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

DIPLOMA THESIS

A Comparative Analysis of Text Representation Methods for Romanian Fake News Detection

Supervisor
Lect. PhD. Lupea Mihaiela-Ana

Author
Pop Răzvan-Gabriel

2025

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

O Analiză Comparativă a Metodelor de Reprezentare a Textului pentru Detectarea Știrilor False în Limba Română

Conducător științific
Lect. Dr. Lupea Mihaiela-Ana

Absolvent
Pop Răzvan-Gabriel

2025

ABSTRACT

This thesis presents a comprehensive methodology for detecting fake news in the Romanian language, covering the entire process from in-depth data analysis to the development of a practical web application. The core contribution of this work lies in its systematic comparison of text representation techniques and the creation of a hybrid model that combines semantic and linguistic features. The research begins with a detailed analysis of a Romanian news dataset, exploring its lexical, syntactic, affective, and named-entity characteristics. This phase revealed measurable differences between factual news and disinformation, particularly in article length, emotional language, and the use of more descriptive language. Following this, an evaluation of various text embedding models, including static Word2Vec and Doc2Vec, as well as contextualized RoBERT representations was conducted to understand their ability to capture the semantic structure of the data. Building on these insights, a series of classification experiments were performed using an SVM classifier. The first experiment compared different text representation methods, where a traditional TF-IDF model achieved the highest performance with a mean F1-Score of 95.7%. However, a final hybrid model that combined semantic Doc2Vec vectors with relevant features derived from the initial dataset analysis also achieved a very strong mean F1-Score of 91.6%, demonstrating the value of feature synergy. The practical outcome of this research is a fully developed web application. The application provides a user-friendly interface for classifying Romanian news articles from text input and implements a feature for analyzing word similarities. It also allows users to download the two custom-trained, domain-specific, and lemmatized Word2Vec models created during this research.

This work is the result of my own activity, and I confirm I have neither given, nor received unauthorized assistance for this work.

Contents

Chapter 1

Introduction

1.1 Problem statement and motivation

Fake news is a growing issue, especially as digital platforms and social media have become primary sources of information. Misleading content can spread rapidly, making it difficult for people to distinguish fact from fiction. While many studies focus on detecting fake news in English, there are fewer tools and less research dedicated to the Romanian language.

In Romania, fake news has influenced public opinion on topics like elections, public health, and the economy. There are not many datasets or tools designed specifically for the Romanian language, which makes automatic detection more difficult. Traditional fact-checking methods cannot keep up with the speed at which misinformation spreads, so automated solutions using artificial intelligence and natural language processing (NLP) are needed.

This thesis works to solve this problem by building and testing a complete system for identifying fake news in Romanian texts. The research starts with a detailed analysis of a news dataset to find the specific language-based and emotional patterns of different types of articles. Based on these findings, it tests different text representation methods with a Support Vector Machine (SVM) classifier, resulting in a functional web application that serves as a practical tool.

1.2 Objectives and Scope

The main goal of this thesis is to find effective methods for detecting fake news in the Romanian language. There are three key objectives. The first is to perform a detailed analysis of a Romanian news dataset to find the language-based and emotional traits that separate real news from disinformation. The second is to test and compare different ways of representing text, such as TF-IDF, Word2Vec, Doc2Vec,

and a hybrid model. The third objective is to build a working web application that uses these findings to show how they can be applied in practice.

The scope of this research is clearly defined. It focuses only on the Romanian language and the domain of news articles. The task is a five-category classification problem (**fake news**, **real news**, **misinformation**, **propaganda**, and **satire**). To ensure a fair comparison, all experiments use a Support Vector Machine (SVM) classifier and are evaluated with a 5-fold cross-validation method. The final web application serves as a proof-of-concept for the research.

1.3 Personal Contributions

This thesis offers several original contributions to the field of fake news detection for the Romanian language.

First, this research provides a detailed, multi-perspective analysis of a Romanian news dataset. It goes beyond using the data for just training by presenting a broad quantitative analysis of its lexical, syntactic, named-entity, and affective characteristics. This foundational analysis helps to create a clear data-driven "fingerprint" for different news categories.

Second, the thesis presents a systematic comparison of different text representation methods within a controlled experimental setup. By using the same SVM classifier to test TF-IDF, Word2Vec, Doc2Vec, and RoBERT-based features extracted from the pre-trained model without fine-tuning, this work provides clear insights into which methods are most effective for this specific task. A key contribution is the development of a high-performing hybrid model, which combines semantic document vectors with a set of custom-engineered features derived from the initial dataset analysis.

Third, this work contributes two new resources for the Romanian language: 150-dimension and 300-dimension Word2Vec models. These models are notable because they are both lemmatized and trained specifically on a news-related corpus, making them a valuable tool for future research in this domain.

Finally, a key contribution is the development of a complete web application that integrates the research findings into a practical tool. While other applications for Romanian fake news detection exist, this project is notable for combining classification with a tool for linguistic exploration.

1.4 Thesis structure

This thesis is organized into seven chapters. Chapter 1 introduces the research problem, motivation, objectives, and personal contributions. Chapter 2 provides a review of related work in the field of fake news detection. Chapter 3 covers the theoretical background of the machine learning models and natural language processing techniques used. Chapter 4 presents a detailed analysis of the dataset, exploring its lexical, syntactic, and affective characteristics, and analyzes the semantic spaces learned by different text embedding models. Chapter 5 details the classification experiments, comparing the performance of various text representation methods. Chapter 6 describes the architecture and implementation of the web application. Finally, Chapter 7 concludes the thesis by summarizing the key findings and suggesting directions for future work.

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

During the preparation of this work the author used Gemini in order to rephrase sentences and improve the text's clarity. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the content of the thesis.

Chapter 2

Related Work

Research on detecting fake news in the Romanian language has advanced through various machine learning and transformer-based methods. In [MMC+24], researchers compared models like Naïve Bayes, Logistic Regression, Support Vector Machines (SVM), BERT-based multilingual cased, and RoBERTa-large. They found that the BERT model trained on the NEW dataset, which was built by the authors, achieved the highest accuracy of 96.5%, while the SVM model also performed well with 94.6% accuracy. This highlights the effectiveness of transformer-based models for Romanian fake news detection.

Another study, [BTMR23], explored different deep learning architectures, including recurrent neural networks (RNN) with long short-term memory (LSTM) and gated recurrent unit (GRU) cells, convolutional neural networks (CNNs), and the RoBERT model, a Romanian-specific BERT model. CNN achieved an accuracy of over 98.20%, outperforming both traditional classification methods and BERT models. This suggests that CNNs can effectively capture patterns in Romanian news articles for the detection of fake news.

In [BD23], researchers addressed the challenge of limited annotated data for Romanian fake news detection by employing back-translation (BT) data augmentation. They used transformer-based models, such as mBART, to generate augmented training data. Although the fake news classifiers themselves were traditional machine learning models like Extra Trees and Random Forest, the study showed that BT augmentation significantly improved performance. This underscores the value of transformer-based data augmentation techniques in low-resource language settings such as Romanian.

The psychological aspect of the consumption of fake news has also been studied. The article [MPR20] investigated how emotional processing affects susceptibility to misinformation. It found that individuals who rely more on emotion are more likely to believe fake news, suggesting that emotional content plays a significant role in the spread and acceptance of misinformation.

Additionally, the study [LSK+24] examined the relationship between emotions and the ability to discern false from real news. The researchers observed that heightened emotional states, particularly anger, were associated with a reduced ability to distinguish between true and false information. This highlights the need to consider emotional factors when developing fake news detection systems.

A foundational resource for the emotional analysis mentioned in this thesis is presented in the paper [LB19]. This work details the development and analysis of RoEmoLex, a comprehensive emotion lexicon for the Romanian language. The authors employed Formal Concept Analysis, a knowledge discovery technique, to examine the lexicon's internal structure and uncover dependencies between words on an emotional level. By building conceptual hierarchies and comparing them with the Romanian RoWordNet, the study establishes the lexicon's validity and structure. This research is crucial as it provides a systematically constructed and analyzed resource, RoEmoLex, which enables the very kind of emotion and sentiment analysis that underpins the fake news detection models developed in this thesis.

These studies provide valuable insights into the development of effective fake news detection systems for the Romanian language and emphasize the importance of considering both linguistic and psychological factors in combating misinformation.

Chapter 3

Theoretical Aspects

3.1 Natural Language Processing Fundamentals

As a field that brings together computer science, artificial intelligence, and linguistics, Natural Language Processing (NLP) focuses on developing systems that can work with human language. The methods from NLP form the technical foundation for text analysis, making them essential for tasks like fake news detection.

3.1.1 Text Preprocessing Techniques

Before applying machine learning algorithms, text data requires preprocessing to convert unstructured text into a format suitable for analysis:

- **Normalization**: This step involves standardizing the text to ensure uniformity. Common normalization tasks include converting all characters to lowercase and removing punctuation and special symbols.
- **Tokenization**: The task of splitting a string of text into smaller pieces, or 'tokens'. These tokens are usually individual words. For example, the sentence "Știrile false sunt o problemă" becomes ['Știrile', 'false', 'sunt', 'o', 'problemă'].
- **Stop Word Removal**: Eliminating common words that appear frequently but often carry little analytical meaning. Examples in Romanian include words like și, cu, or este.
- **Stemming**: A rule-based process of cutting off the beginnings or endings of words to get to a common base form. For instance, the Romanian words cânta (to sing), cântă (sings), and cântare (singing/song) might all be reduced to the common stem cânt.

- **Lemmatization**: A more advanced method than stemming that uses vocabulary and morphological analysis to reduce words to their actual dictionary form, known as the lemma. For example, different forms of a verb like merg, mergi, and merge would all be converted to the lemma "merge".
- **Part-of-Speech (POS) Tagging**: This process analyzes a text to assign a grammatical label to each token. Common labels include noun, verb, adjective, and adverb. It is used to understand the grammatical role each word plays in a sentence.
- **Named Entity Recognition (NER)**: Identifying and classifying named entities in text into predefined categories such as person names, organizations, locations, and dates.

3.1.2 Challenges Specific to Romanian Language Processing

Romanian language presents several unique challenges for NLP systems:

- **Diacritical Marks**: Romanian uses diacritical marks (ă, â, î, ș, ț) that can be inconsistently applied in online text, creating challenges for proper text processing.
- **Rich Morphology**: Romanian has a complex morphological structure with numerous inflectional forms for nouns, adjectives, and verbs.
- **Limited Resources**: Compared to widely-spoken languages like English, Romanian has fewer NLP resources, including smaller corpora, fewer pre-trained models, and limited language tools.
- **Dialectal Variations**: Regional differences in vocabulary and expressions can complicate text analysis.

These fundamental NLP concepts and techniques provide the technical foundation for developing effective fake news detection systems for Romanian text. The preprocessing steps are particularly important for improving the accuracy of subsequent machine learning algorithms applied to text classification tasks.

3.2 Text Representation Methods

For computers to process text data, words and documents must be converted into numerical representations. These representations capture semantic and syntactic information that machine learning algorithms can use. This section explores the main text representation methods used in fake news detection.

3.2.1 Bag of Words and TF-IDF

The simplest text representation methods treat documents as unordered collections of words:

- **Bag of Words (BoW)**: This method converts a text into a numerical vector by counting the occurrence of each word from the vocabulary. A key limitation of this model is that it ignores word order and context, and therefore does not capture the relationships between words.
- **Term Frequency-Inverse Document Frequency (TF-IDF)**: An extension of BoW that assigns a weight to each word based on its importance. A word's importance increases if it appears frequently in a document but is rare across all other documents in the collection. This process gives higher value to distinctive words and less weight to very common terms.

The Term Frequency (TF) measures how often a term t appears in a document d.

$$\text{tf}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{3.1}$$

The Inverse Document Frequency (IDF) measures how important a term is by looking at how often it appears across all documents D in the corpus.

$$\text{idf}(t, D) = \log\left(\frac{\text{Total number of documents in corpus } |D|}{\text{Number of documents containing term } t}\right) \tag{3.2}$$

The final TF-IDF score for a term in a document is the product of these two values.

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D) \tag{3.3}$$

3.2.2 Static Embeddings

Word embeddings are a method of representing words as numerical vectors. The key idea is that words with similar meanings will have similar vector representations.

- **Word2Vec**: This model learns dense vector representations for words based on their neighbors [MCCD13]. It operates using one of two training algorithms. The **Continuous Bag of Words (CBOW)** architecture learns by using the surrounding context of a word to predict the word itself. In contrast, the **Skip-gram** architecture does the opposite: it uses a given word to predict its surrounding context words. A key feature of Word2Vec is its ability to capture

linguistic patterns, making it possible to solve analogies like 3.4 using vector arithmetic.

$$"king" - "man" + "woman" \approx "queen". \tag{3.4}$$

- **Doc2Vec (Paragraph Vectors)**: Extends Word2Vec to generate vector representations for entire documents or paragraphs [LM14]. It achieves this by adding a unique paragraph ID, treated as a special token, which is learned alongside word vectors. This allows the model to capture the semantics of larger text segments, making it suitable for tasks like document similarity and classification.

3.2.3 Contextualized and Dynamic Embeddings

In contrast to static embeddings which provide a single, fixed vector for each word, contextual models produce dynamic representations. This means the vector for a word changes depending on the sentence it is used in.

