The object of this study is the processes of identifying sources and networks of disinformation dissemination in the cyberspace of the world. With the growing influence of social networks on public opinion, the issue of identifying and neutralizing propaganda messages is becoming particularly relevant. Conventional methods of combating propaganda such as manual content moderation have proven to be insufficiently effective due to the large amount of information generated daily.

It is important to use natural language processing and machine learning methods to analyze text, identify sources of disinformation dissemination and inauthentic behavior of bots. Based on the analysis of existing methods of intelligent disinformation search, methods have been devised to identify sources and ways of disinformation dissemination in cyberspace by searching for similar text chains and analyzing the similarity of writing style.

Hybrid vector representation makes it possible to capture surface frequency characteristics of the text and semantic features, which has a positive effect on the quality of classification. Cosine similarity, Jacquard, Levenstein and Word2Vec are used to measure similarity. Clustering (DBSCAN, K-Means) helps group fake messages. Graph analysis detects central accounts and bot networks.

Evaluation of the model's performance by key metrics showed reliable results for identifying sources of disinformation distribution: accuracy – 0.82, F1.3 – 0.8, ROC-AUC – 0.86. The identified differences in lexical patterns for the "fake" and "true" classes confirm the model's ability to capture the content features of texts. The proposed method for detecting disinformation distribution paths serves as the basis for building scalable systems for monitoring the information space and adapting to other text classification tasks

Keywords: disinformation source detection, machine learning, disinformation network, fake news, text similarity

UDC 004.89

DOI: 10.15587/1729-4061.2025.335501

DEVELOPMENT OF AN INFORMATION TECHNOLOGY FOR DETECTING THE SOURCES AND NETWORKS OF DISINFORMATION DISSEMINATION IN CYBERSPACE BASED ON MACHINE LEARNING METHODS

Victoria Vysotska
Doctor of Technical Sciences, Professor*
Mariia Nazarkevych
Corresponding author
Doctor of Technical Sciences, Professor*
E-mail: mariia.a.nazarkevych@lpnu.ua
*Department of Information Systems and Networks
Lviv Polytechnic National University
S. Bandery str., 12, Lviv, Ukraine, 79013

Received 04.03.2025
Received in revised form 01.05.2025
Accepted 16.05.2025
Published 29.08.2025

How to Cite: *Vysotska, V., Nazarkevych, M. (2025). Development of an information technology for detecting the sources and networks of disinformation dissemination in cyberspace based on machine learning methods. Eastern-European Journal of Enterprise Technologies, 4 (2 (136)), 35–51. https://doi.org/10.15587/1729-4061.2025.335501*

1. Introduction

In today's world, disinformation has become one of the key threats to society, as information from unreliable sources spreads rapidly via the internet and social media [1]. False or distorted information can have serious consequences for public opinion, politics, the economy, and security [2]. A variety of actors, from private individuals to government agencies, can deliberately spread disinformation with the aim of manipulation or creating chaos [3]. One of the most difficult aspects of countering disinformation is its detection and separation from reliable facts [4]. In an environment where the volume of information is constantly growing, automating the processes of verifying the veracity of texts is an urgent need [5]. With the increasing influence of social networks on public opinion, the issue of detecting and neutralizing propaganda messages becomes particularly relevant [6]. Propaganda influences political decisions, causes social tension, and spreads disinformation. Conventional methods of combating propaganda, such as manual moderation of content, are not effective enough because of the large volume of information generated every day.

Therefore, research on identifying sources and networks of disinformation dissemination in the global cyberspace is relevant and promising in the modern world of information wars among different segments of the population. This is especially relevant when manipulation of facts and fake news create new realities of the living space for the average citizen of any country. In addition, the spread of disinformation significantly affects the economy, social effects and challenges, politics, public sentiment, and public opinion.

2. Literature review and problem statement

Many studies investigate methods for detecting disinformation using content analysis and user behavioral characteristics [1–6]. One reference is "many studies". Combining different approaches, such as natural language processing and social network analysis, has been shown to increase the effectiveness of fake news detection [7]. However, issues related to adapting models to new disinformation tactics and ensuring scalability of solutions remain unresolved [8]. The likely reason is objective

difficulties associated with the rapid evolution of disinformation methods and the limited available data for training models [9]. An option to overcome these difficulties may be the development of adaptive machine learning algorithms that can respond quickly to new threats [10]. This approach has been used in some modern studies, but its implementation requires significant resources and interdisciplinary collaboration. All this gives grounds to argue that it is advisable to conduct further research on the design of effective and scalable fake news detection systems that can adapt to the rapidly changing information environment.

Study [11] presents an overview of methods for detecting fake news in social networks, including analysis of content, social context, and distribution models. It is shown that combined approaches that take into account both the content of messages and the behavioral characteristics of users can increase the accuracy of disinformation detection. However, issues related to the processing of multimedia content and taking into account cultural characteristics in different regions remain unresolved. The likely reason is associated with the variety of data formats and the lack of universal models for different cultural contexts. An option to overcome these difficulties may be the development of adaptive models that take into account the specificity of a particular region or platform. This is the approach used in [12], in which the authors built a model adapted to detect disinformation in specific cultural contexts. However, the model requires significant amounts of localized data for training, which can be difficult to implement. All this gives grounds to argue that it is advisable to conduct research on the development of universal methods for detecting disinformation that can adapt to different cultural and linguistic contexts with minimal additional training costs.

In [13], the spread of true and false information on Twitter was investigated. It was proven that fake news spreads faster, deeper, and wider than true news. However, the issues related to the mechanisms that contribute to such spread of disinformation remained unresolved. The reason for this is the objective difficulties associated with the complexity of human behavior and psychological factors that affect the perception and spread of news. An option to overcome the difficulties may be the integration of psychological models into the analysis of information spread. This approach was used in [14], which investigated the role of cognitive biases in the perception and spread of fake news. However, the integration of psychological aspects into machine learning models remains a difficult task. This gives grounds to argue that it is advisable to conduct research on the development of interdisciplinary approaches that combine data analysis and psychological theories for a deeper understanding of the mechanisms of disinformation spread.

In study [15], methods for detecting disinformation based on the analysis of social network graphs were proposed. It was shown that taking into account the structural features of the network can improve the efficiency of detecting fake news. However, issues related to the processing of large amounts of data and the dynamism of networks remained unresolved. The likely reason is associated with the scalability of social networks and rapid changes in their structure. An option for overcoming the difficulties may be the use of distributed computing systems and algorithms capable of processing streaming data. This is the approach used in the work [16], in which a scalable system for detecting disinformation in real time was designed. However, the implementation of such systems can be costly in terms of resources and infrastructure. All this gives grounds to argue that it is advisable to conduct a study on the optimization of disinformation detection algorithms for their effective application in large and dynamic networks.

In [17], an approach to detecting disinformation based on fact-checking using natural language processing methods is reported. It is shown that automated systems can effectively identify false statements. However, issues related to the limitations of knowledge bases and the complexity of processing unstructured information remain unresolved. This may be due to objective difficulties associated with the ambiguity of natural language and the lack of complete and up-to-date knowledge bases. An option to overcome these difficulties may be the integration of multiple data sources and the use of deep learning methods to process unstructured content. This is the approach used in [18], which combines several models to improve the accuracy of fact-checking. However, such systems can be difficult to implement and require significant computational resources.

Papers [11–18] report the results of research on the use of artificial intelligence (AI) to detect disinformation. It is shown that existing tools make it possible to use AI to distinguish between organic and coordinated content distribution, detect automated spam distribution systems, assess the impact on the audience of different social media user accounts, distinguish bots from real users, etc. However, issues related to the adaptation of AI to new disinformation tactics and ensuring its effectiveness in a constantly changing information environment remain unresolved. The likely reason is associated with the rapid evolution of disinformation methods and the limited available data for training models. An option to overcome the difficulties may be the development of adaptive machine learning algorithms capable of quickly responding to new threats. This is the approach used in some modern studies, but its implementation requires significant resources and interdisciplinary cooperation. All this gives reason to argue that it is advisable to conduct further research on the development of effective and scalable systems for detecting sources of disinformation that are able to adapt to the rapidly changing information environment.

Work [19] reports the results of research aimed at applying artificial intelligence to design and improve cyber-warfare tools. In particular, the research focuses on combating disinformation, fakes, and propaganda in the Internet space, as well as identifying sources of disinformation and inauthentic behavior (bots) of coordinated groups. It is shown that the use of natural language processing (NLP) and machine learning (ML) methods can be effective in detecting and countering disinformation. However, issues related to contextual ambiguity and the development of linguistic nuances remain unresolved. The likely reason is associated with the constant evolution of language and the adaptation of disinformation to new contexts, which complicates the recognition of such messages. An option to overcome the difficulties may be the integration of multimodal analysis, which combines text and visual elements for a more holistic understanding of the content. This approach was used in [20], but it requires significant computational resources and a complex infrastructure for processing heterogeneous data. All this gives grounds to argue that it is advisable to conduct research on the development of effective and resource-saving methods of integrating multimodal analysis to detect and counteract disinformation in the Internet space.

In [21], various approaches to NLP and machine learning for detecting disinformation in social networks are considered. In particular, linear regression, the *k*-nearest neighbors method (KNN), the support vector method (SVM), long short-term memory (LSTM), artificial neural networks, and many others are considered. It is shown that these methods

are effective for detecting disinformation, but issues related to scalability and real-time analysis remain unresolved. The likely reason is associated with the large volume of data and the speed of their updating in social networks, which makes relevant research impractical without the appropriate infrastructure. An option to overcome these difficulties may be the use of distributed computing systems and optimization of algorithms for real-time operation. This is the approach used in [22], but it requires significant resources and complexity of implementation. All this gives grounds to argue that it is advisable to conduct research on the development of more effective and scalable methods for detecting disinformation in social networks that can operate in real time with limited resources.

Paper [23] analyzes various types of false information spread in modern information systems and characterizes the dangers that the spread of inaccurate information in society entails. It is shown that the spread of disinformation can have serious consequences for society, but issues related to the effectiveness of existing methods for detecting and countering disinformation remain unresolved. The likely reason is related to the constant evolution of methods for spreading disinformation and adaptation to new technologies, which makes relevant research impractical without constant updating and adaptation of methods [24]. An option for overcoming these difficulties may be the development of adaptive disinformation detection systems that can learn and change along with the evolution of methods for spreading disinformation. This is the approach used in [25], but it requires constant monitoring and updating, which can be resource-intensive. All this gives reason to argue that it is advisable to conduct research on the design of automated and self-learning disinformation detection systems that can adapt to new methods of disinformation dissemination with minimal human intervention.

3. The aim and objectives of the study

The aim of our work is to devise an information technology for detecting sources and networks of disinformation dissemination in cyberspace based on NLP and Machine Learning, taking into account the similarity of the content and style of writing text content. This will make it possible to increase the security levels of cyberspace of societies, communities, countries, international platforms, and other subjects of the information space in real time.

To achieve this aim, the following objectives were accomplished:

– to define general functional requirements for the typical architecture of the subsystem for detecting a set of disinformation in the Internet space as the main part of scalable information space monitoring systems;

– to devise a method for detecting sources and networks of disinformation dissemination in cyberspace based on the automation of the search for similar text chains;

– to devise a method for detecting ways of disinformation dissemination in cyberspace based on the definition of stylistically similar content;

– to validate the proposed methods for detecting sources and paths of disinformation based on the developed software.

4. The study materials and methods

The object of our study is the processes for identifying sources and networks of disinformation distribution in the global cyberspace.

