q) or sentences (ID_s) passed the filter whose indices showed a cosine similarity greater than 0.5 compared to the reference question. This also means that in the case of a threshold of 0, everything, and in the case of a threshold

of 1, only the 100% semantic match is accepted. The sentences (ID_s) or paragraphs (ID_q) that passed the filter were packaged in a uniform context in descending order of similarity and handed over to the model to try to answer the given question based on it. If the size of the packed context was more significant than the input of the given model allowed, the context was cut off at the maximum input size.

Through these rules, we controlled the size and quality of the context. We generated answers to all questions in the test dataset using all the indexed databases

$ID_q^{CORN,UB,CORD-19,MedQuAD}$ and $ID_s^{CORN,UB,CORD-19,MedQuAD}$. Finally, we calculated BLEU, ROUGE, and METEOR scores and CS values between the responses generated by the models and the reference responses.

We summarize our measurement results in radar plot 6.1, illustrating the models' relative performance. Furthermore, Table 6.3 presents the average performance of each model approach.

Table 6.3: Average scores of each approach.

Models	ROUGE	METEOR	BLEU	CS
Baseline	0.135164	0.129689	0.002965	0.362269
DFT	0.247349	0.260628	0.060082	0.404662
RAG	0.289601	0.245677	0.068024	0.551685
RAG with DFT	0.229914	0.220869	0.043652	0.366762

The results in Table 6.3, Figure 6.1 show that both DFT and RAG outperformed the baseline, i.e., the version of LLaMA-2-7b run without DFT and RAG. Furthermore, it can be said that overall, the RAG approach without DFT performed the best because models subjected to DFT could no longer produce better results using RAG.

This is further evidenced by the optimal threshold parameters observed during the application of RAG. The best thresholds for the base LLaMA-2-7b model were an average of 0.5, whereas, for the DFT

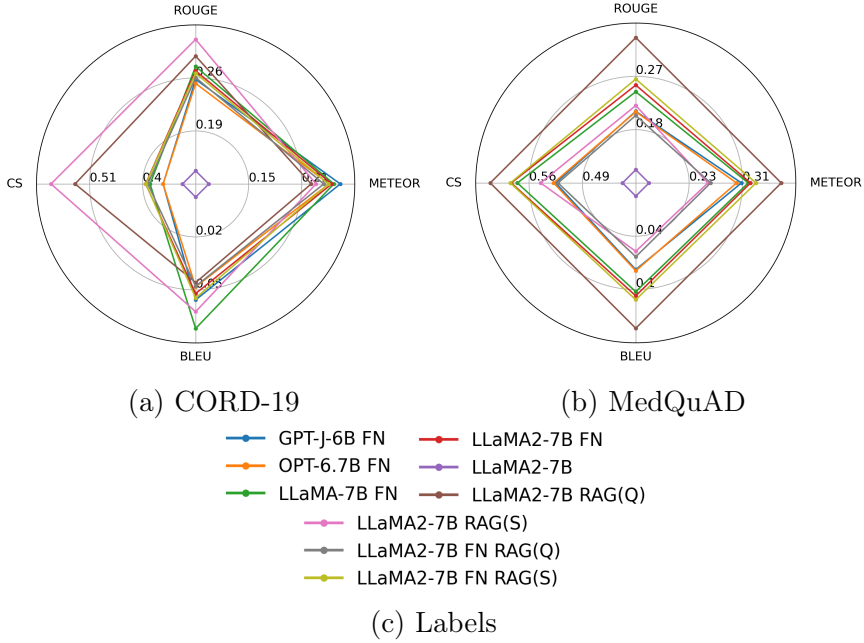


Figure 6.1: Radar plot of the evaluation results of the CORD-19 and MedQuAD models.

LlaMA-2-7b model, the optimal threshold was 1.0, effectively representing a 100% rejection rate. That is, the context injection could no longer help with domain-specific models. Therefore, though the domain-specific fine-tuning provides additional knowledge to the model, in return, it may impair its ability to interpret context, and it is critical for RAG. As a result, compiling a data set that conveys domain-specific knowledge that does not impair the ability of the original model to extract context is a much more difficult task than finding and passing on the appropriate context data to a properly pre-trained model. Moreover, this requires significantly less computing resources. As a result, its ecological footprint is also smaller.

It is important to note that the METEOR and BLEU scores of the fine-tuned models were better than those of the RAG models, but in terms of the ROUGE score, they were already inferior compared to the RAG. Furthermore, the RAG produced a significantly better CS than the fine-tuned models. This shows that RAG significantly improves hallucination, and although the association skills of fine-tuned models may be better, the degree of hallucination of fine-tuned models is significantly more significant.

