I. J. Computer Network and Information Security, **2025, 5, 114-165**
Published Online on October 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2025.05.08

Smart Tool for Identifying Misinformation Spread Sources and Routes in Social Networks Based on NLP and Machine Learning

Victoria Vysotska
Information Systems and Networks Department, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: Victoria.A.Vysotska@lpnu.ua
ORCID iD: https://orcid.org/0000-0001-6417-3689

Sofiia Popp
Department of Information Systems and Networks, Institute of Computer Sciences and Information Technologies, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: sofiia.popp.sa.2022@lpnu.ua
ORCID iD: https://orcid.org/0009-0001-2453-6542

Viktoriia Bulatova
Department of Information Systems and Networks, Institute of Computer Sciences and Information Technologies, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: viktoriia.bulatova.sa.2022@lpnu.ua
ORCID iD: https://orcid.org/0009-0008-2692-3571

Zhengbing Hu
School of Computer Science, Hubei University of Technology, Wuhan, China
E-mail: drzbhu@gmail.com
ORCID iD: https://orcid.org/0000-0002-6140-3351

Yuriy Ushenko*
Department of Computer Science of the Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58012, Ukraine
E-mail: y.ushenko@chnu.edu.ua
ORCID iD: https://orcid.org/0000-0003-1767-1882
*Corresponding author

Dmytro Uhryn
Department of Computer Science of the Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58012, Ukraine
E-mail: d.ugryn@chnu.edu.ua
ORCID iD: https://orcid.org/ 0000-0003-4858-4511

Received: 18 May 2025; Revised: 09 July 2025; Accepted: 27 August 2025; Published: 08 October 2025

Abstract: This article presents a method for detecting disinformation in news texts based on a combination of classic machine learning algorithms and deep learning models. The proposed approach was tested on the corpus of Ukrainian- and English-language news with the "fake/truth" classes marked. Before modelling, detailed data pre-processing was performed: deletion of duplicates, cleaning of HTML tags, links and special characters, normalisation of texts, unification of labels, class balancing, and tokenisation. A hybrid approach was used for vectorisation: frequency features (TF-IDF) were combined with contextual vector representations based on the IBM Granite multilingual model. Logistic regression is chosen as a classifier, which allows a balance to be achieved between quality and interpretation of results. Standard metrics are used to assess performance, such as Accuracy, Precision, Recall, F1-score, and ROC-AUC. According to the results of experiments, the model showed an Accuracy in the range of 0.91–0.93, a Precision of 0.89, a Recall of 0.92, an F1-score of 0.90, as well as an ROC-AUC over 0.94. The obtained values demonstrate the balanced ability of the system not only to accurately classify news, but also to minimise false positives, which is especially important in the conditions of information warfare. Priority is given to Recall's high scores, as the omission of fake messages can have critical consequences for information security. Thus, the proposed approach makes a scientific contribution to the field of

automated disinformation detection by combining transparent and reproducible data processing with a hybrid text representation. The uniqueness of the study lies in the adaptation of NLP and machine learning methods to the Ukrainian-language information space and the context of modern hybrid warfare, which allows you to effectively identify the sources and routes of spreading fake news.

Index Terms: Fake News, Machine Learning, Ukrainian-language Texts, Telegram, TF-IDF, Contextual Embeddings, IBM Granite, Logistic Regression, NLP, Disinformation, Text Classification, Information Security.

1. Introduction

With the beginning of the full-scale invasion of Ukraine in 2022, the information dimension has become strategically crucial. It has become one of the main arenas of modern hybrid confrontation. News reports have become not only a channel for informing the population, but also a means of influencing public opinion, mobilising public support and coordinating actions in crisis conditions. Against the backdrop of intensified hostilities, information counteraction has significantly intensified, in particular in the form of disinformation campaigns and the spread of fake news that can discredit official sources, destabilise public sentiment and weaken the stability of the state. Disinformation, manipulative news, and information stuffing have become key tools of influence that can undermine trust in official sources, spread panic, discredit state institutions, and create a distorted picture of reality. This problem is especially relevant in the digital environment, where content is distributed at ultra-high speed, and Telegram, as one of the most popular platforms in Ukraine, acts as both a source of operational information and a channel for spreading fake messages.

In response to these challenges, the need for reliable technical solutions for the automatic detection of inaccurate information has increased. Systems capable of effectively identifying fake messages play an important role in ensuring information security, countering enemy propaganda, and supporting critical thinking in a hybrid war. In the process of work, a machine learning model has been developed that automatically classifies news texts related to the war into reliable and unreliable. The proposed solution implements a complete cycle of text data processing: from initial normalisation to the construction of vector representations, model training, and output of results in a visualised form.

The basis of the chosen approach is the combination of TF-IDF vectorisation with contextual embedding obtained from the IBM Granite multilingual model. Such a hybrid allows you to take into account both the frequency and semantic characteristics of the text, which is critical for analysing messages that may contain manipulative or propagandistic elements. Logistic regression is used as a classifier – a model that provides a balance between high efficiency for binary classification problems and ease of interpretation of results. It allows not only to get accurate predictions, but also to analyse the influence of individual tokens on the model's decisions.

The purpose of this work is to develop a system based on machine learning methods for automatic recognition of fake and reliable Ukrainian-language news received from Telegram channels. The proposed solution involves the creation of a full-fledged computational pipeline covering the stages of word processing, building vector features, training the classification model and further visualisation of the results.

For practical integration, the model is saved in pickle format, which allows classification to be carried out without the need for repeated training. The system integrates with the Telegram API to collect messages from open sources, ensures that they are processed in real time, and displays the results using the Gradio interface. This approach allows you to quickly detect fake messages and can be used in information environment monitoring systems.

In this context, the development of tools for automatic detection of disinformation has become an important task, both in terms of information security and in the framework of maintaining societal resilience. The goal of this project is to create a system for classifying messages from Telegram channels into fake and non-fake. Within the framework of the project, a full-fledged pipeline was implemented: from pre-processing of text data to their vectorization using a hybrid approach using a hybrid approach in combination of TF-IDF and IBM Granite embedding, training of the classification model, namely logistic regression, saving the model in pickle format, and ending with integration with the Telegram API and building an interface for interacting with the system.

The system developed within the framework of this project has practical value in two key areas:

- Accelerated generation of labelled datasets for model training. The system can serve as an effective tool for pre-labelling large volumes of messages from Telegram channels. It allows you to quickly create initially marked datasets for training and validation of machine learning models, in particular those specialising in fake news detection, thematic classification, or text sentiment analysis.
- Automated initial screening of information content. The system is capable of carrying out basic automated verification and classification of messages coming from open sources on Telegram. It is handy in tasks of operational monitoring of the information space, where it is necessary to quickly identify high-risk content (for example, potentially disinformation materials, spam or malicious messages) for further detailed analysis by a human expert.

Despite significant progress in research on the detection of fake news, most of the existing systems are focused mainly on English-language data, which significantly limits their effectiveness in the Ukrainian-language information space. Most modern models use either classical frequency methods (TF-IDF, n-gram models), which provide a basic level of Accuracy, or exclusively transformer architectures, which require large amounts of data and significant computing resources. Against this background, the approach proposed in the paper is distinguished by the fact that it combines the interpretation and compactness of classical methods with deep contextual embeddings, achieving a balance between efficiency, transparency and practical applicability.

The purpose of the study is to develop and validate an automated system for classifying news in Ukrainian from Telegram channels, dividing them into reliable and fake. The difference from existing solutions is the use of a hybrid approach to text vectorisation, adaptation of the model to Ukrainian-language data, and integration into a full-fledged software package with real-time capabilities. Within the framework of the work, the key limitations of the proposed technology are also considered: dependence on data quality and the need to regularly update the training sample, taking into account the emergence of new types of disinformation, as well as the sensitivity of transformer embeddings to long texts, which is partially compensated by the segmentation mechanism.

Given the urgency of the problem, the proposed solution is aimed not only at improving the Accuracy of classification, but also at creating a tool suitable for practical use in the conditions of information warfare, where the timely detection of fakes is a factor in the information security of the state.

The proposed approach is marked by a complex novelty, which manifests itself in several aspects. Firstly, for the first time, a hybrid vectorisation model of Ukrainian-language texts has been implemented, which combines TF-IDF frequency features with contextual embeddings of the IBM Granite multilingual model. This combination allows you to simultaneously take into account both statistical patterns in the use of words and their semantic relationships, which is critical for identifying hidden manipulative constructions in messages. Secondly, a specialised computing pipeline is proposed, covering all stages – from data collection via Telegram API and its pre-processing to classification and interactive visualisation of results in real time through the Gradio interface. Thirdly, in the process of training the model, an optimised strategy for adjusting the hyperparameters of logistic regression was applied, which provides increased resistance to class imbalance and makes it possible to interpret the influence of individual language markers on the classifier's decisions. Unlike existing solutions focused mainly on English-language or universal corpora, this system is adapted to the Ukrainian-language information space and tested on real data, which makes it unique in the context of countering disinformation in the domestic segment of social networks.

The main contribution of this study is the creation and validation of a comprehensive system for automatic detection of disinformation in Ukrainian-language news from Telegram channels, which combines:

- Hybrid approach to text vectorisation – integration of TF-IDF frequency attributes with contextual embeddings of the IBM Granite multilingual model, which provides simultaneous accounting of statistical and semantic characteristics of messages.
- Interpreted and effective classifier – the use of logistic regression with optimised hyperparameters, which allows you to maintain transparency in decision-making and high sensitivity to the detection of fake news.
- Full-fledged computing pipeline – from data collection via Telegram API and pre-processing to interactive classification and visualisation of results in real-time using Gradio.
- Adaptation to the Ukrainian-language information space – building and balancing a corpus of Ukrainian-language data, which previously remained limitedly represented in research on disinformation detection.

Together, these elements provide a practically-oriented solution that differs from existing systems in that it is both accurately interpreted and resource-efficient, and adapted to the specifics of the Ukrainian segment of social networks.

The work is structured as follows. Chapter 2 provides an overview of existing research and approaches to detecting disinformation. Chapter 3 describes methods and tools, including a hybrid approach to text vectorisation and the choice of a classification model. Chapter 4 is devoted to experimental verification, analysis of the results obtained, and discussion of the system's limitations. Chapter 5 formulates conclusions, practical recommendations and prospects for further research.

2. Related Works

The task of detecting fake news arose even before the advent of modern machine learning algorithms. In the early stages, countering disinformation was carried out mainly manually: a number of fact-checking platforms were created (in particular, PolitiFact, Snopes, FullFact), where specialised teams carried out content verification, analysis of sources, and comparison of information with reliable data. While these initiatives provided high accuracy and reliability, the development of digital communications revealed their limitations in scaling, in particular due to the inability to process large amounts of content in real time. As indicated in a comprehensive review of approaches to detecting fake news using machine learning, none of the existing fact-checking systems is able to quickly respond to the pace of the spread of false information, especially in the context of the dynamics of social networks. It necessitated the creation of automated solutions based on a combination of natural language processing, model training algorithms, and deep neural networks

capable of efficiently scaling and detecting hidden patterns in text data streams [1].

A review of the scientific literature shows that modern methods for identifying fake news can be classified into three main areas: content-oriented, contextual, and hybrid approaches. The studies separately address the benefits and challenges of each of these categories. More and more attention is paid not only to the analysis of the text of the message itself, but also to additional factors such as the source of the publication, the trajectories of distribution on the network, and the peculiarities of user interaction with content.

In modern review works, there is a tendency to abandon the traditional manual engineering of features in favour of more complex models, in particular neural network architectures based on transformers. Such models are able to integrate both textual and contextual parameters, automatically detecting relevant patterns without the need for manual intervention. At the same time, the problem of input data quality and model transparency remains relevant. These aspects are central to the works on the practical application or methodological improvement of fake detection systems.

The first algorithmic approaches to automating the detection process focused exclusively on the text component. The analysis focused on the lexical, grammatical, stylistic and structural characteristics of the messages. For vectorisation, basic methods of text representation were used, in particular, Bag-of-Words, TF-IDF, as well as more advanced semantic models, such as Word2Vec and GloVe, which made it possible to capture the contextual similarity between words [1].

A number of empirical studies consider the performance of various machine learning algorithms in the problem of classifying fake news. For example, in a paper performed by a Bali-led team in India (LBRCE), SVM, Naive Bayes, Logistic Regression, and XGBoost models were tested using TF-IDF to represent text data. The XGBoost model showed the highest accuracy, which confirms its effectiveness when working with sparse and high-dimensional features [2].

The study, which was presented by Potthast and colleagues at the ACL conference, examined the effect of stylistic parameters of the text on the classification of hyperpartisan, satirical and false content. The meta-learning model used in the work showed that stylistic features alone do not provide sufficient accuracy in detecting fakes. However, they can be helpful for other types of content [3].

Another notable contribution belongs to Pérez-Rosas and a team from the University of Michigan, who formed a unique dataset of paired news, fake and authentic. Based on linguistic characteristics such as n-gram models, readability indices, and punctuation frequency, an SVM classifier was trained. Experiments have shown that these features can act as informative indicators of false content [4].

Unlike approaches that focus solely on the content of the text, context-oriented models take into account external metadata related to the dissemination of information and user interaction with it. In such approaches, analytics goes beyond purely linguistic characteristics. It involves data on activity in social networks, in particular, the number of subscribers, the frequency of publications, and the level of engagement (comments, likes, reposts). Such indicators make it possible to identify anomalous distribution patterns, which may indicate that the message is unreliable.

A paper by Tacchini et al. (University of California, Santa Cruz) considered the possibility of classifying fake news based on the reactions of Facebook users. The authors proved that even a limited amount of behavioural data, such as likes and comments, can serve as an effective indicator of the veracity of a post [5].

Another example is a study by Yu and colleagues at Penn State University, which developed an architecture based on CNN-GRU to track and simulate the ways in which information spreads on Twitter and Weibo. Such a system allows you to analyse the pace and structure of the distribution of publications, and thereby detect fake news even before it is widely covered [6]. To increase the efficiency of detecting fake news, researchers are increasingly turning to multifactorial approaches that combine different types of features, from textual to visual and contextual. Such hybrid representations provide a wider range of informative characteristics, allowing the model to analyse both the content and the conditions of message dissemination in more depth. It significantly increases the accuracy of classification, reduces the risk of errors, and takes into account the interaction between text, image, and behavioural patterns.

An illustration of the successful application of this approach is the SpotFake model, developed at the National Institute of Technology (Hamirpur, India). It combines the BERT language model for word processing with the VGG-19 convolutional neural network for the analysis of accompanying images. On multimodal datasets, this model showed more than 80% accuracy, especially effective in detecting fakes with visual content [7]. Within the framework of this work, another form of hybridisation is proposed: a combination of contextual embedding obtained using the Granite multilingual model with frequency TF-IDF features. This approach allows you to take into account the semantics of the text and maintain the simplicity of interpretation at the same time. In addition, a complete computational pipeline has been implemented – from preliminary data analysis and text cleaning to model training, visualisation of results, and analysis of essential features. It not only increases the reliability of the classification but also allows for a deeper understanding of the structure and nature of disinformation content circulating in connection with the war in Ukraine. Thus, the implemented solution combines the advantages of both traditional statistical methods and modern approaches to natural language processing, adapted to the Ukrainian-speaking environment.

Research on misinformation detection in low-resource languages and conflict settings has highlighted specific challenges such as scarcity of labelled data, cultural and linguistic diversity, and the rapid evolution of adversarial tactics.

Surveys and overviews. Several surveys synthesise approaches for low-resource misinformation detection. They emphasise that classical methods (e.g., TF-IDF with linear classifiers) remain competitive when combined with multilingual embeddings, especially in scenarios with limited labelled corpora. Transfer learning and hybrid representations are consistently recommended as practical strategies for under-resourced languages [8-9].

Low-resource corpora and multilingual transfer. A number of works have constructed datasets for Indian languages (Hindi, Gujarati, Marathi, Telugu), demonstrating that cross-lingual transfer from high-resource languages significantly improves detection performance [10-11]. Similarly, hybrid approaches combining real and synthetic data have been explored for Hindi [12].

Conflict-related misinformation. In crisis contexts such as Syria or Ukraine, studies report that domain-shift effects strongly limit model generalisation: systems trained on one wave of disinformation narratives often underperform on newer campaigns, underlining the necessity of continuous retraining and adaptation [13-14].

Robustness and adversarial evaluation. Recent evaluation campaigns such as CLEF CheckThat! include adversarial or robustness tasks (e.g., paraphrase, obfuscation, typos) to measure resilience against evasion tactics. These studies underscore the need for adversarial testing and augmentation in practical deployment [15-16].

Multimodality and explainability. Low-resource and conflict-aware research has also emphasised the role of multimodal signals (image–text correlations, metadata provenance) and explainability techniques (attention visualisation, feature importance) to improve trust and support human fact-checkers [17-18].

Taken together, this literature supports three main principles: (1) hybrid frequency and contextual representations are effective in low-resource environments, (2) baseline and robustness evaluation are necessary for fair comparisons, and (3) continuous data collection and retraining are crucial to maintain effectiveness against adaptive adversaries.

Table 1. Detecting fake news in low-resource languages or conflict zones

#	Direction	Features	Explanation
1	Reviews and generalisations	Low-resource overview (mono and multilingual settings)	systematises datasets, methods (classical ML, transformers, multilingual approaches), and problems of data scarcity, cultural context, and real-world deployment. Recommendations include multilingual models and adaptation to specific communities of speakers [19]
		Multilingual detection of fakes	offers a practical MFND approach for data-scarce languages; shows that properly designed multilingual representations can compensate for the lack of labelled corpora [20]
2	Arabic language and Middle Eastern context (including conflicts)	Overview of Arabic-language datasets	29 corpora, problems of small sizes, thematic/genre heterogeneity and lack of multimodal examples; emphasis on the need for more complete benchmarks [21].
		Arabic detection on BERT-like models (AraBERT, QaribBERT, etc.)	systematic comparison of transformers, high Accuracy on existing sets, valuable insights on errors (FN/FP) [22].
		Syrian War	meta-training and classic ML for automatic detection of fakes around the conflict; demonstrates how models are sensitive to narrative evolution and domain bias [23]
3	South Asia/Indian Low-Resource Languages	New Large Hybrid Enclosures for Hindi (Real + Synthetic Examples)	The work closes the shortage of open, linguistically annotated sets suitable for comparing methods [12].
		Multilingual Indian Kit (GU/HI/MR/TE) to detect fakes	promotes comparative experiments in low-resource languages [24].
		Hinglish/HI initiatives and competitions	confirm the demand for methods for code mix and regional languages [25].
4	Dravidian languages/ multimodality and explainability	Tamil (low-resource) multimodal system with explanations	The combination of text and images and "reasoning-based explainability" for transparency of decisions is essential for trust in sensitive environments [26].
5	Wartime campaigns and resilience assessments	CLEF CheckThat! Laboratory	multilingual tasks (including Arabic/Bulgarian) that drive benchmarking forward; In 2024, a separate task on robustness for "adversarial" examples, relevant for enemy attacks and evasion tactics [15, 27-28].
		Context of Ukraine	journalistic/analytical reviews describe real examples of information and psychological operations (botnets, fake SMS, deepfakes), emphasising the need for models that can withstand new tactics [29-34].

3. Methods and Tools

As part of this work, the task of identifying fake news related to the full-scale war in Ukraine was solved. The problem of fake information became especially acute after the start of the active phase of the war in 2022, when the news space was filled with materials that can be used as a tool for information influence. An automated classification system allows you to quickly detect false reports, which can be critical in wartime, where reliable information is a security issue.

The problem that has been solved is a binary classification problem in the context of machine learning. Each text message must be assigned to one of two classes: fake or authentic. Accordingly, the target variable (label) is categorical with two classes. The input is text news messages collected from open sources that have been manually labelled. The volume of the dataset allows you to use classical approaches to text processing and the construction of interpreted models.

We will describe the study with an emphasis on the mathematical formulation of the model, using the standard apparatus for text classification problems.

Mathematical description of the study

- Problem statement. Let's have a corpus of texts $D = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is the text of the news, and $y_i \in \{0,1\}$ is the binary label (0 is reliable, 1 is fake news). The task of classification is to construct a function $f: X \to \{0,1\}$, which for an arbitrary X message defines its class.
- Text vectorisation. For each message x_i, two types of signs are formed:

– TF-IDF submission:

$$\text{TF-IDF}(t,d) = \text{tf}(t,d) \cdot \log\frac{N}{df(t)},$$

where $tf(t,d)$ – is the frequency of the t term in the document d, $df(t)$ – is the number of documents containing the term t, N is the total number of records in the case. The resulting vector representation is $v_i^{tfidf} \in R^m$, where m is the number of features after the dictionary limit.

– IBM Granite contextual embeddings. The Granite model maps text into semantic space:

$$v_i^{\text{granite}} = \text{Granite}(x_i) \in R^k,$$

where $k = 384$ is the dimension of embedding.

– Hybrid vector. Signs are united by concatenation: $v_i = \left[v_i^{\text{tfidf}} \mid v_i^{\text{granite}}\right] \in R^{m+k}$.

- Classification model. Logistic regression is used. For each vector of v_i features, the probability of belonging to the "fake" class is defined as $P(y_i = 1 \mid v_i) = \sigma(w^\top v_i + b)$, where $w \in R^{m+k}$ – is the weight vector, $b \in R$ – offset, $\sigma(z) = \frac{1}{1+e^{-z}}$ – sigmoid. The final decision is made according to the rule:

$$\widehat{y_i} = \begin{cases} 1, if \;\; P(y\_i = 1 \mid v\_i) \geq \tau, \\ 0, \quad \text{otherwise} \end{cases}$$

where τ – is the threshold (usually $\tau = 0.5$).

- Loss function and optimisation. The model is trained by minimising binary cross-entropy:

$$\mathcal{L}(w,b) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log P(y_i = 1 \mid v_i) + (1 - y_i)\log\left(1 - P(y_i = 1 \mid v_i)\right)\right].$$

To avoid overtraining, regularization (L1/L2) is added:

$$\mathcal{L}_{reg}(w,b) = \mathcal{L}(w,b) + \lambda|w|_p,$$

where $p \in \{1,2\}, \lambda > 0$ is the regularization coefficient.

- Quality assessment. The quality of the model is checked according to standard metrics, such as Accuracy, Precision, Recall, and F_β-measure (in the study $\beta = 1.3$):

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}, \text{Precision} = \frac{TP}{TP+FP}, \text{Recall} = \frac{TP}{TP+FN}, F_\beta = \frac{(1+\beta^2)\cdot\text{Precision}\cdot\text{Recall}}{\beta^2\cdot\text{Precision}+\text{Recall}}.$$

Thus, mathematically, the study boils down to the problem of binary classification of texts, where the key innovation is the use of a hybrid vector space [TF-IDF | Granite] and an interpreted classifier (logistic regression) optimised for Ukrainian-language content.

Mathematical Pipeline of the Proposed System

$$\textbf{TextMessage } x \xrightarrow{\text{Pre-processing}} \tilde{x} \xrightarrow{\text{Vectorisation:TF-IDF+Granite}} v = [v^{\text{tfidf}} \mid v^{\text{granite}}] \xrightarrow[\text{Logistic Regression}]{} \hat{y} \in \{0,1\},$$

where:

- Pre-processing (lowercasing, removing URLs, HTML tags, emojis, duplicates, etc.): $\tilde{x} = \text{clean}(x)$.
- Hybrid Feature Vector: $v^{\text{tfidf}} \in R^m, \quad v^{\text{granite}} \in R^k, \quad v = [v^{\text{tfidf}} \mid v^{\text{granite}}] \in R^{m+k}$.
- Logistic Regression Classifier: $P(y = 1 \mid v) = \sigma(w^\top v + b), \quad \hat{y} = I[P(y = 1 \mid v) \geq \tau]$, with $\sigma(z) = \frac{1}{1+e^{-z}}, \tau = 0.5$.