- **BERT (Bidirectional Encoder Representations from Transformers)**: A foundational model in modern NLP, BERT generates word representations that are dependent on context. Its key innovation is bidirectionality; it examines the full sentence at once to understand a word's meaning based on both the words that precede and follow it. BERT learns these rich representations by being pre-trained on large text corpora using two main tasks: masked language modeling and next sentence prediction.
- **Multilingual BERT**: Extends BERT to multiple languages, including Romanian, allowing for cross-lingual transfer learning despite limited Romanian-specific resources.
- **RoBERT**: A robustly optimized BERT approach with improved training methodology. Romanian adaptations of RoBERT have shown promising results in various NLP tasks.

These text representation methods provide the foundation for transforming Romanian text into numerical features that machine learning algorithms can process. The choice between these methods often involves trade-offs between computational efficiency, semantic richness, and the availability of language-specific resources for Romanian.

3.3 Machine Learning for Text Classification

Text classification is the task of assigning documents to specific categories based on their content. In fake news detection, this typically involves classifying news articles as either authentic or false. This section examines the main machine learning approaches used for this classification task.

3.3.1 Traditional ML Approaches

Traditional machine learning algorithms have been widely used for text classification due to their efficiency and interpretability:

Supervised Learning Approaches

- **Support Vector Machines (SVM)**: This is a powerful classification method that works by defining a boundary or hyperplane that best divides the data points of different categories in the feature space [CV95].
- **Logistic Regression**: This model predicts the probability that a given document belongs to a certain category using a linear equation. Although it is a simple model, it often provides a strong performance baseline for text classification and its results are easy to interpret.
- **Naïve Bayes**: This classification technique applies the principles of Bayes' theorem. The model is called "naïve" because of the core, simplifying assumption it makes, that every feature contributes to the final prediction independently, without influencing the others. Variations like Multinomial Naïve Bayes are well-suited for text classification and are very efficient with high-dimensional data.
- **Decision Trees and Random Forests**: These models use a tree-like structure of decisions to classify data. A single Decision Tree makes predictions based on a series of feature-based rules, while a Random Forest improves on this by combining the outputs of many different trees to produce a more robust and accurate result.

Unsupervised Learning Approaches

- **K-Means Clustering**: This is a popular unsupervised algorithm used to partition a dataset into a predefined number of 'K' clusters [Mac67]. It works by grouping data points such that each point belongs to the cluster with the nearest mean, or "centroid." While K-Means is not a direct classifier, it is a

valuable tool for feature engineering. In this thesis, it is used to group document embeddings into latent topical or stylistic clusters. The ID of the cluster a document belongs to can then be used as a single, powerful feature for a supervised model like an SVM.

3.3.2 Deep Learning and Transformer-based Approaches

Deep learning models have revolutionized text classification by automatically learning hierarchical feature representations:

- **Recurrent Neural Networks (RNNs)**: These are neural networks built to work with sequential information like text. They process a sequence step-by-step and use an internal memory to retain information from previous steps. However, standard RNNs often have difficulty remembering information over long sequences.

- **Long Short-Term Memory (LSTM)**: LSTMs are an advanced type of RNN created to overcome the memory limitations of basic RNNs [HS97]. They use a system of ”gates” and a memory cell to better control what information is kept or forgotten, which allows them to capture dependencies over longer stretches of text. A common enhancement is the **Bidirectional LSTM**, which processes the text in both forward and backward directions to gain a more complete understanding of each word's context.

- **Convolutional Neural Networks (CNNs)**: While initially created for analyzing images, CNNs have been effectively applied to text analysis [Kim14]. This is done by using ”filters” that slide over sequences of word embeddings to identify key local patterns, such as important phrases or n-grams.

- **BERT**: A powerful pre-trained model that has learned deep linguistic patterns from vast amounts of text [DCLT19]. The primary strength of BERT is its adaptability. By adding a small, task-specific layer to the core model, it can be fine-tuned to achieve high performance on many different applications, such as text classification, question answering, and named entity recognition.

- **Multilingual Models**: These are transformer models, such as mBERT or XLM-RoBERTa, that are pre-trained on text from many different languages. This allows them to be effective for languages with fewer dedicated resources, like Romanian, through transfer learning.

- **Romanian-specific Models**: These are transformer-based models that have been specifically trained on a large corpus of Romanian text. A notable exam-

ple is RoBERT [MRD20], which is designed to better understand the specific grammar and nuances of the Romanian language.

Machine learning approaches vary in their complexity, computational requirements, and performance. For Romanian fake news detection, the choice of model often depends on the available training data, computational resources, and the need for interpretability. While transformer-based models generally achieve the highest accuracy, traditional methods may offer advantages in terms of efficiency and explainability in certain contexts.

3.4 Defining News Categories and Disinformation

To build a classifier, it is first necessary to define the categories it will predict. While "fake news" is often used as a general term, it is part of a broader spectrum of misleading information. This thesis uses a five-category classification scheme.

- **Real News**: This category refers to factual, verifiable information presented in a neutral and objective manner. The primary goal of real news is to inform the public. It follows established journalistic standards, such as citing sources and providing balanced perspectives.
  Example: "Primăria a anunțat astăzi că lucrările de reabilitare a podului vor începe pe 1 august, conform calendarului stabilit. Proiectul este finanțat din fonduri europene și are ca termen de finalizare luna decembrie." (The City Hall announced today that rehabilitation work on the bridge will begin on August 1st, according to the established schedule. The project is financed by European funds and has a completion deadline of December.)

- **Fake News**: This refers to content that is intentionally and verifiably false, created with the specific purpose of deceiving its audience, often for financial or political gain. It mimics the style and format of legitimate news to appear credible.
  Example: "Oamenii de știință au confirmat că un asteroid de mărimea unui stadion de fotbal va lovi Pământul săptămâna viitoare, dar guvernele ascund informația pentru a evita panica." (Scientists have confirmed that an asteroid the size of a football stadium will hit the Earth next week, but governments are hiding the information to avoid panic.)

- **Misinformation**: This term describes false information that is shared without a deliberate goal to mislead. Frequently, the person spreading the content believes it to be correct. Misinformation can often stem from a misunderstanding or exaggeration of a real event.

Example: "Am citit pe un forum că dacă bei ceai de ghimbir în fiecare dimineață, te vindeci de orice boală. Prietena mea a încercat și acum se simte mult mai bine." (I read on a forum that if you drink ginger tea every morning, you will be cured of any illness. My friend tried it and now she feels much better.)

- **Propaganda**: This is information, often biased or misleading, that is used to promote a specific political cause or point of view. While it may contain some truthful elements, its primary goal is to influence an audience's opinions and behavior, not to inform them objectively.
  Example: "Conducerea actuală a țării ne-a adus numai probleme și datorii. Doar un lider puternic, care înțelege nevoile poporului, poate salva națiunea noastră de la dezastrul iminent." (The country's current leadership has brought us nothing but problems and debt. Only a strong leader who understands the people's needs can save our nation from the imminent disaster.)

- **Satire**: This type of content uses tools like humor, irony, and over-the-top exaggeration to critique flaws and shortcomings. It often targets political matters or other current topics. While not factually accurate, its purpose is to provide social commentary and entertain, not to mislead the audience
  Example: "Guvernul a anunțat un nou program de combatere a traficului, prin care fiecare mașină va fi echipată cu aripi. Primele teste au arătat că mașinile zboară, dar aterizează în curtea vecinului." (The government has announced a new program to combat traffic, in which every car will be equipped with wings. The first tests showed that the cars fly, but land in the neighbor's yard.)

3.5 Challenges in Fake News Detection

Developing effective fake news detection systems faces several challenges, particularly for languages like Romanian. This section outlines the main obstacles researchers encounter in this field.

3.5.1 Language-specific Challenges

Romanian language presents unique difficulties for automated detection systems:

- **Linguistic Complexity**: Romanian's Latin-based structure with Slavic influences creates complex morphology and syntax that can be difficult for models to process accurately.

- **Diacritics Inconsistency**: The inconsistent use of diacritical marks (ă, â, î, ș, ț)

in online content creates text normalization challenges and affects the performance of NLP algorithms.

- **Language Evolution**: The rapid evolution of Romanian internet language, including slang, abbreviations, and foreign word adoption, makes it difficult for detection systems to stay current.
- **Dialectal Variations**: Regional differences in Romanian vocabulary and expressions can complicate text analysis and model generalization.

3.5.2 Limited Resources for Romanian

The development of NLP systems for Romanian faces resource constraints:

- **Small Datasets**: Compared to widely-spoken languages, Romanian has smaller labeled datasets for fake news, limiting the training data available for machine learning models.
- **Fewer Pre-trained Models**: The scarcity of Romanian-specific pre-trained language models requires either adaptation of multilingual models or resource-intensive development of new ones.
- **Limited Benchmarks**: Fewer established benchmarks make it difficult to compare different approaches and establish state-of-the-art performance for Romanian fake news detection.
- **Sparse Research Community**: A smaller research community focused on Romanian NLP leads to slower advancement in language-specific techniques and resources.

3.5.3 Ethical Considerations

Fake news detection systems raise important ethical questions:

- **False Positives**: Incorrectly labeling legitimate news as fake can harm credible sources and potentially limit freedom of expression.
- **Algorithmic Bias**: Detection systems may inherit biases from training data, potentially affecting certain topics or perspectives disproportionately.
- **Transparency Issues**: Complex models like deep neural networks often function as ”black boxes”, making it difficult to explain their decisions to users.
- **Privacy Concerns**: Analyzing user behavior to detect misinformation spread patterns may raise privacy issues regarding data collection and usage.

3.5.4 Adversarial Challenges

Fake news creators actively work to evade detection:

- **Evolving Techniques**: Misinformation tactics constantly evolve in response to detection methods, creating a technological arms race.
- **Subtle Misinformation**: Some fake news employs sophisticated techniques like mixing factual and false information, making detection more difficult than with obviously false content.
- **Domain Adaptation**: Models trained on one type of misinformation may perform poorly on new domains or topics, requiring continuous updating.

Addressing these challenges requires interdisciplinary approaches combining NLP, machine learning, linguistics, and media studies. For Romanian fake news detection, developing language-specific resources and adapting state-of-the-art techniques to account for the unique characteristics of Romanian language and media ecosystems remains crucial for success.

Chapter 4

Dataset Analysis and Embedding Representations

This chapter presents the analysis of the dataset used for Romanian fake news classification, followed by embedding-based representations using both static and contextualized approaches. It includes linguistic insights, emotion and sentiment distributions and visualization of learned embedding spaces.

4.1 Dataset Overview and Preprocessing

The foundation of this study is a preprocessed dataset of Romanian news articles, named "NEW"[Mih], originally introduced by Moisi et al. [MMC+24]. The dataset combines articles from two primary sources: the FakeRom dataset[Aas] and a collection of articles from Veridica.ro[Ver], a Romanian fact-checking platform. This chapter details the analysis performed on this dataset.

This initial version contained 5,160 news articles stored in a JSON format, each with `content` and `tag` features. The dataset was specifically prepared to be balanced, with data augmentation techniques like synonym replacement applied to underrepresented categories. It also used oversampling methods, such as the `RandomOverSampler` technique, to ensure each of the five categories— `fake news`, `misinformation`, `propaganda`, `real news`, and `satire`—was equally represented with 1,032 samples. Furthermore, the initial text cleaning involved several standard procedures: converting all text to lowercase, removing all non-letter characters and isolated single letters, standardizing white spaces, and trimming any leading or trailing spaces from the text.

Before proceeding with analysis, an additional validation and cleaning phase was performed on the preprocessed dataset to ensure its integrity. This process focused on removing duplicate and low-quality entries. First, duplicate articles were

identified based on their content, leading to the removal of 819 duplicates, but keeping their first appearance. Next, articles with a content length of fewer than 50 characters were considered too short for meaningful analysis and were also removed, resulting in the deletion of 7 articles. These cleaning steps reduced the total dataset size from 5,160 to 4,334 articles.

The removal of duplicates and short articles altered the initial class balance. The new distribution of articles is shown below in Table 4.1.

Table 4.1: Distribution of news categories.

Category	Count	Percentage
Real News	1,032	23.8%
Misinformation	916	21.1%
Satire	870	20.1%
Fake News	812	18.7%
Propaganda	704	16.2%

4.2 Analysis Perspective

Figure 4.1 illustrates the flow of this chapter which starts with the initial dataset and continues with cleaning and preprocessing processes before using the processed dataset for all analysis.

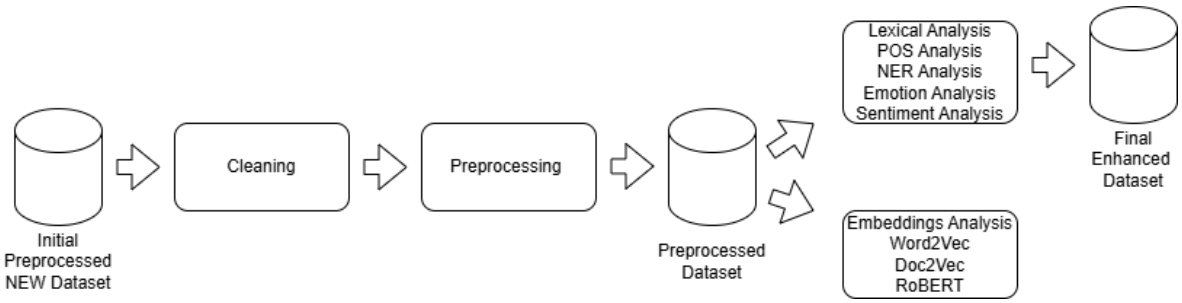


Figure 4.1: Dataset analysis flow.