The best result for the test dataset was obtained using the RAG Llama-2-7b base model with the ID_s dataset. The best approaches' results are ROUGE 0.3, METEOR 0.25, BLEU 0.07, and CS 0.55.

6.5 Conclusion

In this study, we have shown that Retrieval-Augmented Generation consistently outperforms Domain-Specific Fine-Tuning across several evaluation metrics, including ROUGE, BLEU, and our Coverage Score (CS), making RAG a more practical approach for building LLM-based knowledge systems. The enhanced performance of RAG stems from its ability to leverage efficient semantic search within indexed databases, allowing it to retrieve relevant information with higher accuracy. This reduces hallucinations and improves the overall reliability of the model's responses, particularly in knowledge retrieval tasks, offering a clear advantage over the more resource-intensive and specialized DFT approach.

While RAG-based systems excel in these areas, integrating DFT with RAG does not provide a significant performance boost, highlighting the complexity of combining these two methods. Moreover, the RAG approach proves to be more cost-effective and scalable, as it simplifies the expansion of knowledge bases by utilizing embedded vector indexing without the need for constant retraining of fine-tuned models.

Summary

Generative large language models have revolutionized the development of knowledge-based systems, enabling new possibilities in applications like ChatGPT, Bing, and Gemini. Two key strategies for domain adaptation in these systems are Domain-Specific Fine-Tuning and Retrieval-Augmented Generation. In this chapter, we evaluate the performance of RAG and DFT across several LLM architectures, including GPT-J-6B, OPT-6.7B, LLaMA, and LLaMA-2, using metrics such as ROUGE, BLEU, METEOR, as well as our cosine similarity based Coverage Score (CS). Our results, based on experiments across multiple datasets, show that RAG-based systems consistently outperform those fine-tuned with DFT. Specifically, RAG models outperform DFT by an average of 17% in ROUGE, 13% in BLEU, and 36% in CS. At the same time, DFT achieves only a modest advantage in METEOR, suggesting slightly better creative capabilities. We also highlight the challenges of integrating RAG with DFT, as such integration can lead to performance degradation. Furthermore, we propose a simplified RAG-based architecture that maximizes efficiency and reduces hallucination, underscoring the advantages of RAG in building reliable, domain-adapted knowledge systems.

The findings discussed in this chapter were presented in a paper titled "Investigating the Performance of Retrieval-Augmented Generation and Domain-Specific Fine-Tuning for the Development of AI-Driven Knowledge-Based Systems" [134] [P1] in the Machine Learning and Knowledge Extraction journal. They also provided the basis for the results of the "Designing Prompts and Creating Cleaned Scientific Text for Retrieval Augmented Generation for More Precise Responses from Generative Large Language Models" [135] [P9] and 'Investigating the Influence of Hyperparameters on the Optimal Time-Series Prediction Ability of Generative Large Language Models' [136] [P10] conference papers.

Chapter 7

Optimization of theoretical models

7.1 Introduction

Starting the optimization of neural network-based architectures from a random state is often less effective, especially with limited data. Complex models can easily overfit, impairing generalization, and training tends to be more expensive, requiring more epochs to reach optimal performance. The effectiveness of pre-training is supported by various publications, including those related to the Generative Pretrained Transformer (GPT) [4] in natural language processing.

This chapter outlines a procedure for effectively employing non-linear least-squares solver algorithms to optimize a theoretical model. This approach yields better results during training than directly training a neural network. We validate our method using the SIR model, which is commonly used for epidemic prediction. Section 7.2 covers the optimization data, and Section 7.3 details the steps to create the best-fitting model. Neural network approximation issues are addressed elsewhere in a separate Ph.D. dissertation.

7.2 Countries and data

For our research, we used infection data regarding influenza and COVID-19 viruses.

In the case of COVID-19, we examined the data published by the World Health Organization (WHO) for influenza-related studies between 2018 and 2019. WHO makes the data *FluNet* [137] available by year, and every record in the dataset consists of weekly values.

For COVID-19, we were working on data for new cases published by the Johns Hopkins Coronavirus Research Center [138]. Johns Hopkins University updates the data daily and makes it available in the *open humanitarian data sharing platform (HDX) dataset*.

We tested our proposed method using data from several countries, including Austria, China, Croatia, Germany, Hungary, Japan, and Romania. Our primary focus was on Hungary, which we aimed to compare with its neighboring countries. Additionally, we extended the study to include developed countries and leading regions that could provide the most accurate data.