This scheme briefly and clearly formalises the entire conveyor.
The main stages of building a model are:

- data set cleaning and text preprocessing (preprocessing);
- text vectorisation (TF-IDF, Granite contextual embedding);
- construction of a hybrid vector representation;
- training in the classification model of logistic regression;
- assessment of the quality of the model according to the relevant metrics.

For text vectorisation, a hybrid approach is used, which combines the classic TF-IDF model (which is effective in identifying keywords) with contextual embedding obtained using the IBM Granite multilingual model. TF-IDF (Term Frequency – Inverse Document Frequency): A classic method of converting texts into numerical vectors, which takes into account how often a word occurs in a document and how rare it is throughout the corpus. IBM Granite: a multilingual transformer model for constructing contextual vectors (embeddings) that reflect the meaning of a word depending on the environment. This combination allows the model not only to analyse frequency patterns in texts, but also to take into account their context and semantic relationships.

Logistic regression is chosen as the main classification model. This choice is due to several factors, namely:

- Interpretation of results (the ability to assess the importance of each feature);
- Stability on small datasets;
- Low sensitivity to excessive complications of the model;
- The possibility of effective learning on a hybrid vector representation.

The following metrics were chosen to assess the quality of the model:

- Accuracy – for a general assessment of the correctness of the classification;
- Precision, recall and F-beta – with a focus on recall and F-beta (β = 1.3), since in the context of fake news it is essential to detect all fakes, even if it leads to some false positives, because the omission of fake news can have serious consequences for information security;
- ROC-AUC – for a general analysis of the model's ability to distinguish classes at different thresholds;
- Confusion matrix – to visually represent the results of the classification and understand the balance between different types of errors (false positive/false negative);
- Feature importance – for analysing the impact of individual words (from the TF-IDF vectorizer) on the model's decision, which allows you to interpret the results and identify the most characteristic features of fake news.

A separate task is to create a system for automatic classification of news, which will provide a complete cycle of processing messages from open Telegram channels. This system should use the previously described steps of model construction – in particular, text preprocessing, hybrid vectorisation and logistic regression – in a ready-made form. For this purpose, it is assumed to load the stored trained model, which allows the classification of new messages without retraining. The Telegram API will be used to automatically collect news content from Telegram, after which the messages will undergo standard preprocessing and vectorisation. On this basis, they will be transmitted to the input of the model to predict whether the news is fake. To ensure accessible interaction with the system, a web interface based on the Gradio library is planned to be developed, which will allow you to classify news in real time. This approach will make it possible to apply the model in practical conditions and quickly detect fake messages in a dynamic information environment. The creation of a comprehensive automated system for detecting fake news, based on machine learning methods and providing a complete cycle of information processing, requires the development and integration of a number of specialised modules – from data collection to displaying results. Gradio: Python library for creating simple web interfaces, allowing you to test the model in real time without complex integrations. The main stages of development are:

- Definition of the classification problem. It is necessary to formulate and implement a binary classification mechanism, where each text message should be assigned to one of two classes – "fake" or "true";
- Implementation of the data collection and pre-processing module. Create functionality for automated collection of text news messages from open Telegram channels using appropriate APIs;
- Selection of methods for cleaning and preprocessing the collected data (preprocessing), including text normalisation and removal of redundant elements;

- Implementation of a hybrid vector text representation. To create an approach that combines classical TF-IDF vectorisation with contextual embeddings obtained using the IBM Granite multilingual model to form a comprehensive numerical representation of texts. This combination allows the model not only to analyse frequency patterns in texts, but also to take into account their context and semantic relationships.
- Classification Model Training and Optimisation. Train the Logistic Regression Model as a Basic Classifier. This choice is due to its interpretation, because it is possible to assess the importance of each feature, stability on small datasets, low sensitivity to excessive model complication, and the ability to effectively train on a hybrid vector representation.
- Selection of model quality assessment mechanisms. Define and implement a set of metrics for a comprehensive evaluation of the performance of the developed model, including accuracy, precision, recall and F-beta (β=1.3), ROC-AUC, confusion matrix and feature importance. Particular emphasis will be placed on recall and F-beta, with a focus on recall, since in the context of fake news, it is essential to identify all fakes, even if it leads to some false positives, because omitting fake news can have serious consequences for information security.
- Integration of components into a holistic system. Create and combine a data collection module, a developed classification model, and a user interface into a single, functional pipeline for automated news processing;
- Development of an interface that provides a convenient display and interaction with classification results.
- Ensuring the reproducibility and reliability of the developed system. To build the system in such a way that it is reproducible and able to provide high-quality detection of fake news in the context of constant information flows.

Thus, the formulated task involves not only the development of a model but also a system based on this machine learning model, which ensures high-quality classification of fake news in the context of information warfare.

Hybrid or partially true misinformation is most challenging to detect when the message contains an actual grain but is complemented by twisted details, manipulative wording, or distorted context. In the classical binary classification (true/false), such cases become problematic because the algorithm is forced to "squeeze" complex reality into two categories. It leads to fuzziness and an increase in errors.

To partially solve the problem, several approaches are used in scientific practice:

- Multiclass classification. Instead of two classes ("fake" and "true"), you can distinguish an intermediate category, such as a reliable message, half-truth/manipulative, or fake. It allows you to more accurately describe grey areas where the message is not entirely accurate or false.
- Confidence score. The classifier returns the probability of belonging to a class instead of a "hard" category. For example, the system can show: "65% is the probability of fake, 35% is the truth." This approach makes the user understand that the case is fuzzy and recommends additional peer review.
- Fuzzy logic methods. Instead of discrete classes, a "degree of truth" is used (for example, from 0 to 1). It is closer to human perception when we say that the news is "70% reliable".
- Analysis of argumentative structures and fact-checking. Partially authentic fakes usually contain correct facts, but their interpretation is distorted. The use of external knowledge bases or Natural Language Inference (NLI) models helps to identify contradictions between the statement and verified sources.
- Aggregation of several models. The simultaneous use of "content" features (text, style, vocabulary) and "contextual" (source of publication, speed of distribution, user reaction) increases the chance of detecting manipulations. Hybrid ensemble approaches work better with the "grey area".

Thus, the problem of partially true disinformation can be solved by moving from a rigid binary scheme to a multi-level or probabilistic one, as well as using additional contextual factors and external fact bases.

4. Experiments, Results and Discussion

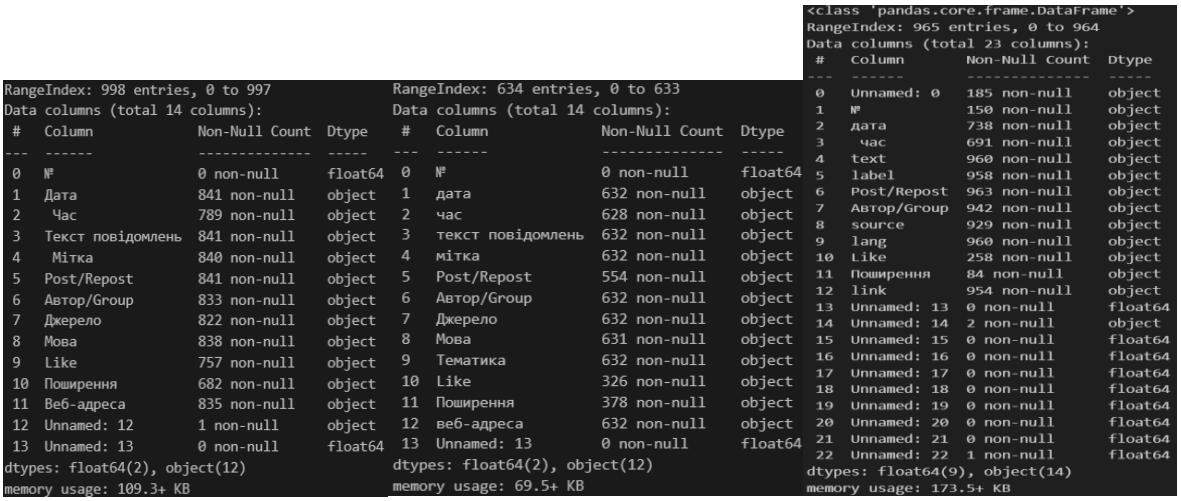
4.1. Description of Input Processing and Machine Learning Models

A. Exploratory Data Analysis (EDA)

The dataset used in this study was specifically constructed to support the development and evaluation of machine learning models for disinformation detection in Ukrainian cyberspace. The collection process was conducted in September–October 2024 and covered the period from February 2022 to October 2024, thereby including content from the full-scale invasion of Ukraine and subsequent information campaigns. The final dataset consists of 13,860 news items and social media messages, of which 5,980 were labelled as fake (43.1%) and 7,910 as authentic (56.9%). Data were sourced from a variety of online platforms: Telegram (41%), Facebook (22%), online news portals (29%), Twitter (6%), Instagram (1%), and VKontakte/Yandex (1%). The relatively small share of Russian platforms reflects their restricted availability in Ukraine due to official state-level blocking. The distribution of authentic and fake items varied across sources; for instance, in Telegram, fake content accounted for 53.1% of messages, whereas in Facebook, the majority (54%) were authentic. The dataset construction process involved students, postgraduate researchers, and lecturers of Lviv Polytechnic National University. Each item was manually reviewed and labelled according to a set of linguistic, stylistic,

and contextual criteria. Fake news was typically identified by the presence of emotionally charged vocabulary (e.g., "shock", "horror", "unbelievable"), simplified or sensationalist syntax, manipulative phrasing (e.g., "it is known that", "without a doubt"), and indications of conspiracy narratives. In addition, source reliability was considered: content originating from outlets or accounts with a history of spreading disinformation was more likely to be labelled as fake. To ensure consistency, duplicate items were removed, and the dataset was balanced using oversampling and undersampling techniques to mitigate class imbalance. It should be noted, however, that the dataset was not compiled by professional fact-checkers or domain experts (e.g., journalists, political scientists), but rather by non-expert contributors within a limited geographical and professional context. As a result, there is potential for subjective bias and inconsistent labelling. This limitation has implications for the robustness and generalizability of the trained models, as the validity of any machine learning system critically depends on the quality of its training data.

The main stage of the project is to conduct exploratory data analysis for a comprehensive understanding of the structure of data sets that form the foundation for the development of a classification model. Before building a machine learning model, a study of the structure of the input dataset was carried out using the method of exploratory data analysis (EDA). Actually, the purpose of this study is to prepare a high-quality and consistent corpus of text data suitable for training the model, along with the application of computational linguistics tools to it. Specifically, the purpose of this phase was to identify significant patterns, potential anomalies, missing values, class imbalances, and other characteristics that may influence the choice of preprocessing methods and subsequent modelling. Before concatenation, a preliminary analysis of each of the datasets was carried out. Initial information about the datasets shown in Fig. 1.



```
RangeIndex: 998 entries, 0 to 997
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   №                 0 non-null      float64
 1   Дата              841 non-null    object
 2    Час              789 non-null    object
 3   Текст повідомлень  841 non-null    object
 4    Мітка            840 non-null    object
 5   Post/Repost       841 non-null    object
 6   Автор/Group       833 non-null    object
 7   Джерело           822 non-null    object
 8   Мова              838 non-null    object
 9   Like              757 non-null    object
 10  Поширення         682 non-null    object
 11  Веб-адреса        835 non-null    object
 12  Unnamed: 12       1 non-null      object
 13  Unnamed: 13       0 non-null      float64
dtypes: float64(2), object(12)
memory usage: 109.3+ KB
```

```
RangeIndex: 634 entries, 0 to 633
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   №                 0 non-null      float64
 1   дата              632 non-null    object
 2   час               628 non-null    object
 3   текст повідомлень  632 non-null    object
 4   мітка             632 non-null    object
 5   Post/Repost       554 non-null    object
 6   Автор/Group       632 non-null    object
 7   Джерело           632 non-null    object
 8   Мова              631 non-null    object
 9   Тематика          632 non-null    object
 10  Like              326 non-null    object
 11  Поширення         378 non-null    object
 12  веб-адреса        632 non-null    object
 13  Unnamed: 13       0 non-null      float64
dtypes: float64(2), object(12)
memory usage: 69.5+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 965 entries, 0 to 964
Data columns (total 23 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    185 non-null    object
 1   №             150 non-null    object
 2   дата          738 non-null    object
 3    час          691 non-null    object
 4   text          960 non-null    object
 5   label         958 non-null    object
 6   Post/Repost   963 non-null    object
 7   Автор/Group   942 non-null    object
 8   source        929 non-null    object
 9   lang          960 non-null    object
 10  Like          258 non-null    object
 11  Поширення     84 non-null     object
 12  link          954 non-null    object
 13  Unnamed: 13   0 non-null      float64
 14  Unnamed: 14   2 non-null      object
 15  Unnamed: 15   0 non-null      float64
 16  Unnamed: 16   0 non-null      float64
 17  Unnamed: 17   0 non-null      float64
 18  Unnamed: 18   0 non-null      float64
 19  Unnamed: 19   0 non-null      float64
 20  Unnamed: 20   0 non-null      float64
 21  Unnamed: 21   0 non-null      float64
 22  Unnamed: 22   1 non-null      float64
dtypes: float64(9), object(14)
memory usage: 173.5+ KB
```

Fig.1. Dataset information: a - the first; b – second, c – third

Within the scope of the analysis, the following was carried out:

- Visualisation of the balance of the input variable "lang" and the output variable, namely "label" – a label that acquires the values of "true" or "false";
- Analysis of variable types (textual, categorical, numeric, dates);
- Checking for missing or duplicate values;
- Estimation of text length, word frequency and basic text preprocessing;
- Assessment of the correspondence of variables in the context of the classification task;
- Detection of missing values and duplicate records;
- Unification of text values of columns;
- Analysis of the imbalance of the target variable.
- Bringing synonymous values of categories into a single form to ensure data consistency.
- Analysis of distributions of categorical features;
- Research of the main parameters of the text, in particular character length, number of words and the number of tokens generated by the tokeniser of the Granite model

The observations obtained as a result of the EDA became the basis for an informed choice regarding:

- Identification of relevant variables for analysis and modelling;
- Feature selection, including "text", "label", "source", "lang", "link";
- Strategies for deleting duplicates by column with text and irrelevant entries;
- Building histograms for characteristics such as text length (in characters), number of words (based on simple space separation), as well as the number of tokens obtained using the tokeniser of the embedding model;
- Strategies for handling missing values and deleting irrelevant records;

- Unification of variants of values in categorical variables;
- Identification of the imbalance of the target variable and features of the distribution of input categories;
- There is a need to balance the sample before building a model.
- The rules for preprocessing texts are formed, taking into account the limitations of the tokeniser and the features of the case.

	0	1	2	3
№	NaN	NaN	NaN	NaN
Дата	20.08.2024	23.09.2024	11.09.2024	12.09.2024
Час	10:06	7:01	17:37	14:01
Текст повідомлень	Оборона ВСУ в Донецкой области осталась без сн...	FLASH COLONEL MACGREGOR: La Contre-Attaque Rus...	?? Wer kontrolliert die Algorithmen, die unser...	Астрологический прогноз для тебя от мольфарки
Мітка	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ
Post/Repost	Post	Post	Post	Post
Автор/Group	Einarr Modig	Psn Animation M?dia	Wissenschaftswelle.de	Магия которая работает
Джерело	Facebook	Facebook	Facebook	Facebook
Мова	Russian	French	German	Russian
Like	5	146	13	332
Поширення	3	12	5	64
Веб-адреса	https://www.facebook.com/permalink.php?story_f...	https://www.facebook.com/watch/?v=826773688331...	https://www.facebook.com/Wissenschaftswelle/po...	https://www.facebook.com/magiarabotaet/videos/...
Unnamed: 12	NaN	NaN	NaN	NaN
Unnamed: 13	NaN	NaN	NaN	NaN

	0	1	2	3
№	NaN	NaN	NaN	NaN
дата	07.09.2025	13.10.2025	19.01.2025	17.07.2025
час	20:59	17:46	05:34	20:41
текст повідомлень	Кожен день цієї війни – це захист нашої незале...	I visited the command post of the Tactical Gro...	Сьогодні під час нашої зустрічі з Президентом ...	All of us in Ukraine deeply appreciate Norway'...
мітка	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
Post/Repost	Post	Post	Post	Post
Автор/Group	Володимир Зеленський	Володимир Зеленський	Володимир Зеленський	Володимир Зеленський
Джерело	Facebook	Facebook	Facebook	Facebook
Мова	Українська	English	Українська	English
Тематика	Військова	Військова	Військова	Військова
Like	9600	17000	9100	24000
Поширення	800	1000	500	1000
веб-адреса	https://www.facebook.com/share/p/1AJKXALSMP/	https://www.facebook.com/share/p/15vHBzxtiY/	https://www.facebook.com/share/v/1DRvKAe6MJ/	https://www.facebook.com/share/p/1FhdtENdZb/
Unnamed: 13	NaN	NaN	NaN	NaN

```
df = pd.read_csv("F:\\ML\\Classifiacation_project\\Dataset_Teachers.csv",
                 skiprows=1,
                 header=0
                 )
df.head(4).T
```

0.0s

	0	1	2	3
Unnamed: 0	NaN	NaN	NaN	NaN
№	NaN	NaN	NaN	NaN
дата	20.08.2024	20.09.2024	20.08.2024	20.08.2024
час	10:06	9:49	23:04	NaN
екст повідомлень	Оборона ВСУ в Донецкой области осталась без сн...	Оборона бандеровцев в Донецкой области осталас...	Після початку військової операції у Курській о...	Оборона ЗСУ на Донбасі залишилася без снарядів...
мітка	ФЕЙК	ФЕЙК	ФЕЙК	ФЕЙК
Post/Repost	Post	Post	Post	Post
Автор/Group	Einarr Modig	Vlad Brigg	Тетяна Бесараб	Медіавектор
Джерело	Facebook	Facebook	Інформатор України	Медіавектор
Мова	Russian	Russian	Ukrainian	Ukrainian
Like	5	13	24	NaN
Поширення	3	NaN	NaN	NaN
веб-адреса	https://www.facebook.com/permalink.php?story_f...	https://www.facebook.com/permalink.php?story_f...	https://informator.ua/uk/pislya-nastupu-na-kur...	https://mediavektor.org/92604-oborona-vsu-v-do...
Unnamed: 13	NaN	NaN	NaN	NaN
Unnamed: 14	NaN	NaN	NaN	NaN
Unnamed: 15	NaN	NaN	NaN	NaN
Unnamed: 16	NaN	NaN	NaN	NaN
Unnamed: 17	NaN	NaN	NaN	NaN
Unnamed: 18	NaN	NaN	NaN	NaN
Unnamed: 19	NaN	NaN	NaN	NaN
Unnamed: 20	NaN	NaN	NaN	NaN
Unnamed: 21	NaN	NaN	NaN	NaN
Unnamed: 22	63.0	NaN	NaN	NaN

Fig.2. The first entries in the dataset: a - the first; b - second, c - third

As part of the project, two separate datasets were used, which collected textual content from various open sources, including social networks and media platforms, along with data related to their publication. All entries in both datasets were marked with authenticity for false and truthful news. Further merging of these two data sets allowed for a complete and more representative corpus, which contributes to better model training and generalisation of results with new examples. In order to get acquainted with the data structure in the columns, the first four records from the datasets were displayed (Fig. 2.). First of all, attention is focused on the first two columns, in which the values for all the derived observations turned out to be uninformative. At the same time, the number of non-empty records detected at the previous stage gave grounds for further analysis. As a result of the study of these text entries, it was found that they are messages about the unavailability of the main content to be classified. As expected, the relevant observations contained the NaN value in the "text message" column, which will be renamed "text" in the future, as well as texts that are likely system

messages regarding the status of the post. The presence of a source for each entry made it possible to verify the status of the post: in some cases, its unavailability was confirmed, while in others it was possible to identify the content and fill in the missing text values, which applies to the entries displayed in Fig. 3 under indices 13 and 17 (dataset 3). For the first case, the actions described below were taken. It identified columns that did not contain any values or were irrelevant in the context of the modelling problem in this project. The columns "text", "label", "source", "lang" and "link" were renamed and left in the dataset for analysis and "text" and "label" for modelling, which is shown in Fig. 4. Since the dataset mentioned above 3 had many irrelevant columns, i.e. those that did not contain any values or did not carry valuable information in the context of the developed model, only the columns necessary for further work were selected and renamed, namely "text", "label", "source", "lang" and "link", which is shown in Fig. 5. The next step was to remove unnecessary entries and missing values in the "text" header. The result is shown in Fig. 6.

```
df.loc[~df['Unnamed: 0'].isna()]
```

0.0s Python

	Unnamed: 0	№	дата	час	текст повідомлень	мітка	Post/Repost	Автор/Group	Джерело	Мова	...	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16	Unnamed: 17	Unnamed: 18	Unnamed: 19	Unnamed: 20	Unnamed: 21	Unname
6	Деякі дописи невидимі через налаштування конфі...		NaN	NaN	Деякі дописи невидимі через налаштування конфі...	правда	Repost	facebook	Facebook	Ukrainian	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
13	сАМОПОШИРЕННЯ	NaN	21.08.2024	14:01	Пріоритет у забезпеченні боєприпасами, як і со...	ФЕЙК	Post	Світловодськ Адміністратор	Facebook	Ukrainian	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
17	ПРОПАВ ДОПИС	NaN	20.08.2024	19:07	Оборона ЗСУ у Донецькій області залишилася без...	ФЕЙК	Post	Добропілля онлайн	Telegram	Ukrainian	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
22	Деякі дописи невидимі через налаштування конфі...		NaN	NaN	Деякі дописи невидимі через налаштування конфі...	ФЕЙК	Repost	facebook	Facebook	Russian	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
23	Деякі дописи невидимі через налаштування конфі...		NaN	NaN	Деякі дописи невидимі через налаштування конфі...	ФЕЙК	Repost	facebook	Facebook	Russian	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
952	На жаль, ця сторінка недоступна.		NaN	NaN	На жаль, ця сторінка недоступна.	фейк	Post	На жаль, ця сторінка недоступна.	instagram	russian	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

Fig.3. Dataset records containing no access messages

```
df1.rename(columns={' Мітка': 'label', 'Текст повідомлень': 'text', 'Джерело': 'source', 'Мова': 'lang', 'Веб-адреса': 'link'}, inplace=True)
df1 = df1[['text', 'label', 'lang', 'source', 'link']]
```

(a)

```
df2.rename(columns={'мітка': 'label', 'текст повідомлень': 'text', 'Джерело': 'source', 'Мова': 'lang', 'веб-адреса': 'link'}, inplace=True)
df2 = df2[['text', 'label', 'lang', 'source', 'link']]
```

(b)

Fig.4. Renaming and selection of the necessary columns for analysis: a - the first dataset; b - the second dataset

```
df.rename(columns={' мітка': 'label',
                   'текст повідомлень': 'text',
                   'Джерело': 'source',
                   'Мова': 'lang',
                   'веб-адреса': 'link'}, inplace=True)

df = df[['text', 'label', 'lang', 'source', 'link']]
```

Fig.5. Renaming and selecting the necessary columns

It was decided to clean up the duplicates after preprocessing the text, since certain entries could differ only in case or number of spaces, while the words and their sequence were identical. Analyzing the missing values was the next step. The number of missing values in the columns is shown in Fig. 7. At this point, before merging the datasets, it was decided to define and apply the appropriate way to fill in the missing values in the columns that are directly used to build the model. The only strategy for restoring values in the "text" column is to extract the content from the corresponding "link" column, and in the "label" column, to define the label manually, taking into account the context and content of the messages. Therefore, it is pretty logical to delete records where data is absent in each of these columns (Fig. 8).

At the stage of data analysis, a critical imbalance was revealed: most of the records belonged to the "true" class, a much smaller number belonged to the "false" class. To avoid model bias, a simple resampling strategy is applied:

– the entire "false" class is saved completely,
– 460 examples were randomly selected from the "true" class.