4.3 Lexical and Syntactic Analysis

To understand the linguistic characteristics that distinguish different news categories, a detailed lexical and syntactic analysis was performed. For this stage, each article in the cleaned dataset was processed through a `spaCy` pipeline configured for Romanian [Exp]. This process involved tokenization, lemmatization, and Part-of-Speech (POS) tagging. To ensure a meaningful analysis of word usage, com-

mon stop words, conjunctions, prepositions, punctuation, and non-alphabetic tokens were filtered out before calculating word counts and vocabulary sizes.

4.3.1 Word Count and Article Length

The analysis began by examining the length of articles, measured by the count of filtered lemmas. The results show a clear difference in length between disinformation-oriented content and factual reporting.

On average, articles in the `fake news` (154 words), `misinformation` (150 words), and `propaganda` (145 words) categories are substantially longer than those in `real news` (83 words) and `satire` (61 words). To put this in perspective, an average **fake news article is approximately 85% longer than an average real news article**. This suggests that deceptive content might employ longer, more elaborate narratives to build a seemingly credible story or to obscure facts, whereas legitimate news reporting tends to be more direct and concise.

4.3.2 Vocabulary Size

Next, the vocabulary size for each category was analyzed by counting the number of unique lemmas. This metric measures how rich or varied the vocabulary is.

The `fake news` category utilizes the largest vocabulary with 11,771 unique lemmas, closely followed by `real news` with 10,699. This indicates that both categories use a broad vocabulary to discuss diverse topics. In contrast, `propaganda` (9,659) and `misinformation` (7,786) use a more limited vocabulary. The most notable difference is with satire, which uses only `4,847` unique lemmas, less than half the vocabulary of fake news. This aligns with the focused and often repetitive nature of satirical writing.

4.3.3 Part-of-Speech (POS) Analysis

To analyze the syntactic structure, the distribution of key Parts-of-Speech (POS), **nouns, verbs, adjectives, and adverbs** was calculated. The average counts per category article reveal a distinct structural pattern, particularly in the use of descriptive language (Figure 4.2).

The results consistently show that the `fake news`, `misinformation`, and `propaganda` categories contain a higher density of certain word types. Comparing `fake news` to `real news`, the differences are significant: an average fake news article contains **77% more nouns** (80 vs. 45) and it features **97% more adjectives** (25 vs. 13).

This significantly higher concentration of nouns and especially adjectives suggests that disinformation relies heavily on descriptive and evocative language. This strategy may be used to create vivid mental images, appeal to emotions, or present opinions as facts. In contrast, the more balanced and concise structure of real news points to a more objective and fact-based reporting style. The high standard deviation in these counts also indicates a wide variety of writing styles within the disinformation categories, ranging from short, punchy articles to long, elaborate narratives.

The `satire` category demonstrates remarkable consistency. Its mean count and median are nearly identical for every POS, and it has the lowest standard deviation across the board. This points to a **highly predictable and formulaic structure** in satirical writing. A more subtle finding lies in the use of adverbs. While `satire` is concise in its use of nouns and adjectives, it employs more adverbs on average (9.2) than `real news` (6.5). This may reflect a stylistic tendency to modify actions and descriptions for comedic or exaggerated effect.

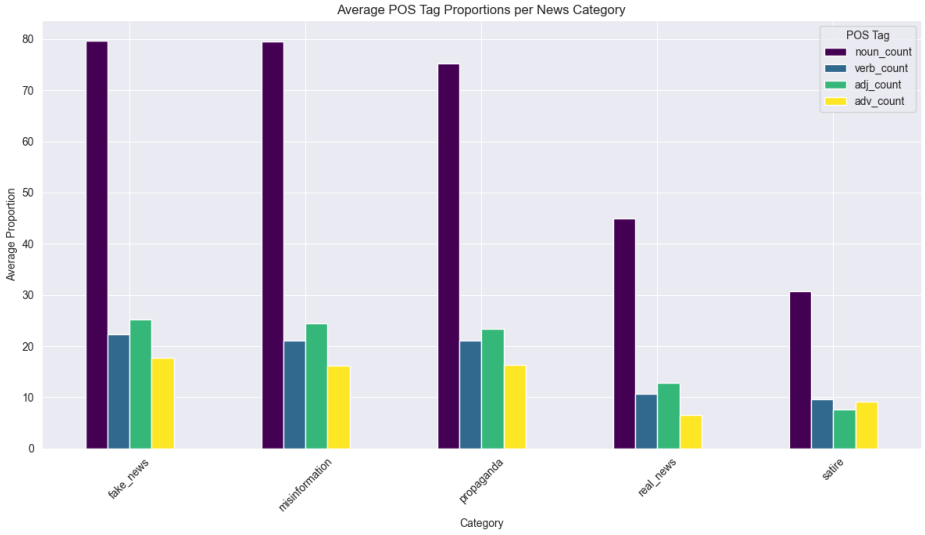


Figure 4.2: Average POS tag proportions per news category.

4.3.4 Prominent Terms

To visually summarize the most prominent terms in each category, word clouds (Figure 4.3) were generated from the filtered lemmas. It is important to note that a significant portion of the dataset originates from the COVID-19 pandemic era. As a result, all five categories share a common thematic core, with terms like `covid`, `vaccin`, `coronavirus`, and `pandemie` being dominant across the board.

While this thematic overlap makes the clouds appear similar at first, a closer look

of the secondary terms reveals subtle but crucial differences in framing and focus, which align with the nature of each category.

- **Real news**: The vocabulary is characterized by institutional and procedural language. Terms like *comunicat* (press release), *guvern* (government), *minister* (minister), *decizie* (decision), and *sursă* (source) are prominent. This reflects a focus on official announcements, journalistic sourcing, and the actions of state bodies.

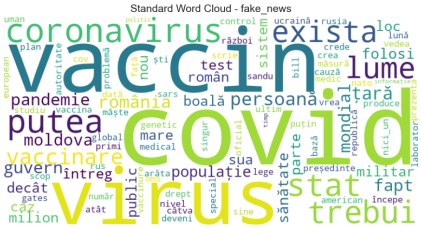
- **Fake news**: This category's vocabulary leans heavily into conspiracy theories. The presence of *Bill Gates*, alongside terms like *control*, *sistem*, and the speculative word *putea* (could/might), points to narratives about hidden control and powerful figures.

- **Propaganda**: The language here attempts to mimic scientific and official discourse to build credibility for a specific agenda. Words like *studiu* (study), *efect*, *cauză* (cause), and references to *OMS* (WHO) appear alongside geopolitical terms like *american*.

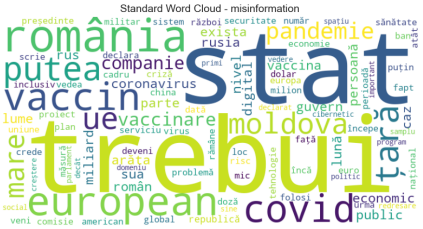
- **Misinformation**: This cloud shows a focus on broader socio-economic and political themes. Prominent words include *trebui* (must/should), suggesting a moralizing tone, as well as *nivel* (level), *digital*, *cibernetic* (cyber), *securitate* (security), *economic*, and *UE* (EU). This indicates a shift from health conspiracies to narratives about policy and technology.

- **Satire**: The satirical vocabulary is the most distinct, collecting well-known conspiracy tropes to create absurdity. The appearance of *cip* (microchip), *Bill Gates*, *Soros*, and even *anti-Dumnezeu* (anti-God) alongside everyday words like *bani* (money) and *casă* (house) clearly signals mockery and exaggeration.

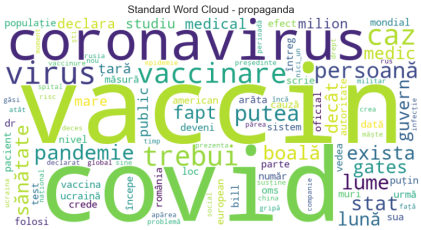
In summary, while the main topic is often shared, the surrounding vocabulary clearly distinguishes each category, exposing its underlying strategy—whether it's factual reporting, pushing conspiracies, using pseudo-scientific arguments, spreading socio-economic fear, or mocking everything outright.



(a) Fake news



(b) Misinformation



(c) Propaganda

(d) Real news

(e) Satire

Figure 4.3: Standard word clouds showing the most frequent terms for each news category.

4.4 Named Entity Recognition (NER)

A Named Entity Recognition (NER) analysis was carried out in order to identify entities in the articles. Entities were extracted and categorized using the `spaCy` model. People, locations, organizations, nationalities/religious/political groups, facilities, events, and money were the main categories that were the focus of this analysis.

The most immediate finding is in the sheer quantity of entities mentioned. The disinformation-focused categories consistently feature a higher number of named entities than factual or satirical content (Figure 4.4). An average `misinformation` article contains 13.4 entities, a `propaganda` article contains 13.3 and a `fake news` article contains 12.7. In contrast, an average `real news` article contains only 7.8 entities, approximately 40% fewer. `Satire` is the most sparse, with only 5.7 entities on average.

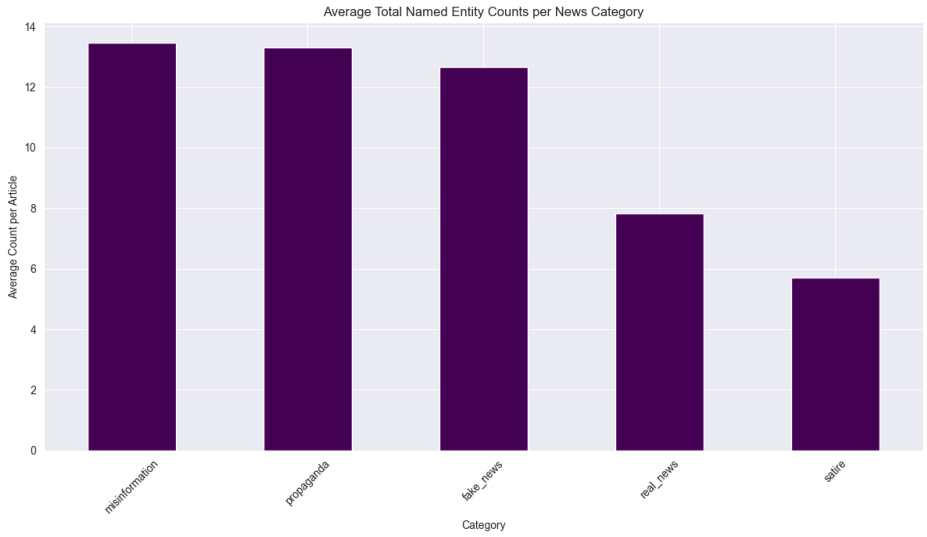


Figure 4.4: Average Total Named Entities per News Category

This implies that dishonest articles frequently contain a lot of references to individuals, organizations, and groups, probably in an effort to construct a complex story and give the impression of authority or credibility.

A closer look at the types of entities reveals further strategic differences.

- **Persons and Organizations**: Disinformation articles, reference significantly more people and organizations than `real news`. This overuse of specific names can be a tactic to attribute blame, fabricate quotes, or lend undue weight to a story by associating it with well-known figures or institutions.

- **Groups**: The most notable distinction appears in the this category. `Misinformation` articles lead with an average of 3.2 such entities per article, more than double

the rate of `real news` (1.5). This highlights a common strategy in misinformation: invoking identities tied to nationality, religion, or politics to create divisive "us vs. them" narratives.

Other entities like locations and events were mentioned less frequently across all categories, making them less distinctive as features.

Finally, the standard deviation results (Figure 4.5) support the lexical analysis. The high variability in entity counts for `fake news` and `propaganda` indicates a lack of stylistic consistency. `Satire`, conversely, shows the lowest standard deviation, reinforcing the idea that it follows a more predictable and formulaic structure, with a minimal focus on naming specific entities.

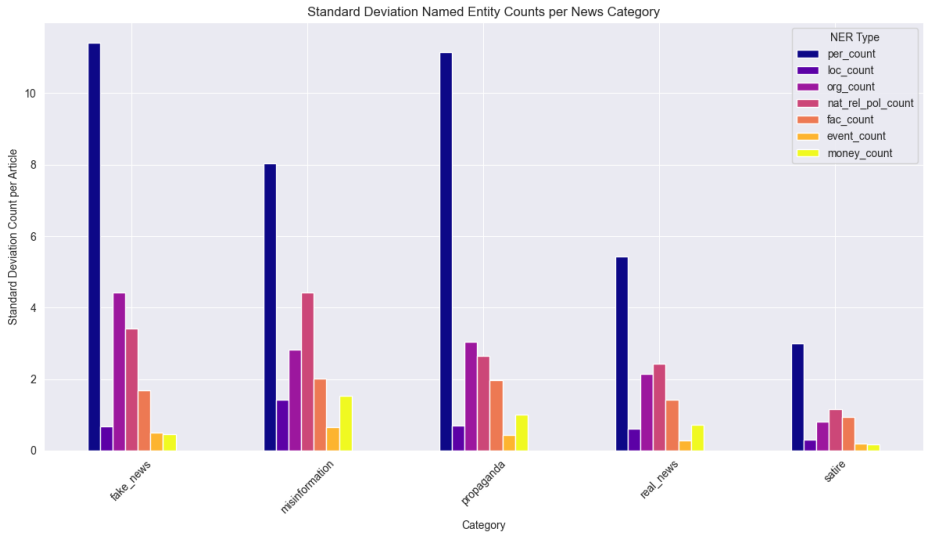


Figure 4.5: Standard Deviation Named Entities Counts per News Category

4.5 Emotion and Polarity Analysis

This section analyzes the emotional and sentimental patterns across the news categories. Using `RoEmoLex` [LB19], a rich Romanian emotion lexicon, words are mapped to eight core emotions (*Anger*, *Anticipation*, *Disgust*, *Fear*, *Joy*, *Sadness*, *Surprise*, *Trust*) and to sentiment polarities (*Positivity*, *Negativity*).