One of the most critical parameters in the SIR model is the population of the studied area. Knowing the population is essential for the SIR model. We collected population data for the countries from a *public (total population) database* [139], which the World Bank currently maintains. In the analysis, we used the most recent population data for 2019 for both viruses.

7.3 Methodology

A properly parameterized SIR model can already exhibit predictive solid capabilities. Therefore, when we use a neural network to approximate a theoretical model, we aim for the neural network to learn the theoretical model.

In the case of SIR, adjusting the best parameterized theoretical model for the actual data is practically a least-squares curve fitting

problem, where points describing the SIR curve are in two-dimensional space. That is, due to the peculiarities of the SIR model, we have a non-linear least-squares minimization problem with ample solution space, as seen in Figure 7.1.

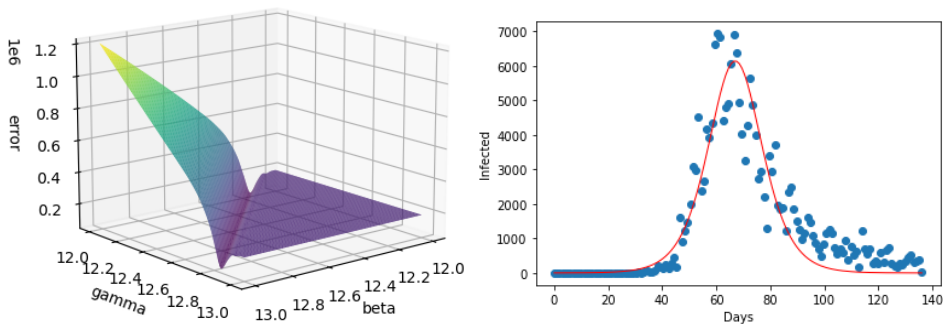


Figure 7.1: A slice from the big solution space (left) and a SIR curve fitted to the German data (right).

Several methods exist for solving such problems, such as gradient descent, the Gauss-Newton algorithm, the Levenberg-Marquardt algorithm [15] [16], or differential evolution [17]. In the experimental phase, we compared several solution methods (BFGS, Newton, brute force, etc.), out of which the Levenberg-Marquardt algorithm (LMA) and differential evolution algorithms (DEA) proved to be the most efficient ones. LMA is a fast and effective algorithm for finding an ideal solution, even if started from initial values relatively far from the optimum. Because of this property, it is applicable to determine proper starting parameters and their associated parameter range. DEA is slower than LMA, but starting with the same initial parameters, it generally finds a better solution than LMA. Over and above, because of its construction, DEA works more efficiently in the case of large solution spaces, like those also occurring in the optimization of SIR models.

Our method is straightforward. We collect the data that the theoretical function can explain well. Next, the LMA is used to find suitable starting parameters for the theoretical model. We further refine these initial parameters using DEA. The optimized theoretical model, created through this process, can be effectively approximated using a neural network. Our article in Replacing the SIR epidemic model with a neural network and training it further to increase prediction accuracy [140] demonstrates the effective operation of this method with measurement results.

Summary

Our experience shows that the LMA and DEA methods apply well to the curve-fitting problem. DEA is especially advantageous because of its speed and the differential evolution algorithm, which effectively refines the existing near-optimal parameters. As a result, a neural network pre-trained on the theoretical function can achieve better results on the final data than a model without pre-training.

The findings discussed in this chapter were presented in a paper titled "Replacing the SIR epidemic model with a neural network and training it further to increase prediction accuracy" [140] [P4] in the Annales Mathematicae et Informaticae journal.

Summary

In my Ph.D. dissertation, I comprehensively explore machine learning and artificial intelligence techniques applied to various domains, aiming to address complex problems and enhance decision-making processes. My introduction sets the stage for my research, outlining the fundamental role of optimization techniques in machine learning. I elucidate the multifaceted nature of optimization, stressing the dependence of optimization process and strategies on the model type and specific nature of the task. In addition, through a series of chapters, my dissertation delves into different methodologies, models, and applications. I explore the different strategies utilized in training, highlighting the applicability of supervised, semi-supervised, and unsupervised approaches subject to the availability and quality of data and the type of problem at hand.

Chapter 3 focuses on the problems of the time-series data, specifically oil price trend prediction, using social media data sentiment analysis. Through the application of natural language processing and machine learning techniques, the study demonstrates the potential of analyzing textual information from platforms like X to forecast price movements accurately. The chapter underscores the importance of leveraging sentiment analysis algorithms and recurrent neural networks.