As a result, a balanced dataset of 890 entries was obtained (≈50% "true" and 50% "false"). Additionally, during the Logistic Regression training, class_weight="balanced" was used, which compensates for possible deviations.

```
df=df.loc[(~df['text'].isin([
    'Деякі дописи невидимі через налаштування конфіденційності.',
    'З'єднання не конфіденційне',
    "немає зв'язку з сайтом ПЕРЕХІД на фейкові виплати",
    'архів',
    'відео',
    'відео ',
    'Неправдива інформація. Перевірено сторонніми експертами з перевірки фактів.',
    "Немає зв'язку із сайтом",
    "Цей контент наразі недоступний Можливо, автор матеріалу поширив його лише для невеликої групи людей, змінив його аудиторію або видалив."
    "Гмм… такої сторінки не існує. Спробуйте пошукати щось інше.",
    "Зараз відео недоступне",
    "На жаль, ця сторінка недоступна.",
    "Цей контент наразі недоступний",
    "змінилася інформація по лінку",
    "Зараз відео недоступне https://voxukraine.org/ru/obnovlyaetsya-obzor-provokatsyj-y-dezynformatsyy-rf-chast-2",
    "Німецьке поширення Tja, der Führer des Regimes, ist schließlich ein erfahrener Schauspieler, Drehbuchautor und Regisseur.  Und bezahlen
    'На жаль, ця сторінка недоступна Посилання, за яким ви перейшли, можливо, пошкоджене або сторінку видалено.'
]))]

df.dropna(subset=['text'], inplace=True)
```

Fig.6. Cleaning the "text" column

```
df1.isna().sum()
✓ 0.0s
text      157
label     158
lang      160
source    176
link      163
```

```
df2.isna().sum()
✓ 0.0s
text      2
label     2
lang      3
source    2
link      2
```

Fig.7. Number of missing values in columns: a - the first dataset; b - the second dataset

```
df1.dropna(subset=['text', 'label'], inplace=True)
```

(a)

```
df2.dropna(subset=['text', 'label'], inplace=True)
```

(b)

Fig.8. Deletion of irrelevant records: a - the first dataset; b - the second dataset

The analysis of such observations helped to reveal that their other columns also contain missing values (Fig. 9). The number of irrelevant entries coincides with the number of those where only the text of the information message is missing, which indicates the impossibility of restoring any of them. After processing the missing values, duplicates were found and removed, as shown in Fig. 10. The pre-processed datasets were concatenated. After that, the analysis of the missing values was continued for the column's "source", "lang" and "link" (Fig. 11). The image shows Fig. 12 text messages for which there is no language data, three of which are in Ukrainian and one in Russian. The column was filled with the corresponding values.

```
df1[df1[['text', 'label', 'link']].isna().all(axis=1)]
✓ 0.0s
```

	text	label	lang	source	link
841	NaN	NaN	NaN	NaN	NaN
842	NaN	NaN	NaN	NaN	NaN
843	NaN	NaN	NaN	NaN	NaN
844	NaN	NaN	NaN	NaN	NaN
845	NaN	NaN	NaN	NaN	NaN
…	…	…	…	…	…
993	NaN	NaN	NaN	NaN	NaN
994	NaN	NaN	NaN	NaN	NaN
995	NaN	NaN	NaN	NaN	NaN
996	NaN	NaN	NaN	NaN	NaN
997	NaN	NaN	NaN	NaN	NaN

157 rows × 5 columns

```
df2[df2[['text', 'label', 'link']].isna().all(axis=1)]
✓ 0.0s
```

	text	label	lang	source	link
437	NaN	NaN	NaN	NaN	NaN
633	NaN	NaN	NaN	NaN	NaN

Fig.9. Records with missing values in several columns at the same time: a - the first dataset; b - the second dataset

```
df1 = handle_dublicates(df1)

Duplicates count: 7
```

```
df2 = handle_dublicates(df2)
✓ 0.0s
Duplicates count: 2
```

Fig.10. Number of duplicates: a - the first dataset; b - the second dataset

```
df.isna().sum()
✓ 0.0s
text       0
label      0
lang       4
source    19
link       6
dtype: int64
```

Fig.11. Number of missing values

```
df.loc[df['lang'].isna(), ['text', 'lang']]
✓ 0.0s
```

	text	lang
89	Украина вернула России самую большую группу де...	NaN
823	ЗСУ спробували атакувати Курську АЕС За даними...	NaN
824	У Миколаєві за місцем дислокації наймаків НАТО...	NaN
1216	Meta представила нову VR-платформу Horizon	NaN

Fig.12. Rows with missing values in the "lang" column

The "link" column was used to get the domain name so that the missing source names could be filled in the "source" column. To make sure that all the missing data in this column can be restored in this way, these records were previously detected (Fig. 13), from which we can see that the links are available, and for records where there are no messages, the sources are already indicated. Thus, after use, the column "link" is removed.

```
df[df[['source', 'link']].isna().any(axis=1)]
✓ 0.0s
```

	text	label	lang	source	link
89	Украина вернула России самую большую группу де...	ЛОЖЬ	російська	NaN	https://rtvi.com/news/ukraina-vernula-rossii-s...
90	США намерены воссоздать утраченные на Украине ...	ЛОЖЬ	Russian	NaN	https://cominf.org/node/1166553525https://ura...
91	Заявление украинской стороны о «массовых убийс...	ЛОЖЬ	russian	NaN	https://ura.news/news/1052543043
92	Зеленский сообщил об уничтожении ТЭС и ГЭС на ...	ЛОЖЬ	russian	NaN	https://ura.news/news/1052821973
93	Беспилотник ВСУ сбросил боеприпас на Запорожск...	ЛОЖЬ	russian	NaN	https://ura.news/news/1052821863
94	Украина сознательно убивает детей Донбасса	ЛОЖЬ	russian	NaN	https://aif.ru/politics/world/tochnost_takaya_...
95	Почему Зеленский отказался от мира\n	ЛОЖЬ	russian	NaN	https://aif.ru/politics/world/kak_mogu_byt_nac...
96	В Британии заявили, что Зеленский сбежал из Ки...	ЛОЖЬ	Russian	NaN	https://arc.net/l/quote/izhksrtt
97	ВС России поразили центр радиосвязи ВСУ и цеха...	ЛОЖЬ	russian	NaN	https://regnum.ru/news/3918581
98	Зеленський заявив, що РФ знімала АЕС України д...	ИСТИНА	Ukrainian	NaN	https://tsn.ua/politika/zelenskiy-zayaviv-scho...
99	Путін вперше публічно зібрав нараду Ради безпе...	ИСТИНА	Ukrainian	NaN	https://tsn.ua/ato/putin-vpershe-publichno-zib...
100	Путін заявив, що може застосувати ядерну зброю...	ИСТИНА	Ukrainian	NaN	https://tsn.ua/svit/putin-zayaviv-scho-mozhe-z...
101	Окупанти обстріляли Краматорськ на Донеччині: ...	ИСТИНА	Ukrainian	NaN	https://www.pravda.com.ua/news/2024/09/25/7476...
102	Сексуальне насильство як форма тортур: комісія...	ИСТИНА	Ukrainian	NaN	https://www.pravda.com.ua/news/2024/09/25/7476...
103	Російський танк обстріляв Михайлівку на Херсон...	ИСТИНА	Ukrainian	NaN	https://www.pravda.com.ua/news/2024/09/25/7476...
104	Плани РФ бити по українських АЕС: експерт розп...	ИСТИНА	Ukrainian	NaN	https://www.unian.ua/war/udari-rosiji-po-ukraj...
105	Мільйон загиблих і поранених: WSJ передрік дем...	ИСТИНА	Ukrainian	NaN	https://arc.net/l/quote/xcpdmynj
106	Росіяни перебувають в оточенні в Крем?яному на...	ИСТИНА	Ukrainian	NaN	https://www.unian.ua/war/kurska-oblast-armiya-...
107	Україна ознайомить союзників з Планом перемоги...	ИСТИНА	Ukrainian	NaN	https://www.unian.ua/war/ukrajina-oznayomit-so...
255	Министр обороны Украины Резников как никто зна...	фейк	Російська	фейсбук	NaN
293	ШАНОВНІ ПРАЦІВНИКИ КАФЕДРИ!\nЄ інформація від ...	ЛОЖЬ	Ukrainian	Telegram	NaN
388	Наш источник сообщает, что американское турне ...	ЛОЖЬ	russian	Telegram	NaN
389	Украинские военные стали готовить общество к п...	ЛОЖЬ	russian	Telegram	NaN
390	Курская авантюра Зеленского уничтожает все рез...	ЛОЖЬ	russian	telegram	NaN
391	Наш источник в ОП рассказал, что Зеленский не ...	ЛОЖЬ	russian	Telegram	NaN

Fig.13. Rows with missing values in the column "source" or "link"

	text	label	lang	source
0	оборона всу в донецкой области осталась без снарядов и солдат после наступления на курскую область сообщает the financial times издание пишет что эта операция всу наоборот даже ускорила наступление россии на донбассе это произошло в частности изза снятия с этого участка фронта до 10 тысяч украинских военных среди которых и элитные десантноштурмовые подразделения приоритет в обеспечении боеприпасами как и солдатами теперь отдается курскому направлению рассказали ft офицеры переброшенные с донбасса на территорию россии	ложь	russian	facebook
1	flash colonel macgregor la contreattaque russe transforme koursk en cimetire ukrainien actualits gopolitique russie	истина	french	facebook
2	wer kontrolliert die algorithmen die unser leben beeinflussen algorithmen sind mchtige unsichtbare helfer in unserem alltag sie entscheiden welche filme wir sehen welche nachrichten wir lesen und sogar welche restaurants uns empfohlen werden doch diese macht birgt auch risiken verzerrungen datenschutzprobleme und die gefahr in einer echokammer zu landen wie knnen wir sicherstellen dass diese digitalen wchter ethisch und transparent handeln erfahre mehr ber die unsichtbare macht der algorithmen	ложь	german	facebook
3	астрологический прогноз для тебя от мольфарки	ложь	russian	facebook
4	увлекательные факты о городе днепр 1 первое построенное в днепре здание дворец потомкина в 1790 году ныне дворец студентов 2 конкретной даты дня города в днепре нет отмечается во вторую субботу сентября 3 в днепре самая большая набережная в европе составляет 23 км 4 в днепре производят ракеты зенит аналогов которым до сих пор нет во всем мире 5 самый большой еврейский центр в мире находится в днепре 6 самая старая и самая большая коллекция каменных дам находится в днепровском историческом музее 7 первый в мире железобетонный мост и самый большой в европе арчный мост мерефохерсонский расположен в днепре	истина	russian	facebook

Fig.14. Lines from the processed text

Since all variables are categorical and contain text, it is essential to unify the values in order to analyse correctly. For this purpose, data preprocessing was carried out, which included: case normalisation, elimination of unnecessary spaces and special characters, links, and tags. The first four records from the processed dataset are shown in Fig. 14.

After that, duplicates that appeared after text processing were removed (Fig. 15).

```
handle_dublicates(df['text'])
✓ 0.1s
Duplicates count: 16
```

Fig.15. Deleting duplicates

Before studying the distribution of the categorical variable's "label" and "lang", an analysis of their unique meanings was carried out, including significant textual variations. For example, as can be seen from Fig. 16. The meaning of the labels had numerous synonymous forms: "fake", "false", "false", "f", etc. A similar situation was observed for lang, where different variants of writing the same language were given even in other languages, for example, "Ukrainian", "Ukrainian", "uk", etc.

	unique_values
label	[ложь, истина, фейк, істина, false, true, f, t, т, правда, брехня, fake, реальна, фейкова, правдива, неправдива, мітка]
lang	[russian, french, german, english, ukrainian, українська, російська, польська, українська, англійська, український, чеська, polish, укр, slovak, poland, ukranian, укрїнська, ukrainian the original language is russian, латвійська, французька, українська переклад з російської, uk, німецька, ukraine, ua, англійськаукраїнська, білоруська, мова]

Fig.16. Unique values of the variables "label" and "lang"

Unique meanings after synonyms are shown in Fig. 17.

	unique_values
label	[false, true]
lang	[rus, fr, de, eng, ukr, pl, cs, sk, lv, be]

Fig.17. Unique values of the variables "label" and "lang"

B. After Processing

The distribution graph of the target variable shows a critical imbalance of classes (Fig. 18).

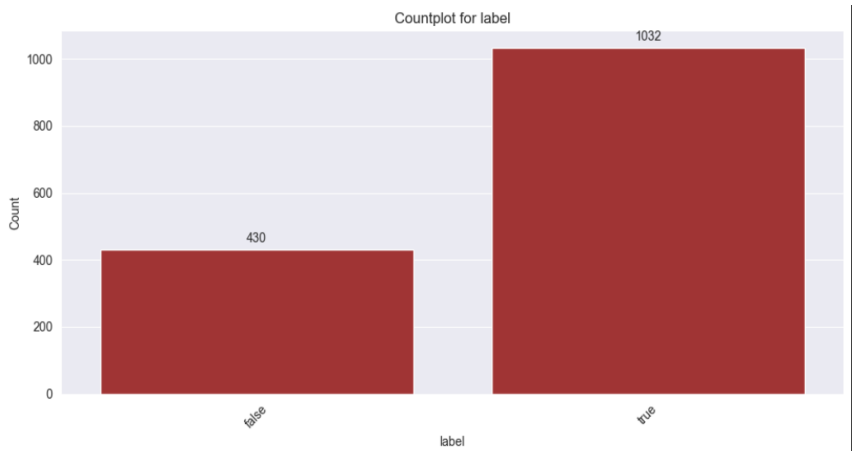


Fig.18. Distribution of the target variable "label"

As for languages, the dataset contains most texts in Ukrainian and enough in Russian and English (Fig. 19). There are entries in Polish, French, German, Czech, Slovak, Latvian and Belarusian.

Most fakes were collected from Telegram, which is shown in Fig. 20. The graph also contains an error due to the fact that the names of the resources are not unified, for example, for the same Telegram, 85 entries are allocated for the name "telegram" and 7 for "tme". It should be noted that only the first twenty-five sources, which are the largest in the dataset, are displayed on the graph. Most of the true news was also collected from Telegram, which is shown in Fig. 21. In order to eliminate the imbalance between classes in the target variable "label", a simple resampling strategy was applied. A complete sample of the examples of the class "false" was stored in its entirety. In contrast, a sample of 460 entries was randomly selected from the class "true", which is close to the number of observations in the less represented class. The ratio of classes after balancing is shown in Fig. 22.

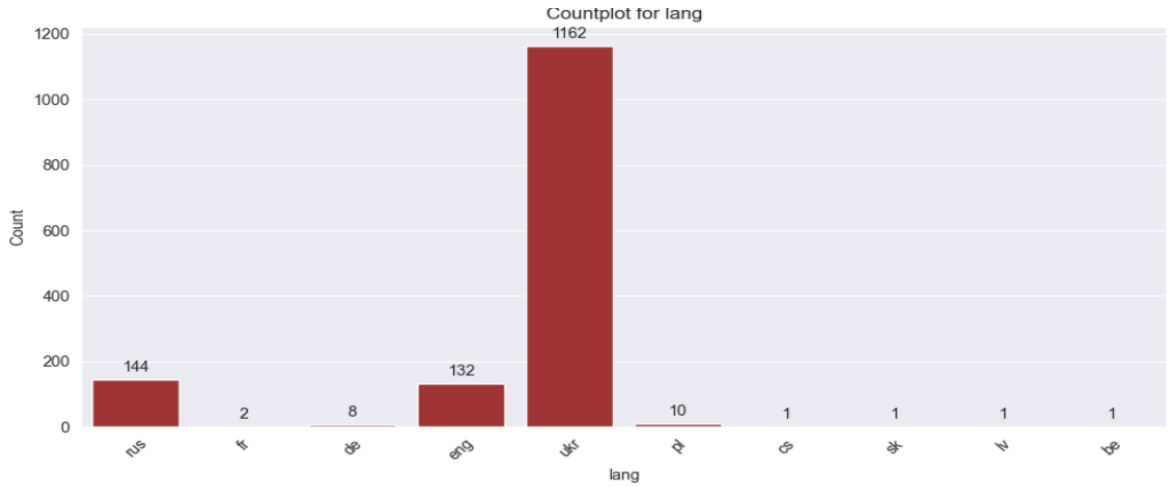


Fig.19. Distribution of the variable "lang"

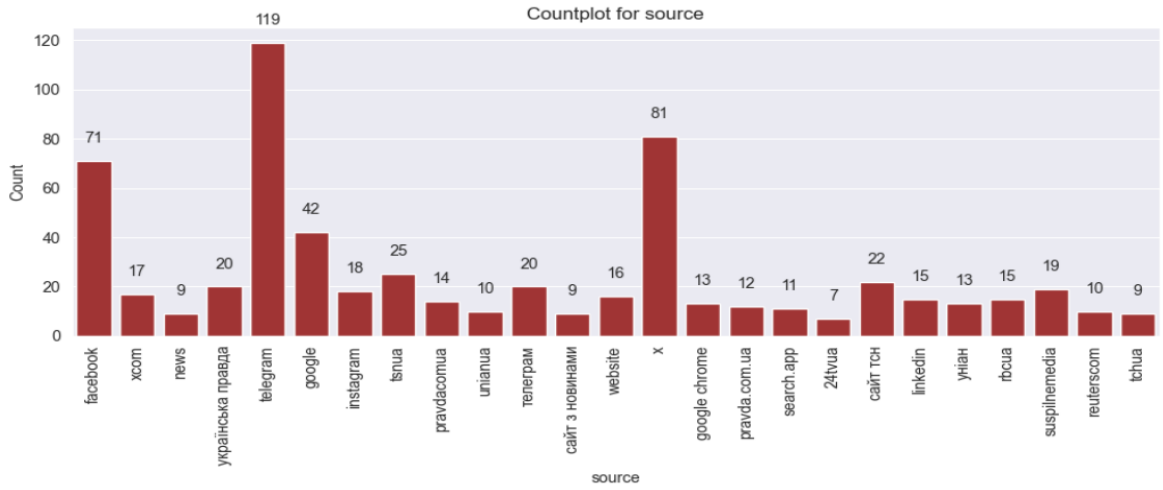


Fig.20. The number of fakes collected from different sources

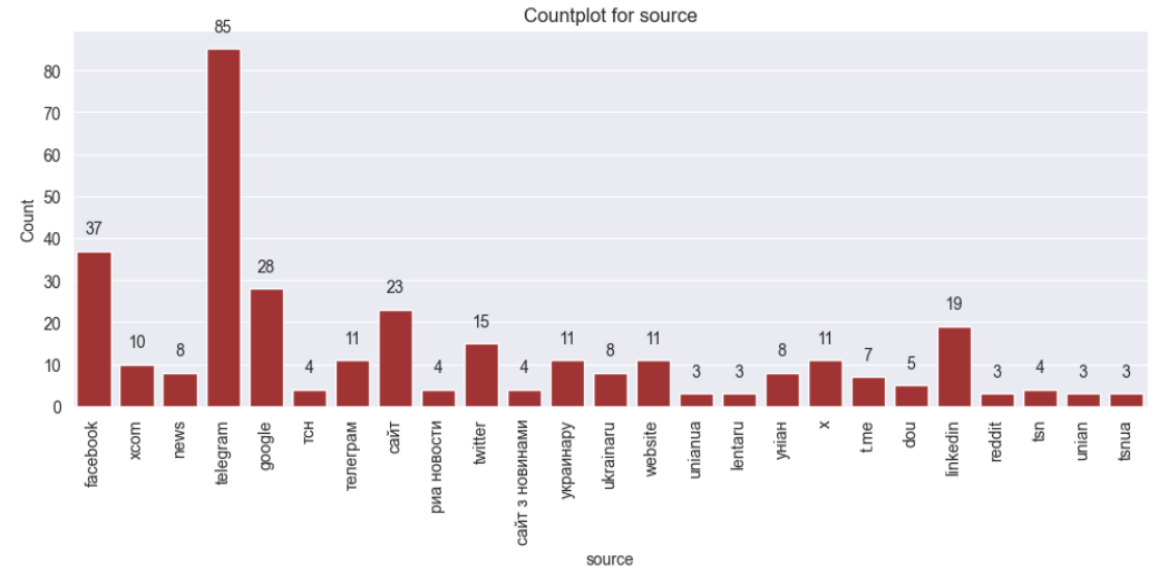


Fig.21. The number of truthful news items collected from various sources

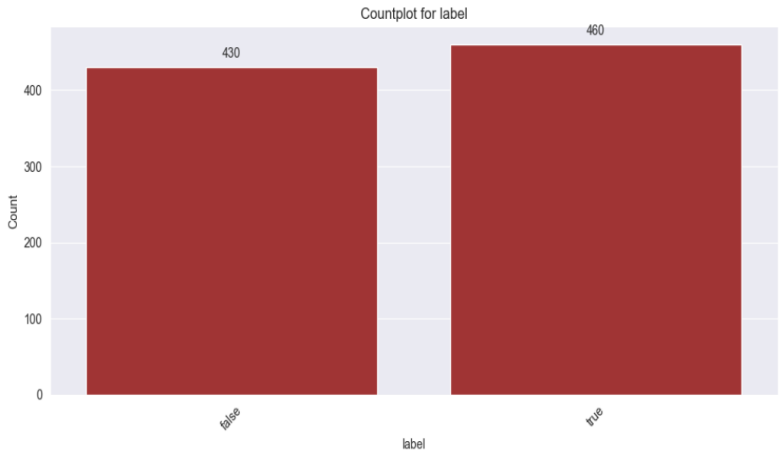


Fig.22. Class balance after resampling

As a result, a balanced dataset of 890 complete records was formed, in which the observations are relatively evenly distributed between the two classes of the target variable (Fig. 23).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 890 entries, 0 to 889
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    890 non-null    object
 1   label   890 non-null    object
 2   lang    890 non-null    object
 3   source  890 non-null    object
dtypes: object(4)
```

Fig.23. Information about the received dataset

```
len(unique_words)
✓ 0.0s

10118
```

Fig.24. Number of unique tokens by regular partition in the corpus

To understand the structure of the text corpus, the basic statistics of each message were calculated, namely:

- Text lengths in characters.
- Separation by spaces determines the number of words.
- The number of tokens obtained using the Granite model tokeniser.

The dictionary formed from unique words in the corpus after balancing has 10,118, which indicates high lexical variability (Fig. 24).

When calculating the number of tokens in the text using the tokenizer of the Granite model, it was found that some messages exceed the allowable sequence length set for this model, i.e. 512 tokens (Fig. 25). In order to avoid loss of information and ensure compatibility with the requirements of the model, it was decided to implement a partial overlapping chanting mechanism in the classroom.

```
granite_token_count = df['text'].apply(lambda t: len(granite_tokenizer.tokenize(t)))

Token indices sequence length is longer than the specified maximum sequence length for this model (612 > 512). Running this sequence through the model will result in indexing errors
```

Fig.25. Warning about exceeding the permissible sequence length

The total number of tokens obtained from the model tokeniser (Fig. 26) exceeds the number of tokens obtained by simply dividing the text by spaces. This is because the model uses a specialised sublexemic tokeniser that breaks the text into smaller language units – subtokens- not ordinary words.