The principal hypothesis of the study assumes that the use of an improved method to search for similar text chains to identify disinformation similar in content and/or writing style could increase the accuracy of identifying sources of disinformation and its distribution network. The accuracy result can also be influenced by professional/expert filling of the dataset for training the model, the choice of the training model, and the design/implementation of effective and scalable systems for detecting sources of disinformation that are able to adapt to the rapidly changing information environment.

The basic assumption in the process of conducting the study is the fact that fake news is written according to thematic narratives using the appropriate set of keywords according to a certain template by a group of people/bots with the appropriate style. Another assumption is that the spread of fake news requires multiple accounts that repost all news from relevant sources at certain intervals or immediately after the news is published in the relevant source.

The main simplification is that bots have a peculiar behavior in social networks, in relation to people, in particular the periodicity and regularity of activity (publication or repost, people are more chaotic in any activity), and they also have a template writing style compared to people (short sentences, uniformity of sentence structure and punctuation, limited word usage, lack of synomization and sarcasm, non-emotional coloring of the text or, conversely, uniform emotional coloring of the texts, for example, only aggressive, etc.). All this significantly simplifies the process of identifying sources/ways of spreading fakes. Therefore, it is important to use NLP and machine learning methods for automated analysis of text data. Before identifying sources and networks of spreading disinformation in cyberspace, it is necessary to first train the model to detect sets of fakes, then form subsets of text content similar in content and/or writing style and record the dates/place of publication. This will subsequently make it possible to build a graph of the distribution of similar content over time and encourage to identify the original sources/authors of the generated fake content.

A typical algorithm for detecting disinformation sets in social networks performs automatic detection of propaganda messages on Twitter. The attention is paid to methods of data collection and preparation, text pre-processing, vectorization, model training, and evaluation of its effectiveness. The basic classical and typical processes of the disinformation set detection subsystem for scalable information space monitoring systems are as follows: Loading and preparing data from the dataset (module 1) → Research of unique characters (module 2) → Search for substrings (module 3) → Pre-processing of text (module 4) → Text vectorization (module 5) → Model training (module 6) → Model effectiveness evaluation (module 7).

Module 1 "Loading and Preparing Data from a Dataset" performs several key operations: it removes unnecessary columns, estimates class balance, detects missing values, analyzes text length and distribution, and normalizes text data. This ensures high-quality data preparation before further processing and training the model. If this stage is performed incorrectly, even the most sophisticated models can demonstrate poor results due to "dirty" or unevenly distributed input data. Unique characters play an important role in text analysis, as they can indicate text features, such as the use of special characters, emojis, punctuation, or different alphabets. Exploring these characters helps better prepare the data for further processing and training the model. Therefore, at the stage of exploring unique characters (module 2), the unique characters used in the texts are first identified. Next, we analyze the

frequency distribution of characters in tweets, examine the use of punctuation, emojis, and different alphabets, and identify possible patterns that may be characteristic of propaganda messages. This analysis helps solve the following:

– Should rare characters be removed?

– What role does punctuation play in manipulative texts?

– Can emojis be markers of propaganda messages?

– Is mixing alphabets a suspicious factor?

The results obtained are used for further text processing and building an effective machine learning model.

Punctuation marks play an important role in text analytics. Frequent use of exclamation marks (!), questions (?), quotation marks (", ') or periods (.) may indicate a certain writing style, emotionality, or manipulative nature of the text. Emojis can be markers of the emotional coloring of the text. It is important to assess their frequency in messages, as they can be significant for classification. Propaganda tweets may contain mixed alphabets (Latin, Cyrillic, Arabic script, etc.).

In text analysis of fake news and disinformation, it is often necessary to find certain substrings in texts to identify specific patterns, words, or symbols. For example, in the task of propaganda detection, keywords, manipulative phrases or specific symbols are searched for.

The substring search module (3) makes it possible:

– to determine the presence of certain words or phrases in texts;

– to calculate the number of occurrences of substrings in texts;

– to build statistics on the use of certain terms.

Substring search methods can be used for:

– detection of keywords (e.g., "fake", "conspiracy", "betrayal", etc.).

– search for specific expressions from propaganda narratives;

– analysis of the emotional coloring of the text (e.g., search for the words "hate", "support", "traitor");

– detection of anomalous use of symbols (e.g., emojis or special marks).

Example of use in the context of propaganda analysis: *keywords* = [«*фейк*», «*маніпуляція*», «*зрада*», «*агент*», «*брехня*»], *df*["*contains_keywords*"] = *df*["*text*"].*apply*(*lambda x*: *any*(*kmp_search*(*x.lower*(), kw) *for kw in keywords*)). The pseudocode adds a column that shows whether a tweet contains at least one of the given keywords. Substring search methods help quickly find keywords or phrases in texts. Naive substring search is simple but slow ($O(nm)$). The Knuth-Morris-Pratt (KMP) algorithm is efficient ($O(n)$) and makes it possible to quickly find matches without unnecessary checks. Substring search is used to detect manipulative phrases, specific vocabulary, and other signs of propaganda. Text preprocessing is a critically important step in NLP tasks, as text data often contains noise, unnecessary characters, different word forms, etc. This step makes it possible to normalize the text and improve the performance of machine learning models. In the work, text data undergo the following preprocessing steps:

– removing special characters (removing URLs, emojis, punctuation);

– tokenizing text (dividing text into individual words);

– changing case (converting all words to lowercase);

– removing stop words (extra words that do not carry significant meaning);

– lemmatizing (reducing words to their basic form).

These steps reduce the dimensionality of the text space and allow for a high-quality representation of the texts before vectorization.

Text vectorization is the process of converting text data into numerical vectors for further use in machine learning models. Since algorithms work with numbers, text data must be converted into a format that can be used for calculations. There are several approaches to text vectorization: One-Hot Encoding (OHE), Bag-of-Words (BoW), TF-IDF (Term Frequency – Inverse Document Frequency), Word Embeddings (Word2Vec, GloVe, FastText). After vectorizing text data into numerical format, a machine learning model can be trained on these features to automatically detect propaganda messages. SVM often gives the best results because it effectively separates the data. Logistic regression works well with a large number of features. Decision trees are well explained but can be overtrained. The best model is chosen based on the F1-measure. The F1-measure is a key metric, as it is important not to miss propaganda, but also not to generate many false alarms. ROC-AUC shows how well the model is able to distinguish between classes. Cross-validation helps assess generalization ability. Error analysis makes it possible to find weaknesses in the model (for example, the model may confuse sarcasm with propaganda).

5. Results of research into the detection of sources and networks of disinformation dissemination in cyberspace based on machine learning methods

5. 1. General functional requirements for a typical architecture of a subsystem for detecting a set of disinformation in the Internet space for scalable information space monitoring systems

Main processes of module 1 "Data loading and preparation": Data loading → Removing unnecessary columns → Analysis of class distribution → Detection of missing values → Analysis of tweet length → Estimation of statistical characteristics of text length → Visualization of length distribution → Pre-normalization of texts. The function (program) "Data loading" works with text data from Twitter, stored in CSV format (comma-separated values). This format is convenient for processing using the pandas library. The main object of work is a data frame, which contains columns with tweets and their labels (whether they are propaganda). To load data, one uses function *df* = *pd.read_csv*(*data.csv*), where *df* is a data frame with the loaded data; *data.csv* is the path to the data file.

Before processing the texts, the "Remove Unnecessary Columns" process analyzes the structure of the dataset. Some columns, such as *Unnamed:* 0, *id*, may not contain useful information for propaganda analysis, so they are removed: *df.drop*(["*Unnamed*: 0","*id*"], *axis* = 1, *inplace* = *True*), where *axis* = 1 indicates that columns (not rows) are removed; *inplace* = *True* means that the changes are applied directly to *df*. Before training the model, it is important to assess the class balance in the data based on the "Class Distribution Analysis" process. Propaganda and non-propaganda tweets should be evenly represented to avoid class imbalance, which can affect the accuracy of the model. The proportion of each class is determined by $P(y_i) = |y_i|/N$, where $P(y_i)$ is the probability of class i in the sample; $|y_i|$ is the number of examples of class i; N is the total number of examples in the dataset. If $P(y_1) \neq P(y_2)$ and one class is significantly dominant, data balancing should be applied, for example, through oversampling (increasing the number of less represented examples) or undersampling (decreasing the number of more represented examples).

If there are missing values (*NaN*) in the text column, they should be removed or filled in through the "Missing Value Detection" process *df.dropna*(*subset* = ["*text*"], *inplace* = *True*). This ensures that the model will not work with empty texts that do not contain useful information.

Texts can have different lengths, which affects the quality of the model. To estimate the length distribution through the "Tweet Length Analysis" process, a new column is added: $df[text_{length}] = df[text].apply(len)$. Mathematically, the average text length (arithmetic mean) is defined as

$$\mu = \frac{1}{N}\sum_{i=1}^{N} l_i, \tag{1}$$

where μ is the average tweet length; N is the number of tweets; l_i is the length of the i-th tweet. If a significant part of the data contains very short messages (for example, less than 3–5 words), such tweets are removed or specially processed because they do not have enough context for the model. The following characteristics are also evaluated in the process "Estimation of statistical characteristics of text length":

– median (the value located in the center of the sorted data)

$$\tilde{x} = \begin{cases} x_{\frac{N+1}{2}}, N\%2 = 1, \\ \dfrac{x_{\frac{N}{2}} + x_{\frac{N}{2}+1}}{2}, N\%2 = 0, \end{cases} \tag{2}$$

– standard deviation, showing the variability of tweet lengths

$$\sigma = \sqrt{\frac{1}{N}\cdot\sum_{i=1}^{N}\left(l_i - \mu\right)^2}, \tag{3}$$

where σ is the standard deviation; l_i is the length of the i-th tweet; μ is the mean value. The higher σ, the more scattered the text lengths are. To assess the nature of the length distribution, the process "Visualization of length distribution" uses a histogram. If the histogram shows a long tail on the right, i.e., there are tweets with a very long length, they can be truncated to a certain maximum threshold, for example, 280 characters, which corresponds to the Twitter limit.

At the final stage "Pre-normalization of texts", the texts are normalized to simplify further processing:

– converting the text to lowercase (to avoid case-sensitive words): *df*["*text*"] = *df*["*text*"].*str.lower*();

– removing extra spaces (removing double spaces, spaces at the beginning and at the END): *df*["*text*"] = *df*["*text*"].*str.strip*().

These steps help unify text data by reducing the number of variations of the same word (for example, "Propaganda", "propaganda", and " P R O P A G A N D A " result in "propaganda").

The main processes of module 2 "Research of unique characters": Identification of unique characters → Frequency of characters → Analysis of punctuation → Analysis of emoji usage → Analysis of usage of different alphabets → Visualization of character distribution.