In Chapter 4, the dissertation further expands its scope by exploring the extraction of insights from customer reviews. By developing a machine learning-based pipeline, the study showcases the effectiveness

of techniques such as topic modeling, keyword extraction, sentiment analysis, and clustering in analyzing textual data.

Chapter 5, related to covert tobacco advertisements, proposes a multimodal deep-learning architecture for smoking detection. By integrating textual and visual data analysis techniques, the study demonstrates the potential of deep learning models in detecting hidden advertising content.

Chapter 6, the final chapter of my dissertation, explores the development of AI-supported knowledge-based systems. This chapter is focusing on comparing the performance of DFT and RAG techniques. Through a comprehensive evaluation of large language models and methodologies, the study highlights the advantages of RAG in terms of efficiency, hallucination reduction, and scalability. The chapter underscores the challenges and opportunities in integrating DFT and RAG techniques, emphasizing the importance of semantic search engines in retrieving relevant information effectively.

Finally, chapter 7 lays the foundation for the dissertation by investigating the optimization of theoretical models, mainly focusing on curve-fitting problems. By leveraging advanced optimization algorithms such as the Levenberg-Marquardt algorithm and differential evolution, the study demonstrates their efficacy in finding optimal parameters for theoretical models like the SIR.

References

- [1] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.
- [3] I. Goodfellow, “Deep learning,” 2016.
- [4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, “Improving language understanding by generative pre-training,” 2018.
- [5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [6] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of machine learning research*, vol. 13, no. 2, 2012.
- [7] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in neural information processing systems*, vol. 25, 2012.

- [8] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2015.
- [9] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.
- [10] T. Harko, F. S. Lobo, and M. Mak, “Exact analytical solutions of the susceptible-infected-recovered (sir) epidemic model and of the sir model with equal death and birth rates,” *Applied Mathematics and Computation*, vol. 236, pp. 184–194, 2014.
- [11] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.
- [12] C. Lemaréchal, “Cauchy and the gradient method,” *Doc Math Extra*, vol. 251, no. 254, p. 10, 2012.
- [13] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2000.
- [14] C. A. Floudas and P. M. Pardalos, *Encyclopedia of optimization*. Springer Science & Business Media, 2008.
- [15] K. Levenberg, “A method for the solution of certain non-linear problems in least squares,” *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.
- [16] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters,” *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.
- [17] R. Storn, “On the usage of differential evolution for function optimization,” in *Proceedings of North American Fuzzy Information Processing*, pp. 519–523, IEEE, 1996.

- [18] J. W. Wilder, “New concepts in technical trading systems,” (*No Title*), 1978.
- [19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [20] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *arXiv preprint arXiv:1409.1259*, 2014.
- [21] L. Atlas, T. Homma, and R. Marks, “An artificial neural network for spatio-temporal bipolar patterns: Application to phoneme classification,” in *Neural Information Processing Systems*, 1987.
- [22] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [23] A. Zeng, M. Chen, L. Zhang, and Q. Xu, “Are transformers effective for time series forecasting? arxiv 2022,” *arXiv preprint arXiv:2205.13504*, 2022.
- [24] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [25] L. Prechelt, “Early stopping-but when?,” in *Neural Networks: Tricks of the trade*, pp. 55–69, Springer, 2002.
- [26] T. Kudo, “Subword regularization: Improving neural network translation models with multiple subword candidates,” *arXiv preprint arXiv:1804.10959*, 2018.

- [27] M. Schuster and K. Nakajima, “Japanese and korean voice search,” in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5149–5152, IEEE, 2012.
- [28] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” *arXiv preprint arXiv:1508.07909*, 2015.
- [29] T. Kudo and J. Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” *arXiv preprint arXiv:1808.06226*, 2018.
- [30] H. P. Luhn, “A statistical approach to mechanized encoding and searching of literary information,” *IBM Journal of research and development*, vol. 1, no. 4, pp. 309–317, 1957.
- [31] K. S. Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of documentation*, 1972.
- [32] D. Jurafsky, “Speech and language processing,” 2000.
- [33] Y. Goldberg and O. Levy, “word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method,” *arXiv preprint arXiv:1402.3722*, 2014.
- [34] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.
- [35] K. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

- [36] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” 2010.
- [37] A. Szlam, Y. Kluger, and M. Tygert, “An implementation of a randomized algorithm for principal component analysis,” *arXiv preprint arXiv:1412.3510*, 2014.
- [38] N. D. Lawrence, “A unifying probabilistic perspective for spectral dimensionality reduction: Insights and new models,” *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 1609–1638, 2012.
- [39] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.
- [40] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.,” *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [41] M. F. Dicle and B. Dicle, “Content analysis: frequency distribution of words,” *The Stata Journal*, vol. 18, no. 2, pp. 379–386, 2018.
- [42] S. Kübler, R. McDonald, and J. Nivre, “Dependency parsing,” *Synthesis lectures on human language technologies*, vol. 1, no. 1, pp. 1–127, 2009.
- [43] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, “Fasttext. zip: Compressing text classification models,” *arXiv preprint arXiv:1612.03651*, 2016.
- [44] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, “Stochastic variational inference.,” *Journal of Machine Learning Research*, vol. 14, no. 5, 2013.
- [45] D. Arthur and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” tech. rep., Stanford, 2006.