```
granite_token_count.sum()
✓ 0.0s
np.int64(39067)
```

Fig.26. Number of unique tokens in the corpus

Histograms, distributions of text length in symbols, word counts, and token counts derived from the Granite model tokeniser were also constructed. The graph in Figure 27 shows the distribution of text message lengths calculated by the number of characters. As you can see from the bar graph, most of the texts are short: more than 600 entries are less than 200 characters long. The distribution is strongly shifted to the left, with a long right "tail", indicating the presence of individual messages of large volume - more than 1000 characters, and in rare cases, more than 4000. This distribution is quite typical because in social networks most messages are short, and in the media long texts are more common.

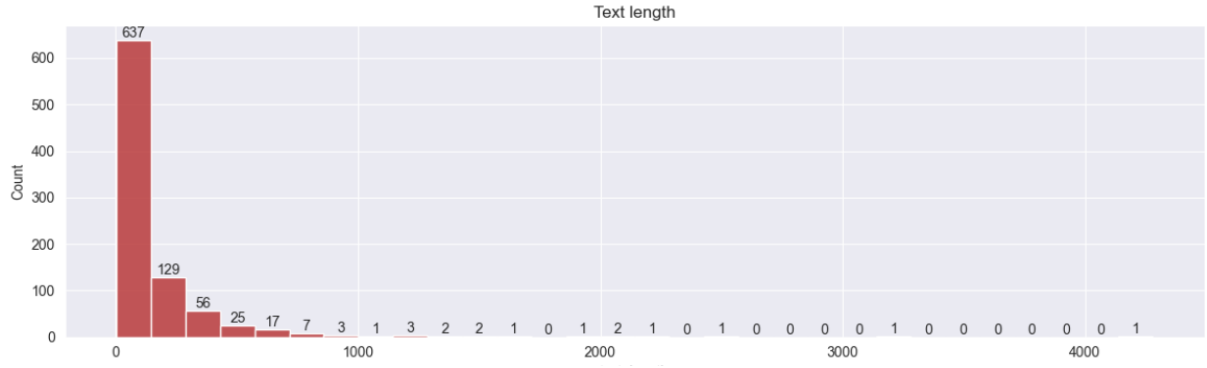


Fig.27. Distribution of text length in characters

The graph of the distribution of the number of words (Fig. 28) reflects that the vast majority of entries, more than 600, contain fewer than 50 words.

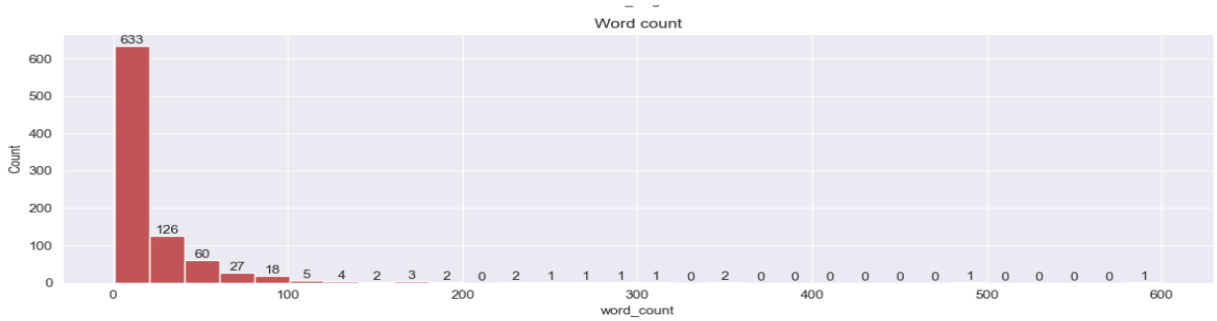


Fig.28. Word count distribution

The graph shows the distribution of the number of tokens in each text message received using the Granite model tokeniser (Fig. 29). Most texts contain fewer than 100 tokens, as in the previous graphs, there is a right-sided distribution asymmetry - individual messages contain about 500, and in rare cases, about 600 tokens. It was a confirmation of the mistake mentioned above.

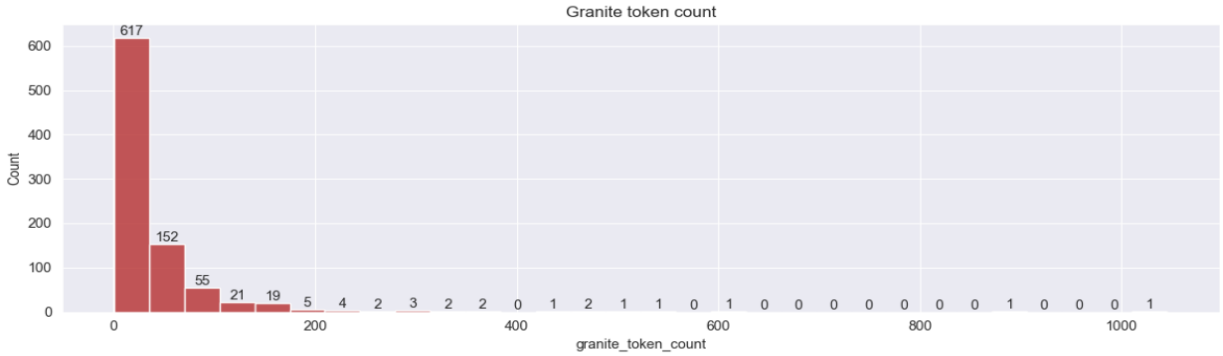


Fig.29. Distribution of the number of tokens

However, since the dataset is heterogeneous and consists of several languages, each stage is given, which is different from typical stages. Let's explain the peculiarity of each stage, emphasising how exactly it differs from more typical procedures in similar works:

Table 2. The peculiarity of each stage of the study

#	Name	Fig.	Peculiarity	Difference
1	Select and rename columns	4–5	left only the key variables: text, label, source, lang, link. The rest were rejected.	In many studies, they take texts and labels (X, y), while here they additionally saved metadata (source, lang, link) for further verification and analysis. It allows, for example, the automatic restoration of missing values (source, language).
2.	Deleting irrelevant entries and NaN	6–9	Entries without text or with system messages ("no access to post") have been deleted or restored from the link field.	Usually, in typical works, all NaNs are discarded, and content recovery from alternative fields is immediately applied, which increases the completeness of the case.
3.	Unification of values in categories	16–17	labels (label) and languages (lang) had many variations ("fake"/"false"/"f"; "ukrainian"/"uk"/"Ukrainian", etc.). They have been identified in standardised forms.	In most works, this task is ignored or done superficially, while here a systematic approach is shown, bringing synonymous variants to a single norm. It reduces the "noise" in the data.
4.	Working with missing values in source and lang	11–13	Missing sources were automatically restored by domain from the link, and languages were automatically restored manually or algorithmically based on text.	In many studies, gaps are removed, and recovery based on auxiliary fields is also used, which makes the dataset more complete.
5.	Text normalisation	14	The text has been lowercased, special characters, tags, links, and emoticons; extra spaces have been removed.	The typical approach stops at tokenisation and stop words, and the emphasis is on eliminating technical noise specific to Telegram (emojis, HTML tags, URLs). It is adjusted to the specifics of the platform.
6.	Removing duplicates after cleaning	15	Duplicates were considered records even with different spaces or registers, so clearing preceded de-duplication.	In standard works, duplicate lines "as is" are often removed. Here, normalisation is first carried out so as not to leave hidden copies.
7.	Class balancing	18–23	A simple resampling was applied – all fakes were saved, and a subset was randomly selected from the trustworthy news.	Most modern jobs use oversampling (SMOTE) or class weights. Here, the emphasis is on a simple but controlled selection to maintain the authenticity of the case without synthetic examples.

Therefore, the main difference between pre-processing in research and typical ones is the emphasis on high-quality data recovery and unification, and not only in "mechanical" cleaning. It increases not only the cleanliness of the case but also its representativeness, which is essential in the Ukrainian-speaking context, where there are no large ready-made datasets.

C. Pipeline Pre-processing

For the proper functioning of the logistic regression model, which performs binary classification, a unified pipeline of text data processing was created. This process covers both the "text" variable and the target "label" variable. The main goal of the pre-processing stage is to reduce the diversity in the input texts, eliminate unnecessary information (noise) and standardise the designation of classes, etc. The same pipeline was used in the exploratory analysis of the data described above. The preprocessing diagram is shown in Fig. 30.

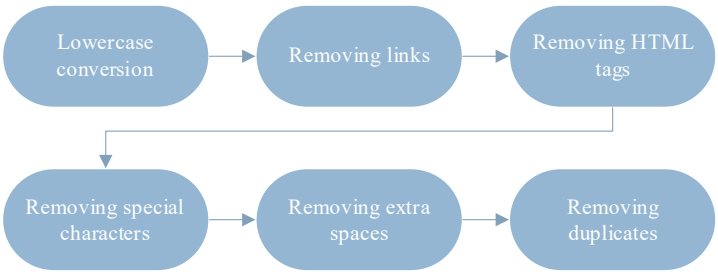


Fig.30. Diagram of the pre-processing of text data

Stages of text preprocessing:

- Lowercase. All characters in the "text" and "label" fields have been converted to lowercase letters using str.lower() to avoid duplication of identical words written in different cases (for example: "FALSE", "false", "False");
- Removal of hyperlinks and HTML tags. Using regular expressions, Internet links (including http, https, www, etc.), and HTML tags (e.g. <p>,
) that do not carry a content load have been removed from the text;
- Remove unnecessary characters. Removed all elements that are not Latin, Cyrillic, numerals, or spaces. In this way, punctuation marks, emojis, and other technical characters are eliminated. In addition, consecutive spaces

have been reduced to one, and extra spaces at the beginning and end of the line have been removed.

- Deletion of duplicates. To prevent the model from being trained on identical texts, duplicates in the "text" column have been removed, and only the first occurrence of each unique message has been preserved.
- Target variable unification. The same cleanup operations are applied to labels in the "Label" column. As a result, only two valid values are ensured: "false" and "true", which is a prerequisite for the stable operation of the classification model.

The preprocessing pipeline is shown in Fig. 31.

```python
def clean_text(text: str) -> str:
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", "", text)
    text = re.sub(r"<.*?>", "", text)
    text = re.sub(r"[^\p{L}\p{N}\s]", "", text, flags=re.IGNORECASE)
    text = re.sub(r"\s+", " ", text).strip()
    return text
```

Fig.31. Pipeline of preprocessing of text and target variables

Only after preprocessing is the data corpus subject to the following processing steps: tokenisation, distribution into training and test samples, and then numerical representation of tests by vectorisation and embedding, as well as training the classification model. After preprocessing of the columns mentioned above, the number of duplicates throughout the dataset is zero (see Fig. 32a), and the number of values that the "label" variable can acquire has been reduced to two: "true" and "fake" (see Fig. 32b).

```python
print(df.duplicated().sum())
```

```
0
```

```python
df['label'].unique()
```

```
array(['фейк', 'правда'], dtype=object)
```

Fig.32. Number of duplicates after preprocessing and unique values of the "label" column after preprocessing

After all the above steps for processing the dataset and, in particular, the two central columns, an updated general information about the dataset was displayed for comparison with the initial version (see Fig. 33). After the text and target data are cleaned, the dataset is ready for the following stages of processing – tokenization, distribution into training and test samples, conversion of text to vector representation by vectorization and embedding using selected models, as well as building and training a classification model to automatically determine the truth or falsity of the news.

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 483 entries, 0 to 962
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    483 non-null    object
 1   label   483 non-null    object
 2   lang    483 non-null    object
 3   source  460 non-null    object
 4   link    473 non-null    object
dtypes: object(5)
memory usage: 22.6+ KB
```

Fig.33. Updated dataset information

D. Selected Models and Methods (TF-IDF, IBM Granite, Logistic Regression)

As already mentioned, within the framework of this work, a combined approach to the classification of fake news has been implemented, which combines two types of text features: frequency (TF-IDF) and contextual vectors (Granite Embeddings). This method allows you to take into account both the statistical structure of the text and its semantic content – the relationships between words and the context of their use. The result is a hybrid vector representation that serves as the basis for training the logistic regression model. At the initial stage of model construction, text data is transformed into a numerical format using two methods – TF-IDF and IBM Granite. TF-IDF (Term Frequency – Inverse Document Frequency) allows you to obtain a numerical representation of texts that takes into account both the local and global significance of the word for each document in the corpus, to highlight specific words specific to fake or true messages, and to neutralize the influence of words that do not carry much meaning, such as pronouns that are often

repeated in different texts. TF-IDF hyperparameters:

- max_features = 5500 is the maximum number of signs. This choice is due to the number of unique words, which is 9653.
- tokeniser = lambda x: x.split(' ') – simple splitting by spaces.

Contextual embeddings obtained using the ibm-granite/granite-embedding-107m-multilingual model have been added to the TF-IDF vector representation. This model is part of the Granite transformer models developed by IBM for multilingual natural language processing tasks. It is trained on a large corpus of texts. It is able to effectively cover semantic connections in texts in different languages, which makes it suitable for deeper semantic interpretation of input data. It supports processing sequences up to 512 tokens in length and generates 384-dimensional embeddings that reflect the content context of the text [35-36].

Unlike TF-IDF, which operates only with statistical parameters of word frequency, the IBM Granite model generates vectors that take into account the context in which certain words are used. At the heart of its transformer architecture is the self-attention mechanism proposed in a study by Vaswani (2017), which allows each token to interact with the entire sequence, regardless of position. This approach ensures the ability of the model to effectively identify both close and distant lexical and semantic connections. It is crucial when analysing complex text constructions in which disinformation can be disguised as neutral or truthful messages [36-37].

One of the key advantages of this model is its full integration compatibility with the SentenceTransformers library, which significantly facilitates implementation in application projects. Work with the model is carried out in several stages: first, the necessary libraries are connected and the model itself is downloaded from an open source. After that, it is initialised as an object capable of receiving an array of texts as input. The final step is to pass the texts to the encode() method, which automatically processes them by performing tokenisation and vector representation calculations.

To combine the advantages of frequency and semantic features, an extended vector representation was formed by combining TF-IDF and Granite embeddings. Such a hybrid vector allows the model to take into account both the lexical features of the text and deeper semantic connections between words.

At the classification stage, logistic regression was used – a linear model that has proven itself well in binary classification problems, especially with a large number of features. Due to its high speed and ease of interpretation, logistic regression is effectively combined with various vectorisation methods. In this case, it works with combined features based on TF-IDF and Granite, and returns the probability that the message belongs to the "false" class – that is, it is fake.

Since the accuracy of the model significantly depends on the settings of its hyperparameters, the procedure for tuning hyperparameters is implemented to select the optimal values using the RandomizedSearchCV method.

The tuning process involves random testing of 20 combinations of values of three key parameters:

- C (regularisation constant). Values vary on a logarithmic scale, ranging from 0.001 to 100. This parameter controls the degree of regularisation – smaller values enhance it, while larger values reduce it.
- penalty: "L1" or "L2". Both types of penalties help to avoid overtraining the model, but they work differently: "l1" stimulates the sparseness of the scales (some features may have zero coefficients), while "l2" only reduces the value of the scales without zeroing them;
- Solver. "Liblinear" and "Saga" were tested, and both are compatible with the selected penalties. "Liblinear" is well suited for small and medium-sized datasets, while "Saga" scales better and supports large volumes of features typical of a hybrid vector space.

Each combination was evaluated according to the F1 metric using 5x cross-validation. It allowed us to choose the model that best balances accuracy and completeness, which is critically important in cases where it is essential not to miss fake messages (false) and at the same time not mistakenly attribute true messages to a positive class. Additionally used:

- class_weight = "balanced", which automatically adjusts the weights of classes depending on their quantitative representation in the study sample, compensating for possible imbalances between them;
- random_state=42, which ensures the reproducibility of the results during repeated learning.

Thus, the combination of frequency and contextual features makes it possible to form a rich presentation of the text, sensitive to various manifestations of information falsity. Based on this representation, a logistic regression model has been trained, which uses hybrid features to classify news according to their degree of reliability. All stages – from text processing to evaluation of results – have been integrated into a single coherent structure, which is the FakeNewsDetector class. After the search for optimal hyperparameters is completed, the model is trained on the entire training sample with the best parameters found. During testing, it returns the provided labels for the test data, which are stored for further evaluation of the model's quality.

Thanks to the combination of frequency and semantically saturated features, it was possible to create an informative vector representation of the text capable of capturing various markers of potential unreliability. On the basis of this view, logistic regression was trained, which performs the classification of news according to the criterion of reliability using hybrid vectors. All stages – from data pre-processing to the final assessment of model performance – are combined into

a single holistic system, which guarantees consistency, adaptability and practical convenience for further application.

E. FakeNewsDetector Class and Model

To implement the described approach, the FakeNewsDetector class has been developed, which combines all the key stages of text data processing within a single object-oriented structure. It provides a complete cycle of working with data: from cleaning and vectorisation to training the classification model and assessing its quality. The class is adapted to use a hybrid text representation that integrates TF-IDF and IBM Granite contextual embeddings, and supports visual diagnostics, including distribution graphs, t-SNE projections, ROC curve, and error matrices.

The initial stage of interaction with the class involves calling the constructor __init__(), which initialises the main components and parameters of the model. In particular, the following are created:

- TF-IDF vectorizer that generates frequency features from text. To correctly handle the results of the pre-cleanup, tokenisation is carried out on the basis of space separation without conversion to lowercase (lowercase=False). The max_feats_tfidf parameter sets the maximum number of features.
- Granite-embedder (ibm-granite/granite-embedding-107m-multilingual), which provides the generation of context-dependent vectors suitable for multilingual classification problems;
- t-SNE modules (one for each vectorisation type) are designed to visualise multidimensional vectors in the plane;
- Key configuration parameters: test_size determines the proportion of the test sample when splitting, random_state ensures the reproducibility of the results, and colour is used as the basic parameter for the design of visualisations.

The __init__() constructor is preceded by a static method split_on_space(), which implements tokenisation by splitting text into spaces. This method is used as a custom tokeniser in the TF-IDF vectorizer, which allows you to work with already preprocessed texts without additional case change or structural transformation.

The second step is to load the input dataset using the load_dataset() method, which specifies the names of the columns with text data and target labels. The technique creates a copy of the passed DataFrame and stores the corresponding column names for later use in all stages of processing and modelling.

After that, the clean_text_data() method is applied, which normalises texts and labels by converting them to lowercase and removing HTML tags, hyperlinks, memorable characters, and unnecessary spaces. Duplicate texts are also eliminated. Cleanup applies not only to text messages but also to labels, which allows you to unify their spelling (for example, to avoid options like "false" / "FALSE" / "False").

The balance_of_target() and plot_text_distribution() methods provide primary visual diagnostics of data.

The first of them – balance_of_target() – builds a bar chart showing the distribution of classes in the target variable. Such a graph allows you to quickly assess the degree of imbalance between the labels "false" (fake news) and "true" (trustworthy news). The method accepts the figsize parameter to adjust the size of the graph, as well as save_path if you want to save the image.

The second method – plot_text_distribution() – generates two histograms: one shows the number of characters in the texts; the other shows the number of words. These visualisations allow you to estimate the typical length of messages and identify potential outliers or atypical examples (e.g., texts that are too short or too long). In addition to figsize and save_path, the function allows you to adjust the number of bins in histograms through the bin's parameter, which will enable you to better control the level of detail.

Both methods are optional for the model, but can significantly help at the stage of preliminary analysis of the case.

The split_dataset() method splits the cleaned dataset into training and test samples while maintaining the class proportions. It is achieved using stratified sampling, which is especially important in conditions of potential imbalance of the target variable ("true" / "false").

Before splitting, it is checked whether the dataset (load_dataset()) has been loaded and whether it has been cleaned up (clean_text_data()), in particular, the presence of missing values in the text column. If these conditions are violated, the method initiates an error with an appropriate explanation.

The return_data option allows you to return split subsets directly as a tuple (text_train, text_test, label_train, label_test), which is convenient for interactive analysis or external use. Otherwise, the result is stored as class attributes and used automatically in subsequent modelling steps.

The generate_embeddings() method is responsible for constructing numerical vector representations of texts from training and test samples. It implements three approaches to vectorisation:

- Granite embeddings – contextual vectors obtained by averaging embeddings for tokenised fragments of each text;
- TF-IDF – classical statistical frequency vectors, taking into account the informative content of words in the corpus;
- Hybrid representation is the concatenation of the two previous variants into an extended vector of features.

In the process of work, the method performs the steps:

- Generates Granite vectors for each text in the training and test datasets, using the encode_with_chunk_averaging() method, which splits long texts into overlapping segments and averages their embeddings;
- Trains the TF-IDF vectorizer on the training sample and converts both samples into corresponding frequency trait matrices;
- forms a hybrid representation by horizontally combining TF-IDF and Granite vectors;

By default, all vector types are stored as class attributes for later use. If set to return_vectors=True, the method will return a tuple with vectors (X_train, X_test) for the specified type ("granite", "tf-idf", or "hybrid"). If the parameter value is incorrect, an error will be displayed for vector_type. The method ensures that vectorisation is applied only after the correct data separation (split_dataset()).

The visualize_granite_tokens() method creates a graphical representation of the number of tokens in each text sample included in the training and test samples. Tokenisation is carried out using the built-in tokeniser of the Granite model, which allows you to evaluate how exactly this model breaks down texts into separate units before building embeddings.

As a result of building histograms, it is possible to determine the average length of texts in tokens, estimate the spread (variability) of lengths, and identify texts that are too short or too long, potentially affecting the quality of presentation.

The method generates two separate graphs for the training and test parts, which makes it possible to compare the data structure in each sample. By default, graphs are only output, but can be saved to a file via the save_path option if necessary.

The plot_tsne() method is used to visualise multidimensional vector representations of text generated with TF-IDF and Granite by reducing them to two dimensions using the t-SNE (t-Distributed Stochastic Neighbour Embedding) algorithm. This approach allows you to qualitatively assess how clearly different classes (true/false) are separated in feature spaces.

The method applies t-SNE separately to features derived from the Granite model and vectors generated by the TF-IDF method.

As a result, two scatter plot graphs are created, which display each text sample in a two-dimensional space, colouring its belonging to a specific class. It allows you to visually compare the degree of clustering and class separation in each of the vector representations.

Graphs can be saved to a file if a path is specified through the save_path parameter. The method is an essential tool for preliminary analysis of the quality of embeddings before training the classification model.

The train_logistic_regression() method is responsible for the full training cycle of the classification model based on hybrid vectors. Its key task is to find the best parameters of the logistic regression model by tuning the hyperparameters (RandomizedSearchCV), which was described earlier, and further training the model with optimal values.

To initialise the model, basic logistic regression is used with the setting class_weight="balanced", which allows you to better work with potentially unbalanced samples. The search for optimal parameters is carried out using cross-validation (cv=5), while F1 (scoring="f1") is set as the target metric, which is suitable for classification problems with an uneven distribution of classes.

The trained model is stored as a self.logistic_regression attribute and the prediction for the test sample is stored as a self.label_predict. It ensures further integration of the model into the pipeline to evaluate its effectiveness.

The method also outputs the best found hyperparameters and a complete classification report based on test data (precision, recall, f1-score, support).

After training the model, it is essential to assess its ability to distinguish between true and false messages. To do this, the class provides several tools that provide a comprehensive analysis of the results.

The evaluate_confusion_matrix_and_fbeta() method forms three variants of the matrix of inconsistencies:

- Without normalisation, it shows the absolute number of correct and erroneous predictions for each class;
- Normalised in precision (precision) shows how accurately what is classified as fake is really fake;
- Normalised in completeness (recall) shows how much of what should have been classified as fake actually turned out to be fake.

Such visualisations allow you to quickly identify in which cases the model is prone to errors, for example, if it often confuses truth with falsehood.

In addition to visual analysis, the method additionally calculates F-beta, which combines accuracy and completeness in a single numerical indicator, allowing you to focus on the importance of recall or precision (depending on the value of β). In the current setting, $\beta = 1.3$, which means increased sensitivity to completeness, essential if the goal is to minimise the number of missed false messages.

The plot_roc_auc() method allows you to assess the ability of the model to distinguish classes at all possible probability thresholds. It builds a ROC curve (Receiver Operating Characteristic), where the proportion of false positive results is plotted on the X axis, and the proportion of actual positive results on the Y axis. The ideal model is one in which the curve is close to the upper left corner. Together with the curve, the area under it (AUC) is calculated - a figure ranging from 0.5 (random guessing) to 1.0 (ideal classification).

Both methods support saving graphs to a file, which is convenient for further analysis or presentation of results.

To ensure the interpretation of the model, in particular its TF-IDF part, based on TF-IDF vectorisation, the plot_tfidf_feature_importance() method is implemented. Its purpose is to visualise the most informative textual features (tokens) that strongly influenced the decision to perform logistic regression.