A dictionary was created from `RoEmoLex` containing **6,330 unique lemmas**, which was then used to score each article based on the emotional and sentimental associations of its words. In this process, 2156 duplicate lemmas were removed, keeping only their first occurrence.

4.5.1 The Emotional Intensity of News

The significant difference in the amount of emotional language used is one of the main findings (Figure 4.6). Articles in the fake news category contained an average of 54 emotionally-charged words. This is more than double the average of 25 emotional words found in real news. This indicates that disinformation relies heavily on emotional language as a core component of its strategy.

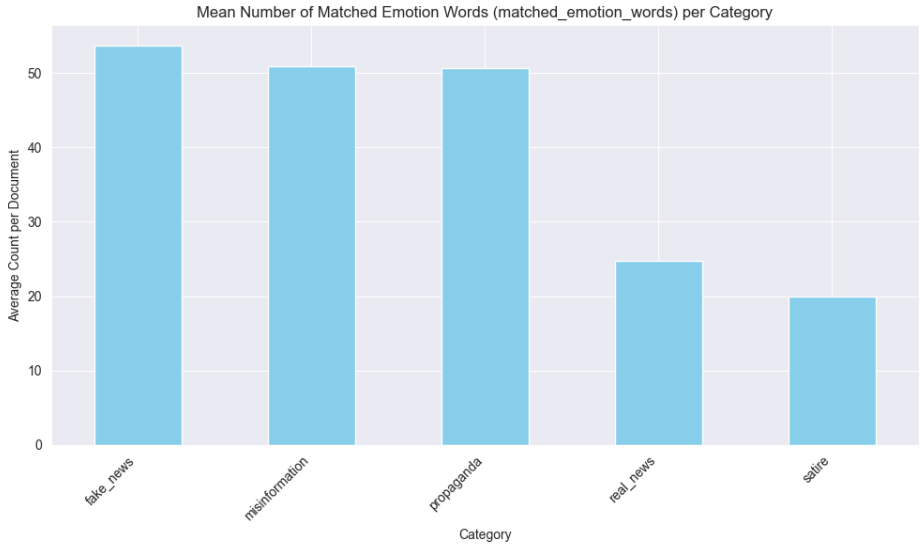


Figure 4.6: Mean number of emotion words per news category

Diving deeper into the specific emotions, a clear pattern emerges (Figure 4.7):

- **Negative Emotions**: The disinformation categories (fake news, propaganda) consistently score the highest in negative emotions like Fear, Anger, Sadness, and Disgust. The average Fear score in fake news (10.4) is nearly three times that of real news (3.7).
- **Trust and Anticipation**: Paradoxically, fake news and misinformation also score highest in Trust and Anticipation. The high Trust score (17.6 for fake news) is particularly notable. This seemingly contradictory signal suggests a sophisticated rhetorical strategy: building a sense of rapport and shared understanding with the reader while simultaneously presenting a world full of threats (Fear) and injustices (Anger).

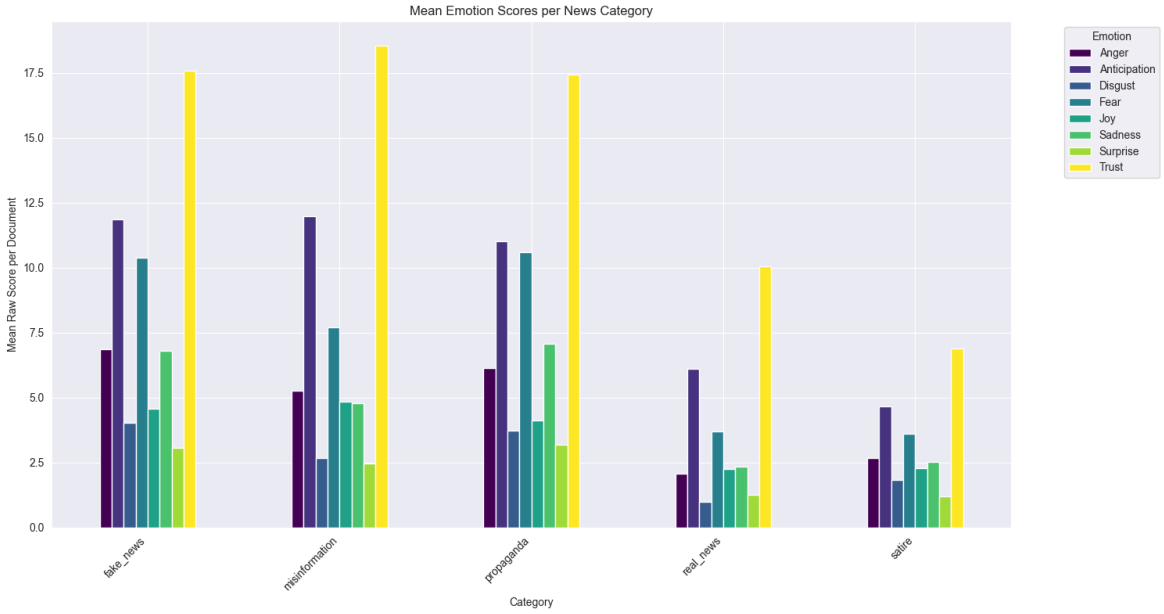


Figure 4.7: Mean emotion scores per news category

Real news and satire exhibit much more emotional restraint, with significantly lower scores across almost all categories, pointing to a more detached and objective tone.

4.5.2 Emotional Word Clouds

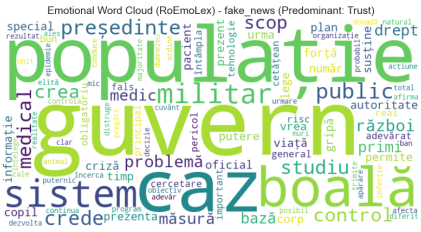
To give a clearer picture of the language behind the emotional scores, word clouds were created for each category (Figure 4.8) using words found in the `RoEmoLex` lexicon. An interesting pattern, shown in each cloud's title, is that *trust* stands out as the dominant emotion across all categories. However, the way trust-related words are used shows very different rhetorical strategies.

- **Real news**: The vocabulary of trust is institutional and procedural. Words like *guvern* (government), *comisie* (commission), *oficial*, and *securitate* (security) are central. This reflects a reliance on the language of established authorities and processes, aiming to build credibility through factual reporting on civic matters.

- **Fake news and propaganda**: These categories often mimic the formal language of real news—using words like *guvern* (government), *studiu* (study), and *medic* (doctor)—but place them in a context of conflict and control. They add terms like *putere* (power), *control*, *militar* (military), and *pericol* (danger) to shift the message. This tactic builds trust by sounding authoritative, then redirects that trust toward a narrative of hidden threats and power struggles.

- **Misinformation**: This category also uses trust-related words like *guvern* (government), *public*, and *proiect* (project), but places them in a socio-economic context. Terms like *economic*, *criză* (crisis), and *război* (war) point to a strategy where trust is used to make stories about major economic and geopolitical instability seem more credible.

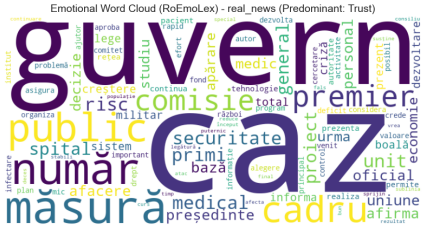
- **Satire**: In the satire cloud, the language of trust is completely subverted. While it contains words like *crede* (believe) and *bun* (good), they are surrounded by absurd or mundane terms (*șobolan* - rat, *delfin* - dolphin) and words of judgment (*prost* - stupid, *păcat* - sin). This contrast does not aim to build real trust, but to mock the very act of belief by placing trust-related words in ridiculous contexts.



(a) Fake news

(b) Misinformation

(c) Propaganda



(d) Real news

(e) Satire

Figure 4.8: Emotional word clouds for each news category based on the RoEmoLex lexicon.

In essence, this analysis shows that *Trust* is a central theme in the information landscape. It is used legitimately in real news, weaponized to create false credibility in disinformation and propaganda, and satirized for comedic effect.

4.5.3 Sentiment Polarity

While raw emotion scores are insightful, analyzing the proportion of positive versus negative words within the emotional content of an article reveals the underlying sentiment.

Interestingly, when considering only the emotionally-charged words, all categories have a high proportion of positive sentiment, with real news and misinformation averaging around 45% positive words (Figure 4.9).

The key differentiator is the proportion of negative sentiment. satire is the most negative category, with 30% of its emotional words being negative. It is closely followed by fake news at 27%. In contrast, real news is the least negative, with only 18% of its emotional words carrying a negative connotation. This supports the idea that, while real news does express emotion, it does so with a much more neutral and balanced tone than fake news or satire.

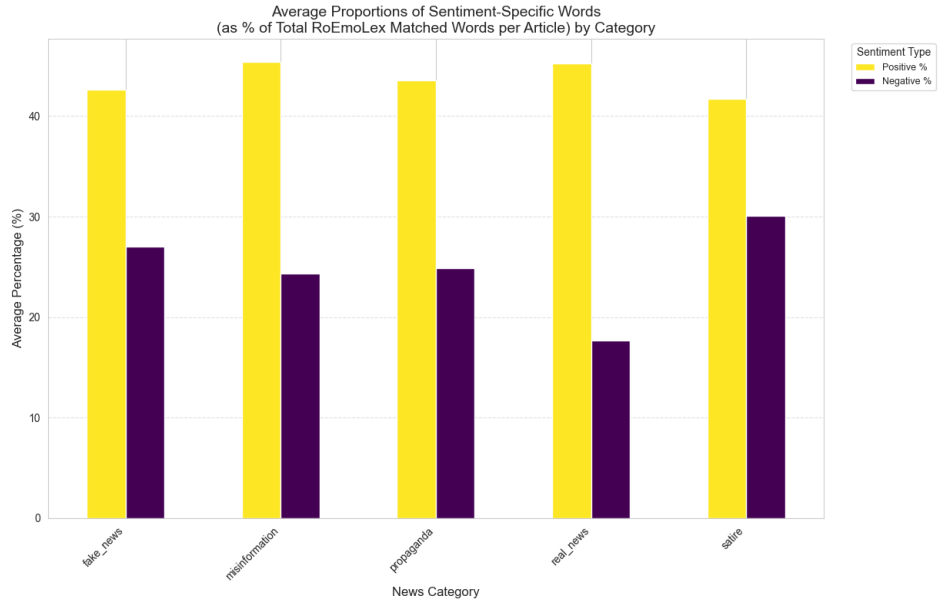


Figure 4.9: Average proportions of sentiment specific words per news category

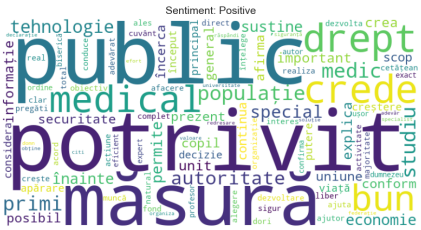
4.5.4 Sentiment Word Clouds

To better understand the vocabulary that constitutes these positive and negative sentiments across the dataset, word clouds were generated for each polarity (Figure 4.10). These visualizations reveal the core themes associated with each sentiment.

- The **Positive** sentiment vocabulary is strongly associated with institutions, order, and expertise. Prominent words include *potrivit* (suitable/according to), *măsură* (measure), *autoritate* (authority), *public*, *drept* (right), and *susține* (sup-

ports). This is the language of governance, civic society, and scientific credibility, used to frame arguments as well-founded and beneficial.

- In contrast, the **Negative** sentiment vocabulary is centered on themes of threat, conflict, and suffering. It is dominated by words like *boală* (disease), *criză*, *problemă*, *război* (war), *pericol* (danger), *fals*, and *control*. This vocabulary is clearly used to generate alarm, highlight danger, and frame topics in terms of harm and instability.



(a) Positive (b) Negative

Figure 4.10: Word clouds showing the most frequent terms for positive and negative sentiment polarities.

This confirms that in the context of this dataset, positive language is used to establish authority and correctness, while negative language is used to frame threats and crises.

4.6 Embedding Representations and Visualization

This section describes how embedding models are used to go beyond surface-level statistics and capture the semantic relationships within the dataset. Text is represented by these models as dense numerical vectors. Three different strategies will be examined: static word-level embeddings with `Word2Vec`; static document-level embeddings with `Doc2Vec`; and dynamic and contextualized embeddings based on `RoBERT`.

4.6.1 Word2Vec: Static Word Embeddings

The first approach to understanding the semantic space of the dataset involved creating word-level embeddings using the `Word2Vec` model. This model learns vector representations of words from a text corpus, placing words with similar meanings in close proximity within a continuous vector space.

For this task, the `gensim` library ([RS10]) was used to train two separate models on the lemmatized content of the 4,334 cleaned documents: one creating 150-dimension vectors, and a more complex one creating 300-dimension vectors. Both

models shared the following configurations: the model used was **Skip-gram**; the **window size** was set to **5**, meaning the model learned a word's context by considering five words preceding and succeeding it; the **minimum count** was set to **5** to exclude very rare words from the vocabulary; and the model was trained for **10 epochs**.

This process resulted in a final vocabulary of **10,771 unique lemmas**. Upon evaluation, the 300-dimension model was found to capture richer and more nuanced semantic relationships. Therefore, it was selected for all subsequent analyses and for use in the final application.