- [46] R. Xu and D. Wunsch, *Clustering*, vol. 10. John Wiley & Sons, 2008.
- [47] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *kdd*, vol. 96, pp. 226–231, 1996.
- [48] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, “Hierarchical density estimates for data clustering, visualization, and outlier detection,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 1, pp. 1–51, 2015.
- [49] D. Angelov, “Top2vec: Distributed representations of topics,” *arXiv preprint arXiv:2008.09470*, 2020.
- [50] M. Grootendorst, “Bertopic: Leveraging bert and c-tf-idf to create easily interpretable topics,” 2020.
- [51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [52] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.
- [53] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (P. Isabelle, E. Charniak, and D. Lin, eds.), (Philadelphia, Pennsylvania, USA), pp. 311–318, Association for Computational Linguistics, July 2002.
- [54] S. Banerjee and A. Lavie, “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments,” in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or*

Summarization, (Ann Arbor, Michigan), pp. 65–72, Association for Computational Linguistics, June 2005.

- [55] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*, (Barcelona, Spain), pp. 74–81, Association for Computational Linguistics, July 2004.
- [56] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [57] Twitter, “Twitter,” 2006.
- [58] Yahoo, “Crude oil.”
- [59] Yahoo, “Brent crude oil.”
- [60] M. G. Kendall, “A new measure of rank correlation,” *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.
- [61] F. Galton, “Regression towards mediocrity in hereditary stature.,” *The Journal of the Anthropological Institute of Great Britain and Ireland*, vol. 15, pp. 246–263, 1886.
- [62] J. L. Myers, A. D. Well, and R. F. Lorch Jr, *Research design and statistical analysis*. Routledge, 2013.
- [63] M. Heitmann, C. Siebert, J. Hartmann, and C. Schamp, *More than a feeling: Benchmarks for sentiment analysis accuracy*. Ssrn, 2020.
- [64] F. Barbieri, J. Camacho-Collados, L. Neves, and L. Espinosa-Anke, “Tweeteval: Unified benchmark and comparative evaluation for tweet classification,” 2020.
- [65] E. Loper and S. Bird, “Nltk: The natural language toolkit,” 2002.

- [66] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, “Huggingface’s transformers: State-of-the-art natural language processing,” 2020.
- [67] R. Lakatos, G. Bogacsovics, and A. Hajdu, “Predicting the direction of the oil price trend using sentiment analysis,” in *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*, pp. 177–182, IEEE, 2022.
- [68] D. Falush, M. Stephens, and J. K. Pritchard, “Inference of population structure using multilocus genotype data: linked loci and correlated allele frequencies,” *Genetics*, vol. 164, no. 4, pp. 1567–1587, 2003.
- [69] J. K. Pritchard, M. Stephens, and P. Donnelly, “Inference of population structure using multilocus genotype data,” *Genetics*, vol. 155, no. 2, pp. 945–959, 2000.
- [70] C. Zhai, “Statistical language models for information retrieval,” *Synthesis lectures on human language technologies*, vol. 1, no. 1, pp. 1–141, 2008.
- [71] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a review,” *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.
- [72] J. Ni, J. Li, and J. McAuley, “Justifying recommendations using distantly-labeled reviews and fine-grained aspects,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 188–197, Association for Computational Linguistics, 11 2019.