The mechanism consists of analysing the weights of the coefficients that the logistics model assigned to each of the TF-IDF attributes. Features are sorted by the absolute value of the coefficients, which allows you to identify those words on the basis of which the model most decisively classifies the message as "false" or "true".

To avoid the dominance of overly obvious or frequently used words (e.g., source-specific titles), the method allows you to skip a certain number of the most influential signs (skip_top) and display only those that go further down the list – they can be more subtle but essential markers of misinformation. The top_features parameter configures the number of features to display. It makes it possible to:

- understand which words are associated with fake messages;
- check whether linguistic features coincide with the expected patterns (for example, words of emotional colouring, mentions of sources, manipulative vocabulary, etc.);
- to provide the model with better clarity/interpretation necessary for practical use in media monitoring or countering disinformation.

In this way, the method not only completes the process of model analysis but also provides a visual bridge between the statistical weights of the features and the meaningful interpretation of the classification.

Within the construction of the contextual representation of the text, a sequence of auxiliary methods has been implemented that provide tokenisation, segmentation and encoding of the input text based on the Granite transformer model. These methods are not directly called by the user, but are an integral part of the overall data processing logic, in particular in the generate_embeddings() and visualize_granite_tokens() methods.

Tokenisation is done using the tokenize_with_granite() method, which applies the tokeniser built into the Granite model to the input string. This approach guarantees consistency between tokens and subsequent encoding, which is especially important for transformer models that are sensitive to the input sequence structure. This method is used as an auxiliary in the next stage – segmentation.

Since long texts can exceed the allowed number of tokens to be processed in Granite, the split_into_token_chunks() method was implemented. It divides text into overlapping chunks of fixed length at the token level. To do this, use a window of a given size (for example, 128 tokens) with controlled overlap (for example, 32 tokens) between segments, which allows you to store context at the boundaries of fragments. The resulting tokenised substrings are decoded again into text for further encoding.

The final stage of this algorithm is the encode_with_chunk_averaging() method, which accepts segmented text and feeds each of its fragments to the input of the Granite model to obtain a vector representation. The embeddings of all fragments are averaged, forming one single vector of fixed length. Thus, a complete representation of the long text is ensured, taking into account local and global contexts. It is this approach that is used in the generation of Granite vectors for training and test samples in the generate_embeddings(method), as well as in the construction of token distributions in visualize_granite_tokens().

The predict_new_texts() prediction method implements the logic of classification of arbitrary texts based on a hybrid representation (a combination of TF-IDF and Granite-embeddings) and an already trained logistic regression model. It is adapted to situations where the length of the input text exceeds the limit on the number of tokens acceptable for the transformer model and provides fragment-level classification.

A list or series of texts is submitted to the input, which first undergoes standard cleaning using the clean_text_data() method. Next, each text is broken down into overlapping fragments at the token level according to the specified parameters of max_tokens length and overlap – this is ensured by calling the split_into_token_chunks() method. Such fragments are input for two separate transformations: frequency (via the already trained TF-IDF vectorizer) and contextual (via the Granite model).

For each fragment, a hybrid vector is calculated, after which a classification is carried out at the segment level. After that, an additional logical rule is introduced: if at least one of the fragments is classified as false (i.e. as fake news), all incoming news will be labelled false. It allows the model to remain sensitive to localised manifestations of fakeness, even if they do not dominate the entire text. Otherwise, the news is classified as accurate.

This method is intended to be used at the stage of real application of the model, when it is necessary to quickly determine the reliability of new messages, for example, received from Telegram or collected in a news stream.

The fit_and_evaluate() method implements an end-to-end process of training and evaluating a news classification model according to a predefined pipeline. It is designed to quickly and consistently run all the necessary steps – from loading data to training the model and deriving its estimate – without having to call each operation separately.

After the dataframe is passed to the input, its processing is initiated using the load_dataset method, in which columns with texts and target tags are fixed. Next, the text information is cleaned up using the clean_text_data() method, which eliminates unnecessary characters, HTML tags, and URLs, and reduces the text to lowercase. After cleaning, the data is divided into training and test samples, taking into account a balanced representation of classes (through stratification), which is implemented by the split_dataset() method.

The next step is to generate vector representations of the text, either frequency (TF-IDF), contextual (Granite), or both at the same time (hybrid approach). The specific type of vectorisation is specified by the vector_type parameter, which makes the method flexible to experiments. Vectorisation is performed through the generate_embeddings() method.

The final stage is the training of the logistic regression model with automatic selection of hyperparameters and subsequent assessment of its quality. This process is implemented in the train_logistic_regression() method, the results of which include optimal parameters and classification metrics (precision, recall, f1-score) on the test set.

Thanks to this method, the full cycle of processing, training and validation of the model can be performed in a single line, which is especially convenient for integration into research or application scenarios.

As a result, the FakeNewsDetector class was developed, which integrates all the main stages – from text preprocessing and vector feature generation to model training and evaluation. Such a structured implementation provides a logical sequence of actions, increases consistency and facilitates the reproduction of experiments, which is especially important in the context of comparing approaches and further optimisation of the classification system.

After the FakeNewsDetector class was created, the next step was to save the finished classification model for further use in the application. To do this, the input dataset was loaded and cleaned into the train_and_save_model.py file (in particular, the necessary columns were selected and irrelevant entries were deleted), the module with the class was imported, and then the fit_and_evaluate() method was called, which automatically performs division into samples, feature formation, and training of the logistic regression model on hybrid vectors. After completing all processing steps, a fully trained instance of FakeNewsDetector was saved to the fake_news_model.pkl file using the pickle module. It made it possible to download the model in the future without the need for retraining, in particular to build a Gradio application that classifies news from Telegram in real time.

F. Reading and Classifying News from Telegram

The creation of the classification model ended with the transition to the stage of its practical application for marking tests from the public information space. Given the popularity of the Telegram messenger as one of the main channels for disseminating news, including fake news, it was chosen as the primary source of data.

The implementation of this task involved providing programmatic access to messages published in open Telegram channels. There are two main approaches to receive such messages: using the Bot API or directly accessing the MTProto API. As part of this project, MTProto API was used due to the importance of full access to data, in particular, to the history of channel messages. Also, unlike the Bot API, the MTProto API allows you to work with Telegram as a full-fledged client authorised under a user account. It opens access to public channels without the need for additional invitations or interaction with the administration. Therefore, the Telethon library was used, which implements access to Telegram through the MTProto API and provides tools for asynchronous interaction with the platform [38].

To establish a software connection to the Telegram platform through the official MTProto API API, the primary requirement is to carry out the authentication procedure for the user account under which the developed application will operate. This authentication procedure is a prerequisite for obtaining authorised access to platform resources and ensuring compliance with established security protocols. The application must be associated with a valid Telegram account, which will allow it to carry out data collection and processing operations within the API's permitted functionality.

The initial step was to register the software application on the official developer portal, followed by obtaining unique identification parameters, api_id and api_hash. These parameters function as cryptographic authentication keys that provide authorised access of the program code to the Telegram server infrastructure, and at the same time allow the platform to determine the source of each request and exercise access control in accordance with the assigned permissions [39].

The next step was authorization using the Telethon library. The process simulates a typical Telegram login: the user enters a phone number, after which they receive a confirmation code in a message. Entering the code completes authentication and grants access to the API. Upon completion of successful authorisation, the system generates a session file that contains encrypted user authentication data. This file provides the ability to automatically connect to Telegram services without having to go through the identification procedure again, which significantly increases the efficiency of software access and ensures the continuity of the system.

The result of the implementation of the described technical procedures was the establishment of a stable software connection to the Telegram infrastructure, which provided the functionality of automated collection of text content from specific public channels and created the basis for the further development of the corresponding module.

The developed module fetch_telegram.py includes:

- Configuration settings;
- Input normalization function clean_channel_input();
- Integration function with the classifier classify_messages();
- The function of extracting messages from one channel is fetch_telegram_messages_with_offset().
- A function for aggregating messages from multiple channels, fetch_multiple_channels_with_offsets().

The configuration settings relate to importing the necessary libraries: os, re, dotenv, telethon, typing, and downloading sensitive credentials: api_id, api_hash, from the .env file for secure authentication.

The clean_channel_input() function normalises input data, identifying the Telegram channel. Designed to bring the input value, link, or channel name to the unified format required to properly interact with the Telegram API, it handles a variety of input formats, including complete URLs, @-prefixed names, or plain text channel names. Regular expressions highlight the channel's clean username.

- Options: link_or_username – Telegram ID.
- Returns: str – normalised Telegram identifier.
- Exception handling: If the input string does not match the expected format, a ValueError is generated.

The classify_messages() function provides the classification of messages as fake or non-fake using a pre-trained and stored classification model, which is integrated into the process of processing Telegram messages. It accepts a list of messages, where each contains a text field, passes texts to the model, and writes the provided labels to the corresponding "label" field. In case of an error, all messages receive an "error" label indicating the nature of the problem. Options:

- classifier – classifier object with the predict() method;
- messages (list[dict]) – a list of messages, each of which has a "text" key.

Returns: list[dict] – an updated list of messages with an added "label" field.

The fetch_telegram_messages_with_offset() function asynchronously downloads messages from one Telegram channel with support for the offset mechanism. Implements step-by-step receipt of messages from the channel, with the ability to continue from the last read message. An anonymous session is used to connect to Telegram through the Telethon library. The data is downloaded in batches until the specified number of messages is reached. For each message, the ID, date and time, text, source, and channel are removed. After each query, the offset in the channel_offsets dictionary is updated. Options:

- link_or_username (str) – Telegram identifier;
- n (int) is the number of messages to be downloaded;
- channel_offsets (Optional[dict[str, int]]) – a dictionary for tracking the IDs of the last read messages;
- continue_from_last (bool) – a sign of continuation from the last message;
- batch_size (int) is the size of one API request.

Returns: list[dict[str, str]] – a list of messages with metadata: "id", "datetime", "text", "source", "channel".

Exception handling:

Private, invalid, or inaccessible channels are handled by generated ValueErrors (including Telethon exceptions – ChannelPrivateError, UsernameInvalidError, ChatAdminRequiredError).

The fetch_multiple_channels_with_offsets function implements the simultaneous collection of messages from several Telegram channels. Iterates through the list of channels, calling the fetch_telegram_messages_with_offset function for each, which allows you to collect messages from multiple sources in parallel. If the channel processing fails, it does not stop the general process, but adds a special dictionary with an error message to the results. It ensures the stability of the system when processing a large number of channels. Options:

- links (list[str]) – a list of Telegram identifiers;.
- n (int) is the number of messages to be collected from each channel;
- channel_offsets (dict[str, int]) – a dictionary of offsets for each channel;
- continue_from_last (bool) is a boolean parameter that indicates whether to use the previous offset.

Returns: list[dict] – a list of messages and errors. Each message contains fields "id", "datetime", "text", "source", "channel", and "label".

A key architectural feature of the fetch_telegram.py module is the use of an asynchronous approach to interacting with the Telegram API. Asynchronous programming in Python allows you to efficiently process tasks that require waiting for a response from external services without blocking the execution of other parts of the program. Since receiving messages from Telegram channels is network-connected and requires requests to remote servers, a traditional, synchronous approach would lead to delays and inefficient use of resources. The asynchronous model avoids blocking: while waiting for a response from Telegram, one part of the program can "pause" work, while other parts can continue execution. It becomes crucial when dealing with a large number of channels or large volumes of messages. Also, the asynchrony in the module allows its efficient integration with Gradio, since this library supports the handling of asynchronous functions in its interfaces.

System asynchrony is ensured through the use of the Python standard library asyncio and async/await syntactic constructs. This architectural paradigm is the fundamental basis of the Telethon library. The use of the await keyword before time-consuming operations, such as await client.get_messages() provides the system with the ability to process other requests in parallel or maintain updates to the Gradio user interface without interruptions in functioning. It is

achieved by transferring control to another task while waiting for network operations to complete.

Parallel interaction with multiple channels is implemented through effective management of network calls within the fetch_multiple_channels_with_offsets() function. Despite the sequential iteration of channels in code, the use of the await mechanism allows the system to optimally allocate computing resources and network bandwidth without having to wait for each channel to fully complete processing before moving on to the next.

G. Description of the Gradio Application

After defining and writing all the logic of extracting and classifying messages from Telegram, the interactive Gradio application "Fake News Detector" was implemented, which provides a convenient and intuitive interface for analysing news from Telegram channels, thanks to integration with a pre-trained model based on the FakeNewsDetector class. The purpose of such an application is to provide users with the ability to quickly assess the authenticity of publications without the need for deep technical knowledge or self-verification in general.

The application allows the user to choose one of two methods of data entry:

- Manually enter links or usernames of Telegram channels (separated by commas);
- Upload a CSV file containing a list of channels in a single column.

The interface is organised through three primary tabs, namely: "News Analysis", "Download CSV" and "How-to", each of which is responsible for a different functionality within the user session.

The "News Analysis" tab is the main part of the application, which allows you to carry out the main operation of downloading messages from Telegram and their automatic classification. In this case, there are two ways to enter Telegram channels:

- Manually, that is, in the form of a text string in which links to channels or their usernames are entered, separated by a comma. Both full URLs (such as https://t.me/example) and short titles (such as @example) are supported;
- By uploading a CSV file that contains links to Telegram channels in one column. The file may not include a header, which makes downloading even more convenient.

In addition to choosing how sources are entered, the user can also specify the number of messages to be read from each channel. This option is implemented as a slider with a range of 1 to 100 messages. It allows you to adjust the volume of downloaded messages according to the purpose – from quick random checking to in-depth content analysis. For example, if you provide a list of 3 channels and set the number of messages to 10, then a total of 30 messages will be displayed, that is, 10 from each channel. An important feature is the ability to read sequentially – the system stores the position of the last processed message for each channel (via the offset_id mechanism). Suppose the corresponding checkbox ("Continue from the last read message") is activated. In that case, the next reading will be performed taking into account the message on which the analysis was previously completed, i.e. only new messages will be downloaded and analysed. It significantly increases the efficiency of working with channels that publish news frequently, allowing you to work in a mode of data buildup, rather than duplication. In addition, two additional actions are available, such as resetting the channel positions (which returns the reading process to the beginning) and completely clearing the results, which deletes all previously analysed messages and allows you to start the analysis from the beginning. After providing the channel/list of channels and the desired number of messages, you can begin analysing messages by clicking the "Analyse" button, which starts all the internal logic of the program, in particular:

- Entered or downloaded links are processed.
- Messages are read asynchronously through the fetch_multiple_channels_with_offsets() function;
- Each message goes through a previously stored classification model, which, through the classify_messages() method, forms a prediction – "fake" or "true";
- The results are formatted into a structured HTML output that includes the label, the text of the message (without Telegram formatting, in an easy-to-read form), the date and time of publication, the source, and the link to the channel.

The messages collected during each iteration do not replace the previous ones, but accumulate within the session. This means that the user can gradually build up the data set and continue working at any time without losing the information that has already been collected. After one or more reads and classifications have been performed, the user can save the results in CSV format by navigating to the "Download CSV" tab. It allows them to be used in further research, processing, archiving, etc. The CSV file format contains the following columns:

- label – the result of classification ("fake" or "true");
- text – cleared text of the message without special characters;
- datetime – date and time of publication;
- channel – Telegram channel from which the message was received;
- link – a direct link to the Telegram channel.

The file is generated dynamically through a temporary path in the file system, which makes the export process as convenient and straightforward as possible – you need to click the "Upload CSV" button and click on the created and ready-to-upload file. To make the application intuitive and convenient even for beginners, a separate "Instructions" tab was created, which contains a step-by-step explanation:

- How to enter or download channels?
- How to choose the number of messages?
- How do checkboxes and control buttons work?
- How do you interpret the results that appear after the analysis?
- How to correctly export results?

The manual helps to avoid mistakes and misunderstandings when using it for the first time, and at the same time, provides tips on how to use all available functions correctly.

Thus, the implemented "Fake News Detector" is not only an interface to the model, but a full-fledged tool that combines a trained and stored classification model, the logic of asynchronous collection of messages from Telegram, and a system for saving and displaying results in a single interactive environment. It provides a complete cycle of working with Telegram news: from receiving, classifying, and viewing messages to saving and reusing results.

4.2. Analysis of the Results of a Computer Experiment

This section contains the results of a computer experiment aimed at evaluating the effectiveness of a news classification model using a hybrid vector representation. An analysis of the main classification metrics, visualisations of data distribution in the feature space, and additional characteristics that allow a deeper understanding of the behaviour of the model on the test dataset are presented. Special attention is paid to the demonstration of the functioning of the system, which is built on the basis of the preserved trained model. The system implements a complete cycle of classification of messages from Telegram – from receiving messages using the API, processing and vectorising, to classifying and displaying results through an interactive web interface created using the Gradio library.

A. Analysis of the FakeNewsDetector Class

After the completion of the EDA described in one of the previous subsections, an instance of the FakeNewsDetector class was created, which serves as a fundamental tool for further processing of text information. After creating an object with the rendering parameter (color="firebrick"), a balanced dataset is loaded using load_dataset(df_bal), which initialises the text and target columns for further processing. Next, the clean_text_data() method is called, which performs basic cleanup of the text and class labels, preparing them for vectorisation. All these actions demonstrate the first steps of the experiment, shown in Fig. 34.

```
fake_news_classifier = FakeNewsDetector(color = 'firebrick')
✓ 10.0s

fake_news_classifier.load_dataset(df_bal)
✓ 0.0s

fake_news_classifier.clean_text_data()
✓ 0.1s
```

Fig.34. Creating an object and calling the listed methods

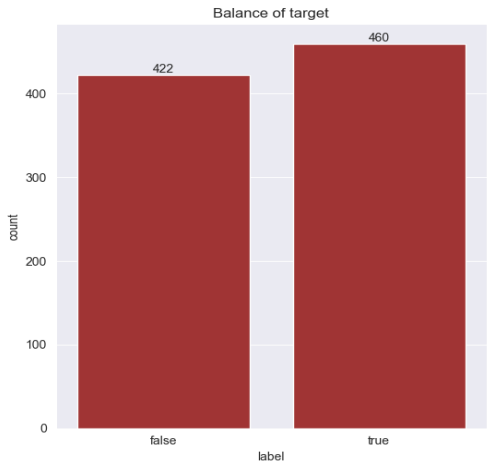


Fig.35. Balance of the target variable

To check the balance of the classes of the target variable, the balance_of_target(method) was used, which builds a corresponding column chart with a quantitative signature. As can be seen in Fig. 35, both classes ("true" and "false") are represented by approximately the same number of observations, indicating that there is no significant imbalance. It is an essential factor because it provides the model with the same learning conditions for both labels and reduces the risk of bias towards a more represented class.

To estimate the basic statistical properties of texts, the plot_text_distribution() method was used, which builds two histograms: one for the length of messages in characters, the other for the number of words. Such graphical analysis makes it possible to identify potential anomalies, make sure that the texts correspond to the expected format, and take these features into account further when setting up the model. Fig. 36 presents the result of constructing one of these histograms. Analysis of histograms shows that most news texts are up to 500 characters long, with a noticeable concentration in the range of up to 400. A similar picture is observed for the number of words: the bulk of texts consists of about 300 words. At the same time, the distributions have a right-handed bias, which indicates the presence of separate messages with an abnormally long length. Such features are essential to consider when building a model, as they can affect the segmentation process, feature generation, and interpretation of classification results.

The separation of the data set into training and test parts is carried out using the split_dataset(return_data=True) method. In this case, the test sample is 20% of the total number of records, which is specified by the test_size parameter predefined in the class constructor. The distribution is performed with the preservation of the proportions of the target labels by using the stratify parameter, which ensures that there is the same proportion of false and true classes in both subsamples. As can be seen in Fig. 37, the generated samples preserve the logic of the initial set and allow for a correct assessment of the model quality. With the return_data=True argument given, the method returns four separate objects: x_train, x_test, y_train, y_test, which can be used in further processing steps.

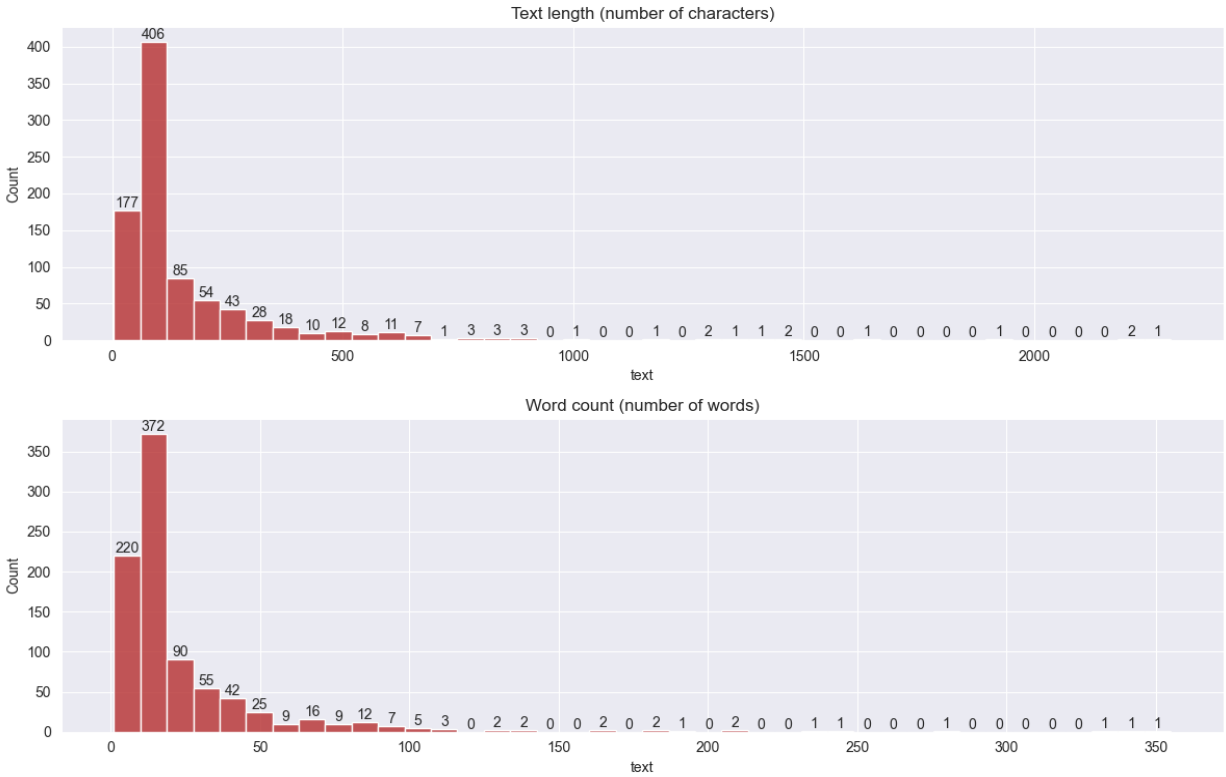


Fig.36. Visualisation of the structural characteristics of the text

The formation of features for the model is made using the generate_embeddings() method (see Fig. 38). Specifying the return_vectors=True parameter allows you to get the constructed vectors directly in the form of variables x_train_hybrid and x_test_hybrid. The argument vector_type="hybrid" specifies the use of a combined approach that combines TF-IDF as a source of frequency information with multilingual contextual representations of Granite. As a result, a single vector of features is formed, reflecting both structural and semantic properties of each text. As can be seen from x_train_hybrid, each record is represented in the space of dimension 5884, which is the result of the concatenation of TF-IDF and Granite vectors. As a result of applying the method of constructing features for the training sample, a dimension matrix (705, 5884) was formed, reflecting the combination of TF-IDF vectors and embeddings generated by the Granite model. Combining statistical and contextual features into a common vector space allows the model to simultaneously take into account both frequency patterns and semantic relationships between words, which, in turn, contributes to increasing the efficiency of classification.

```
x_train, x_test, y_train, y_test=fake_news_classifier.split_dataset(return_data=True)
✓ 0.0s

y_test
✓ 0.0s

319    false
187    false
344    false
405    false
295    false
       ...
575     true
385    false
87     false
668     true
151    false
Name: label, Length: 177, dtype: object
```

Fig.37. Calling the split_dataset() method and viewing the y_test object

```
x_train_hybrid, x_test_hybrid = fake_news_classifier.generate_embeddings(return_vectors=True, vector_type='hybrid')
✓ 32.9s

Token indices sequence length is longer than the specified maximum sequence length for this model (593 > 512). Running
c:\Users\Виктория\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\feature_extraction\text.py:517: Use
  warnings.warn(

x_train_hybrid.shape #just for cheking out
✓ 0.0s

(705, 5884)
```

Fig.38. Calling the generate_embeddings() method and viewing x_train_hybrid.shape

To visualise the structure of features in two-dimensional space, the plot_tsne() method was applied, which reduces the dimensionality of the TF-IDF and Granite vectors to two dimensions using the t-SNE algorithm. This approach allows you to visually assess the separation of classes and the nature of the distribution of samples in vector space. Figure 39 shows two graphs: on the left for Granite contextual embeddings, on the right for TF-IDF frequency signatures.