To explore the learned semantic space and uncover the specific biases of this news-focused corpus, two qualitative analyses were designed:

1. Semantic Opposition Analysis: This involves comparing the semantic neighbors (i.e., the closest words in the vector space) of two words with opposing meanings. This test helps to visualize how well the model has learned concepts like polarity and context.

2. Contextual Bias Analysis: This involves comparing the neighbors of a specific target word from the custom model against the neighbors of the same word from a pre-trained, general-purpose Romanian language model. This comparison is designed to highlight how the meaning and associations of words can shift within the specialized context of fake news.

The results of these analyses will be presented in the following sections.

Analysis of Semantic Opposites

To validate the semantic space learned by the Word2Vec model, a series of tests were conducted comparing the neighborhoods of words with opposing meanings. The t-SNE algorithm ([vdMH08]) was used to visualize the high-dimensional vectors by projecting them into two-dimensional space. The results show that the model successfully learned not only the definitions of words but also the specific context in which they are used within the news dataset.

- Știință (science) vs. conspirație (conspiracy): The model effectively distinguishes between these two domains. The neighbors of "știință" include words like *rațiune* (reason) and *filosof*, framing it as a respected intellectual pursuit. The neighbors of "conspirație" describe its nature, including *teorie* and words like *verificabil*, which reflects how conspiracy theories are often presented as "verifiable" to their audience.

- Drept (right/law) vs. control: This visualization (Figure 4.11) shows a separation between civic ideals and conspiratorial fears. "Drept" is associated with

libertate (freedom) and *cetățenesc* (civic). "Control" is associated with forced obligation (*obligare*), plots (*complot*), and futuristic technology (*nanoroboțică*), which are common themes in disinformation narratives.

Figure 4.11: t-SNE visualization of the neighborhoods for **drept** and **control**.

Overall, these visualizations demonstrate that the `Word2Vec` model has successfully captured the nuanced and often biased contexts in which words are used throughout the dataset. The model learned not just dictionary definitions, but the specific rhetorical pairings and framings that characterize the different news categories.

Contextual Bias Analysis

The second analysis was designed to highlight the contextual biases learned by the `Word2Vec` model. To achieve this, the neighbors of key terms were compared against those from a general-purpose Romanian model provided by the CoRoLa project ([VD18]). This general model is much larger, with a vocabulary of over 250,000 words, but it is not lemmatized.

This difference in lemmatization is immediately apparent in the results. The general model's neighbors for a word are often just different morphological variations

of that same word. In contrast, the custom lemmatized, domain-specific model provides neighbors that reveal the specific discourse and context surrounding a term within the news dataset. This highlights the value of creating specialized, lemmatized models, which is why the models developed in this thesis are made available for download to contribute to the field of NLP for the Romanian language.

The following comparisons reveal how the meaning of words can shift from a general definition to a specialized, often biased, contextual one.

Target Word: `vaccin`

The General Model provides neighbors that are simply variations of the target word (*vaccinul*, *vaccine*, *vaccina*), defining the word in a closed, dictionary-like loop. The Fake News Model, however, associates "vaccin" with its real-world pandemic context. The neighbors are specific vaccine manufacturers (*Moderna*, *Janssen*), related viruses (*rubeolă* - rubella), and even brand names (*Vaxzevrie*). This shows a much richer, more applied understanding of the term.

Target Word: `expert`

The General Model defines "expert" by professional role, with neighbors like *consultant*, *consilier* (advisor), *evaluator*, and *specialist*. The Fake News Model defines "expert" by association within the articles. The neighbors are proper names of individuals cited as experts (*Stephen*, *Voicu*, *Boyle*), as well as unusual or satirical terms like *șobologie* (the study of rats), which was likely picked up from a specific satirical article. This demonstrates that the model learns to associate "expert" with whoever is named as one, regardless of the context's validity.

Target Word: `guvern`

The General Model links "guvern" to state institutions, with neighbors like *parlament*, *coaliție*, *miniștrii* (ministers). The Fake News Model captures a context of political action and conflict. The neighbors include *ordonanță* (emergency decree), *autoritar*, and *cerem* (we demand), reflecting a discourse of governmental overreach and public opposition.

In conclusion, this comparative analysis shows that the custom-trained Word2Vec model effectively captured the specific, nuanced, and often biased language of the news dataset. Unlike a general model that reflects broad word meanings, this domain-specific model learns how words are used to shape arguments, build narratives, and add context.

4.6.2 Doc2Vec: Static Document Embeddings

Moving from word-level to document-level representations, the next step involved training a `Doc2Vec` model. The goal of this model is to create a single, fixed-size vector for each article, capturing its overall semantic meaning. This allows for direct comparison and clustering of entire documents.

Similar to the `Word2Vec` approach, two models and were initially trained using the `gensim` library. Early tests showed that the **150-dimension** model failed to produce meaningful separations between the document categories. Therefore, the **300-dimension** model was used for all subsequent analysis.

The model was configured as follows: the **PV-DM** model was used (Distributed Memory Model of Paragraph Vectors); the **window size** was set to **5**; the **minimum count** was set to **5**; and the model was trained for **20 epochs**. The training data contained 4,334 cleaned and lemmatized documents.

To analyze the resulting document vectors, the t-SNE algorithm was again used to visualize the high-dimensional space in two dimensions.

Analysis of Document Clusters

The first visualization (Figure 4.12) plots the document vectors for all five news categories. The plot shows that the categories do not form distinct, well-separated clusters. Instead, there is a significant amount of overlap. An interesting pattern emerges where `real news` articles (**red**) form a relatively dense core near the center of the plot. The other categories, especially `fake news` (**blue**) and `satire` (**purple**), are more scattered around this central core. This may suggest that `real news` articles in the dataset share a more consistent thematic space, while the other categories are more diverse in their semantic structure or are defined by their deviation from this factual core.

To investigate further, the three disinformation categories—`fake news` (**blue**), `misinformation` (**orange**), and `propaganda` (**green**)—were plotted alone (Figure 4.13). Even when isolated, these categories show considerable overlap. This finding suggests that the different types of disinformation in the dataset are semantically very similar, likely employing related topics and rhetorical strategies, which makes them difficult to distinguish from one another using a single document vector.

The final and most direct comparison was between `fake news` (**blue**) and `real news` (**orange**) (Figure 4.14). This visualization confirms that there is no simple, clean boundary to separate the two categories. However, it reveals a subtle and important structural difference. The `real news` articles form a dense semantic core, which suggests a high degree of stylistic and topical consistency among them. In contrast, the `fake news` articles are more diffuse, forming a larger cloud that sur-

rounds this factual core. This indicates that while fake news often discusses the same topics as real news, it does so with much greater stylistic and narrative variety.

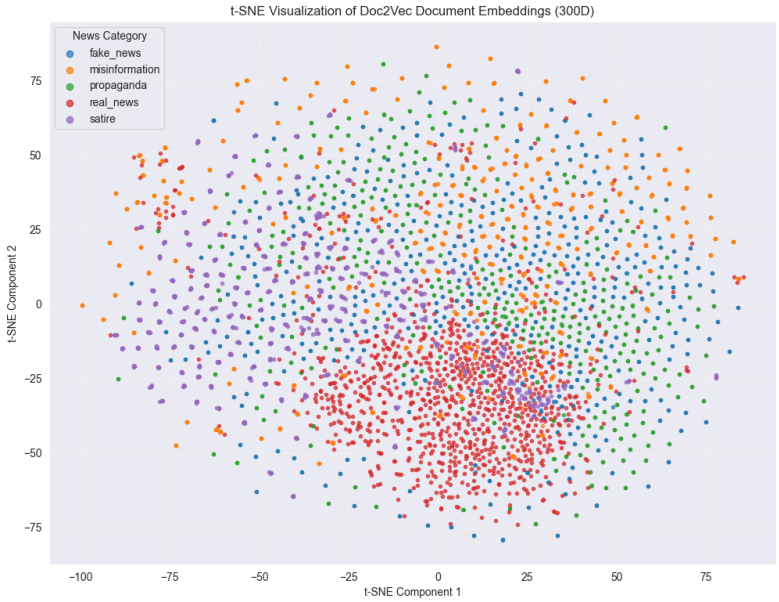


Figure 4.12: t-SNE visualization of Doc2Vec embeddings for all news categories.

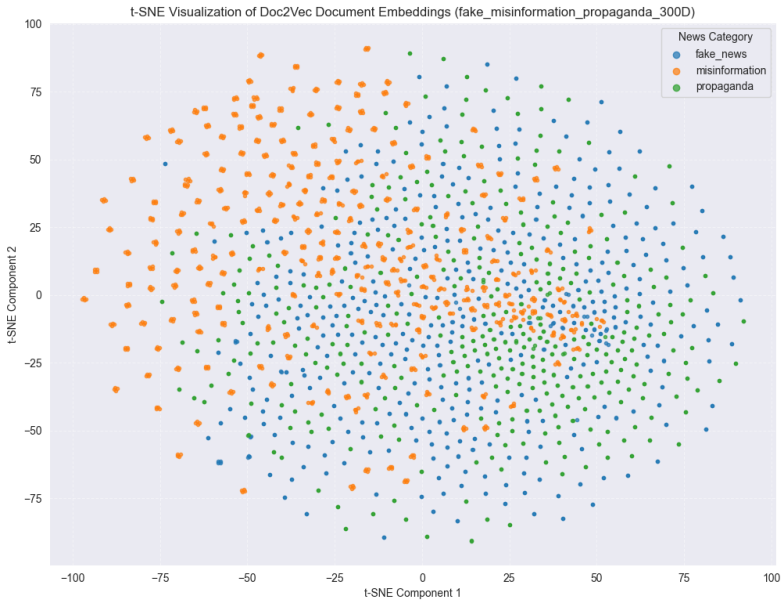


Figure 4.13: t-SNE visualization of Doc2Vec embeddings focusing on the disinformation categories.

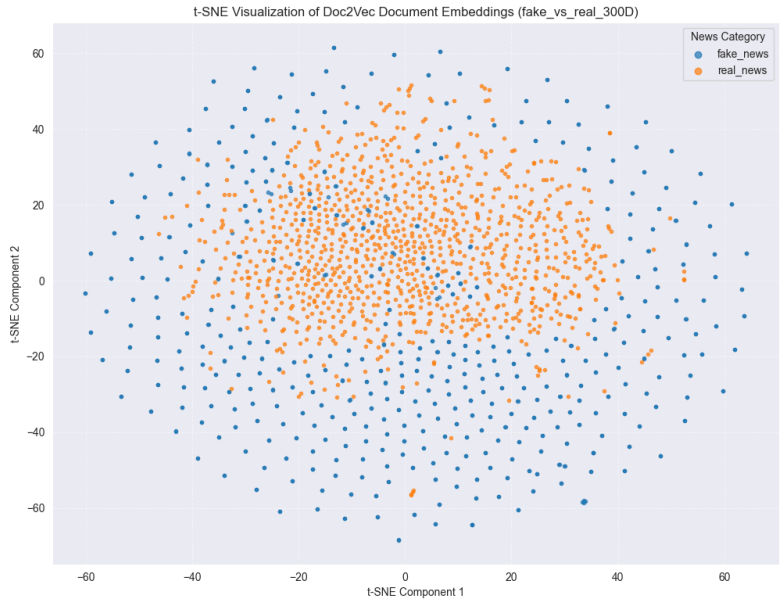


Figure 4.14: t-SNE visualization of Doc2Vec embeddings focusing on fake and real news.

This structural difference is a crucial insight. It demonstrates that although `Doc2Vec` on its own is insufficient for creating a simple classifier, due to the heavy overlap, it successfully reveals a key characteristic of the dataset. The model captures that `real news` has a more consistent semantic "fingerprint", while `fake news` is more varied. This finding highlights the need for more sophisticated classification models that can learn the complex, non-linear boundaries between these overlapping but structurally different classes.

4.6.3 RoBERT: Contextualized Embeddings

The approach of text representation explores contextualized embeddings generated by the `RoBERT-base` model [MRD20]. Unlike Word2Vec and Doc2Vec, which generate a single, static vector for each word or document, RoBERT creates dynamic representations where the vector for a word changes based on its surrounding context.

Tools and Frameworks: `AutoTokenizer`, `AutoModel`, `transformers`, `torch`

This analysis uses the pre-trained RoBERT model as a **feature extractor without fine-tuning**. The goal is to evaluate the raw, out-of-the-box capabilities of its contextualized representations on this specific dataset.

Analysis of Contextualized Word Embeddings

To demonstrate contextualization at the word level, an analysis was conducted to show that RoBERT generates different embeddings for the same word depending on its usage. This was achieved through the following process:

1. A target word, "*vaccin*", was selected based on its presence across all three categories in the word cloud analysis. For each category, the first article containing this word was chosen to ensure systematic sampling.

2. Articles where this word appeared—from the real news, fake news, and satire categories—were chosen.

3. For each article, the text was tokenized and passed through the pre-trained RoBERT model to extract the last hidden layer representations for all tokens.

4. The hidden state vectors for each "*vaccin*" instance were extracted, with subword tokens averaged when necessary to create a single word-level representation.

The cosine similarity between the three distinct embeddings was then calculated. The similarity between the real news and fake news contexts was 0.6655, between real news and satire it was 0.6953, and between fake news and satire it reached 0.7224. Figure 4.15 illustrates the t-SNE visualization of the contextualized embeddings of the word vaccin in real, fake, and satire news contexts.