Let D be the set of all text data, where each tweet is represented as a sequence of characters $D = \{T_1, T_2, ..., T_N\}$, where T_i is an individual tweet, and N is the total number of tweets. The set of characters in the dataset is defined as

$$S = \bigcup_{i=1}^{N} C_i, \tag{4}$$

where C_i is the set of all characters contained in the tweet T_i. This makes it possible to detect all characters in the process of "Determining unique characters" that occur in texts, including Latin, Cyrillic, special characters, emojis, and other characters. To understand which characters are most often found in texts, we calculate the frequency of occurrence of each character in the process of "Character occurrence frequency"

$$f(s) = \frac{\sum_{i=1}^{N} count(s, T_i)}{M} \text{ and } M = \sum_{i=1}^{N} |T_i|. \tag{5}$$

where $f(s)$ is the relative frequency of the symbol s, $count(s,T_i)$ is the number of occurrences of symbol s in the tweet T_i, M is the total number of all symbols in the dataset. This approach makes it possible to identify the dominant symbols that are most often used in texts. To study the use of punctuation, the total proportion in texts was calculated in the process of "Punctuation Analysis"

$$P_{punct} = \frac{\sum_{i=1}^{N}\sum_{s\in S_{punct}} count(s, T_i)}{M}, \tag{6}$$

where S_{punct} is the set of punctuation characters (.,!?;:-), P_{punct} is the total proportion of punctuation characters in the dataset. If P_{punct} is too large, this may indicate emotionally charged or manipulative texts, which are more common in propaganda.

To highlight emojis, the emoji library is used in the "Emoji Usage Analysis" process. The proportion of texts containing at least one emoji

$$P_{emoji} = \frac{\left|\left\{T_i \in D : E(T_i) \neq \varnothing\right\}\right|}{N}, \tag{7}$$

where $E(T_i)$ is the set of emojis in tweet T_i. If P_{emoji} is high, emojis may be important for analyzing propaganda texts.

For the process "Analysis of the use of different alphabets", the distribution of characters by alphabets is applied. Let S_{latin}, $S_{cyrillic}$, S_{arabic} be the sets of characters of the corresponding alphabets. Then the share of each alphabet in the dataset is calculated

$$P_{latin} = \frac{\sum_{s\in S_{latin}} f(s)}{|s|} \text{ and } P_{cyrillic} = \frac{\sum_{s\in S_{cyrillic}} f(s)}{|s|}. \tag{8}$$

If the texts contain a mixture of alphabets, this may be a sign of manipulative content. In the process "Visualization of character distribution", a character distribution graph is built to analyze the obtained data.

The main processes of module 3 "Substring search" are based on a linear/optimization substring search algorithm, for example, on the Knuth-Morris-Pratt (KMP) algorithm. Let us have a text T of length n, i.e., $T = \{t_1, t_2, t_3, ..., t_n\}$, where each character t_i belongs to the set of characters of some alphabet Σ. Let us also have a given substring (template) P of length m: $P = \{p_1, p_2, p_3, ..., p_m\}$. The task of the substring search module is to find all indices i, where substring P completely coincides with some subsequence in T: $T[i:i+m] = P$. That is, it finds all i such that $t_i = p_1, t_{i+1} = p_2, ..., t_{i+m-1} = p_m$. The simplest way to find a substring is a naive algorithm that checks all possible positions of i in text T. Its complexity in the worst

case is $O(nm)$. This algorithm goes through all positions of i in text T and checks whether substring P matches a part of the text. Since the naive algorithm has complexity $O(nm)$, it can be improved to $O(n)$ using the Knuth-Morris-Pratt (KMP) algorithm. The main idea of KMP is to avoid unnecessary comparisons by using a prefix function π that, for each prefix of a substring P, determines the largest proper suffix that is also a prefix. Formally, the prefix function is defined as $\pi[j] = \max\{k|P[0{:}k] = P[j-k+1{:}j]\}$, where $P[0{:}k]$ is a prefix of length k, $P[j-k+1{:}j]$ is a suffix of length k. After constructing the prefix function, the KMP algorithm performs a substring search in $O(n)$. This algorithm is significantly faster than the naive method since it does not perform unnecessary comparisons. Detailed analysis of the results makes it possible to improve text classification and automatically detect potentially manipulative messages.

The main processes of module 4 "Text Preprocessing": Removing special characters → Text tokenization → Case changing → Stop word removal → Lemmatization. Text data often contains characters that do not carry useful information for analysis, for example:

– special characters (@, #, !, %, &, *, etc.);

– URLs (https://example.com, www.test.com, etc.);

– emojis (☺ ☻ ♥☼♫, etc.).

Such characters can complicate the analysis and therefore should be removed. Tokenization is the process of splitting the text into individual words (tokens). Formally, for text T consisting of a sequence of characters $T = t_1,t_2,t_3,...,t_n$, a list of words W is formed: $W = \{w_1,w_2,...,w_m\}$, where w_i are the words after splitting the text. Since machine learning models do not distinguish between the words "Propaganda" and "propaganda", all words are reduced to lower case $w_i' = lower(w_i)$. Stop words are frequently used words (for example, "this", "that", "and", "we") that do not carry essential information. Formally, for a set of words in the text $W = \{w_1,w_2,...,w_m\}$, after removing stop words, $W' = W - S$ is calculated, where S is the set of stop words. Lemmatization is the reduction of words to their base form (lemmas). For example: "learning" → "learn", "working" → "work". Formally, lemmatization function L transforms each word w_i into its normal form $L(w_i)$: $W'' = \{L(w_1), L(w_2),..., L(w_m)\}$. After all stages of text preprocessing, each document d_i is represented as a set of lemmatized tokens $d_i = \{w_1,w_2,...,w_k\}$, where k is the number of words left after filtering. Now text data can be converted into numerical representations for models to work with. Text preprocessing improves the quality of analysis and increases the efficiency of models. Filtering out unnecessary elements (URLs, emojis) helps clean up the data. Lemmatization and removal of stop words reduce the size of the text while preserving the content. After processing, the text becomes more structured and ready for further analysis and vectorization.

The main processes of module 5 "Text Vectorization": *One-Hot Encoding* (*OHE*)/*Bag-of-Words* (*BoW*)/*TF-IDF* (*Term Frequency – Inverse Document Frequency*)/*Word Embeddings* (*Word2Vec, GloVe, FastText*). The program uses the TF-IDF method since it makes it possible to assess the importance of words in texts and works well for classification tasks. The TF-IDF method calculates the weight of each word in the text depending on its frequency in the document and rarity in the entire corpus of texts. The formula for calculating TF-IDF for term t in document d: $TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t)$, where TF (Term Frequency) is the frequency of the word in a specific document $TF(t,d) = n_t/N_d$, n_t is the number of occurrences of word t in document d, N_d is the total number of words in document d. IDF (Inverse Document Frequency) – inverse document frequency for a word $IDF(t) = \log(D/(1+d_t))$, where D is the total number of documents, d_t is the number of documents in which word t occurs. As a result, a matrix is built, where each row corresponds to a document, and each column to a specific word. The values in the matrix are the TF-IDF weights for each word in each document. Suppose there are three documents with the following words: d_1 = "propaganda analysis dangerous", d_2 = "analytics machine learning", d_3 = "propaganda machine text". After calculating TF-IDF, a matrix of size $D \times T$ is built, where D is the number of documents, T is the number of unique words, and each value $X_{i,j}$ is the TF-IDF weight for word j in document i

$$X = \begin{bmatrix} 0.5 & 0.7 & 0.0 & 0.0 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.7 & 0.7 & 0.0 & 0.0 \\ 0.5 & 0.0 & 0.0 & 0.0 & 0.7 & 0.5 \end{bmatrix}. \quad (9)$$

TF-IDF takes into account the frequency of words in the entire corpus (IDF), which helps reduce the influence of common words. It is easy to implement and works well on small to medium-sized datasets. It is also easy to interpret. However, it does not take into account word order (it ignores syntax and grammar). It is also not context-sensitive (words with the same spelling but different meanings will have the same weight). TF-IDF is sensitive to unbalanced datasets. In addition to TF-IDF, neural network methods can be used to represent text:

– Word2Vec trains word vectors based on their context;

– GloVe builds word vectors based on a co-occurrence matrix;

– FastText takes into account the morphological structure of words.

However, for the propaganda detection problem, TF-IDF is an effective method because it works well with short texts such as tweets. TF-IDF makes it possible to transform text data into a numerical matrix that can be used in machine learning. The method estimates the importance of words, taking into account their frequency in documents. The vectorized data will be used in the next step – training a propaganda detection model. After vectorizing text data into a numerical format, a machine learning model can be trained on these features to automatically detect propaganda messages.

Main processes of Module 6 "Loading and Preparing Data": *Selecting machine learning algorithms [Logistic Regression* (*LR*)*/ Support Vector Machine* (*SVM*)*/Decision Trees* (*DT*)*/...] → Sample partitioning → Selecting the best model.* Different machine learning algorithms are used to classify texts. Three main approaches LR, SVM, and DT were tested in the program. These algorithms are used to solve a binary classification problem, where each input text has label $y \in \{0,1\}$, where 0 means "not propaganda" and 1 means "propaganda". Logistic regression is one of the simplest and most efficient methods for classification. It is based on the use of a sigmoid activation function $\sigma(z) = 1/(1+e^{-z})$, where z is a linear combination of input features $z = w_1x_1 + w_2x_2 + ... + w_nx_n + b$, w_i is the feature weights, x_i is the feature value (TF-IDF weights), b is the bias. The loss function for logistic regression is

$$J(w) = -\frac{1}{m} \cdot \sum_{i=1}^{m} \left[y_i \cdot \log\left(\hat{y}_i\right) + \left(1-y_i\right) \log\left(1-\hat{y}_i\right) \right], \quad (10)$$

where m is the number of examples in the sample, $\hat{y}_i$ is the predicted value of the model.

The support vector method searches for a hyperplane that maximally separates two classes in the feature space. Formally, it solves the problem

$$\min_{w,b}\left(\frac{1}{2}\cdot\|w\|^2\right) \text{ provided } y_i\cdot\left(w^T\cdot x_i+b\right)\geq 1,\ \forall i, \tag{11}$$

where w is the model weights, x_i is the feature vector, y_i is the class (0 or 1), b is the bias.

SVM uses kernels to work in nonlinear space. For example, the radial basis kernel (RBF kernel)

$$K\left(x_i,x_j\right)=\exp\left(-\gamma\cdot\|x_i-x_j\|^2\right). \tag{12}$$

Decision trees partition the feature space into regions using conditional branches: if $x_i \leq \theta$, then it is classified into C_1, otherwise C_2. The algorithm works using the information entropy metric

$$H\left(S\right)=\sum_{i=1}^{c}p_i\cdot\log_2\left(p_i\right), \tag{13}$$

where p_i is the probability that the object belongs to class i. When constructing the tree, the information gain criterion is used

$$IG\left(S,A\right)=H\left(S\right)-\sum_{v\in V}\frac{|S_v|}{|S|}\cdot H\left(H_v\right), \tag{14}$$

where $IG(S, A)$ is the information gain when partitioning by attribute A, $H(S)$ is the entropy of the initial set, $H(S_v)$ is the entropy of the subsets after partitioning. Before training, the model needs to split the data into training and test samples. Usually, a ratio of 80/20 or 70/30 is used.

The main processes of module 7 "Model evaluation": Error matrix → Classification metrics → ROC-curve and AUC → Cross-validation → Error analysis. After training the model, it is necessary to evaluate its quality on test data. This helps understand how well the model can generalize knowledge on new examples. The evaluation is performed using classification metrics such as accuracy, completeness, F1-measure, and confusion matrix analysis. The confusion matrix shows how the model predicts correct and incorrect classes based on metrics such as

– TP (True Positive) – correct prediction of class "1" (propaganda);

– TN (True Negative) – correct prediction of class "0" (not propaganda);

– FP (False Positive) – falsely predicted "1" (false alarm);

– FN (False Negative) – falsely predicted "0" when class "1" (missed propaganda case).