- [73] J. Rennie, “20 newsgroups dataset,” 05 2012.
- [74] M. V. Mäntylä, D. Graziotin, and M. Kuutila, “The evolution of sentiment analysis—a review of research topics, venues, and top cited papers,” *Computer Science Review*, vol. 27, pp. 16–32, 2018.
- [75] M. Grootendorst, “Keybert: Minimal keyword extraction with bert.,” 2020.
- [76] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” *arXiv preprint arXiv:1908.10084*, 2019.
- [77] R. Lakatos, G. Bogacsovics, B. Harangi, I. Lakatos, A. Tiba, J. Tóth, M. Szabó, and A. Hajdu, “A machine learning-based pipeline for the extraction of insights from customer reviews,” *Big Data and Cognitive Computing*, vol. 8, no. 3, p. 20, 2024.
- [78] O. for Economic Co-operation and Development, “Daily smokers (indicator),” 01 2023.
- [79] W. H. Organization, “Tobacco,” 05 2022.
- [80] S. Chapman and R. M. Davis, “Smoking in movies: Is it a problem?,” *Tobacco Control*, vol. 6, no. 4, pp. 269–271, 1997.
- [81] C. Pechmann and C. Shih, “How smoking in movies and anti-smoking ads before movies may affect teenagers’ perceptions of peers who smoke,” *Irvine, California: Graduate School of Management, University of California, Irvine*, 1996.
- [82] G. Kong, A. S. Schott, J. Lee, H. Dashtian, and D. Murthy, “Understanding e-cigarette content and promotion on youtube through machine learning,” *Tobacco Control*, 2022.

- [83] R. Fielding, Y. Chee, K. Choi, T. Chu, K. Kato, S. Lam, K. Sin, K. Tang, H. Wong, and K. Wong, “Declines in tobacco brand recognition and ever-smoking rates among young children following restrictions on tobacco advertisements in hong kong,” *Journal of Public Health*, vol. 26, no. 1, pp. 24–30, 2004.
- [84] R. Fu, A. Kundu, N. Mitsakakis, T. Elton-Marshall, W. Wang, S. Hill, S. J. Bondy, H. Hamilton, P. Selby, R. Schwartz, *et al.*, “Machine learning applications in tobacco research: a scoping review,” *Tobacco Control*, vol. 32, no. 1, pp. 99–109, 2023.
- [85] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [86] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.
- [87] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [88] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books,” in *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.
- [89] “Common crawl,” 2022. Accessed: 2022-06-01.
- [90] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

- [91] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.
- [92] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.
- [93] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.
- [94] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.
- [95] J. Lin, Y. Chen, R. Pan, T. Cao, J. Cai, D. Yu, X. Chi, T. Cernava, X. Zhang, and X. Chen, “Camffnet: A novel convolutional neural network model for tobacco disease image recognition,” *Computers and Electronics in Agriculture*, vol. 202, p. 107390, 2022.
- [96] S. Ali, K. Masood, A. Riaz, and A. Saud, “Named entity recognition using deep learning: A review,” in *2022 International Conference on Business Analytics for Technology and Security (IC-BATS)*, pp. 1–7, IEEE, 2022.
- [97] Y. Liu, Y. Guo, L. Liu, E. M. Bakker, and M. S. Lew, “Cyclematch: A cycle-consistent embedding network for image-text matching,” *Pattern Recognition*, vol. 93, pp. 365–379, 2019.
- [98] Z. Liu, F. Chen, J. Xu, W. Pei, and G. Lu, “Image-text retrieval with cross-modal semantic importance consistency,” *IEEE*

Transactions on Circuits and Systems for Video Technology, 2022.

- [99] N. Reimers and I. Gurevych, “Making monolingual sentence embeddings multilingual using knowledge distillation,” *arXiv preprint arXiv:2004.09813*, 2020.
- [100] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.
- [101] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, “Unsupervised cross-lingual representation learning at scale,” *CoRR*, vol. abs/1911.02116, 2019.
- [102] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” *ArXiv*, vol. abs/1910.01108, 2019.
- [103] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Advances in neural information processing systems*, vol. 26, 2013.
- [104] A. Khan, “Dataset containing smoking and not-smoking images (smoker vs non-smoker),” *Mendeley Data*, vol. 1, 2020.
- [105] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.
- [106] OpenAI, “Chatgpt,” 2024. Accessed on March 12, 2024.

- [107] T. Viola, *Ellentétes jelentésű szavak adatbázisa*. Tinta Könyvkiadó, 2012.
- [108] G. AI, “Gemini,” 2024. Accessed on March 12, 2024.
- [109] D. M. Nemeskey, “Introducing huBERT,” in *XVII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2021)*, (Szeged), p. TBA, 2021.
- [110] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018.
- [111] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” *arXiv preprint arXiv:1908.10084*, 2019.
- [112] T. G. Dietterich, “Ensemble methods in machine learning,” in *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, 2000 Proceedings 1*, pp. 1–15, Springer, 2000.
- [113] H. P. Center, “Egészség elvitelre,” 2023.
- [114] R. Lakatos, P. Pollner, A. Hajdu, and T. Joó, “A multi-modal deep learning architecture for smoking detection with a small data approach,” *Frontiers in Artificial Intelligence*, vol. 7, p. 1326050, 2024.
- [115] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” *arXiv preprint arXiv:1901.02860*, 2019.
- [116] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