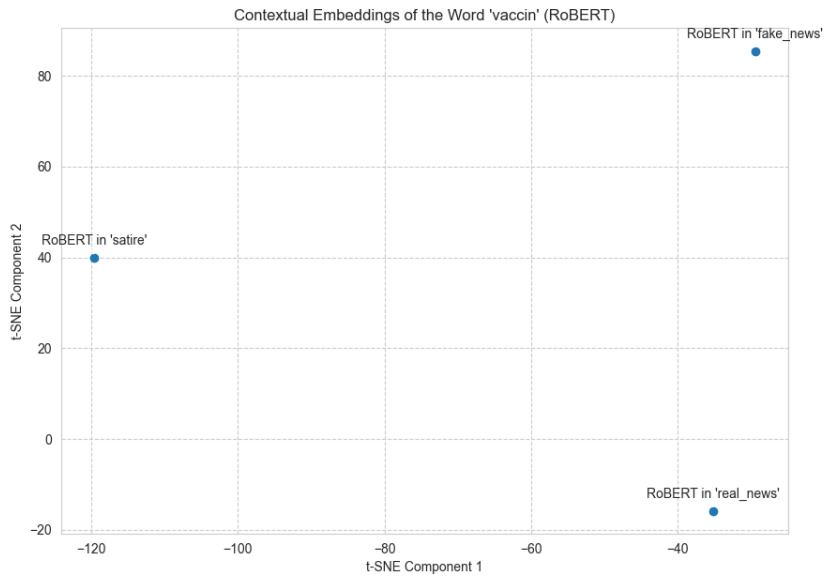


Figure 4.15: t-SNE visualization of contextualized embeddings of the word **vaccin**.

The moderate similarity scores show meaningful contextual differences, confirming that the model is indeed generating unique, context-dependent vectors for the same word. This is a fundamental advantage over static models like Word2Vec, which would have produced a similarity of 1.0 in all cases.

Document-Level Feature Extraction

In addition to the word-level analysis, a document-level representation was generated to serve as features for a traditional classifier in a later experiment. This was done to establish a performance baseline for a non-fine-tuned transformer.

The standard method was used: for each of the 4,334 articles, the final hidden state of the special [CLS] token was extracted from the pre-trained RoBERT model. This process resulted in a single 768-dimensional vector for each document. These vectors were then saved to be fed into an SVM classifier, and the results of that experiment are detailed in the experiments chapter.

Chapter 5

Classification Experiments and Results

5.1 Introduction

This chapter presents the results of several classification experiments. The main goal was to find the most effective method for representing text to automatically detect fake news in Romanian articles.

To achieve this, two approaches were tested and compared:

- First, an evaluation of four text representation techniques: **TF-IDF** (high-dimensional and sparse), **Word2Vec**, **Doc2Vec**, and **RoBERT**.
- Second, a **hybrid model** was developed and tested. This model combines semantic document embeddings with a custom set of engineered linguistic and emotional features.

To ensure a fair comparison, all methods were evaluated using the same Support Vector Machine (SVM) classifier and the same testing process. The following sections will describe the experimental setup, detail each experiment, and discuss the final results.

5.2 Experimental Setup

This section describes the setup used for all classification experiments. This includes the dataset, the evaluation method, the chosen classifier, and the metrics used to measure performance.

5.2.1 Dataset

The dataset used in this research consists of 4,334 Romanian news articles. Each article is assigned one of five labels: `real`, `fake`, `misinformation`, `propaganda`,

and `satire`. A more detailed description of the data collection and preprocessing steps is available in Chapter 4.1.

5.2.2 Evaluation Methodology

To get a reliable measure of each model's performance, a method called **5-fold stratified cross-validation** was used. This process splits the entire dataset into five equal parts, or "folds".

The experiment is run five times. In each run, four folds (80% of the data) are used for training the model, and the remaining one fold (20% of the data) is used for testing. The process then rotates which fold serves as the test set. This ensures that every article is used for testing exactly once.

The "stratified" part is important because it makes sure that each fold has the same percentage of each class as the full dataset. This prevents one fold from having an unfairly high number of samples from a single class and leads to more trustworthy results.

5.2.3 Base Classifier

A **Support Vector Machine (SVM)** was used as the classifier for all experiments. By using the same classifier for every test, we can be sure that any differences in performance are due to the text representation method being tested, not the classifier itself.

5.2.4 Performance Metrics

To evaluate the performance of the models, four standard metrics were calculated. For a multi-class problem like this one, these metrics are derived by considering each class individually and then calculating a weighted average to form a single score.

First, for any given class l from the set of all classes L, we define the following terms:

- **True Positive (TP_l)**: The number of samples correctly classified as belonging to class l.
- **False Positive (FP_l)**: The number of samples incorrectly classified as class l, which actually belong to a different class.
- **False Negative (FN_l)**: The number of samples belonging to class l that were incorrectly classified as a different class.

Based on these counts, the performance metrics are calculated as follows:

Overall Accuracy This is the fraction of all predictions that the model got right, regardless of class.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Weighted Precision, Recall, and F1-Score These metrics are calculated by finding the per-class scores and then averaging them based on the number of true samples for each class.

The Precision and Recall for a single class l are defined as:

$$\text{Precision}_l = \frac{TP_l}{TP_l + FP_l} \quad \text{and} \quad \text{Recall}_l = \frac{TP_l}{TP_l + FN_l}$$

These per-class scores are then combined to get the final weighted metrics:

$$\text{Weighted Precision} = \sum_{l \in L} w_l \times \text{Precision}_l$$

$$\text{Weighted Recall} = \sum_{l \in L} w_l \times \text{Recall}_l$$

$$\text{Weighted F1-Score} = \sum_{l \in L} w_l \times \left(2 \times \frac{\text{Precision}_l \times \text{Recall}_l}{\text{Precision}_l + \text{Recall}_l} \right)$$

Where L is the set of all class labels, and w_l is the weight of class l, calculated as the proportion of true samples for that class in the dataset. This weighting scheme accounts for class imbalance.

The final reported result for each metric is the **mean and standard deviation** across all five folds of the cross-validation.

5.3 Experiment A: High-Dimensional Text Representations

5.3.1 Objective

The first experiment, Experiment A, was designed to test and compare four text representations techniques. The goal was to see which of these representations works best as a feature set for the SVM classifier.

5.3.2 Methodology

Four different methods were used to create feature vectors for each document:

- **TF-IDF**: This method calculates the importance of words based on their frequency. A `TfidfVectorizer` was used to create the feature vectors. The configuration was set as follows: `ngram_range=(1, 2)` to include both single words and two-word phrases, `max_features=10000` to limit the vocabulary to the most important words, `max_df=0.95` to ignore overly common words, and `min_df=5` to ignore very rare words.
- **Word2Vec (Averaged)**: The second method used the Word2Vec model detailed in Chapter 4.6.1. A single vector for each document was created by averaging the 300-dimensional vectors of all its constituent words.
- **Doc2Vec**: The third method used the 300-dimensional document vectors from the pre-trained Doc2Vec model, which was also described in Chapter 4.6.2. These vectors were used directly as features.
- **RoBERT** *[CLS]*: Finally, the fourth method utilized the pre-trained *RoBERT-base* model (see Chapter 4.6.3 for details). The 768-dimensional embedding of the special *[CLS]* token was extracted from the model's final layer to represent each document.

5.3.3 Results

The performance of the SVM classifier using each of the four feature representation methods was evaluated using 5-fold cross-validation. The mean and standard deviation for each performance metric are presented in Table 5.1.

Table 5.1: Performance Comparison of High-Dimensional Feature Representations (Experiment A)

Method	Mean Accuracy	Mean Precision	Mean Recall	Mean F1-Score
TF-IDF	0.9571 $\pm$0.0068	0.9578 $\pm$0.0061	0.9571 $\pm$0.0068	0.9570 $\pm$0.0067
Word2Vec (Averaged)	0.7702 $\pm$0.0106	0.7682 $\pm$0.0119	0.7702 $\pm$0.0106	0.7686 $\pm$0.0113
Doc2Vec	0.8087 $\pm$0.0109	0.8057 $\pm$0.0114	0.8087 $\pm$0.0109	0.8046 $\pm$0.0117
RoBERT (CLS)	0.7425 $\pm$0.0056	0.7371 $\pm$0.0063	0.7425 $\pm$0.0056	0.7376 $\pm$0.0072

The results clearly show that the TF-IDF method achieved the highest performance across all evaluation metrics, establishing a strong baseline for this classification task.

5.4 Experiment B: Hybrid Feature Model

5.4.1 Objective

The second and final experiment was designed to test a hybrid model. The objective was to investigate if combining the rich semantic representations from Doc2Vec with a set of custom-engineered features could achieve a higher performance than the single representation methods evaluated in Experiment A.

5.4.2 Methodology

This model used a feature set created by concatenating three distinct groups of features for each document:

- **Doc2Vec Embeddings**: The full 300-dimension vector from the pre-trained Doc2Vec model.
- **Engineered Features**: A set of 21 custom-engineered features was calculated **for each individual article** to capture various properties of the text. These included **lexical features** to measure the document's length and vocabulary diversity; **syntactic features** to analyze grammatical structure by calculating part-of-speech proportions; **named entity features** to measure the count and density of recognized entities; and a suite of **affective features**. This last group quantified the emotional and sentimental tone by calculating: the mean score for each of eight emotions, the total number of words found in the emotion lexicon, the total positive and negative sentiment scores, and the percentage of words with an exclusively positive or negative sentiment.
- **Cluster ID Feature**: An additional feature was created using the Doc2Vec embeddings. The k-Means clustering algorithm was used to group the documents into ”latent” topical or stylistic groups. The optimal number of clusters was found to be 8 using the Elbow Method, as shown in Figure 5.1. Each document was then assigned the ID of the cluster it belonged to.

This process resulted in a final feature vector with **322 dimensions** for each document.

To maximize the performance of the SVM classifier with this complex feature set, **GridSearchCV** was first used to find the optimal hyperparameters for **C**, **gamma**, and **kernel**. These best-found parameters were then used in the final 5-fold cross-validation evaluation to ensure a robust performance measure.

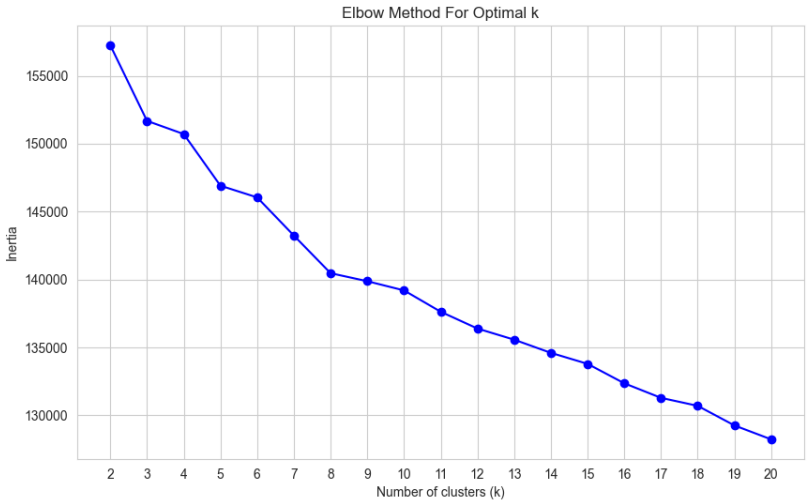


Figure 5.1: Elbow Method for determining the optimal k for K-Means clustering on Doc2Vec embeddings.

5.4.3 Results

The SVM classifier, configured with its optimal hyperparameters, was evaluated using the hybrid feature set. The final performance metrics from the 5-fold cross-validation are shown in Table 5.2.

Table 5.2: Performance of the Hybrid Feature Model (Experiment B)

Method	**Mean Accuracy**	**Mean Precision**	**Mean Recall**	**Mean F1-Score**
Hybrid Model	0.9174 $\pm$0.0139	0.9188 $\pm$0.0135	0.9174 $\pm$0.0139	0.9169 $\pm$0.0141

The hybrid model achieved a very strong Mean F1-Score of **0.9169**. While this result did not surpass the top-performing TF-IDF model from Experiment A, it represents a significant improvement over the pure Doc2Vec model and demonstrates the considerable value of combining different feature types.

5.5 Comparative Analysis and Discussion

This section synthesizes the results from the two main experimental approaches to identity the most effective strategies for Romanian fake news detection.

For a direct comparison, Table 5.3 shows the performance of the best model from Experiment A (TF-IDF), the hybrid model, and the embedding models.

These results reveal several important insights:

First, the strong performance of the **TF-IDF model** strongly suggests that this classification task relies heavily on **lexical cues**. The presence or absence of specific

Table 5.3: Final Performance Comparison of All Experimental Approaches

Experimental Method	Mean F1-Score
Experiment A: TF-IDF	0.9570 $\pm$0.0067
Experiment B: Hybrid Model	0.9169 $\pm$0.0141
Experiment A: Doc2Vec	0.8046 $\pm$0.0117
Experiment A: RoBERT (CLS)	0.7376 $\pm$0.0072

keywords, names, and phrases are very strong indicators for this classification problem. The TF-IDF method, which directly represents this information, proved to be the most effective single approach.

Second, the success of the **Hybrid Model** demonstrates the power of feature synergy. While the pure Doc2Vec model achieved an F1-score of 0.8046, combining its semantic vectors with the 22 engineered features improved the F1-score to 0.9169. This is direct, quantitative proof that the engineered linguistic and affective features provided valuable, complementary information that the semantic vectors alone did not capture.

Finally, the results highlight a key theme in applied machine learning: sometimes a simpler, feature-rich model like TF-IDF can outperform more complex ones (like RoBERT used for feature extraction) if the problem is more about specific signals than deep semantic understanding. It underscores that the full potential of advanced models like RoBERT is typically realized in a fine-tuning framework, which was outside the scope of this comparative experiment.