To select the best model, performance metrics such as Accuracy, Recall, and F1-measure are used:

$$\text{Accuracy}=\frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}},\ \text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}}, \tag{15}$$

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}},\ F1=2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}. \tag{16}$$

The Accuracy metric measures the overall proportion of correct predictions. It is good if the sample is balanced but can be misleading if one class is significantly dominant. Recall measures the proportion of positive cases that the model correctly finds. The higher the recall, the fewer real cases of propaganda the model misses. Precision measures how true the positive predictions are. The higher the precision, the fewer false positives the model produces. The F1 measure is a balance between precision and completeness. If the F1 measure is high, the model balances well between finding all cases of propaganda and minimizing errors. The ROC curve (Receiver Operating Characteristic) shows how the accuracy of predictions changes when the probability threshold is changed. The Area Under Curve (AUC) measures the quality of the model: AUC = 1 means a perfect model, AUC = 0.5 means a random guess. To avoid dependence on a single sample partition, k-fold cross-validation is used. It divides the data into k parts and trains the model k times on different parts

$$\text{TPR}=\frac{\text{TP}}{\text{TP}+\text{FN}} \text{ and } \text{FPR}=\frac{\text{FP}}{\text{FP}+\text{TN}}. \tag{17}$$

To understand the model, one needs to revise the most erroneous predictions. The next step is to optimize the model to improve the results.

5. 2. Method for detecting sources and networks of disinformation dissemination in cyberspace based on the automation of the search for similar text chains

Discovery of a disinformation dissemination network involves the automated search for text messages that have a common origin or structure. This helps:

– identify the original source (source of fake news);

– identify bot networks that spread propaganda;

– build a graph of information dissemination.

The method is based on the comparison of text messages, authors, date and time of publication, style of writing the text, and the order of appearance to detect networks of propaganda, disinformation, and fake news dissemination. The main stages of automating the detection of sources and networks of fake news dissemination:

Stage 1. Text preprocessing (cleaning, tokenization, lemmatization) based on modules 1–4 of the subsystem for detecting disinformation in cyberspace as part of scalable information space monitoring systems.

Stage 2. Calculation of similarity between texts (cosine similarity metrics, Jacquard, Word2Vec).

Stage 3. Clustering of similar texts (K-Means, DBSCAN algorithms).

Stage 4. Construction of an information dissemination graph (network analysis).

Stage 5. Analysis of the results to identify sources of disinformation dissemination.

At Stage 2 "Calculation of similarity between texts", the first step is to measure the similarity between text messages. The code uses various metrics such as Cosine Similarity, Jaccard Similarity, Levenstein Distance, and embedding similarity (Word2Vec Cosine Similarity). Cosine Similarity (TF-IDF, Count Vectorizer) works well for short texts. Jaccard Similarity is effective for simple comparison by words. Levenstein is more suitable for analyzing similarity at the symbol level. Word2Vec makes it possible to take into account the semantic content of words. A similarity matrix S is defined for all texts $S(i,j) = Similarity(T_i,T_j)$, where T_i, T_j are two texts, and Similarity() is one of the metrics (TF-IDF Cosine Similarity, Word2Vec, Jaccard). A similarity matrix between texts is built, for example

$$S=\begin{bmatrix} 1 & 0.85 & 0.45 & 0.20 \\ 0.85 & 1 & 0.50 & 0.25 \\ 0.45 & 0.50 & 1 & 0.60 \\ 0.20 & 0.25 & 0.60 & 1 \end{bmatrix}.$$

This matrix shows how similar the texts are to each other. Cosine similarity or editing metrics are used to compare the texts. Cosine similarity determines the similarity of two texts based on the angular distance between their vectors. CountVectorizer and TfidfVectorizer are used. The formula for the cosine coefficient is $\cos(\theta)=A\times B/(||A||\times||B||)$, where A and B are vector representations of the two texts; $||A||$ та $||B||$ are the norms (lengths) of the vectors; $A\times B$ is the scalar product of the vectors. Cosine similarity works well if the texts contain common words, but it does not take into account synonyms and word meanings.

Jaccard similarity is based on the ratio of common words to the total number of words in the two texts. The formula: $J(A,B)=|A\cap B|/|A\cup B|$, where A and B are sets of words in the two texts; $|A\cap B|$ is the number of common words; $|A\cup B|$ – the total number of unique words. The Jaccard coefficient is suitable for short texts but does not work well with paraphrased sentences. The Levenstein metric determines how many insertion, deletion, or replacement operations are required to transform one text into another. The formula for the recursive approach

$$D(i,j)=\begin{cases}\max(i,j),\text{if}\,(i=0)\,||\,(j=0),\\ \min\begin{cases}D(i-1,j)+1,\\ D(i,j-1)+1, \quad \text{else},\\ D(i-1,j-1)+cost,\end{cases}\end{cases} \tag{18}$$

where cos$t=0$ if the characters are the same, otherwise 1. Levenstein is used to check for grammatical errors and spelling differences.

Embeddings transform words into multidimensional vectors that take into account their semantic meaning. The cosine similarity formula for embeddings is

$$\cos(A,B)=\frac{\sum_{i=1}^{n}A_i\cdot B_i}{\sqrt{\sum_{i=1}^{n}A_i^2}\cdot\sqrt{\sum_{i=1}^{n}B_i^2}}. \tag{19}$$

Word2Vec makes it possible to detect similarities even between synonyms and paraphrased texts. The choice of metric depends on the specific task. If the texts are short like posts on social networks – Jacquard or Levenstein are better suited for analysis. If you need to compare long documents like articles in the media – TF-IDF+Cosine Similarity is the best choice. If semantics are important when analyzing disinformation – choose Word2Vec.

At Stage 3, the grouping of similar messages into propaganda networks is further clustered (K-Means, DBSCAN). Clustering is the process of combining similar texts into groups to highlight the centers of disinformation. One of two approaches is used: K-Means (effective if the number of clusters is specified) and DBSCAN (automatically finds clusters of any shape, convenient for disinformation networks). K-Means divides texts into k clusters using the Euclidean distance between vectors. The centroid update formula

$$C_i=\frac{1}{|s_i|}\cdot\sum_{x_j\in s_i}x_j, \tag{20}$$

where C_i is the center of cluster i, S_i is the set of texts belonging to cluster i.

DBSCAN is well suited for finding botnets that spread fakes. The main idea is that texts with similarity $>\varepsilon$ form a cluster. If a point has $<$*min_samples* of neighbors, it is an outlier, i.e., $N_\varepsilon(x)=\{y\in X|distance(x,y)\le\varepsilon\}$, where ε is the similarity threshold between texts, *min_samples* is the minimum number of points in the cluster. Clusters of similar messages are obtained, where each cluster can indicate a botnet.

At Stage 4, a graph is constructed to visualize the disinformation network. The spread of disinformation can be represented as a graph, where nodes are individual messages, users, or accounts that spread information, and edges are connections between vertices that reflect the similarity of texts or the fact of reposting (high similarity). This approach makes it possible to identify sources and find networks that spread disinformation. Graph G can be represented as an undirected or directed graph containing a set of vertices V and a set of edges E: $G=(V,E)$, where $V=\{v_1,v_2,...,v_n\}$ is the set of vertices (for example, texts or users), $E=\{(v_i,v_j)|Similarity(T_i,T_j)>threshold\}$ is the set of connections between vertices. Depending on the type of connections, the graph can be directed (if the direction of propagation is important, for example, retweets) and undirected (if only the similarity between texts is taken into account). The adjacency matrix A is an $n\times n$ square matrix, where n is the number of texts or accounts

$$A(i,j)=\begin{cases}1,\text{if}\,S(i,j)>\tau,\\ 0,\text{else},\end{cases} \tag{21}$$

where $S(i,j)$ is the similarity measure between texts (e.g., cosine similarity), τ is the similarity threshold (e.g., 0.8), $A(i,j)=1$ means that there is a connection between the texts. The NetworkX library was used to construcy a graph based on the similarity matrix (simulation) $S=np.array([[1, 0.9, 0.2, 0.0], [0.9, 1, 0.3, 0.1], [0.2, 0.3, 1, 0.85], [0.0, 0.1, 0.85, 1]])$. A graph was built where the vertices are texts, and the connections between them show the level of similarity. Next, an analysis of key nodes was performed (identification of sources of disinformation). Some accounts or texts may be central in the spread of propaganda. The following vertices are identified using the metrics Degree Centrality and Betweenness Centrality. Degree Centrality shows the number of connections of a node $C_D(v)=deg(v)/(n-1)$. The higher the degree, the more this account spreads information. Betweenness Centrality shows how often a node is on the shortest paths between other nodes

$$C_B(v)=\sum_{s,t\in V}\frac{\sigma(s,t|v)}{\sigma(s,t)}, \tag{22}$$

where $\sigma(s,t)$ is the number of shortest paths between vertices s, t; $\sigma(s,t|v)$ is the number of paths passing through v. The graph helps identify central accounts that spread information (finding the most connected nodes). It also makes it possible to detect automated campaigns (for example, groups of accounts with strong connections between identical texts). Analysis of the graph structure and the formed connections encourages one to understand the structure of propaganda distribution.

If accounts form dense clusters, this may be a bot network. The process of detecting bot networks is implemented based on the clustering coefficient and spectral partitioning of the graph (detection of subgroups). The clustering coefficient shows how connected a node is to other nodes

$C(v)$ = number of triangles with v/maximum number of possible triangles.

The spectral decomposition of the graph is based on the DBSCAN method. If dense groups of accounts publishing the same content are detected, this may indicate an automated bot network. Automation of the analysis of similar texts allows for the effective detection of fake networks. Cosine similarity, Jaccard, Levenstein, and Word2Vec are used to measure similarity. Clustering (DBSCAN, K-Means) helps group fake messages. Graph analysis detects central accounts and bot networks. Further improvements will consist in temporal analysis (the dynamics of the spread of fakes over time). Neural networks are also used to automatically classify texts before building the graph. It is advisable to use the main manipulators through betweenness and clustering analysis.

5. 3. Method for identifying ways of disinformation dissemination in cyberspace based on the identification of stylistically similar content

The graph of disinformation dissemination can be supplemented with an analysis of text authorship, which will make it possible to identify bots that use template phrases and automatic generators. It will also help identify accounts of individual people or groups that write disinformation in a characteristic style. Methods for linking the graph to authors are based on stylometry (analysis of text style), generative patterns (detection of template content of bots), graph connectivity of accounts (who interacts with whom), and temporal analysis (when and how often content is disseminated). The main stages and steps of analyzing text authorship through the graph of disinformation dissemination:

Stage 1. Stylometry for authorship detection:

Step 1. 1. Frequency analysis of word and symbol usage.

Step 1. 2. Determination of average sentence and word length.

Step 1. 3. Punctuation and grammar analysis.

Step 1. 4. Vectorization of author style (TF-IDF) and clustering of authors.

Step 1. 5. Analysis of clustering results and prediction on test samples based on deep neural network models for stylometry.

Stage 2. Detection of template content of bots:

Step 2. 1. Detection of repeating patterns in texts.

Step 2. 2. Analysis of syntactic structure of text.

Step 2. 3. Analysis of text length.

Step 2. 4. Detection of cyclic publications (timer bots).

Step 2. 5. Visualization of bot network.

Stage 3. Linking the graph to authors through shared sources:

Step 3. 1. Building a graph of shared sources.

Step 3. 2. Identifying common phrases between authors.

Step 3. 3. Analyzing the author's style (stylometry).

Step 3. 4. Building a graph of authors.

Step 3. 5. Analyzing the most influential sources of disinformation.

Stage 4. Interaction analysis and the time factor:

Step 4. 1. Analyzing the interaction graph.