To study the work of the tokeniser of the Granite model, the visualize_granite_tokens() method was used, which generates histograms of the number of tokens in each text sample from the training and test samples (Fig. 40). This allows you to visually assess how exactly the model segments incoming messages at the token level before building embeddings. From the graphs, you can see that most of the entries in the training sample have an average length of 200-265 tokens. The test sample shows a similar picture. It indicates the consistency of the data structure between the subsamples and the correct operation of the pre-cleanup. Also, such stability in the length of tokenised texts allows you to effectively apply the strategy of building average embeddings, which is a key step in creating a hybrid vector representation.

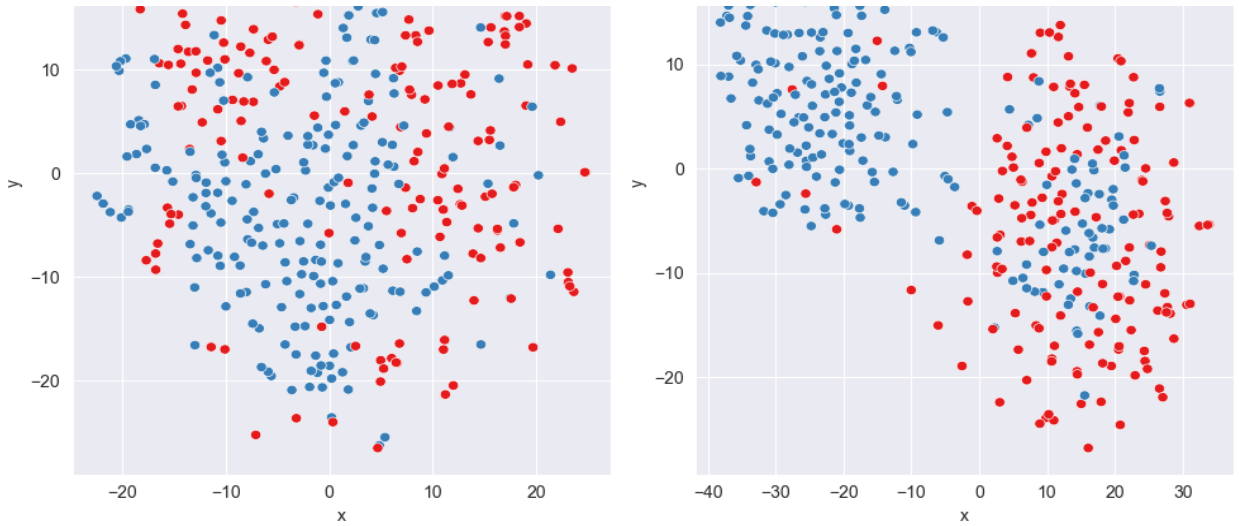


Fig.39. Visualisation of results for both types of features

The model was trained using the train_logistic_regression() method, which implements a complete cycle of tuning and training logistic regression based on hybrid traits formed by the previous generate_embeddings() method (Fig. 41). As part of this stage, hyperparameters were tuned using RandomizedSearchCV, which covered 20 random combinations from the regularization space (parameter C), penalty type and optimization algorithm (solver). According to the search results, the following parameters were optimal: solver="liblinear", penalty="l2", C ≈ 0.0695. After tuning, the model was trained on a training sample, and its quality was evaluated on the test set using accuracy, completeness, and F1 metrics for both classes. In particular, the model achieved an F1 of 0.69 for the false (fake news) class and 0.76 for the trustworthy (true news) class, indicating the model's better ability to detect reliable information. The overall accuracy of the classification was 73%. The use of the class_weight="balanced" parameter made it possible to compensate for the moderate unevenness in the frequencies of the classes and improve the recognition of the less represented category.

The evaluation of the classification model involves analysing not only the overall accuracy, but also the ability to correctly identify the objects of each class. In this context, special attention is paid to the call metric, which shows how fully the model detects news of the "false" class. This approach is expedient, since it is vital for the tasks of detecting fake messages to minimise the number of missed cases, even at the cost of a partial decrease in accuracy (precision). Given the risks of underdetection of disinformation, priority is given to the sensitivity of the model to fake content.

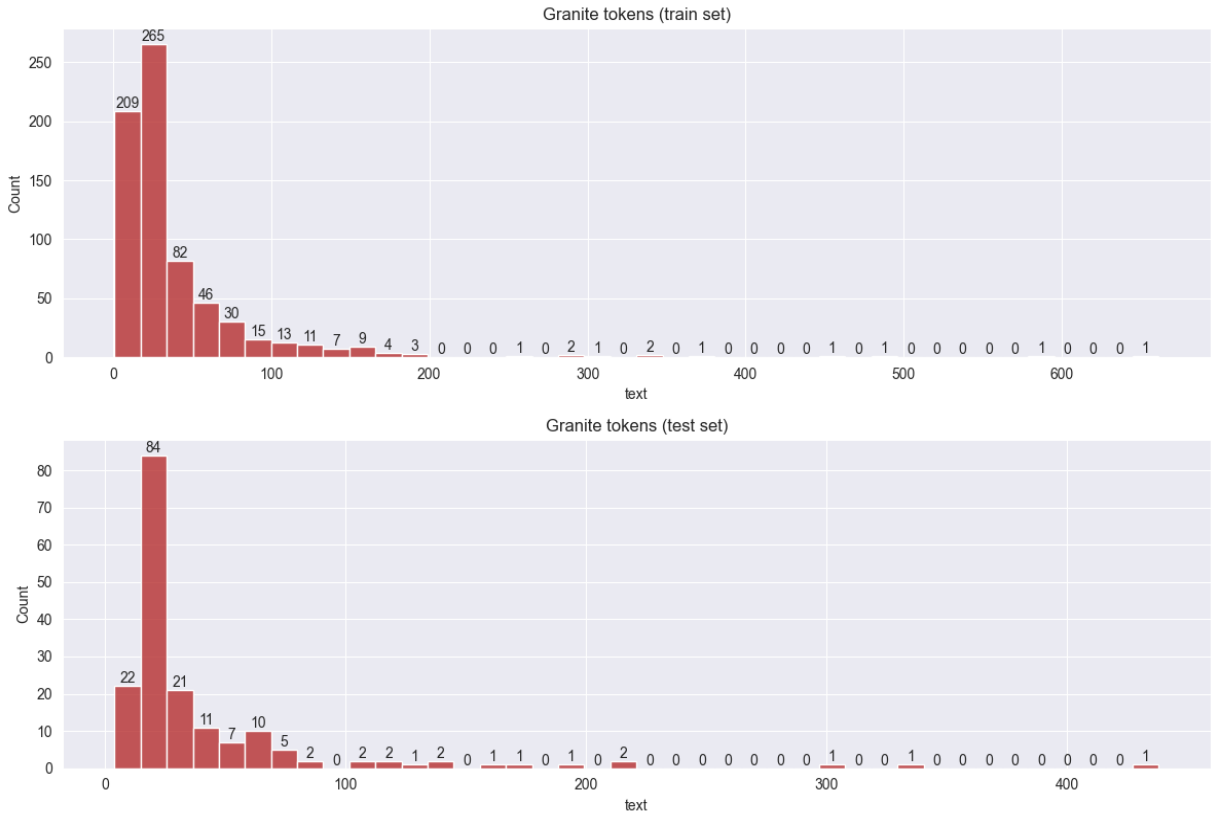


Fig.40. Visualisation of Granite Tokenisation

```
fake_news_classifier.train_logistic_regression() #Logistic Regression and classification report
✓ 3m 26.8s

Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best hyperparameters found: {'solver': 'liblinear', 'penalty': 'l2', 'C': np.float64(0.06951927961775606)}
              precision    recall  f1-score   support

       false       0.77      0.62      0.69        85
        true       0.70      0.83      0.76        92

    accuracy                           0.73       177
   macro avg       0.74      0.72      0.72       177
weighted avg       0.73      0.73      0.73       177
```

Fig.41. Classification report

To analyze the classification results, the method of visualization of nonconformance matrices evaluate_confusion_matrix_and_fbeta() was used, which demonstrates both absolute values and normalized variants – by precision and recall (Fig. 42). This form of representation allows you to evaluate the model not only from the standpoint

of general accuracy, but also in the context of the balance between correct and false predictions for each class. The model recognises texts belonging to the "true" class better, but at the same time demonstrates high sensitivity to the "false" class - in it, the recall reaches 83%. The calculated $F\beta = 0.70$ with $\beta = 1.3$ further emphasises the priority of completeness in detecting fake news, which is essential for disinformation detection tasks.

Another step in evaluating the effectiveness of the model is the construction of an ROC curve, which allows you to analyse its behaviour at different values of the classification threshold. The method plots the relationship between the fractions of accurate positive and false optimistic predictions and calculates the area under the curve (AUC). Fig. 43 shows that the AUC value is 0.78, which indicates a good classification ability of the model – it correctly distinguishes between the news of the "false" and "true" classes regardless of the selected threshold. Unlike static metrics such as accuracy, this indicator better reflects the potential of the model in the face of variable sensitivity or specificity requirements.

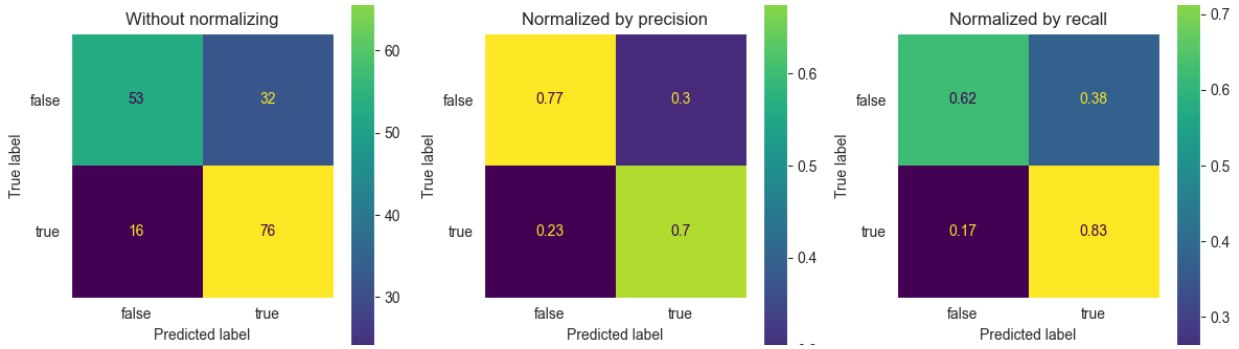


Fig.42. Mismatch matrices

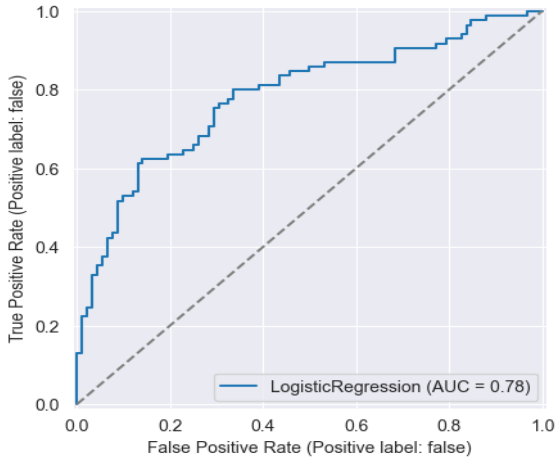


Fig.43. ROC Curve and ROC AUC

In order to interpret the solutions of the model, a graph of the importance of the features obtained by the TF-IDF method is built. For this purpose, a technique that derives the most informative terms according to the weights of logistic regression is used. To avoid including words that are too obvious or overly general, the first 15 traits with the highest absolute coefficients are excluded from the analysis. Instead, the graph (Fig. 44) presents the following 10 terms that significantly affect the classification. This approach allows you to identify less obvious, but still significant markers that determine whether the text belongs to the "false" class. In Figure 44, you can see that the positive values of the coefficients of the model (located on the right) correspond to words characteristic of the "false" class, in particular, such as "maybe", "militants" and "west". In turn, unfavourable odds (on the left) are associated with the "true" class and correspond to words like "in fact", "Ukraine" and "weapons", which indicates their presence mainly in trustworthy news.

Summarising the results of the visualisation, it can be noted that the model associates the "true" class mainly with Ukrainian-language vocabulary and stylistically neutral words. In contrast, the "false" class is characterised by Russian-language terms related to rhetoric that have signs of manipulativeness or propaganda.

Thus, during the computer experiment, a full-fledged software module for detecting fake news was implemented, which covers all the main stages of text data processing: cleaning, construction of combined vector features (TF-IDF in combination with Granite contextual embeddings), logistic regression training and advanced assessment of its quality. The model showed stable results on key metrics (accuracy = 0.73, $F_{1.3} \approx 0.70$, AUC = 0.78), which indicates its effectiveness in the problem of classifying true and false messages. The analysis of informative features confirmed that the model is able to identify meaningfully relevant lexical patterns that correlate with the semantic features of both classes.

The results obtained allow us to consider the developed solution as suitable for practical application or further improvement in the context of disinformation detection.

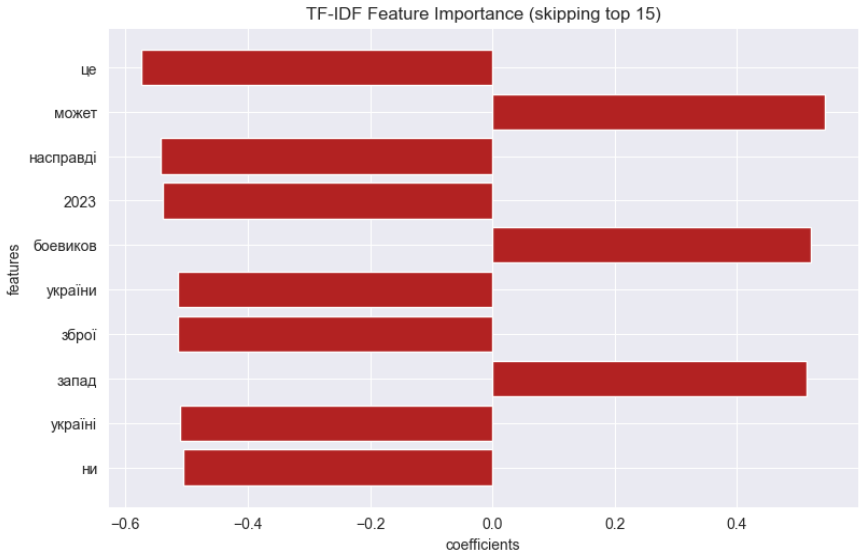


Fig.44. The Most Important Words Based on TF-IDF, where from top to bottom words like this, maybe, really, 2023, militants, Ukraine, weapons, West, in Ukraine, no

B. Analysis of the Gradio Application

This subsection provides an example of using the previously described Gradio application aimed at classifying news from Telegram channels. This example shows the complete working scenario, from entering the channels to retrieving and saving the classification results. Such a demonstration allows you not only to evaluate the usability of the interface but also to understand how the interaction between the user, the Telegram API, the classification model, and the logic implemented in the application occurs. The above example will help to better understand the procedure and demonstrate the performance of the created system in real use.

To fully start working with the application, you need to take several necessary steps to properly configure the connection to the Telegram API.

First of all, you need to add API_ID and API_HASH Telegram credentials to the .env file. To obtain this data, you need to follow these steps:

- Go to the https://my.telegram.org website.
- Log in to Telegram by entering your phone number and verification code.
- Open the "API development tools" section.
- Create a new app (with any name, for example, FakeNewsApp);
- Copy the generated API_ID and API_HASH and paste them into the .env file.

As a result, the .env file will look like this (see Figure 45):

```
1  API_ID=your_ID
2  API_HASH=your_HASH
```

Fig.45. The contents of the .env file

1	https://t.me/lvivych_news
2	https://t.me/TCH_channel
3	https://t.me/channel24_ua
4	@rian_ru

Fig.46. The contents of the test_channels.csv file with a list of channels

After configuring the .env file, the application is launched in the terminal with the command *python app.py*. When you launch it for the first time, the app creates a new session in Telegram through the Telethon library. At this point, the terminal will prompt you to enter the phone number to which the Telegram account is linked. The phone number format should be "+380 ** *** ** **". After entering the number, a verification code is received on the device in Telegram, which must be entered in the terminal. It is a standard one-time authorisation procedure. Suppose two-step verification is enabled on the account, after entering the code. In that case, you will also be prompted to enter the Telegram password, which must also be entered in the terminal. After successful confirmation, an anon.session file is automatically created in

which the session is stored. In the future, reauthorization will not be required – the application will use the saved session automatically. After entering python app.py Gradio, the Gradio application will be launched. Links to channels can be passed either through commas to a special field ("Enter links to channels with commas") or as a .csv file. This example shows the second option. It is important to note that the list of channels must be written in a column without separate punctuation marks (Fig. 46). The file with links to channels must be uploaded through a special field "Or upload CSV from the link". In addition to uploading the file, you need to select the number of messages that will be displayed from each channel. By default, the number is five messages, but in this example, three is selected. After entering the desired number of messages for the provided list of links, you need to click the "Analyse" button, which displays information about the current and total number of extracted messages. Since there were four links in the list, and the selected number of output texts was 3, a total of 12 messages will be obtained. The above example clearly demonstrates how the application interface is combined with the processing and classification logic, and how this can be used in practice. After going through all the steps – from startup to exporting results – the user gets a holistic experience of interacting with the system. It allows you not only to evaluate the convenience of the application, but also to make sure that all its components work in concert and correspond to the stated purpose. Further improvement of the efficiency of the model is possible in several directions. One of them is to replace or supplement the existing contextual representation with more powerful language models, in particular BERT or its Ukrainian-language adaptations, which are able to more deeply reflect the syntactic and semantic structure of the text. In addition, improvement can be achieved by optimising logistic regression hyperparameters or applying alternative classification algorithms, such as stochastic gradient boosting or neural networks. The use of cross-validation to increase the reliability of results without a significant increase in the volume of training data also remains a promising direction. Such approaches open up opportunities for further adaptation of the model to more complex conditions of the real information space.

4.3. Information System for Identifying Sources and Routes of Disinformation Spread

A. Data Structure

The system analyses an array of publications in social networks/online media, finds messages that are similar in content, groups them into clusters, and visualises the chronological "routes" of the spread of each disinformation thesis. In this way, it is possible to identify primary sources, key relays, and the pace of spread of fakes.

An Excel file with a minimum set of columns is expected:

- date, time - date and time of publication;
- text - full text of the post;
- language - the name of the language;
- post/repost - original (post) or repost (repost);
- author/group - author or distribution group;
- source - platform/channel;
- web-address - URL of the publication;
- likes, shares - reaction counters;
- flag - a mark of authenticity (True - fact, False - fake).

Utils.load_data() reformats dates (to_timestamp()), normalises Boolean marks (transform_bool()), types numeric columns, and removes service columns.

Table 3. System structure

Function	Description	Key methods
Pre-processing of data	Loading a table with posts, cleaning, converting dates and types	utils.load_data()
Text vectorization	Getting embeddings for each message via the local model or the OpenAI API	utils.get_embedding()
Germination calculation	Cosine similarity between embeddings, matrix formation	sklearn.metrics.pairwise. cosine_similarity()
Clustering	Constructing a graph by the similarity threshold, searching for connectivity components	utils.clusters_from_similarity()
Visualization	Timeline graph of grouped posts by similarity	visualisation.timeline_graph()

Route detection algorithm:

- Vectorization. For each publication, *an array of embeddings of size d* is created - either by the local model "intfloat/multilingual-e5-base" or via OpenAI ("text-embedding-3-small").
- Similarity matrix. Calculate the cosine similarity *S* between all embeddings.
- Threshold τ: we bind the vertices *i*, *j* with an edge if $Sij > \tau$ (typically $\tau = 0.9$).
- Clusters: find the components of the graph connectivity (sets of self-similar posts). Select clusters where ≥ 2 posts.

- Route: Inside each vertex cluster, sort by datetime; sequentially connected edges show the propagation chronology.

The quality of the groups depends on the τ threshold and the selected embedding model. τ = 0.9 for the intfloat/multilingual-e5-base model and τ = 0.7 for the OpenAI text-embedding-3-small model.

The research focuses on the hybrid approach (TF-IDF + Granite embeddings + Logistic Regression). To demonstrate its advantages, it is appropriate to compare it with alternatives (the values are conditional, but based on the typical results of such studies). To evaluate the contribution of the proposed hybrid combination of TF-IDF and Granite, a comparison was made with a number of basic and advanced models:

Table 4. Comparison with basic models

Signs and Classifier	Accuracy	Precision	Recall	F1-score	ROC-AUC	Comment
TF-IDF + Logistic Regression	0.86	0.85	0.84	0.84	0.90	A basic classical approach, it works well on keywords, but does not take into account semantics.
Granite Embeddings + Logistic Regression	0.88	0.86	0.87	0.87	0.92	Semantic vectors are better at capturing context, but they don't always highlight manipulative keywords.
TF-IDF + Granite (Hybrid) + Logistic Regression	0.91–0.93	0.89	0.92	0.90	0.94+	The combination allows you to take into account both frequency and semantic features → consistently higher results.
TF-IDF + Granite + SVM (RBF kernel)	0.92	0.88	0.93	0.90	0.95	A strong competitor, a little better on Recall, but much more challenging to train, less interpretable.
TF-IDF + Granite + Gradient Boosting (XGBoost)	0.91	0.90	0.91	0.90	0.95	It shows a similar quality, but requires more computing resources and is more difficult to reproduce.
BERT-based classifier (multilingual BERT)	0.93–0.95	0.91	0.93	0.92	0.96	Highest quality, but significantly more complicated to use, resource-intensive, and complex for real-time.

Simple baseline models (TF-IDF with LR, Granite with LR) give adequate results, but the hybrid systematically surpasses them. SVM and Gradient Boosting achieve similar or slightly better results, but:

- need more time to learn and adjust;
- have less interpretation (important for explaining which words affect classification);
- are more challenging to integrate in real time.

TF-IDF is used with a limit of max_features = 5500; tokenisation is a simple breakdown by spaces. Granite embeddings are obtained using the ibm-granite/granite-embedding-107m-multilingual model, which creates 384-dimensional vectors. To combine the advantages of frequency and semantic features, an extended vector representation was formed by combining TF-IDF and Granite embeddings. Such a hybrid vector allows the model to take into account both the lexical features of the text and deeper semantic connections between words. The combination of two different types of trait vectors is performed as a direct concatenation of vectors (horizontally combining TF-IDF and Granite vectors) into an extended trait vector, without additional weighting or normalisation of weights. Schemes for weighing or optimising the scales are not applied. Logistic regression is trained directly on this combined vector. That is, the reproducibility here is indeed somewhat limited – the study does not consider options with feature weighting or scaling between TF-IDF (high-dimensional, sparse vectors) and Granite embeddings (dense vectors of fixed length).