5.6 Conclusion

This chapter evaluated two main strategies for text representation in the context of Romanian fake news detection. The experiments revealed two strong but different approaches that are well-suited to this task.

The findings clearly show that the traditional **TF-IDF model**, which captures a broad range of specific keyword features, offers the most effective and reliable baseline, delivering the best overall performance.

However, the research also demonstrated that a **sophisticated hybrid model**, which combines semantic vectors from Doc2Vec with engineered linguistic and affective features, can achieve similarly strong results. This confirms that combining different types of information can be a powerful alternative to relying solely on lexical features.

The results provide a strong foundation for future work, where end-to-end fine-tuning of contextual models like RoBERT represents a promising direction to potentially surpass the current performance benchmarks.

Chapter 6

Application Development

This chapter presents the development process of the Fake News Detection web application. The process begins with defining the system's requirements, followed by the architectural design, and the implementation of the core components. These components include the backend logic for classification and the frontend user interface. Finally, the chapter covers the testing methods used to ensure the application works correctly and provides a user manual for guidance.

6.1 System Requirements

This section outlines the specific requirements that guided the development of the web application. These are divided into functional requirements, which describe the system's behaviors, non-functional requirements, which define its qualities, and user interface requirements, which detail the user's interaction with the system.

6.1.1 Functional Requirements

- **FR1: Text Input**: The user must be able to enter a block of text (a news article) into the application for analysis.
- **FR2: Text Classification**: The system must process the submitted text and classify it into one of five predefined categories: *fake news*, *misinformation*, *propaganda*, *real news*, or *satire*.
- **FR3: Conditional Result Display**: The system's output must change based on the classification model's confidence score:
  - If confidence is **above 80%**, the system displays a single, definitive category as the result.

  - If confidence is between **60% and 80%**, the system displays the most likely category along with a list of all possible categories and their respective probabilities.
  - If confidence is **below 60%**, the system displays a message stating that the result is inconclusive and requires manual verification, without showing a predicted category.

- **FR4: Word Embedding Analysis**: The system must provide a separate tool for users to perform a neighbor analysis on the Word2Vec models. This includes accepting a target word, allowing model selection (150D or 300D), and displaying a list of the most similar words and their scores.
- **FR5: Model Download**: The system must provide users with the ability to download the trained Word2Vec models (150D and 300D).

6.1.2 Non-Functional Requirements

- **NFR1: Performance**: The application should process the user's text and return a classification result in a timely manner, ideally under 5 seconds for a standard-length article.
- **NFR2: Usability**: The user interface must be intuitive, easy to use, and presented in English.
- **NFR3: Reliability**: The application must handle invalid inputs, such as empty text submissions, gracefully by displaying a clear error message without crashing.
- **NFR4: Maintainability**: The codebase must be well-structured to allow for easy updates and future expansion. This was achieved by separating the user interface logic from the core classification pipeline and by implementing established design patterns like the Facade and Singleton.

6.1.3 User Interface Requirements

- **UIR1: Classification Page**: The main page must feature a large, clear text area for article submission and a button to initiate the classification.
- **UIR2: Results Page**: A separate page must be used to display the classification results, with its layout adapting to the three confidence scenarios (high, medium, and low).

- **UIR3: Analysis Page**: A dedicated page must exist for the Word2Vec neighbor analysis tool, with its own input form.
- **UIR4: Navigation**: The application must provide clear and easy-to-use links for navigating between the main classification page and the Word2Vec analysis page.
- **UIR5: Visual Feedback**: The classification result should use distinct colors (e.g., red for ”FAKE”, green for ”REAL”) to provide immediate visual feedback to the user.

6.2 Architecture and Design

This section details the application's high-level architecture, the technologies used, and the underlying software design principles. It provides a blueprint of the system, covering the main components and how they interact.

6.2.1 System Overview and Technology Stack

System Overview

The web application is built with Django framework. The overall system flow is straightforward: a user accesses the web interface through their browser, where they can submit Romanian news text for classification or analyze word similarities. The request is sent to the Django backend server, which handles all the core logic. This includes preprocessing the text, feeding it into the trained machine learning pipeline (TF-IDF and SVM), and determining the classification result. Finally, the server renders an HTML page with the results and sends it back to the user's browser.

Technology Stack

The selection of technologies was based on their suitability for web development, data science, and natural language processing in Python.

- **Backend**: Python, Django, Scikit-learn, Gensim, Spacy.
- **Frontend**: HTML5, CSS3.
- **Data Manipulation**: Pandas, NumPy.

6.2.2 Use Cases and Use Case Diagram

The application allows general users to classify news articles through text input, analyze word similarity using Word2Vec models, and download pre-trained Word2Vec models.

The use case diagram in Figure 6.1 illustrates the interaction between the user and the main functionalities of the system.

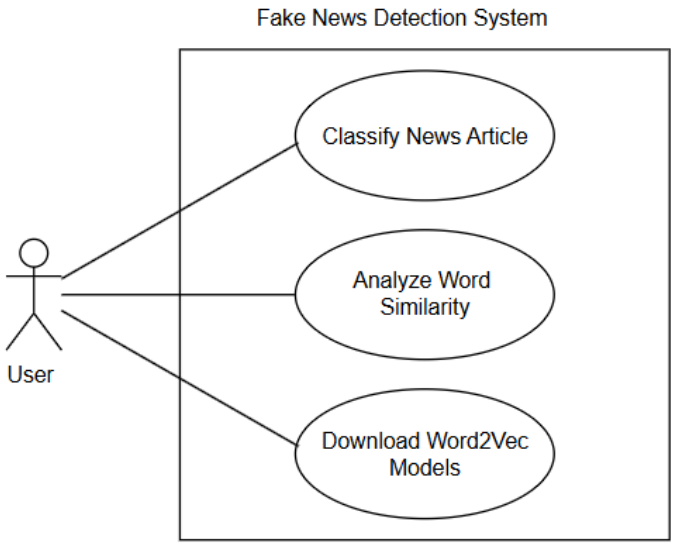


Figure 6.1: Use Case Diagram showing user interactions with the system.

6.2.3 Design Patterns

To ensure the application is maintainable, scalable, and well-structured, two key software design patterns were implemented.

- **Facade Pattern**: This pattern was used to create a single, simplified interface for the classification process. This facade coordinates the multiple steps required for a prediction—including text preprocessing, TF-IDF vectorization, and SVM model inference—behind a single method call from the user interface layer. This approach decouples the web logic from the underlying machine learning pipeline, making the system easier to maintain and modify in the future.

- **Singleton Pattern**: This pattern was used to manage the loading of resource-intensive assets. It ensures that the SpaCy model for preprocessing text, the TF-IDF vectorizer, the SVM classifier, and the Word2Vec models are loaded only once when the application starts. By guaranteeing there is only one instance of each of these components in the application, this pattern significantly

improves performance and reduces memory usage, as it avoids the costly process of reloading these models on every user request.

6.2.4 Class and Sequence Diagrams

To further illustrate the system's design, class and sequence diagrams are presented.

Class Diagram

The class diagram in Figure 6.2 shows the structure of the application's backend.

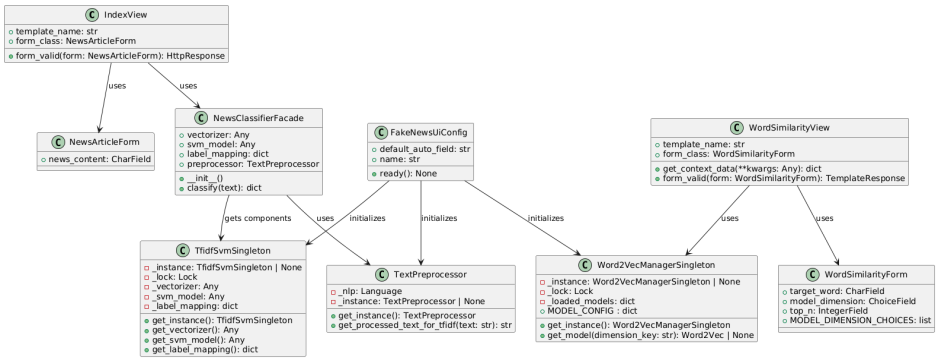


Figure 6.2: Class Diagram of the core application components.

Sequence Diagram

The sequence diagrams presented below show the sequential interactions for the classification of a news article and the neighbor analysis tasks within the application. The first, diagram (Figure 6.3) shows the classification flow of a news article. It describes the interaction between the main architectural layers: the User's Web Browser, the Web Application, and the Backend. In the second diagram (Figure 6.4), the sequence for the word similarity analysis is detailed. This flow begins with the user submitting a target word and selecting a model. The diagram then shows how the Web Application retrives the appropriate Word2Vec model and uses it to find the neighboring words. Furthermore, it shows the two possible outcomes: the successful return of a list of similar words, or the return of an error message if the target word is not found in the model's vocabulary.

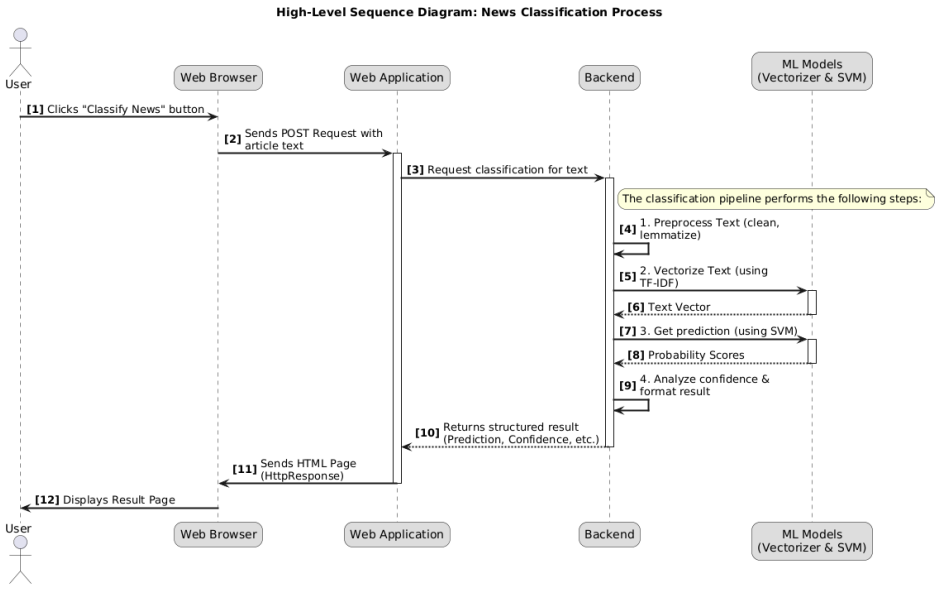


Figure 6.3: Sequence Diagram for the classification process.

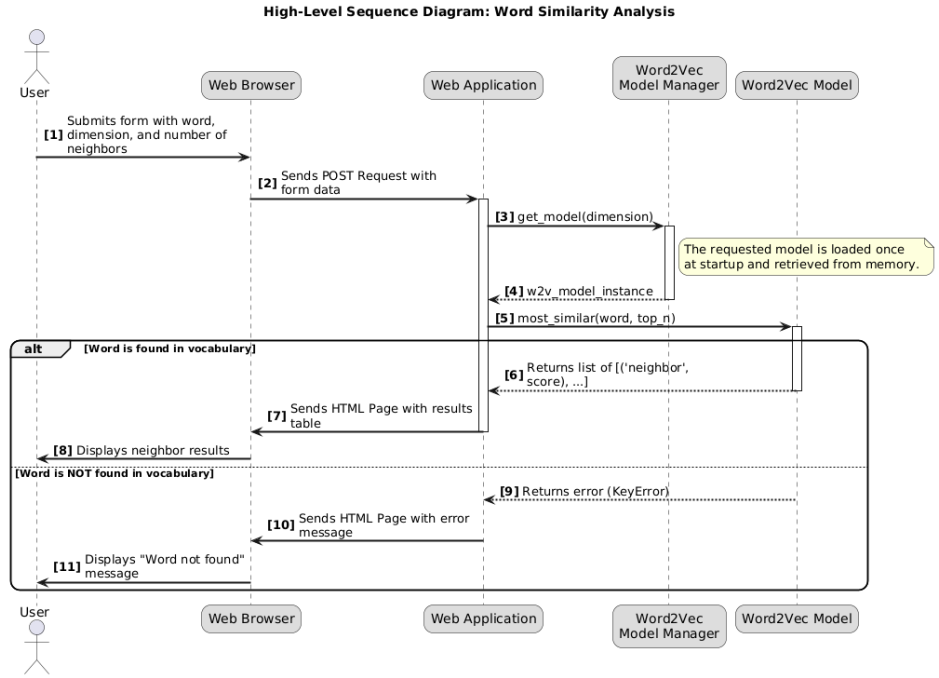


Figure 6.4: Sequence Diagram for the word similarity analysis.

6.3 Implementation

This section describes the practical implementation of the fake news detection application. It details the construction of the backend, which handles the classification logic, and the frontend, which provides the user interface.

6.3.1 Backend Implementation

The backend handles user requests and runs the machine learning predictions and word similarity analysis.

Models & Preprocessing

The models were created during the experimental phase in Jupyter Notebooks and are loaded into the application when it starts.

- **Model Loading**: The TF-IDF vectorizer and the trained SVM model are loaded from disk using Python's pickle library. Similarly, the Word2Vec models are loaded using the gensim library. This entire process is managed by the Singleton classes to ensure these large files are loaded into memory only once, which significantly improves the application's performance.
- **Text Preprocessing**: Before text can be classified, it must be processed in the exact same way as the data used to train the models. This critical step is handled by the TextPreprocessor class, which uses the `ro_core_news_lg` SpaCy model.