Step 4. 2. Analyzing the time of publication of messages.

Step 4. 3. Combining the analysis of style and time of publication.

Step 4. 4. Visualizing the graph with temporal relationships.

Stage 5. Visualizing the distribution graph with authors:

Step 5. 1. Identifying the most important authors (centrality in the graph).

Step 5. 2. Detecting bots in the graph.

Step 5. 3. Combining the graph with the analysis of the author's style.

Step 5. 4. Visualizing the graph with distribution routes.

Description of stage 1. Stylometry is a method of analyzing the writing style of texts to identify the author or group of authors. It is based on the fact that each person has a unique vocabulary, grammatical constructions, punctuation, and other language features. People and bots use different word patterns. The author is characterized by unique word combinations, sentence structure, and punctuation. For each author, a vector characteristic F_A is defined, which contains the frequency distribution of words, the average length of sentences, and an analysis of the use of punctuation marks. For each author, the distribution of words used (word frequency distribution) can be calculated. The probability of the occurrence of word w_i in text d is determined from the following formula

$P(w_i)$ = number of occurrences of word w_i/total number of words in text d.

If certain words occur more often for one author, this may be his/her lexical signature. In addition to word frequency, one can analyze the use of symbols, for example: the number of punctuation marks, the use of capital letters, the frequency of specific symbols, such as "!" or "...". The formula for calculating punctuation

P_A = number of punctuation marks in text A/total number of characters in text A.

If the author often uses "..." or "!!!", this may be his/her stylistic feature. People also have different writing styles. Some write long sentences using complex constructions. Others write short, using simple sentences. The average length of sentences L_A and words W_A, respectively

$$L_A = \frac{\sum_{i=1}^{n} |s_i|}{n} \text{ and } W_A = \frac{\sum_{i=1}^{m} |w_i|}{m}, \quad (23)$$

where L_A is the average sentence length of author A, S_i is the length of the i-th sentence, n is the total number of sentences in the text, W_A is the average word length in the texts by author A, w_i is the i-th word, m is the total number of words in the text. If a person writes long sentences and a bot writes short ones, this will be noticeable. TF-IDF helps find unique words that the author uses more often than others $TF\text{-}IDF(w,d) = n_w/N_d \times \log(D/d_w)$, where n_w is the number of occurrences of word w in document d, N_d is the total number of words in document d, D is the total number of documents, d_w is the number of documents containing word w. When clustering author styles, a cluster of authors who write in a similar style is determined. To group authors, one can use K-Means or DBSCAN. The formula for updating the centers of K-Means clusters

$$C_i = \frac{1}{|s_i|} \cdot \sum_{x_j \in s_i} x_j. \quad (24)$$

Authors have unique language patterns that can be identified. Word and character frequency analysis makes it possible to find unique features of authors. TF-IDF and clustering help combine texts of the same author into groups. If accounts have the same style, they may belong to the same source. The next steps of the research are based on deep neural networks for stylometric analysis. It will be necessary to analyze the

time patterns of writing texts and compare author patterns with the database of known accounts.

Description of Stage 2. Bots often generate texts according to the same/similar template for mass distribution or use GPT models, which allows them to be identified. To identify them, the number of repeating patterns in texts, sentence structure (frequent use of the same constructions) and distribution of the same text by different accounts were analyzed. The main detection methods:

– analysis of repeated phrases using N-grams;

– detection of syntactic and semantic matches;

– analysis of sentence structure and message length;

– detection of cyclic publications (posting on a timer).

To analyze similar texts generated by a template, N-gram analysis is used – splitting the text into sequences of N words. The N-gram frequency formula

$P(N)$ = the number of occurrences of N-grams/the total number of N-grams in the text.

If certain phrases are frequently repeated among bots, it means that they are using a text generator (☺ humans → a wide variety of phrases| ☻ bots → similar patterns in most messages). The formula for the frequency of grammatical structures

$F(S)$ = number of sentences with a certain grammatical structure/total number of sentences.

If a bot uses repetitive sentence structures, it is different from a human. Bots often have repetitive sentence structures (e.g., "subject + predicate + object"). This can be done by parsing the text into parts of speech (POS tags) and comparing patterns. Bots also often generate short, clearly structured messages, and are often of a fixed length or range of lengths (e.g., 10–15 words). Humans have variable sentence lengths (short and long messages) and can use more detail. The formula for average text length is

$$L_T = \frac{\sum_{i=1}^{N} |T_i|}{N}, \tag{25}$$

where $|T_i|$ is the length of text i, N is the total number of messages.

Bots often publish messages at certain time intervals. This can be determined by analyzing time patterns. The formula for distributing posts over time is

R_A = number of posts per time period/duration of the period.

If R_A is constant, then the account may be a bot. If the posts are made evenly (for example, every 4 hours) – it is a bot. If bots and template content are identified, a graph of fakes is built, for example, red nodes are bots, blue nodes are the same content that they spread, and connections between bots indicate a coordinated campaign. N-gram analysis reveals template phrases of bots. POS tags help find a repeating sentence structure. The length of bot texts is often the same. Time analysis makes it possible to detect timer bots. Graph analysis shows connections between bots and template content. The next steps of the study are to analyze the emotional coloring of bots. It is necessary to use GPT detectors to detect artificially created texts and implement deep neural networks for stylometric analysis of bots.

Description of Stage 3. After clustering styles, one can add authors as attributes to the graph. This will make it possible to identify the source if the same text styles belong to multiple accounts (perhaps the same person or group) and bot accounts are spreading texts with template N-grams. To link the graph of disinformation spread to specific authors, it is necessary to consider:

– writing style (who created the text);

– distribution graph structure (who retweeted or copied the text);

– reuse of phrases and patterns (is the content repeated across different accounts).

Then the source of disinformation is identified, and which groups of accounts work together are determined. Each account or text can be represented as a graph, where the vertices are authors/texts and the edges are the similarity between texts or actual interaction (retweet, copy). Let there be a set of authors $A = (a_1, a_2,..., a_n\}$ and a set of texts $T = \{T_1, T_2,..., T_m\}$. Then graph G can be described as $G = (V,E)$, where $V = A \cup T$ is the set of authors and texts, $E = \{(a_i, T_j)|$*author* a_i *wrote or distributed* $T_j\}$, $E = \{(T_i, T_j)|$ *Similarity*$(T_i, T_j) > \tau\}$ (if the texts are similar). If two different accounts use the same phrases, this may mean that they copy each other and/or they belong to the same bot network. N-gram analysis was used to detect such connections. The formula for similarity between texts

$$S\left(T_i,T_j\right) = \left|\frac{N\left(T_i\right) \cap N\left(T_j\right)}{N\left(T_i\right) \cup N\left(T_j\right)}\right|, \tag{26}$$

where $N(T_i)$ is the set of N-grams in text T_i, $|N(T_i) \cap N(T_j)|$ is the number of common N-grams between the texts. If accounts A and B publish similar texts, an edge is created between them in the graph. If accounts use the same phrases regularly, they may be part of the same network. Stylometric methods are used to detect authors using the same style. If two authors use similar phrases and style, they may work together. After finding similar texts, a graph of common sources can be constructed. Betweenness Centrality (22) is used to find key authors. If an account has high betweenness centrality, it is a likely source of disinformation. The graph makes it possible to detect networks of accounts using similar texts. N-gram analysis detects common phrases between bots and people. Stylometry helps find authors with similar styles. Graph analysis of betweenness centrality identifies the most influential sources of fakes. If accounts have the same style, similar phrases, and related texts, they can work together. The next step of the research is automated analysis using neural networks (LSTM, BERT).

Description of Stage 4. Identifying sources of disinformation is possible not only through the analysis of text content but also through the analysis of interactions between users and the rate of publication of messages. Key methods:

– analysis of interactions in the graph (who retweets or copies whom);

– temporal analysis of message distribution (activity spikes);

– coincidence of authorship style over time (similar style + simultaneous publication = coordinated attack);

– detection of bots through uniform intervals of posts.

The key signal of coordinated attacks is simultaneous publications of similar texts. Bots act quickly and synchronously, people act chaotically. One can track the time of publication of texts to find suspicious patterns. The formula for the intensity of posts

R_A = number of posts per period of time/duration of the period,

where R_A is the author's posting rate. If R_A is large and stable, it may indicate an automated account. Bots often have a high R_A, which distinguishes them from humans. If posts are made at a fixed interval, it is likely a bot. If we have an information dissemination graph, then nodes can be ☺ sources of fakes (authors who create misinformation), ☻ reposters (those who spread), and ◙ intermediate accounts (may be bots). The graph $G(V, E)$ contains V – a set of users, E – a set of links (u, v), where user u spread information from v. The adjacency matrix for the graph is given by formula (21). The central node (the largest number of outgoing links) is a likely source of disinformation. If accounts spread messages at the same time, this is a coordinated campaign. If several accounts simultaneously publish similar text, they can be bots or participants in an information campaign. Publication time correlation formula

$$C(T_i,T_j)=\frac{\sum_{k=1}^{n}\left(t_{i,k}-\bar{t}_l\right)\cdot\left(t_{j,k}-\bar{t}_j\right)}{\sqrt{\sum_{k=1}^{n}\left(t_{i,k}-\bar{t}_l\right)^2}\cdot\sqrt{\sum_{k=1}^{n}\left(t_{j,k}-\bar{t}_j\right)^2}}, \quad (27)$$

where $t_{i,k}$ is the time of the k-th publication of account i, t_l is the average publication time of account i. If the correlation of publication times is high ($C > 0.9$), this indicates coordinated dissemination. To determine the relationship between style and publication time, we compare:

– time intervals between posts (bots act evenly);

– stylistic similarity of texts (similar style = probable common author). Calculation of stylistic similarity between accounts

$$S(A,B)=\frac{\sum_{i=1}^{m} w_i\cdot f_{i,A}\cdot f_{i,b}}{\sqrt{\sum_{i=1}^{m}\left(f_{i,A}\right)^2}\cdot\sqrt{\sum_{i=1}^{m}\left(f_{i,B}\right)^2}}, \quad (28)$$

where $f_{i,A}$ is the frequency of word i in the texts by author A, $f_{i,B}$ is the frequency of word i in the texts by author B, $S(A, B)$ is the cosine similarity between styles. If two accounts have a similar style ($S > 0.8$) and a synchronous posting time ($C > 0.9$), they may be part of the same network. Connections between accounts that post at the same time and have a similar style indicate a network of disinformation distribution. If accounts post at the same time, this may be a coordinated attack. Graph analyses makes it possible to determine connections between authors through the style and time of posts. Posting frequency analysis detects bots (uniform time intervals). The combination of stylometry and temporal analyses indicates the real authors of fakes. The next step is to use LSTM or Transformer models to detect anomalies in the style and time of posts.

Description of stage 5. The information dissemination graph allows one to:

– identify the main sources of disinformation (key authors);

– find bots and coordinated groups;

– trace the distribution chains (who reposts whom, who creates content);

– combine the style of the text and the structure of the graph to establish a connection between authors.

Recommended designations on the graph:

– ☺ Source of fakes → accounts with many outgoing links. Central nodes are possible sources of fakes.

– ☻ Reposters and/or real accounts of users copying the text.

– ◙ Bots → accounts acting in a coordinated manner.