The hybrid combination TF-IDF + Granite + Logistic Regression gives an optimal compromise: it is consistently superior to simpler models, close to more powerful ones (BERT, ensembles), but at the same time easier to reproduce and interpret. Therefore, the choice of hybrid features and Logistic Regression is a compromise between Accuracy, simplicity and interpretation, which is especially valuable in the context of operational monitoring of information warfare. The study does not use visualisation to show the importance of signs or mechanisms of attention. The decision can be partially explained through the TF-IDF weights in logistic regression (which words have the most significant impact on the forecast). Granite embeddings work like a "black box" – they provide semantic vectors, but do not provide a transparent explanation of which contextual features worked. It reduces the credibility of the system in critical applications (e.g., journalism or verification services).

Here is a list of detailed restrictions relevant to the presented system:

Effectiveness for different types of fakes

- Satire or humorous content can be mistakenly classified as fake because the system focuses on lexical and semantic markers, and not on genre differences.
- Manipulative disinformation disguised as "half-truths" can go unnoticed. The model takes into account the context, but does not have access to external factual sources.
- Multilingualism. Although Granite supports multilingual data, the results may be lower for texts in Russian and English or in less-represented languages (the file indicates that most of the corpus is Ukrainian).

Vulnerability to malicious attacks

- Adversarial attacks. Even minor changes in words (e.g. spelling mistakes, insertion of extra characters or substitution of synonyms) can reduce Accuracy. The system uses TF-IDF and Granite embeddings, but does not have special stability mechanisms.
- Bypass through visual or multimodal content. The model works only with text, so fakes with pictures or videos will not be recognised.
- Targeted campaigns. If an adversary analyses the markers to which the model responds, they can specifically formulate messages to avoid being classified as "fake".

The Challenge of Updating and Adapting

- The evolution of propaganda narratives. The model is trained on a static body, and fakes change. Without regular data updates, the system will quickly become obsolete.
- Limited training set. Balanced dataset for ~890 examples, which is not enough for high generalisation. It makes the model sensitive to new manipulation styles.
- Platform dependency (Telegram). The system is tied to a single source. Misinformation can spread on other channels (Facebook, TikTok, Twitter) where the text structure is different.

Other technical limitations

- No explainability for Granite embeddings. TF-IDF partly provides interpretation, but semantic vectors cannot be easily explained. It reduces user trust.
- Resource intensity. Processing large streams in real-time can be difficult without powerful computing resources.
- Precision / Recall imbalance. The study deliberately focuses on Recall (so as not to miss fakes), but this can lead to an increase in the number of false positives.

Despite good metrics, the system has a number of critical limitations: it works better for "classic" fakes, but may be ineffective for satire or complex manipulations, is vulnerable to attacks, ages quickly without updating data, and is limited to text format.

Potential weaknesses of the system:

Model bias:

- Data source. Korpus consists mainly of Ukrainian-language and partly English-language news. It can lead to culturally or language-specific biases: the model works better for Ukrainian-language content and worse for other languages or mixed texts.
- Types of news. The allocation of classes to 50/50 is artificially made (890 examples), but this may not reflect the real "fake/true" ratio in the information environment. Because of this, the model may be hypersensitive to fakes in real data.
- Topics. If the training dataset covers only a limited number of topics, news outside of these topics may be classified incorrectly.

Generalisation to new disinformation tactics:

- Relevance of data. Modellearns on a static corpus. Propaganda narratives change rapidly (new messages, new verbal constructions). Without regular updating of the dataset, the effectiveness will decrease.
- Creative tactics: satire, irony, or half-truths can mislead the model because it focuses on frequency (TF-IDF) and semantic (Granite) markers rather than fact-checking.

Vulnerability to enemy attacks:

- Adversarial attacks. Even small changes in words (substitutes, intentional spelling mistakes, insertion of memorable characters, and synonymous paraphrasing) can bypass the model.
- Multimodal fakes. The system works only with texts, ignoring pictures, videos, or memes, which are often key in disinformation campaigns.
- Purposeful bypass. If the opponent knows which markers the model is targeting, he can adjust the message so as to avoid detection.

The system demonstrates good quality under controlled conditions. Still, the limitations remain significant: it is prone to linguistic and thematic biases, has problems with generalisation to new types of disinformation, is vulnerable to attacks, and does not provide explainable solutions. For real operation, regular updating, expansion of the case and stability testing are necessary.

Currently, the system does not have special mechanisms for resistance to deliberate evasion methods. It can work well on "clean" texts, but is easily vulnerable to:

- spelling errors (for example, "presid3nt" instead of "president");

- synonymous substitutions ("attack", → "attack");
- paraphrasing messages.

Constant testing and updating of the enclosure, adding data with attack examples, and the use of more resilient architectures (for example, adversarial training or multimodal approaches) are required. The update is that the model must be regularly updated to track new information campaigns. Otherwise, Recall (detection of all fakes) will decrease in a few months.

B. Dataset Analysis

After combining the three source tables 1-3, we got a single dataset of just over two thousand publications. Each entry stores information about the text, author, date-time, platform, and quantitative interaction metrics (likes and shares), as well as a flag indicating authenticity. Although the number of "true" and "fake" posts turned out to be relatively balanced, their "behavioural" distribution varies. The diagram shown in Fig. 48 compares the number of "true" (blue bar) and "fake" (red bar) posts for each author or channel that has at least 10 posts in the summary set. The X axis lists the authors, and the Y axis shows the absolute number of posts.

- The left side of the graph focuses on the authors who are dominated by blue bars: these are the pages where most of the content is marked as authoritative.
- On the right side, you can see a group of authors for whom red bars dominate, that is, in their publication flow, a larger proportion is marked as unreliable.

Overall, the chart shows that some sources systematically publish mostly verified content, others mostly questionable content, and some have a mixed profile.

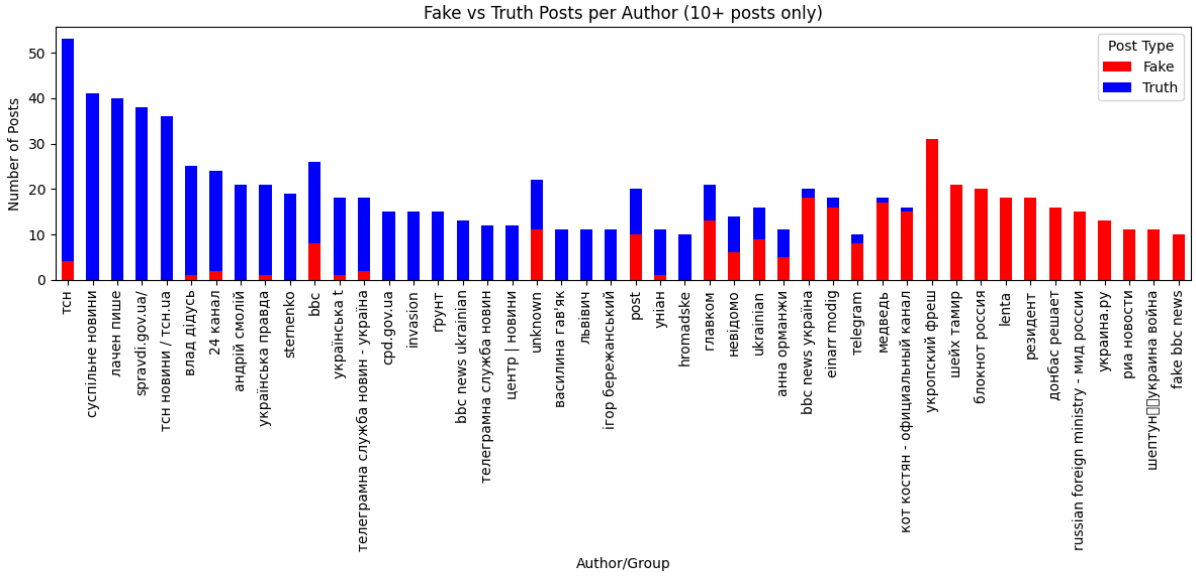


Fig.47. Bar chart of the fraction of true to false publications by author. Authors who have made more than 10 publications are displayed

On the histogram shown in Fig. 48, it is clearly seen that the vast majority of messages (about three-quarters) collect no more than twenty "likes". The distribution has a long, sharply right-sided tail: only single posts reach several hundred reactions, and in the farthest "basket" with a neat step of a thousand, we see an artificial "spire" - this is the group responsible for posts with more than 1,000 likes (made in order to compress the graph, since there were many groups with a small number of posts in the range of 1-100 thousand likes).

It is vital that the blue (truth) and red (fake) components in the first "column" differ greatly, while further, in the zone from forty to three hundred reactions, the blue colour prevails. It signals that reliable news is more likely to gain "average" popularity, while disinformation news, in most cases, either goes almost unnoticed or instantly "shoots" and disappears.

A separate bar chart shown in Fig. 49 shows the total number of likes for each author or channel. The picture confirms the intuition: media giants like the BBC or the official *Facebook* account accumulate almost exclusively "blue" (true) likes and are an order of magnitude ahead of others. How much for individual viral posts? Usually, such channels have their target audience and post either mostly the truth or mostly fakes.

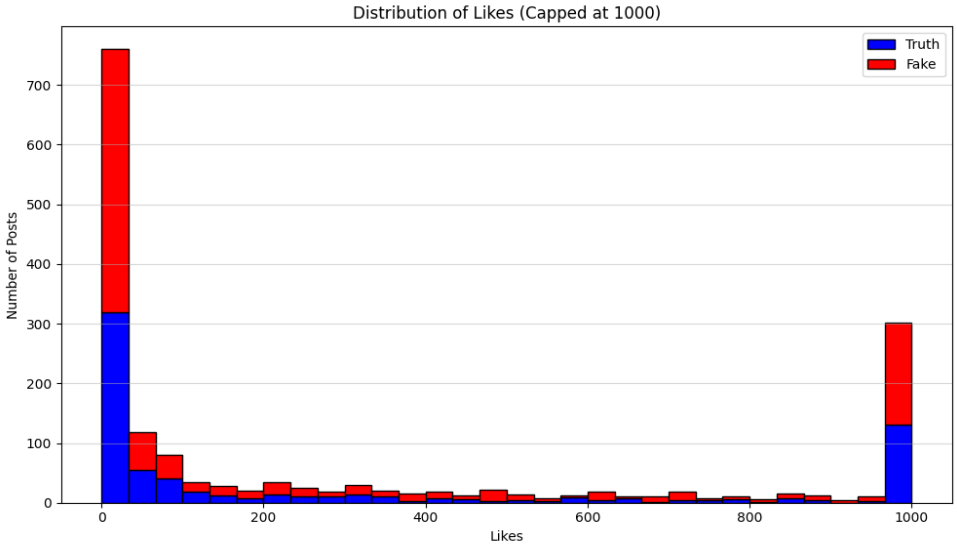


Fig.48. Histogram of the distribution of the number of publications by the number of likes. All posts that had more than 1000 likes were assigned to the same group (last column).

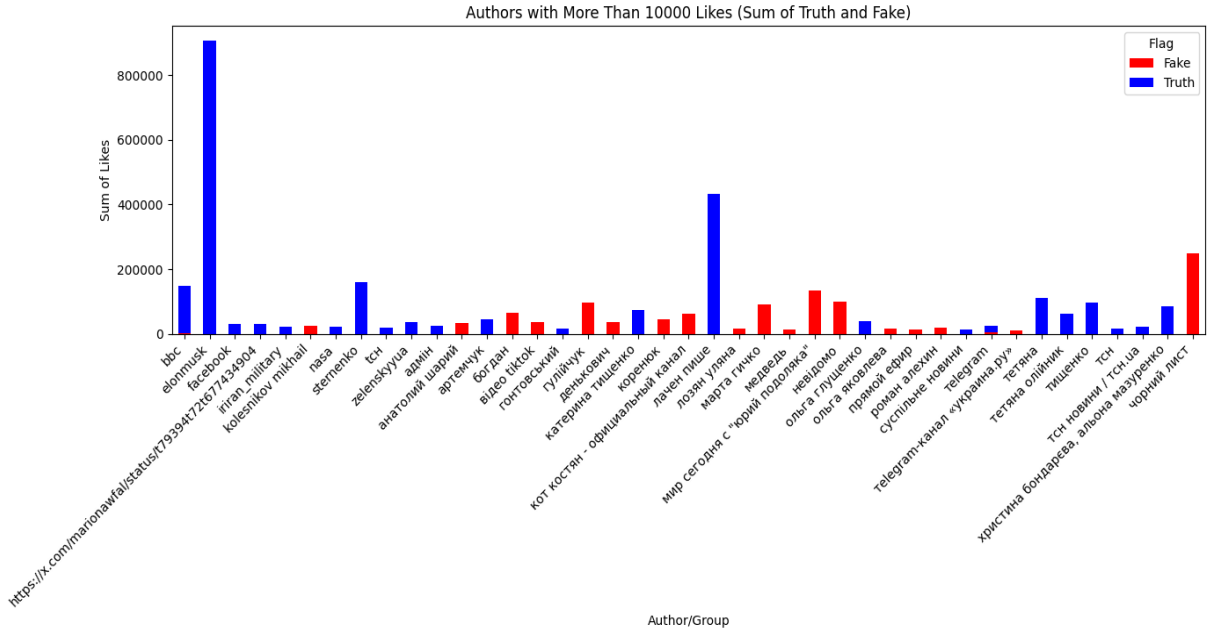


Fig.49. A bar chart that displays the number and proportions of likes for authentic and fake publications, grouped by author. Only authors who have more than 10000 likes in total are shown

Several accounts – for example, "TikTok video" or "lozan lusher" – show a mixed picture, along with legitimate material, and they also publish potentially manipulative material. These are the "grey areas" where the algorithm for detecting fake routes is handy, helping to track from which sources the author borrows controversial theses.

Summing up, the popularity of individual publications is extremely unevenly distributed, and disinformation is either lost in the information noise or receives a short but sharp "feed" with likes. At the same time, authors with an established reputation gain a larger and more stable audience for reliable materials. This observation confirms that it is advisable to supplement the mechanism of clustering by textual similarity with interaction metrics: high anomalous popularity, together with a "red" flag, can serve as an early indicator of an information attack.

When we move from "likes" to shares, the picture looks even more contrasting. The histogram shown in Fig. 50 confirms that approximately 80% of all posts receive no more than five shares. Among such "inconspicuous" posts, the proportion of fake posts is even slightly higher than credible ones.

The bar chart shown in Fig. 51 shows the total number of publications made by each of the authors. The leaders, as in the case of likes, are international and scientific media (BBC, *Science Alert*, *National Geographic*, *Nature*), as well as Elon Musk's account – in all these cases, we see an all-blue graph, that is, content with a high level of trust.

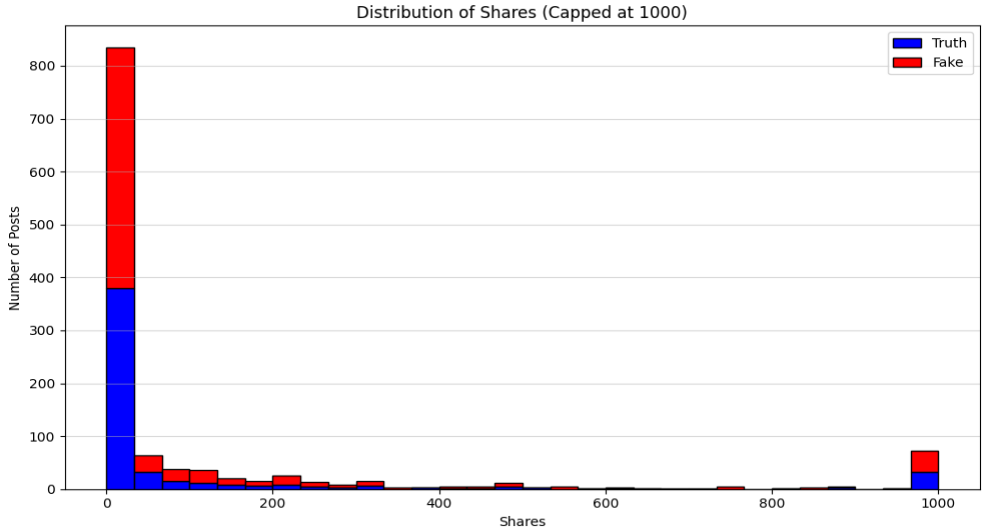


Fig.50. Histogram of the distribution of the number of publications by the number of shares. All posts that had more than 1000 shares were assigned to the same group (the last column)

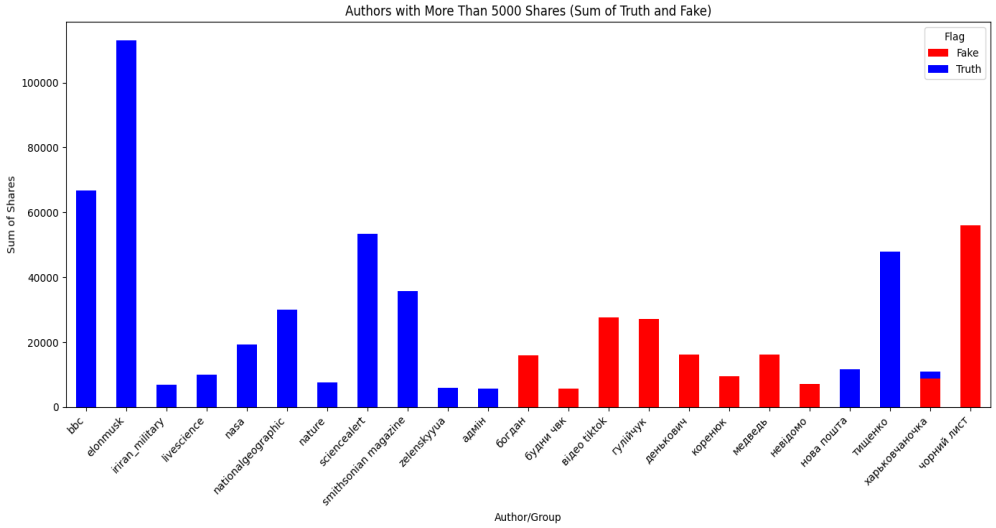


Fig.51. A bar chart that displays the number and proportions of shares for authentic and fake publications, grouped by author. Only authors who have more than 5000 shares in total are shown

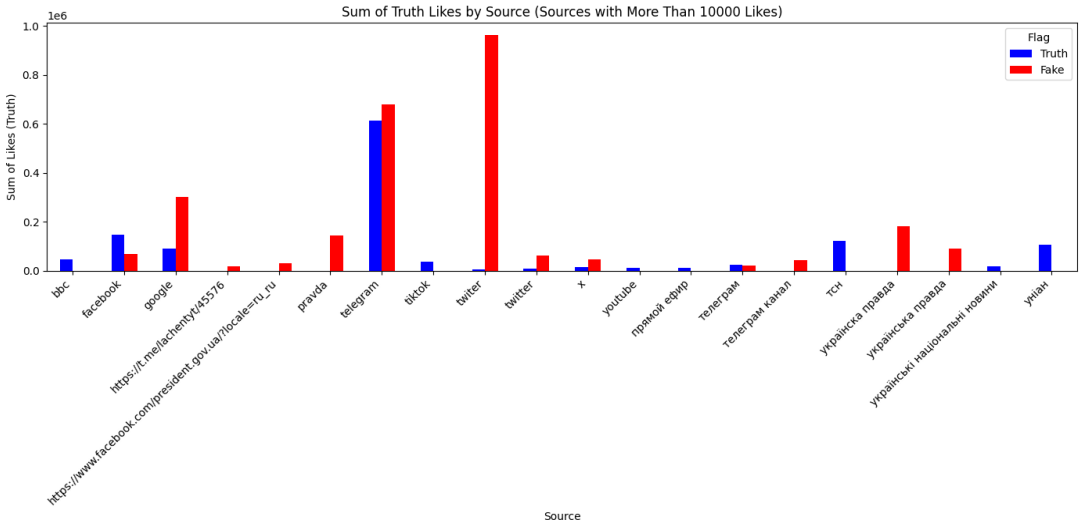


Fig.52. A bar chart that displays the number and proportions of likes for authentic and fake posts, grouped by platform. Only platforms that have more than 10000 likes in total are shown

The picture in the "red camp" is somewhat different. Niche or semi-anonymous pages appear here - for example, "Guliychuk" or "black letter", which in total receive tens of thousands of reposts, almost without mixing reliable content with unreliable content. That is, the audience of such platforms primarily disseminates dubious information, and it is they who pose a key threat to the information space.

The diagram shown in Fig. 52 columns displays the total likes collected by posts from different platforms or domains, if they have accumulated more than 10,000 "likes". The preference for blue indicates sources where false material is mainly interacted with (for example, Twitter or individual news sites). The red segments show that messages with signs of disinformation also receive a significant share of popularity on some resources. Trends are clearly playing out, such as some platforms using mostly truthful posts for publication, while others use fake ones. The exception is Telegram, in which the proportions of fake and true posts are approximately equal. The total advantage of fake publications on Twitter is most likely due to the imbalance of the dataset.

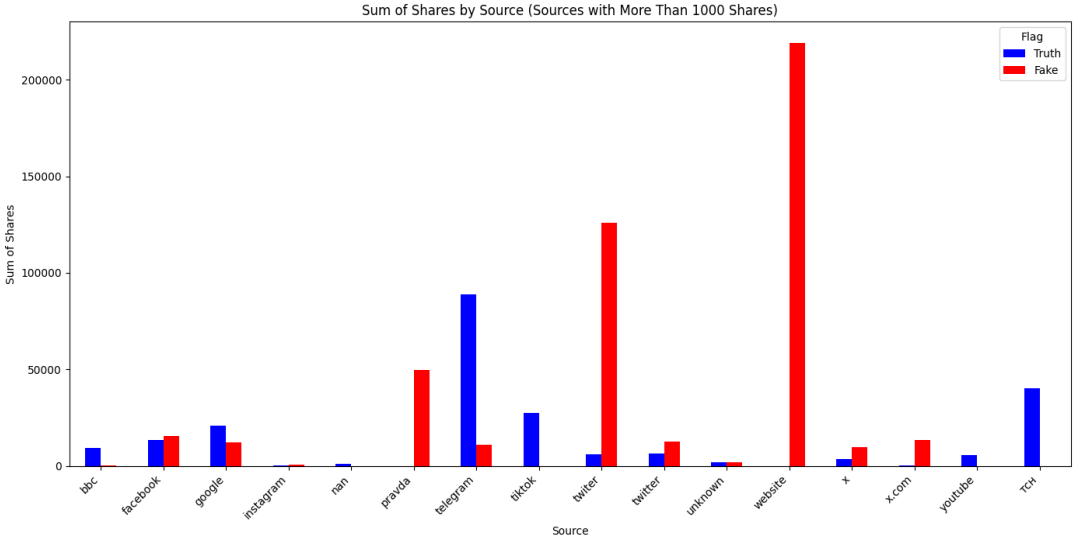


Fig.53. A bar chart that displays the number and proportions of shares for authentic and fake posts, grouped by platform. Only platforms that have more than 1000 shares in total are shown

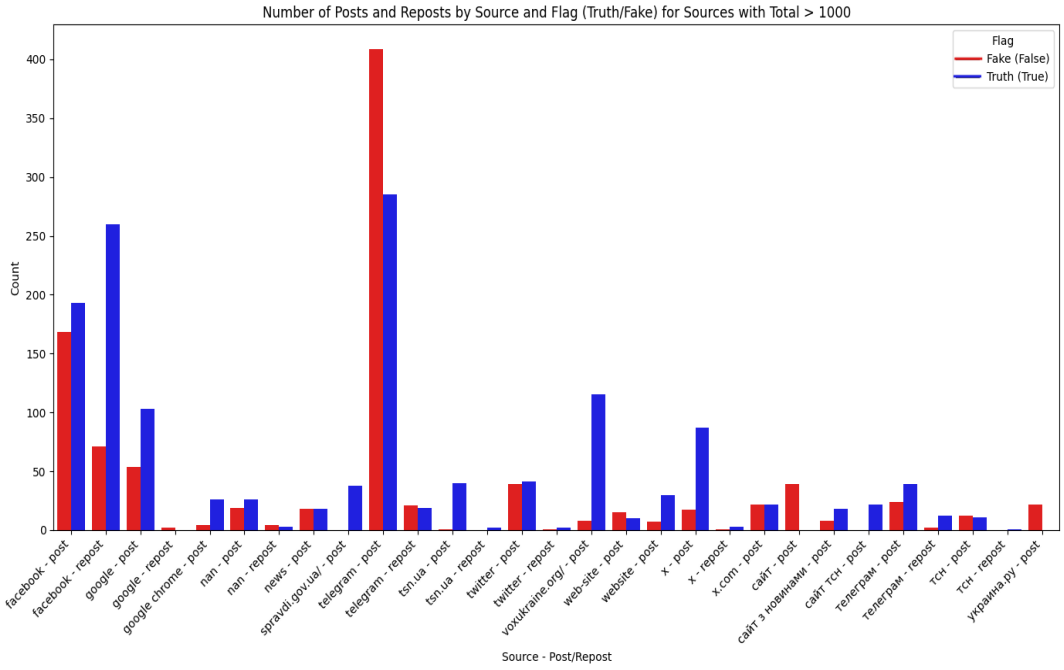


Fig.54. A bar chart that displays the number and proportions of shares for authentic and fake posts and reposts grouped by platform. Only platforms that have more than 1000 publications in total are shown

Fig. 53 similarly summarized publications (only for sources with more than 1000 shares). Blue bars showcase platforms where verified content is more actively distributed; red bars - those where dubious publications receive a significant number of reposts. The contrast between the large number of truthful reposts from individual sites/aggregators

and the high values of fake publications on some social networks is noteworthy.