- [117] Microsoft, “Microsoft copilot,” 2024. Accessed on March 12, 2024.
- [118] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, “Palm: Scaling language modeling with pathways,” 2022.
- [119] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, “Llama: Open and efficient foundation language models,” 2023.
- [120] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, “Language models as knowledge bases?,” *arXiv preprint arXiv:1909.01066*, 2019.
- [121] M. A. K. Raiaan, M. S. H. Mukta, K. Fatema, N. M. Fahad, S. Sakib, M. M. J. Mim, J. Ahmad, M. E. Ali, and S. Azam, “A review on large language models: Architectures, applications, taxonomies, open issues and challenges,” *IEEE Access*, 2024.
- [122] J. Zheng, H. Hong, X. Wang, J. Su, Y. Liang, and S. Wu, “Fine-tuning large language models for domain-specific machine translation,” *arXiv preprint arXiv:2402.15061*, 2024.

- [123] C. Jeong, “Fine-tuning and utilization methods of domain-specific llms,” *arXiv preprint arXiv:2401.02981*, 2024.
- [124] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, “Retrieval augmented language model pre-training,” in *International conference on machine learning*, pp. 3929–3938, PMLR, 2020.
- [125] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Burdick, D. Eide, K. Funk, Y. Katsis, R. Kinney, *et al.*, “Cord-19: The covid-19 open research dataset,” *ArXiv*, 2020.
- [126] A. Ben Abacha and D. Demner-Fushman, “Medquad: A medical question answering dataset containing diverse healthcare topics,” *Studies in Health Technology and Informatics*, pp. 307–311, 2019.
- [127] A. Ben Abacha and D. Demner-Fushman, “A question-entailment approach to question answering,” *BMC Bioinform.*, vol. 20, no. 1, pp. 511:1–511:23, 2019.
- [128] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.
- [129] B. Wang and A. Komatsuzaki, “GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model.” <https://github.com/kingoflolz/mesh-transformer-jax>, May 2021.
- [130] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, *et al.*, “Opt: Open pre-trained transformer language models,” *arXiv preprint arXiv:2205.01068*, 2022.

- [131] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, “Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers,” 2020.
- [132] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, “Stanford alpaca: An instruction-following llama model.” https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [133] Voidful, “Context only question generator.”
- [134] R. Lakatos, P. Pollner, A. Hajdu, and T. Joó, “Investigating the performance of retrieval-augmented generation and domain-specific fine-tuning for the development of ai-driven knowledge-based systems,” *Machine Learning and Knowledge Extraction*, vol. 7, no. 1, p. 15, 2025.
- [135] R. Lakatos, E. K. Urbán, Z. J. Szabó, J. Pozsga, E. Csernai, and A. Hajdu, “Designing prompts and creating cleaned scientific text for retrieval augmented generation for more precise responses from generative large language models,” in *2024 IEEE 3rd Conference on Information Technology and Data Science (CITDS)*, pp. 1–6, 2024.
- [136] A. Pandey, R. Lakatos, P. Pollner, E. Csernai, and A. Hajdu, “Investigating the influence of hyperparameters on the optimal time-series prediction ability of generative large language models,” in *2024 IEEE 3rd Conference on Information Technology and Data Science (CITDS)*, pp. 1–6, 2024.
- [137] “World health organization, flunet.”
- [138] “Johns hopkins coronavirus resource center.”
- [139] “World bank total population data.”

- [140] G. Bogacsovics, A. Hajdu, R. Lakatos, M. Beregi-Kovács, A. Tiba, and H. Tomán, “Replacing the sir epidemic model with a neural network and training it further to increase prediction accuracy,” in *Annales Mathematicae et Informaticae*, vol. 53, pp. 73–91, Eszterházy Károly Egyetem Líceum Kiadó, 2021.