When building a graph to identify disinformation authors, the results of the analysis of several parameters and criteria are taken into account. In particular, clustering by style (stylometry) of the text helps find one person or group. If many accounts have the same style, they may be bots (pattern analysis). If accounts with the same style closely interact, they may work together (connectivity graph). Bots act faster than people (temporal analysis). When identifying a network of bots and the source of fakes, the results of the analysis of several parameters and criteria are also taken into account. In particular, the most central accounts in the graph are likely primary sources. Groups of accounts with the same style and graph connections are possible coordination. Bots have short intervals between publications and a template structure of messages. The graph G consists of nodes $V = A \cup T$, where A are authors, T are texts, and edges $E = \{(a_i, T_j) | \textit{author } a_i \textit{ wrote or distributed } T_j\}$. The graph is represented as a directed graph $G = (V, E)$, where the edges *from the author to the text are → who created the content and the edges between the texts are → who copied whom*. The adjacency matrix of the graph and is calculated based on the formulas:

$$A(i,j)=\begin{cases} i, \text{if } \textit{author } A_i \textit{ spread text } T_j, \\ 0, \text{else}, \end{cases} \quad (29)$$

$$C_B(v)=\sum_{s,t\in V}\frac{\sigma(s,t|v)}{\sigma(s,t)}, \quad (30)$$

where $\sigma(s, t)$ is the number of shortest paths between vertices s and t; $\sigma(s, t|v)$ is the number of paths passing through v. If an account has high betweenness, it is a likely source of misinformation. If an account has high betweenness, it may be the main source of misinformation. Clustering Coefficient

$C(v)$ = number of triangles with v/maximum number of possible triangles.

If accounts form very dense groups, it may be a bot network. Bots often have a high clustering coefficient (closely interconnected). To link the graph structure to the author's style, style similarity between accounts is added:

1. Style similarities (TF-IDF+cosine measure) are calculated according to (28).

2. If $S(A, B) > 0.8$ and there is a connection in the graph, coordination is possible. If two accounts have a similar style and are connected in the graph, they may be part of the same network. Red nodes are authors, and blue nodes are their texts. If the texts are connected, the authors may be part of the same network.

For more accurate analysis and identification of sources of disinformation distribution, it is necessary to use GPT detectors to detect generated content. It is also necessary to analyze the emotional tone of texts (neutral, aggressive, manipulative) and add time graphs to see when a wave of disinformation begins. Graph visualization shows sources of disinformation. Betweenness indicates the most important authors. Clustering coefficient helps find bot networks. The combination of stylometry and graph shows who works together. If accounts have a similar style, spread the same content and are closely connected in the graph, they may be part of a coordinated information attack. The next step is to use LSTM/BERT for extended/deeper analysis of styles/graphs.

5. 4. Validation of the proposed methods for identifying sources and paths of disinformation based on the developed software

The program interacts with a dataset containing tweets classified as propaganda or neutral. The input data is provided in CSV format, which contains the main columns: text (tweet content) and label (class label, where 0 is not propaganda, 1 is propaganda). In addition, there are additional technical parameters, such as tweet identifier, publication date, publication author and publication time. The pandas library is used to process the data. At the initial stage, the following operations are performed:

1. Removing unnecessary columns that are not relevant for the analysis.

2. Checking the balance of classes in order to assess the evenness of the representation of each category. In case of significant unevenness, balancing methods such as oversampling or undersampling can be applied. The current state of the dataset consists of 61.7% of records with label 1 and 38.3% with label 0. 10 experiments were conducted, the description of which is given in Table 1.

Table 1

Description of experiments to identify sources of disinformation

No.	Machine learning	Vectorization	Cleanup
1	ComplementNB	TF-IDF	– Remove HTML tags; – Remove Special Characters; – Convert to Lowercase; – Normalize Whitespace; – Tokenize; – Stem Words (UkrStemmer lib)
2	GaussianNB	FastText	
3	ComplementNB	TF-IDF	– Convert to Lowercase; – Tokenize; – Remove stopwords; – Lemmatize(spaCy lib)
4	GaussianNB	W2V	
5	ComplementNB	TF-IDF	– Remove punctuation; – Replace numbers with words; – Convert to Lowercase; – Remove stopwords; – Translates English words to Ukrainian
6	HistGradient Boosting Classifier	Glove	
7	RandomForest		
8	MultinominalNB		– Remove stopwords; – Lemmatize; – Remove emojis
9	RandomForest		
10	Logistic Regression	TF-IDF	

3. Checking for missing values in the text column. If they are found, they can be deleted or filled in depending on the context of the task.

4. Adding a new column containing the length of the tweet, which makes it possible to assess the possible impact of short/long messages on the effectiveness of the model.

Since Twitter supports a large set of characters, including emojis and special characters, it is important to take into account their presence in texts. For this purpose, a set of unique characters is formed, which makes it possible to identify potential problems when processing the text. The analysis reveals that the following are often found:

– emojis, which can carry the emotional context of the message;

– characters from other alphabets, which may indicate multilingualism of the dataset;

– special characters and punctuation marks that affect tokenization.

Based on this analysis, a decision is made on further processing of such characters (deletion, replacement, or inclusion in the analysis). To identify thematic keywords related to propaganda, the substring_check(substring) function has been implemented, which makes it possible to find certain words or phrases in tweets and analyze their frequency in different classes. This makes it possible to:

– identify patterns of keyword usage in fakes;

– analyze the impact of certain terms on the classification result;

– improve the model by expanding the set of features.

Tweet texts undergo several stages of processing:

– removal of special characters, links, emojis and punctuation;

– tokenization – dividing the text into separate elements;

– converting all words to lowercase;

– removal of stop words (for example, "and", "this", "or");

– lemmatization – reducing words to their base form.

Various algorithms were used to train the model (Table 2), in particular:

– Complement Naïve Bayes (ComplementNB) – a variant of Multinomial Naïve Bayes, adapted for processing unbalanced classes in classification problems;

– Gaussian Naïve Bayes – a variant of Naïve Bayes, which assumes a normal (Gaussian) distribution of features;

– HistGradientBoostingClassifier – a powerful gradient boosting algorithm based on an ensemble of decision trees using histogram binning;

– Multinomial Naïve Bayes – a Naïve Bayes algorithm for classification;

– RandomForest algorithm for detecting complex nonlinear dependences.

Table 2

Algorithm comparison table

Algorithm	Advantages	Purpose	Disadvantages
Complement NB	Resistant to unbalanced classes	Text data	Not for numeric features
Gaussian NB	Simple, fast	Numerical data	Does not work well with non-Gaussian distributions
HistGradient Boosting	Fast, robust	Large data sets	Complex setup
Random Forest	Scalable, flexible	Different types of features	Difficult to interpret
Multinomial NB	Fast, good at word frequencies	Text classification	Does not support numeric features

The ratio of 80:20 was used to form the training and test samples. The final results for class F (Fake) are shown in Fig. 1. The best indicators at the moment are demonstrated by experiment 5, which is based on the TF-IDF and ComplementNB methods. At the same time, anomalies are observed in the series of experiments (in particular, in experiment 7, which uses Glove and RandomForest), which require further analysis. For processing text data, the most suitable methods are MultinomialNB and ComplementNB. For working with numerical features, it is recommended to use GaussianNB or ensemble methods, such as RandomForest. For large volumes of data,

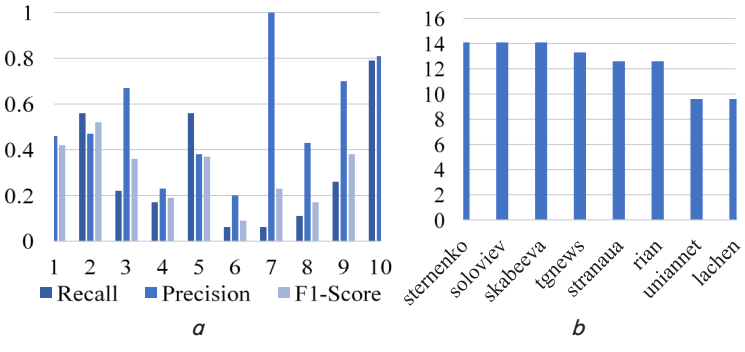
the use of HistGradientBoosting is effective. RandomForest is optimal for working with combined features that include both numerical and categorical data. The plot (Fig. 2) shows the change in the number of fakes, propaganda, and manipulations during 2024. Such visualization makes it possible:

– to determine periods with a peak level of disinformation distribution;

– to compare trends in the dynamics of different types of disinformation;

– to use the obtained data for further training of classification models, forecasting and construction of graphs (networks) of the spread of fake information (Fig. 3).

The proposed module demonstrates high efficiency in detecting propaganda. Further improvement is possible by expanding the dataset and adapting the model to multilingual analysis.



a *b*

Fig. 1. Analysis plots of metrics for detecting sources of disinformation: *a* – Recall, Precision, and F1-Score; *b* – sources of fakes

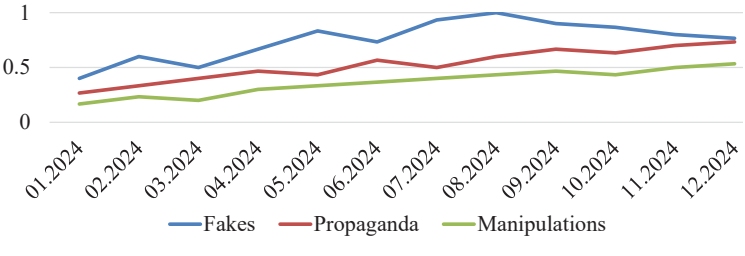
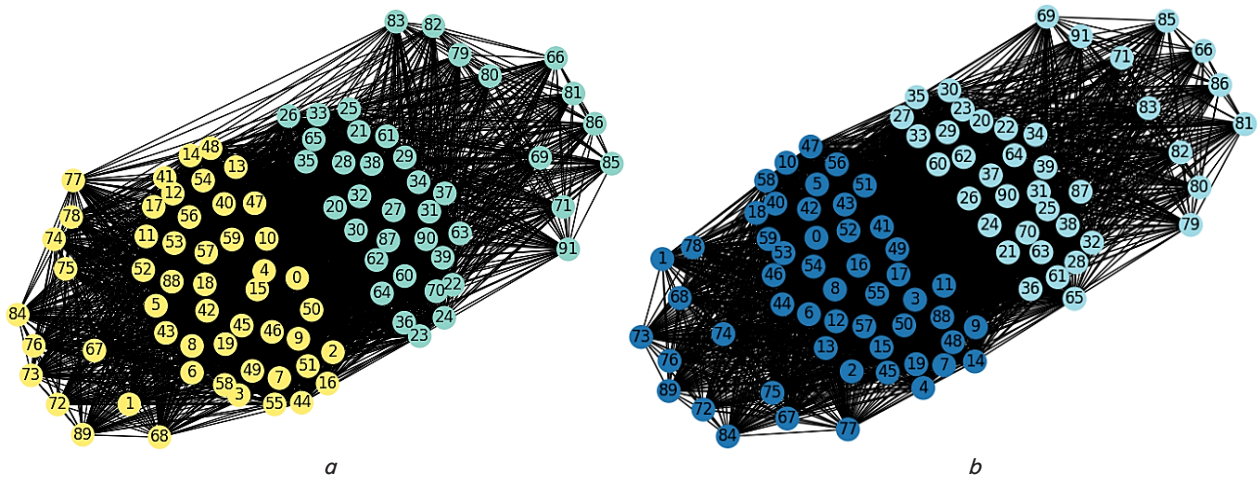


Fig. 2. Changes in the number of fakes, propaganda, and manipulations over time

6. Discussion of results related to identifying sources of disinformation in cyberspace based on machine learning

The bulk of news messages have a length of 50–500 characters, and the number of words in most texts is from 24 to 41. The distribution of text lengths is characterized by a right-side shift. This is explained by the fact that a significant proportion of messages are short, although there are individual cases with an abnormally long length. Such characteristics are essential for further data processing, in particular during model training, analysis of its errors, and assessment of classification quality. By default, the test_size parameter in the FakeNewsClassifier class constructor is set to 0.2, which corresponds to 20% of the total number of records – a typical ratio for machine learning tasks. The distribution is implemented taking into account the preservation of proportions between classes, which makes it possible to maintain a balance between fake and authentic news in both samples. In most texts of the training sample, the number of tokens is in the range from 18 to 42. Moreover, the maximum concentration is observed in the range of 37–42 tokens. In contrast, the distribution of the lengths of the tokenized texts in the test sample is more variable and is characterized by a less pronounced peak value (Fig. 1, *b*). These results indicate the relative constancy of the lengths of the texts after tokenization. This indicates the homogeneity of the corpus and the proper quality of the data preprocessing before their transfer to the Granite model.