Classification Logic (Facade)

The `NewsClassifierFacade` class acts as the main controller for the classification pipeline, offering a simple interface to manage the process. It receives raw text from the web view, cleans and lemmatizes it using a text preprocessor, then passes it to the TF-IDF vectorizer that converts it into a numerical vector. This vector is fed into the SVM model, which calculates the probabilities for each news category and returns the results as a dictionary to the view.

Views

The application's views, are responsible for handling all web requests and connecting the user interface to the backend logic. The implementation uses Django's Class-Based Views, specifically the generic `FormView`, to provide a structured and maintainable approach. The primary `IndexView` class handles requests for the main

classification page. It automatically displays the submission form on a GET request and, upon a valid POST submission, passes the user's text to the classification pipeline before rendering the results template with the returned data. Similarly, a `WordSimilarityView` class manages the interface for the word embedding analysis tool.

6.3.2 Frontend Implementation

The frontend was built using Django's templating system with standard HTML and CSS.

Template & Forms

The user interface is composed of three primary HTML templates, with the application's forms to handle data structure and validation. The main page, presents the user with a large text area for article submission, which is rendered using the `NewsArticleForm`. After classification, the result page displays the outcome, using conditional logic to adapt its view based on the model's confidence: it shows a single definitive result for high confidence, a list of all potential categories for medium confidence, or a manual verification message for low confidence. Additionally, the word similarity analysis page, provides the Word2Vec analysis tool, which includes the `WordSimilarityForm` for word submission, a table to display the similarity results, and a section with details for downloading the Word2Vec models.

6.4 Testing

A set of unit tests was built to ensure the application works correctly. Each main component was tested in isolation, making it easy to detect any issues and verify that every part behaves as expected before integration.

The tests were implemented using Python's built-in `unittest` framework, along with Django's `TestCase` and `Test Client` classes for testing web-related components. A key technique used was **mocking**, which replaces external dependencies–such as machine learning models–with simulated objects. This keeps the tests fast, reliable, and focused on the logic of the component being tested.

The testing process was divided into three main areas: forms, backend logic (Facade), and views.

Testing the Forms

The application's Django forms, `NewsArticleForm` and `WordSimilarityForm`, were tested to ensure data validation works correctly. Test cases verified that the forms are correctly identified as valid when provided with appropriate data, and as invalid when required fields are submitted empty. For invalid submissions, the tests also confirmed that the correct error messages are generated, ensuring the application properly rejects incorrect user input.

Testing the Classification Logic

The core classification logic within the `NewsClassifierFacade` class was tested using mocked versions of its dependencies to isolate its behaviour. The tests verified that the facade correctly implements the confidence-based display rules by checking its output in three key scenarios: returning a single prediction for high confidence (>80%), providing a list of all probabilities for medium confidence (60-80%), and issuing a manual verification status for low confidence (<60%). An additional test confirmed that the facade handles missing or unloaded models, by returning a clear error message instead of crashing.

Testing the Views

The application's views were tested using Django's `Test Client` to simulate user interactions and ensure they correctly handled web requests. Tests for GET requests confirmed that visiting the application's URLs successfully renders the appropriate pages with a 200 OK status code. For POST requests, which simulate form submissions, the facade dependency was mocked. This allowed for focused testing to verify that the view logic correctly validates user input, calls the appropriate backend service with the submitted data, and renders the correct template with the context returned from the service, correctly handling all three confidence scenarios for display.

This suite of unit tests provides confidence that each component of the application functions as intended, leading to a more robust and reliable final product.

6.5 User Manual

This user manual provides instructions for operating the Fake News Detection web application. It covers the two main features of the system: classifying a news article and analyzing word similarities with Word2Vec.

How to Classify a News Article This is the primary function of the application. The process involves three simple steps.

Step 1: Navigate to the Classification Page Open a web browser and go to the application's main page (Figure 6.5). You will be presented with the main interface for news submission.

Figure 6.5: The main user interface for article submission.

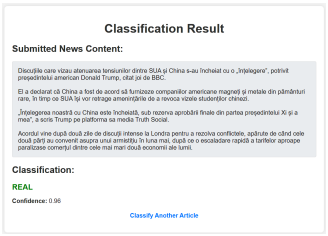
Step 2: Enter the News Article Text Copy the full text of the Romanian news article you wish to analyze and paste it into the large text area labeled "News Article".

Step 3: Receive and Interpret the Result Click the "Classify News" button. The system will process the text and take you to a results page. The result will be displayed in one of three ways, depending on the model's confidence in its prediction:

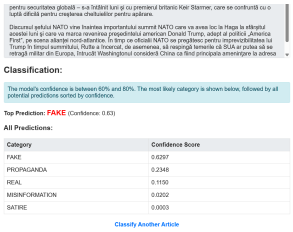
- **Scenario 1: High Confidence (Above 80%)**
  If the model is very confident, a single, clear classification will be displayed (e.g., "REAL" in green or "FAKE" in red) along with the confidence score (Figure 6.6a).

- **Scenario 2: Medium Confidence (Between 60% and 80%)**
  If the model is less certain, the page will show the most likely prediction but will also display a table of all possible categories and their corresponding probabilities. This indicates to the user that there is some ambiguity in the result (Figure 6.6b).

- **Scenario 3: Low Confidence (Below 60%)**
  If the model's confidence is too low to make a reliable prediction, no category will be shown. Instead, a message will appear advising that the article requires **manual verification** (Figure 6.6c).



(a) High-confidence result.



(b) Medium-confidence result.

(c) Low-confidence result.

Figure 6.6: Classification results user interface.

How to Analyze Word Similarity This feature allows for the exploration of the Word2Vec embedding models.

Step 1: Navigate to the Analysis Page From the main classification page, click the **"Word2Vec Neighbor Analysis"** link in the top-right corner.

Step 2: Enter Your Query On the analysis page (Figure 6.7), fill out the form:

- **Target Word (Lemma)**: Enter the single Romanian word you want to analyze.
- **Select Word2Vec Model**: Choose between the `300D` and `150D` dimensions.
- **Number of Neighbors**: Specify how many similar words you want to see.

Step 3: View Results Click the "Find Neighbors" button. The results will appear in a table showing the neighboring words and their similarity scores (Figure 6.8a). A higher score indicates a closer semantic relationship in the model's vector space.

How to Download Models On the Word2Vec Neighbor Analysis page, there is a "Download Word2Vec Models" section (Figure 6.8b). To download a model for your own use, simply click the corresponding "Download Model" button.

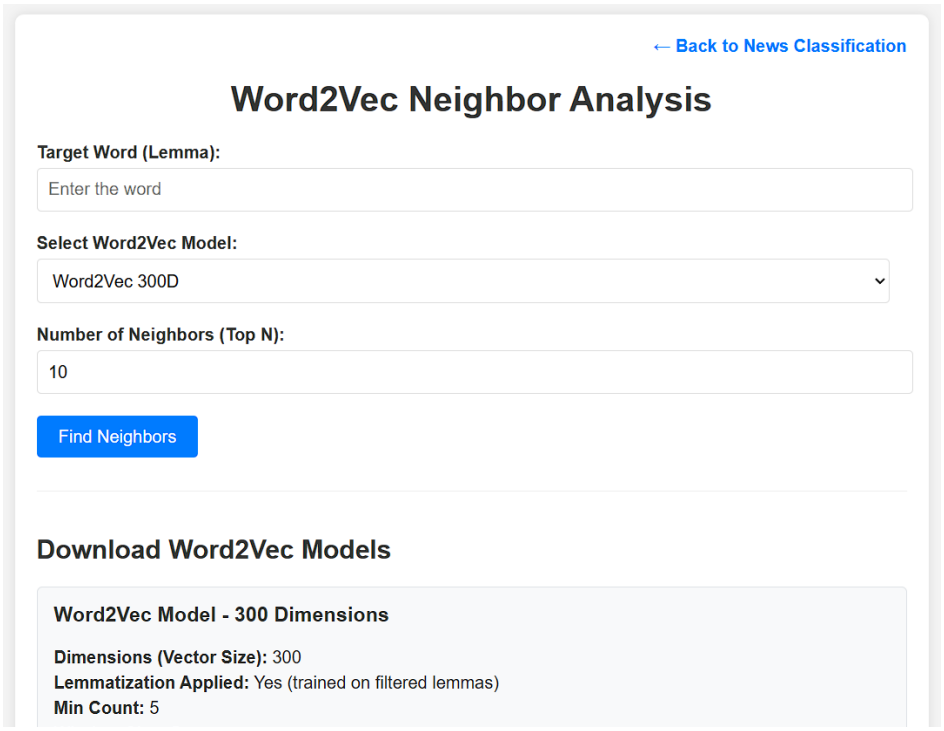
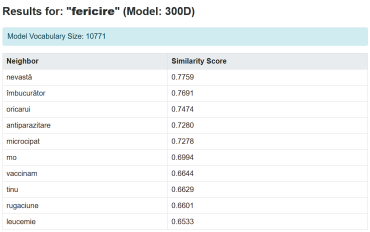
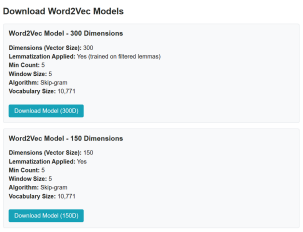


Figure 6.7: The user interface for neighbor analysis.



(a) Example of Word2Vec neighbors result.



(b) Download Word2Vec models section.

Figure 6.8: Neighbor analysis user interface.

Chapter 7

Conclusions

This thesis explored different methods for detecting fake news in the Romanian language, covering everything from data analysis to building a working application. The research began with a detailed analysis of a news dataset, which found clear, measurable differences between real news and disinformation. The analysis showed that deceptive articles are often longer, use more descriptive language, and rely on a specific emotional strategy that mixes a feeling of trust with negative emotions like fear and anger.

The classification experiments tested several ways of representing text. The most important finding was that a traditional **TF-IDF model performed the best**, with a mean F1-Score of 95.70%. This result shows that for this dataset, the specific words used in an article are the strongest signal for distinguishing between the news categories. The research also showed that combining different types of information is a powerful strategy. A **hybrid model**, which used both Doc2Vec static document vectors and a custom set of engineered features, also achieved a very strong mean F1-Score of 91.69%. The practical result of this work is a web application that uses these findings to provide a useful tool for classifying news articles.

This work faced several challenges common for the Romanian language, such as the lack of large datasets and publicly available, lemmatized Word2Vec models. These limitations highlight important directions for future research, including creating a more diverse dataset, fine-tuning a large language model like RoBERT to achieve better results, and deploying the current application to a cloud service for public access.

In summary, this thesis provides a solid foundation for fake news detection in Romanian. By continuing to build better datasets and explore more advanced models, the field can continue to address the important challenge of disinformation.

Bibliography

[Aas] Matei Aass. Fakerom: Romanian fake news dataset. `https://huggingface.co/datasets/mateiaass/FakeRom`. Accessed on June 7, 2025.

[BD23] Marian Bucos and Bogdan Drăgulescu. Enhancing fake news detection in romanian using transformer-based back translation augmentation. *Applied Sciences*, 13(24):13207, 2023.

[BTMR23] Marius Cristian Buzea, Stefan Trausan-Matu, and Traian Rebedea. Automatic fake news detection for romanian online news. *Information*, 13(3):151, 2023.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[Exp] Explosion. spacy romanian language model (ro_core_news_lg). `https://spacy.io/models/ro#ro_core_news_lg`. Accessed on June 7, 2025.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, 2014.

[LB19] Mihaiela Lupea and Anamaria Briciu. Studying emotions in romanian words using formal concept analysis. *Computer Speech Language*, 57:128–145, 2019.

[LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning, ICML 2014*, pages 1188–1196, 2014.

[LSK+24] Jula Lühring, Apeksha Shetty, Corinna Koschmieder, David Garcia, Annie Waldherr, and Hannah Metzler. Emotions in misinformation studies: Distinguishing affective state from emotional response and misinformation recognition from acceptance. *Cognitive Research: Principles and Implications*, 9(12), 2024.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press, 1967.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations, ICLR 2013, Workshop Track*, 2013.

[Mih] Bogdan Cornel Mihalca. Fakero_updated: Romanian fake news detection dataset. `https://huggingface.co/datasets/mihalca/FakeRO_updated`. Accessed on June 8, 2025.

[MMC+24] Elisa Valentina Moisi, Bogdan Cornel Mihalca, Simina Maria Coman, Alexandrina Mirela Pater, and Daniela Elena Popescu. Romanian fake news detection using machine learning and transformer-based approaches. *Applied Sciences*, 14(24):11825, 2024.

[MPR20] Cameron Martel, Gordon Pennycook, and David G. Rand. Reliance on emotion promotes belief in fake news. *Cognitive Research: Principles and Implications*, 5(47), 2020.

[MRD20] Mihai Masala, Stefan Ruseti, and Mihai Dascalu. Robert – a romanian bert model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6626–6637. International Committee on Computational Linguistics, 2020.

[RS10] Radim Rehůřek and Petr Sojka. Gensim – python framework for vector space modelling. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010) Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 2010. ELRA, European Language Resources Association.

[VD18] Păiș Vasile and Tufiș Dan. Computing distributed representations of words using the corola corpus. *Proceedings of the Romanian Academy, Series A*, 19(1):25–32, 2018.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[Ver] Veridica. Veridica.ro – romanian fact-checking news platform. `https://www.veridica.ro/`. Accessed on June 6, 2025.