List of Publications

Journal articles in English

- [P1] Lakatos, R., Pollner, P., Hajdu, A., and Joó, T. (2025). Investigating the Performance of Retrieval-Augmented Generation and Domain-Specific Fine-Tuning for the Development of AI-Driven Knowledge-Based Systems. *Machine Learning and Knowledge Extraction*, 7(1), 15. <https://doi.org/10.3390/make7010015>
- [P2] Róbert Lakatos , Péter Pollner, András Hajdu, and Tamás Joó. 2024. A multimodal deep learning architecture for smoking detection with a small data approach. In *Frontiers in Artificial Intelligence* (Vol. 7). Frontiers Media SA. <https://doi.org/10.3389/frai.2024.1326050>
- [P3] Róbert Lakatos, Gergő Bogacsovics, Balázs Harangi, István Lakatos, Attila Tiba, János Tóth, Marianna Szabó, and András Hajdu. 2024. A Machine Learning-Based Pipeline for the Extraction of Insights from Customer Reviews. *Big Data and Cognitive Computing* 8, no. 3: 20. <https://doi.org/10.3390/bdcc8030020>
- [P4] Gergo Bogacsovics, András Hajdu, Róbert Lakatos, Marcell Beregi-Kovács, Attila Tiba, Henrietta Tomán. *Replacing the SIR epidemic model with a neural network and training it further to increase prediction accuracy*. In: *Annales Mathematicae et Informaticae*. Eszterházy Károly Egyetem Líceum Kiadó, 2021. p. 73-91. doi.org/10.33039/ami.2021.02.003

Journal articles in Hungarian

- [P5] Nagy János, Hajdu András, Bogacsovics Gergő, Bojtor Csaba, Illés Árpád, Lakatos Róbert, Mészáros László Precíziós gazdálkodásban használható adatelemzés alapú növénytermesztési döntéstámogató rendszer fejlesztés In: Pőr, Csilla; Szabó-Soós, Adrienn; Szabó, Péter (szerk.) LXV. Georgikon Napok Tudományos Konferencia [65th Georgikon Days Scientific Conference] Keszthely, 2024. május 17–18. : Kivonatkötet Keszthely, Magyarország : Magyar Agrár- és Élettudományi Egyetem, Georgikon Campus (2024) 206 p. pp. 133-134., 2 p.
- [P6] Bogacsovics Gergo, Hajdu András, Harangi Balázs, Lakatos István, Lakatos Róbert, Szabó Marianna, Tiba Attila, Tóth János. Napelemlétfarmok Magyarország területén történő elhelyezését segítő döntéstámogató rendszer fejlesztése. KözigazgatásTudomány, 2021, 1.2: 134-145. doi.org/10.54200/kt.v1i2.23
- [P7] Bogacsovics Gergo, Hajdu András, Harangi Balázs, Lakatos István, Lakatos Róbert, Szabó Marianna, Tiba Attila, Tóth János, Tarcsi Ádám. Adatelemzési folyamat és keretrendszer a közigazgatás számára. KözigazgatásTudomány, 2021, 1.2: 146-158. doi.org/10.54200/kt.v1i2.24
- [P8] Bátfa, M., Lakatos, R., Molnár, G., Takácsné Bubnó, K. BPMN 2.0 a könyvtári minőségbiztosításban, Tudományos és Műszaki Tájékoztatás, 63(8), p. 305–316, 2016.

Conference papers

- [P9] Róbert Lakatos, Eszter Klára Urbán, Zoltán János Szabó, János Pozsga, Eszter Csernai and András Hajdu. 2024. Designing Prompts and Creating Cleaned Scientific Text for Retrieval Augmented Generation for More Precise Responses from Generative Large Language Models. In 2024 IEEE 3rd Conference on Information Technology and Data Science (CITDS) (pp. 1-6). IEEE. [10.1109/CITDS62610.2024.10791382](https://doi.org/10.1109/CITDS62610.2024.10791382)
- [P10] Árpád Pandy, Róbert Lakatos Péter Pollner, Eszter Csernai and András Hajdu. 2024. Investigating the Influence of Hyperparameters on the Optimal Time-Series Prediction Ability of Generative Large Language Models. In 2024 IEEE 3rd Conference on Information Technology and Data Science (CITDS) (pp. 1-6). IEEE. [10.1109/CITDS62610.2024.10791385](https://doi.org/10.1109/CITDS62610.2024.10791385)
- [P11] Gergő Bogacsovics, Balázs Harangi, Marcell Beregi-Kovács, Dávid Kupás, Róbert Lakatos, Norbert Dávid Serbán, Tiba Attila, János Tóth. 2024. Assessing Conventional and Deep Learning-Based Approaches for Named Entity Recognition in Unstructured Hungarian Medical Reports. Presented at the 2024 IEEE 22nd World Symposium on Applied Machine Intelligence and Informatics (SAMI). doi.org/10.1109/sami60510.2024.10432795
- [P12] Róbert Lakatos, Gergo Bogacsovics, András Hajdu. Predicting the direction of the oil price trend using sentiment analysis. In: 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS). IEEE, 2022. p. 177-182. doi.org/10.1109/CITDS54976.2022.9914158