The classification model was trained using the log_reg_and_report() method, which implements logistic regression based on the hybrid feature space generated using embedding(). Within the framework of the approach, the model is trained on the matrix x_train_hybrid and the corresponding labels y_train, after which class prediction is performed for the test sample. Logistic regression is initialized with the parameters penalty = 'l2', solver = 'lbfgs' and class_weight = 'balanced'. The latter parameter makes it possible to compensate for a slight imbalance between classes, ensuring automatic scaling of weights according to their representation in the training set.



a *b*

Fig. 3. Example of a fake news distribution network: *a* – ○ source of fake news and ○ reposters → accounts that copy the text; *b* – ○ Bots → accounts that act in coordination and ○ real users → accounts that copy the text

This helps increase the sensitivity of the model to the less represented class (Fig. 1, *a*). Within the framework of classification tasks, it is advisable to evaluate the effectiveness of the model not only by the general accuracy indicators but also by taking into account its ability to correctly identify each class separately (Tables 1, 2). For this purpose, metrics such as recall and precision are widely used, which reflect the quality of the model from different analytical perspectives. The recall indicator shows what proportion of objects of the positive class, which in this study is fake news, the model was able to correctly classify. That is, how effectively it detects all relevant examples, without allowing them to be missed. In contrast, precision shows the proportion of objects that truly belong to the "fake" class among all those that were assigned to it. In other words, this indicator characterizes the accuracy of the model's prediction in terms of avoiding false positive solutions. Depending on the specific requirements of the applied problem and the possible consequences of errors, one of these metrics is often preferred in machine learning practice. This necessitates a comprehensive analysis.

Unlike the results reported in [19], in which the focus is on the analysis of the accuracy of disinformation detection at the level of English-language articles/sentences, the main emphasis in this experiment is on recall. This is explained by the fact that the goal of the model is to timely and fully detect multilingual fake news (in Ukrainian, English, and Russian) (Fig. 2). Even with a partial decrease in precision, such a compromise is justified within the framework of the task since the priority is to detect as many fake news as possible. According to the results of the classification analysis, the overall accuracy of the model is 0.82. This indicates a high level of consistency between the model predictions and the actual class labels. The F1-measure for the "true" class is 0.84, and for the "fake" class is 0.80. This demonstrates a relatively balanced effectiveness of the model in recognizing both types of news. The values of the macro average and weighted average metrics are also at the level of 0.82. This indicates a stable classification quality under the condition of both uniform and weighted consideration of classes. For a deeper assessment of the classification quality, a method was used that generates three types of discrepancy matrices: without normalization, normalized by precision, and normalized by recall. Such a multidimensional approach allows for a comprehensive analysis of the model's behavior with respect to both classes, revealing a tendency to false positive or false negative decisions.

Within the framework of this method, the value of the Fβ-measure with the parameter $\beta = 1.3$ is also calculated. This provides a weight shift to the recall indicator – a critically important aspect in the context of detecting fake messages. The obtained values: precision for the "fake" class – 81%, recall – 79%. This is an acceptable balance for the task, where the priority is to minimize the omission of fakes. The corresponding value of the Fβ-measure is approximately 0.79, which confirms the effectiveness of the model in detecting the target (positive) class. To assess the discriminatory ability of the model, regardless of the classification threshold, the method of constructing the ROC-curve (Receiver Operating Characteristic) and calculating the area under it (AUC) was used. In the study, the "fake" category was defined as the positive class (via the pos_label parameter). When the method is called, the probabilities of belonging to this class are calculated, which are used to construct the ROC-curve. The obtained AUC value = 0.86 indicates a high ability of the model to differentiate between fake and true news. The closer the area under the curve is to 1.0, the higher the quality of the classification distinction. It should be emphasized that, unlike the accuracy metric, which is based only on classification decisions at a fixed threshold, the ROC-AUC indicator makes it possible to evaluate the performance of the model over the entire range of threshold values. This approach is especially valuable in cases where errors of different types have unequal weight or when it is necessary to adapt the decision-making system to specific conditions. To analyze the impact of individual terms on the classification results, the tfidf_feature_importance() function was used. It displays the logistic regression weights associated with TF-IDF features. This makes it possible to identify which words have the greatest impact on the model's decisions. To avoid including obvious predictors such as functional or very frequent words, the 20 most important features were removed from the output. Signs with positive coefficients indicate an association with the "fake" class – among them, in particular, the words "boevik", "zapad" and "ocherednoy". In contrast, negative coefficients correspond to terms characteristic of the "truth" class, such as "ukraine", "actually", "video", "also", "untrue". This makes it possible to conclude that the model identifies the "truth" class with neutral or Ukrainian-language vocabulary, while the "fake" class is characterized by Russian-language words, often colored by manipulative or propaganda rhetoric.

Unlike [20], in which the study focuses only on identifying sources of English-language disinformation, this study is not limited to one language. Usually, when identifying sources and networks of disinformation, a combination of different approaches is used, such as natural language processing and social network analysis. However, the use of well-known machine learning models is justified only if there is English-language content and standard disinformation distribution tactics. In particular, this is confirmed by the content and results of studies in [11–18]. When trying to overcome these limitations to increase the speed, accuracy, and quality of the processes of identifying sources and networks of disinformation distribution in the global cyberspace, objective difficulties arise, which are associated with the uncertainty of the mechanisms of the process of adapting models to new disinformation distribution tactics and ensuring the scalability of solutions. Our study proposes a way to overcome these difficulties. It is based on the fact that the procedure for determining the mechanism should be preceded by a stylistic analysis of the content, an analysis of inauthentic behavior of chatbots, a temporal analysis of information dissemination and an analysis of the graph (routes) of dissemination (Fig. 3), which is the basis for increasing the accuracy of identifying the distribution networks of the corresponding set of fake news from one group of authors. This method allowed us to obtain a graph of the spread of disinformation by time slices in social communities. This means that the scientific result in the form of an information technology for identifying sources and networks of disinformation dissemination in cyberspace based on machine learning methods is interesting from a theoretical point of view.

From a practical point of view, the identified mechanism for finding text content similar in content and writing style in certain time periods in the relevant social networks (Fig. 3) allows for the identification of central accounts and botnets for the spread of disinformation in the cybersecurity technology of a country/company. Thus, the applied aspect of using our scientific result is the possibility of improving the typical technological process of identifying information threats and supporting the country's information defense capability.

This creates the prerequisites for the transfer of the obtained technological solutions in the cyberspace of the country and the world. All this gives grounds to assert that the goal of the study has been fully achieved.

The study identifies mechanisms for identifying sources and networks for the spread of disinformation in the global cyberspace, which can adapt to different cultural and linguistic contexts with minimal additional training costs. This makes it possible to reasonably approach the identification of a set of central accounts-distributors of fake news and bot networks and to obtain certain effects from the implementation in production. In particular, the accuracy of the technological process of detecting sources and networks of disinformation distribution in cyberspace of the world can be increased and the cost of production of such systems can be reduced. Also, additional results of the study were the detection and analysis of inauthentic behavior of chatbots, which differs from the content and results of research reported in [21–25]. This significantly improves the results of detecting sources and networks of fake news distribution.

Our solutions and research results based on the developed information technology for detecting sources and networks of disinformation resolve the task set. The influence of the substantive and stylistic features of the text content of disinformation, written according to methodological narratives and with certain keywords, on the mechanisms and processes (frequency and period of publication, order and frequency of reposts, methodical patter, etc.) of the spread of disinformation in the Internet space has been proven. This is important because the introduction of additional parameters and coefficients for identifying disinformation distribution networks significantly changes the mechanism and accuracy of their detection. But there are objective difficulties associated with the rapid evolution of disinformation methods and the limited availability of data for training models because there is a lack of adaptation of models to new tactics.

The limitations of our study are the imbalance of classes (since propaganda tweets are rare) and the large resource consumption in identifying primary sources of disinformation using retweet analysis.

The shortcomings of the study include the lack of a detailed analysis of the time of message distribution to build a dynamic graph. It is necessary to analyze the time patterns of writing texts and compare author's patterns with a database of known accounts.

The development of our research implies adding deep neural networks (LSTM, BERT) to improve the quality of training and stylometric analysis. It is necessary to investigate the results of using neural network models (BERT, GPT-4) to improve classification. It is also necessary to use class balancing and optimize hyperparameters (GridSearch, RandomizedSearch). It is desirable to analyze the emotional coloring of bots. It is necessary to use GPT detectors to detect artificially created texts and implement deep neural networks (LSTM, BERT, Transformer) for stylometric analysis of bots and detect anomalies in the style and time of posts.

7. Conclusions

1. Based on our analysis of existing information technologies for identifying fake news, a set of general functional requirements for a typical architecture of a subsystem for detecting multiple disinformation for scalable information space monitoring systems has been determined. That has made it possible to define a set of criteria and parameters for detecting sources and networks of disinformation distribution. Pre-processing of text improves the quality of analysis and increases the efficiency of models. Filtering of unnecessary elements (URLs, emojis) helps clean the data. Lemmatization and removal of stop words reduce the volume of text while preserving its content. After processing, the text becomes more structured and ready for further analysis and vectorization. TF-IDF takes into account the frequency of words in the entire corpus (IDF), which helps reduce the influence of common words. It is easy to implement and works well on small and medium-sized data sets. It is also easy to interpret. But it does not take into account word order (it ignores syntax and grammar). It also does not distinguish between contexts (words with the same spelling but different meanings will have the same weight). TF-IDF is sensitive to unbalanced datasets. In addition to TF-IDF, the following neural network methods can be used to represent text: Word2Vec (learns word vectors based on their context), GloVe (constructs word vectors based on a co-occurrence matrix), and FastText (takes into account the morphological structure of words). However, for the problem of propaganda detection, TF-IDF is an effective method because it works well with short texts such as tweets. TF-IDF makes it possible to transform text data into a numerical matrix that can be used in machine learning. The method estimates the importance of words, taking into account their frequency in documents. The vectorized data will be used in the next step – training a model for propaganda detection. After vectorizing the text data into a numerical format, a machine learning model can be trained on these features to automatically detect propaganda messages. The highest efficiency at the current stage is demonstrated by the fifth experiment, which uses the Complement Naive Bayes model based on TF-IDF features. At the same time, some deviations were found among the results, in particular in the seventh experiment, which combines GloVe embeddings with the Random Forest algorithm, which require additional analysis to clarify the reasons for their occurrence. Our experimental data can be used as the basis for further research aimed at improving approaches to identifying sources of disinformation, inauthentic activity in network communications, as well as malicious content in order to increase information security and the defense capability of the state.

2. Based on our analysis of existing methods of intelligent search for disinformation, a method for detecting sources and networks of disinformation distribution in cyberspace has been devised through the automation of the search for similar text chains. Automation of the analysis of similar texts allows for the effective detection of fake networks. Cosine similarity, Jaccard, Levenstein, and Word2Vec are used to measure similarity. Clustering (DBSCAN, K-Means) helps group fake messages. Graph analysis detects central accounts and bot networks. In particular, the graph model allows for the analysis of connections between accounts and messages. The use of centrality makes it possible to identify the main sources of disinformation. Clustering methods detect bot networks and groups of accounts with similar texts. The use of DBSCAN or other algorithms helps identify propaganda centers. The proposed solution implements a module that combines TF-IDF statistical features with Granite contextual embeddings. This hybrid approach to vectorization makes it possible to take into account both the surface frequency characteristics of the text and its semantic aspects, which significantly improves the efficiency of classification. The functionality of the class includes

the entire process of processing text data: from loading and cleaning the corpus to vectorization, training the logistic regression model, visualization of the feature space and generation of a classification report. The model parameters are selected taking into account the possible imbalance of classes, which ensures the stability of the quality of predictions. The mechanism for analyzing the importance of TF-IDF features provides additional transparency of the results. The assessment of the model's effectiveness by the main metrics showed satisfactory results: accuracy – 0.82, F1.3 – about 0.8, ROC-AUC – 0.86. The differences found in the lexical patterns for the "fake" and "true" classes confirm the model's ability to distinguish the content characteristics of texts. The proposed approach can become the basis for the design of scalable systems for monitoring the information space or adaptation to other text classification tasks.

3. Based on our analysis of the results of identifying sources and networks of disinformation distribution in cyberspace, a method has been devised to identify ways of disinformation distribution in cyberspace by identifying stylistically similar content. Authors have unique language patterns that can be identified. Frequency analysis of words and symbols makes it possible to find unique features of authors. TF-IDF and clustering help combine texts of the same author into groups. If accounts have the same style, they may belong to the same source. N-gram analysis detects template phrases of bots. POS tags help find a repeating sentence structure. The length of bot texts is often the same. Temporal analysis makes it possible to detect timer bots. Graph analysis shows connections between bots and template content. The graph makes it possible to detect networks of accounts that use similar texts. N-gram analysis detects common phrases between bots and people. Stylometry helps find authors with similar styles. Graph analysis of betweenness reveals the most influential sources of fakes. If accounts have the same style, similar phrases, and related texts, they can work together. If accounts post at the same time, this can be a coordinated attack. Graph analysis makes it possible to determine the connections between authors through the style and time of posts. Posting frequency analysis detects bots (even time intervals). The combination of stylometry and time analysis indicates the real authors of fakes.

4. Experiments were carried out on the constructed dataset using machine learning algorithms. The results of experimental testing of the proposed methods for identifying sources and ways of disinformation dissemination based on the developed software modules were analyzed. The system is designed to automatically detect the probability of disinformation in texts, messages, and ways of dissemination. It is of great importance for the modern information space, in particular under the conditions of a large volume of unreliable or manipulative information on the Internet. The system helps users to quickly and effectively determine the level of trust in texts distributed from various sources. This is especially relevant for journalists, researchers, media analysts, and ordinary users of social networks. The ability to automate text analysis based on comparison with a database of trusted and unreliable sources makes this tool useful for everyone who seeks to obtain reliable information. The system combines modern NLP technologies, machine learning algorithms, and effective search, which makes it possible to quickly analyze texts in the Ukrainian language. The use of the devised methods makes it possible to correctly work with texts in several languages, which expands the scope of their application.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The research was carried out with the grant support from the National Research Foundation of Ukraine, project registration number 33/0012 dated 3/03/2025 (2023.04/0012) "Design of an information system for automatic detection of sources of disinformation and inauthentic behavior of chat users" under the contest "Science for Strengthening Ukraine's Defense Capability".

Data availability

The data will be provided upon reasonable request.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Acknowledgments

The research was carried out with the grant support from the National Research Foundation of Ukraine, project registration number 33/0012 dated 3/03/2025 (2023.04/0012) "Design of an information system for automatic detection of sources of disinformation and inauthentic behavior of chat users" under the contest "Science for Strengthening Ukraine's Defense Capability".

References

1. Kaliyar, R. K., Goswami, A., Narang, P., Sinha, S. (2020). FNDNet – A deep convolutional neural network for fake news detection. Cognitive Systems Research, 61, 32–44. https://doi.org/10.1016/j.cogsys.2019.12.005
2. Sahoo, S. R., Gupta, B. B. (2021). Multiple features based approach for automatic fake news detection on social networks using deep learning. Applied Soft Computing, 100, 106983. https://doi.org/10.1016/j.asoc.2020.106983
3. Xue, J., Wang, Y., Tian, Y., Li, Y., Shi, L., Wei, L. (2021). Detecting fake news by exploring the consistency of multimodal data. Information Processing & Management, 58 (5), 102610. https://doi.org/10.1016/j.ipm.2021.102610
4. Wang, X., Xie, H., Ji, S., Liu, L., Huang, D. (2023). Blockchain-based fake news traceability and verification mechanism. Heliyon, 9 (7), e17084. https://doi.org/10.1016/j.heliyon.2023.e17084

5. Martín, A., Huertas-Tato, J., Huertas-García, Á., Villar-Rodríguez, G., Camacho, D. (2022). FacTeR-Check: Semi-automated fact-checking through semantic similarity and natural language inference. Knowledge-Based Systems, 251, 109265. https://doi.org/10.1016/j.knosys.2022.109265
6. Liu, X., Qi, L., Wang, L., Metzger, M. J. (2023). Checking the Fact-Checkers: The Role of Source Type, Perceived Credibility, and Individual Differences in Fact-Checking Effectiveness. Communication Research. https://doi.org/10.1177/00936502231206419
7. Guo, Z., Schlichtkrull, M., Vlachos, A. (2022). A Survey on Automated Fact-Checking. Transactions of the Association for Computational Linguistics, 10, 178–206. https://doi.org/10.1162/tacl_a_00454
8. Shahbazi, Z., Byun, Y.-C. (2021). Fake Media Detection Based on Natural Language Processing and Blockchain Approaches. IEEE Access, 9, 128442–128453. https://doi.org/10.1109/access.2021.3112607
9. Elzayady, H., S. Mohamed, M., M. Badran, K., I. Salama, G. (2022). Detecting Arabic textual threats in social media using artificial intelligence: An overview. Indonesian Journal of Electrical Engineering and Computer Science, 25(3), 1712. https://doi.org/10.11591/ijeecs.v25.i3.pp1712-1722
10. Saquete, E., Tomás, D., Moreda, P., Martínez-Barco, P., Palomar, M. (2020). Fighting post-truth using natural language processing: A review and open challenges. Expert Systems with Applications, 141, 112943. https://doi.org/10.1016/j.eswa.2019.112943
11. Shu, K., Sliva, A., Wang, S., Tang, J., Liu, H. (2017). Fake News Detection on Social Media. ACM SIGKDD Explorations Newsletter, 19 (1), 22–36. https://doi.org/10.1145/3137597.3137600
12. Nguyen, V. H., Sugiyama, K., Nakov, P., Kan, M. Y. (2022). FANG: Leveraging social context for fake news detection using graph representation. Communications of the ACM, 65 (4), 124–132. https://doi.org/10.1145/3517214
13. Vosoughi, S., Roy, D., Aral, S. (2018). The spread of true and false news online. Science, 359 (6380), 1146–1151. https://doi.org/10.1126/science.aap9559
14. Pennycook, G., Bear, A., Collins, E. T., Rand, D. G. (2020). The Implied Truth Effect: Attaching Warnings to a Subset of Fake News Headlines Increases Perceived Accuracy of Headlines Without Warnings. Management Science, 66 (11), 4944–4957. https://doi.org/10.1287/mnsc.2019.3478
15. Zhou, X., Zafarani, R. (2020). A survey of fake news: Fundamental theories, detection methods, and opportunities. ACM Computing Surveys (CSUR), 53 (5), 1–40. https://doi.org/10.1145/3395046
16. Shu, K., Bernard, H. R., Liu, H.; Agarwal, N., Dokoohaki, N., Tokdemir, S. (Eds.) (2019). Studying Fake News via Network Analysis: Detection and Mitigation. Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining. Cham: Springer, 43–65. https://doi.org/10.1007/978-3-319-94105-9_3
17. Thorne, J., Vlachos, A. (2018). Automated fact checking: Task formulations, methods and future directions. Proceedings of the 27th International Conference on Computational Linguistics, 3346–3359. Available at: https://aclanthology.org/C18-1283/
18. Hanselowski, A., Zhang, H., Li, Z., Sorokin, D., Schiller, B., Schulz, C., Gurevych, I. (2018). UKP-Athene: Multi-Sentence Textual Entailment for Claim Verification. Proceedings of the First Workshop on Fact Extraction and VERification (FEVER), 103–108. https://doi.org/10.18653/v1/w18-5516
19. Vysotska, V., Przystupa, K., Kulikov, Y., Chyrun, S., Ushenko, Y., Hu, Z., Uhryn, D. (2025). Recognizing Fakes, Propaganda and Disinformation in Ukrainian Content based on NLP and Machine-learning Technology. International Journal of Computer Network and Information Security, 17 (1), 92–127. https://doi.org/10.5815/ijcnis.2025.01.08
20. Vysotska, V., Chyrun, L., Chyrun, S., Holets, I. (2024). Information technology for identifying disinformation sources and inauthentic chat users' behaviours based on machine learning. CEUR Workshop Proceedings. Available at: https://ceur-ws.org/Vol-3723/paper24.pdf
21. Martseniuk, M., Kozachok, V., Bohdanov, O., Iosifov, I., Brzhevska, Z. (2023). Analysis of methods for detecting misinformation in social networks using machine learning. Cybersecurity: Education, Science, Technique, 2 (22), 148–155. https://doi.org/10.28925/2663-4023.2023.22.148155
22. Berrondo-Otermin, M., Sarasa-Cabezuelo, A. (2023). Application of Artificial Intelligence Techniques to Detect Fake News: A Review. Electronics, 12 (24), 5041. https://doi.org/10.3390/electronics12245041
23. Saeidnia, H. R., Hosseini, E., Lund, B., Tehrani, M. A., Zaker, S., Molaei, S. (2025). Artificial intelligence in the battle against disinformation and misinformation: a systematic review of challenges and approaches. Knowledge and Information Systems, 67 (4), 3139–3158. https://doi.org/10.1007/s10115-024-02337-7
24. Vysotska, V., Przystupa, K., Chyrun, L., Vladov, S., Ushenko, Y., Uhryn, D., Hu, Z. (2024). Disinformation, Fakes and Propaganda Identifying Methods in Online Messages Based on NLP and Machine Learning Methods. International Journal of Computer Network and Information Security, 16 (5), 57–85. https://doi.org/10.5815/ijcnis.2024.05.06
25. Vysotska, V., Nazarkevych, M., Vladov, S., Lozynska, O., Markiv, O., Romanchuk, R., Danylyk, V. (2024). Devising a method for detecting information threats in the Ukrainian cyber space based on machine learning. Eastern-European Journal of Enterprise Technologies, 6 (2 (132)), 36–48. https://doi.org/10.15587/1729-4061.2024.317456