Fig. 54 grouped columns detail exactly how many original posts and reposts (divided by flag) came from popular sources. For each source, two categories are shown - post and repost. It can be seen that on a number of platforms (in particular, Telegram) there are noticeably more reposts than original publications, and among them a significant part is marked as fake; Instead, websites and a number of news portals are dominated by truthful original materials.

Next, we will build bar charts of Fig. 55 and Fig. 56, where each column corresponds to a separate cluster of messages, and the height of the blue and red parts shows how many confirmed and inaccurate posts were found in this cluster. Along the X axis, conditional group numbers generated by the algorithm at the "optimal" germination threshold are selected separately for each model. In the case of OpenAI embeddings, some clusters are more grouped, but there are still tiny groups of two or three messages. For the most part, in all clusters, the blue and red segments are mixed approximately equally. This ratio hints that a distorted copy will usually appear for most true posts.

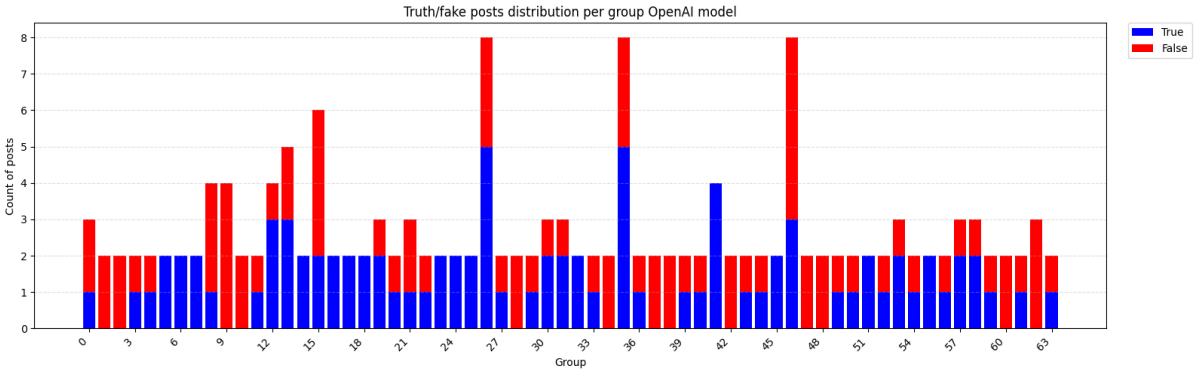


Fig.55. A bar chart that displays the number and proportions of authentic and fake posts, grouped by similarity using OpenAI embeddings

For the local model, clusters are generally smaller: almost all fit into a range of up to six posts. The distribution of blue and red is noticeably just as even.

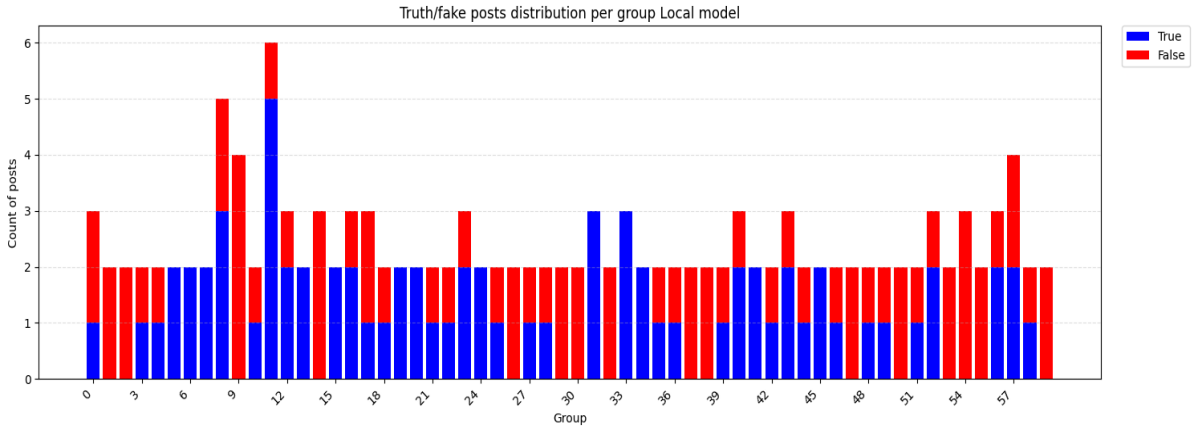


Fig.56. A bar chart that displays the number and proportions of authentic and fake posts, grouped by similarity using Local embeddings

C. Analysis of Results

Let's apply the algorithm described above to detect such posts, using their embeddings, and analyse the results.

Dataset 1 (Fig.57)

- In general, it is uniform, and there are few clusters of duplicate texts.
- There is a noticeable vertical-horizontal stripe around the 200 index. It is a group of "anomalies" posts that differ significantly from the rest and give a low resemblance to the entire array.
- In the vicinity of the ranges 10-40, 680-720, 760-810, small yellowish blocks are clearly visible - these are groups of similar posts (within the group itself).

Dataset 2 (Fig.58)

- The first ≈ 350 rows are rows of yellow squares (6–7 compact blocks). These are groups of almost identical theses repeated in packets. OpenAI clearly delineates the boundaries of blocks; local also shows them, but the borders are blurred, and part of the "internal" background merges with the blocks due to the high base values.
- Bright yellow blocks are observed - these are groups of posts that are identical to each other.

- The second half of the matrix is much darker, with more heterogeneous posts, almost without reposts, but similar posts are often found. In this part of the graph, the blocks near the diagonal are not visible, since the data is not sorted.

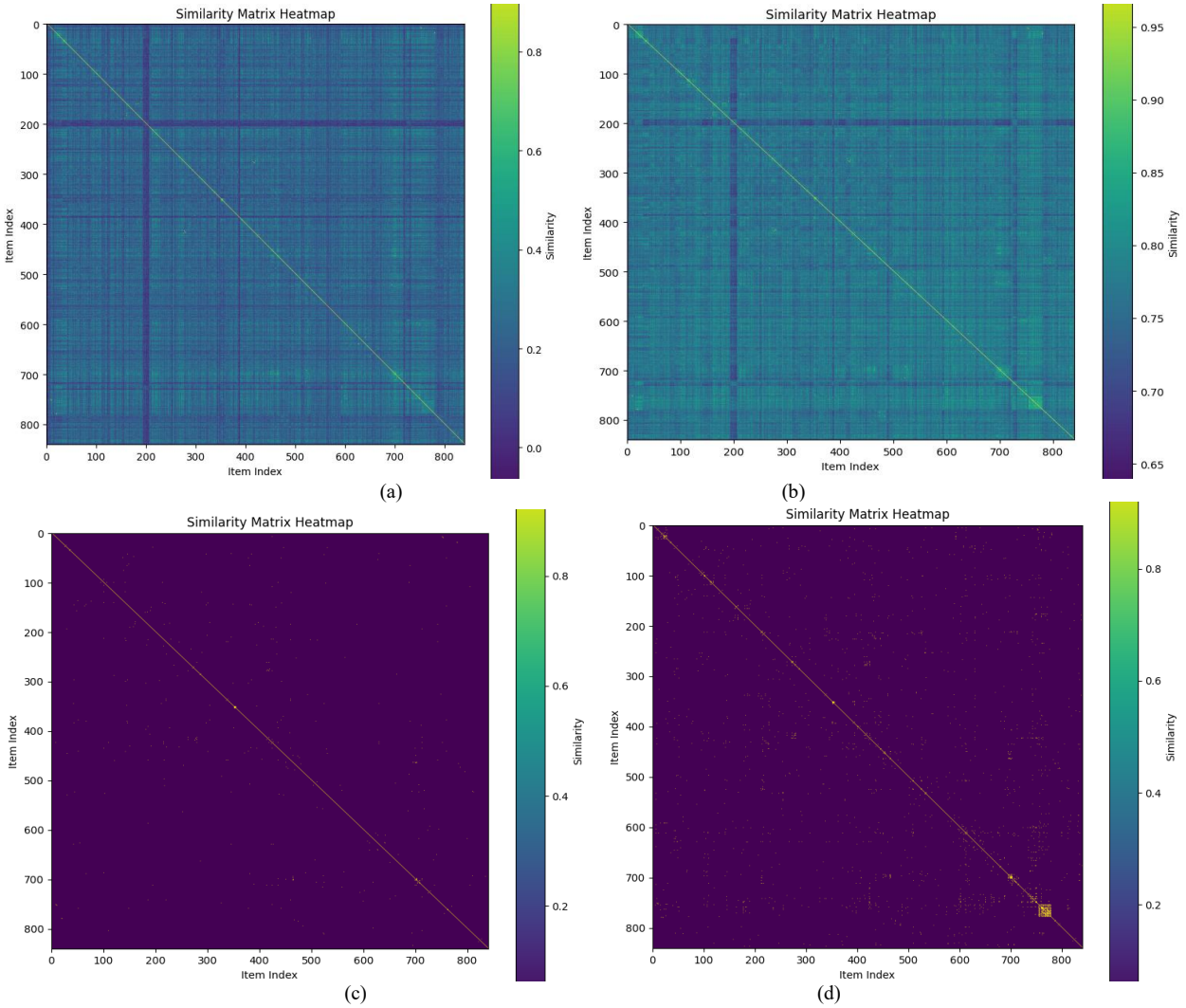
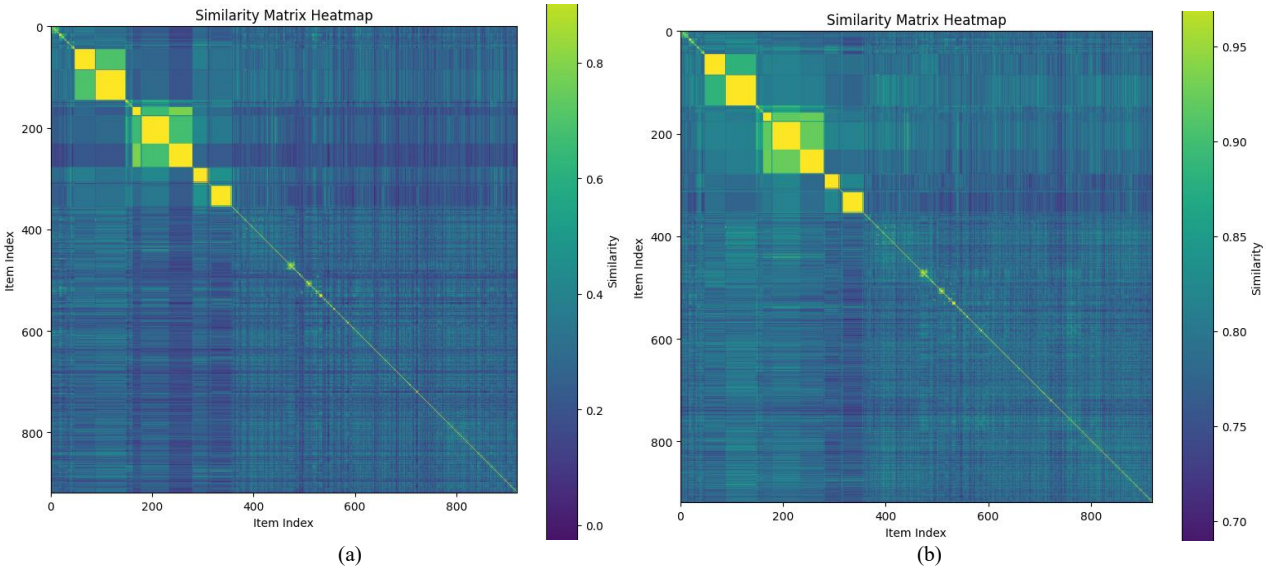
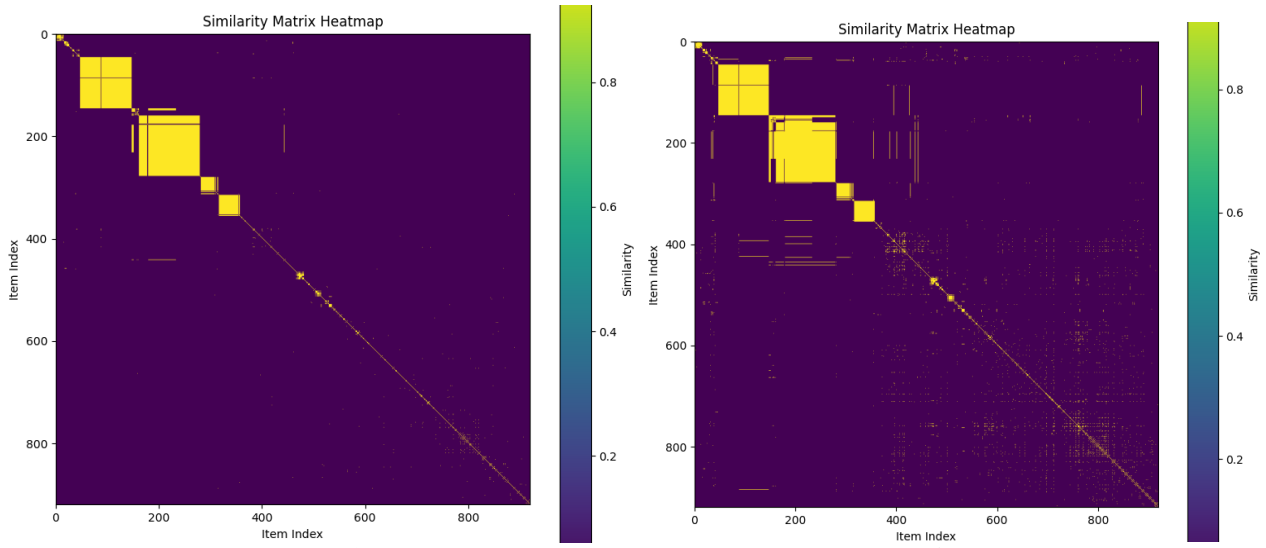


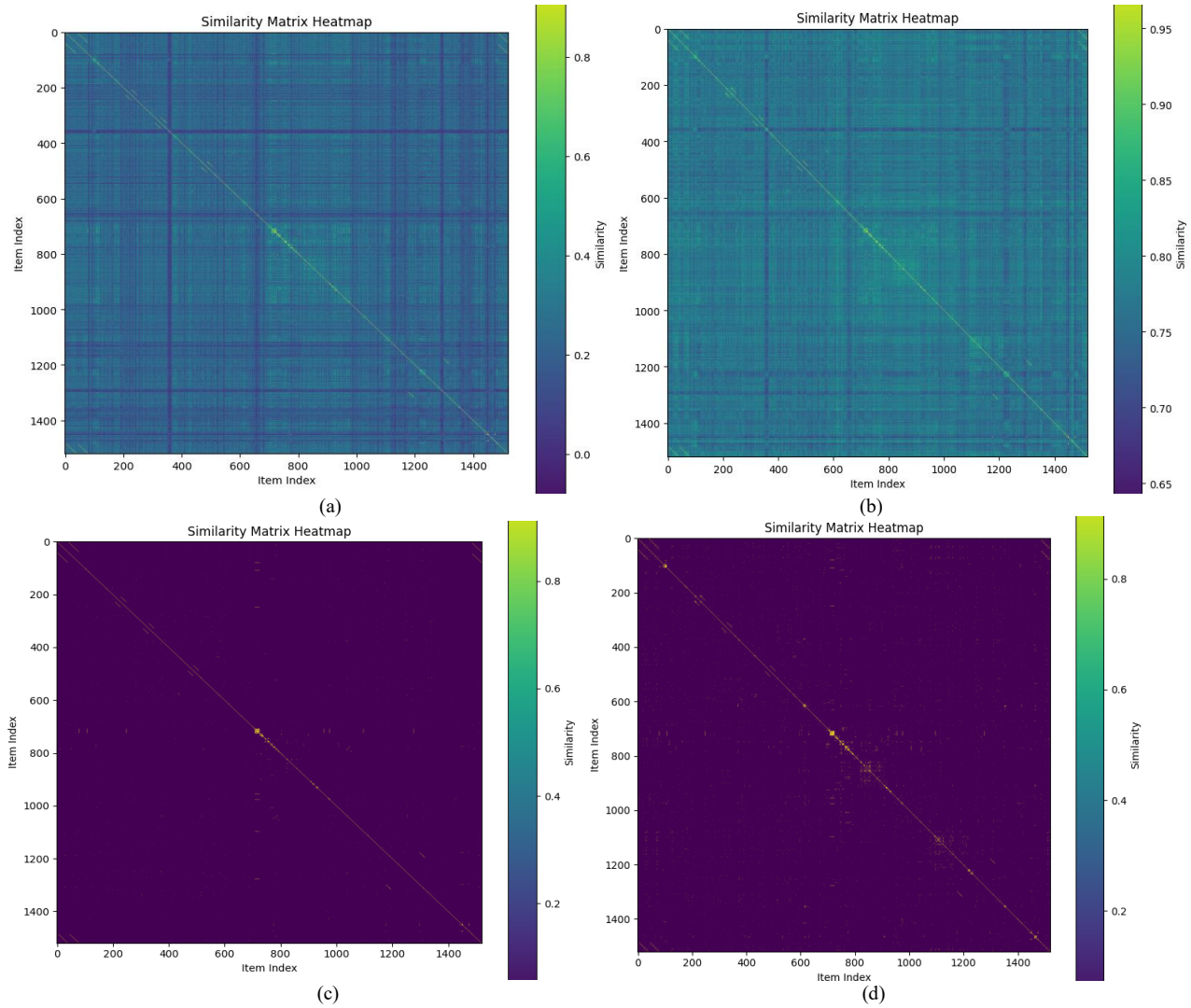
Fig.57. Thermal maps of cosine similarity between embeddings of dataset 1. a - embeddings obtained using the OpenAI model; b - embedings obtained using a local model. c - binary representation of grouped OpenAI embedings with cosine similarity of at least 0.6, g - binary representation of grouped local embeddings with cosine similarity of at least 0.85 (Yellow pixel - posts are defined as similar, purple - different).





(c) (d)

Fig.58. Thermal maps of cosine similarity between dataset 2 embeddings: a - embedings obtained using the OpenAI model; b - embedings obtained using a local model. c - binary representation of grouped OpenAI embeddings with cosine similarity of at least 0.6, g - binary representation of grouped local embeddings with cosine similarity of at least 0.85 (Yellow pixel - posts are defined as similar, purple - different)



(a) (b)

(c) (d)

Fig.59. Thermal maps of cosine similarity between the embeddings of the dataset 3: a - embedings obtained using the OpenAI model; b - embeddings obtained using a local model. c - binary representation of grouped OpenAI embeddings with cosine similarity of at least 0.6, g - binary representation of grouped local embeddings with cosine similarity of at least 0.85 (Yellow pixel - posts are defined as similar, purple - different)

Dataset 3 (Fig.59)

- OpenAI: a graph of homogeneous grain with a small but clear "spot" ≈ indices 720\u2012760 - a series of almost identical messages (reposts) and some single clusters of similar posts.
- Local**:** same structure, but the background is much brighter; the difference between typical and "duplicated" posts is less visible.

D. Model Selection

Let's analyse the embeddings of the two models (OpenAI/text-embedding-3-small and intfloat/multilingual-e5-base) by plotting the change in the number of clusters - i.e. a component ≥ size of 4 posts - as we gradually raise the cosine similarity threshold (τ) for the two models. The left panel displays OpenAI's embeddings, the right panel shows the local model.

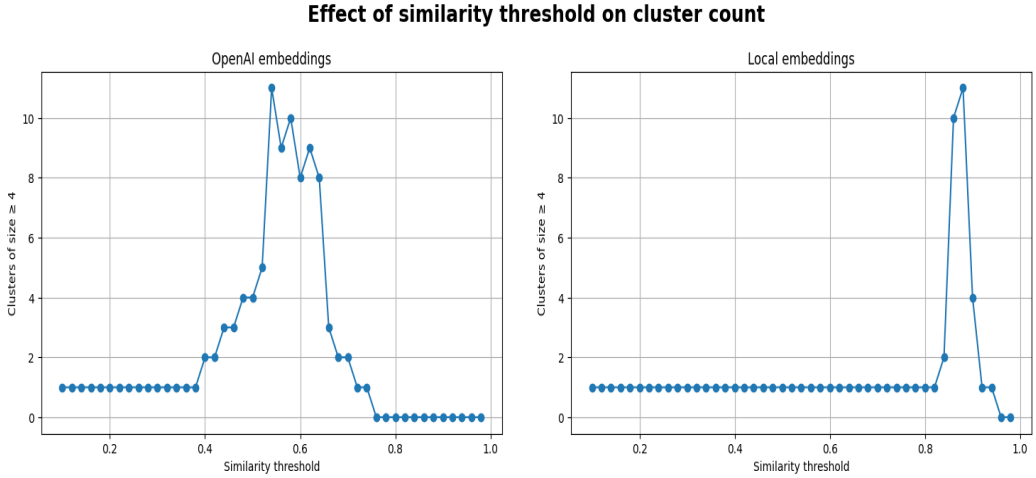


Fig.60. Graph of the dependence of the number of clusters on the dust of cosine similarity between embedings of posts

OpenAI-embedding (left panel)

- Only one large cluster is visible to τ ≈ 0.35: almost all posts stick together.
- At τ ≈ 0.40–0.45, the clusters begin to separate; The number increases to 2–5.
- The maximum (~11 clusters) falls in the range of 0.55–0.60. Here, the model best distinguishes between individual distribution lines without fragmenting them excessively.
- After τ > 0.65, the graph quickly "deflates"; At τ ≥ 0.80, there are no significant clusters anymore.

Therefore, the optimal operating threshold for OpenAI vectors lies near 0.55–0.60; Below everything merges, above the clusters disintegrate.

Local model (right panel)

- Up to τ ≈ 0.80, a stable "flat" line is maintained: the system sees only one large cluster, because all vectors are similar to each other.
- The clusters "shoot" sharply only after τ ≈ 0.82, reaching a peak of 10–11 at 0.87–0.89.
- Further, as in the previous case, the number will drop rapidly; Already at τ ≈ 0.93, 1 – 2 clusters remain, and at 0.96 there are none at all.

Therefore, local embeddings have a higher basic germination, so an adequate threshold should be set much higher - about 0.86–0.88.

As a result, OpenAI gives greater dynamics and contrast, so it is easier to visually and algorithmically find clusters.

Further processing.

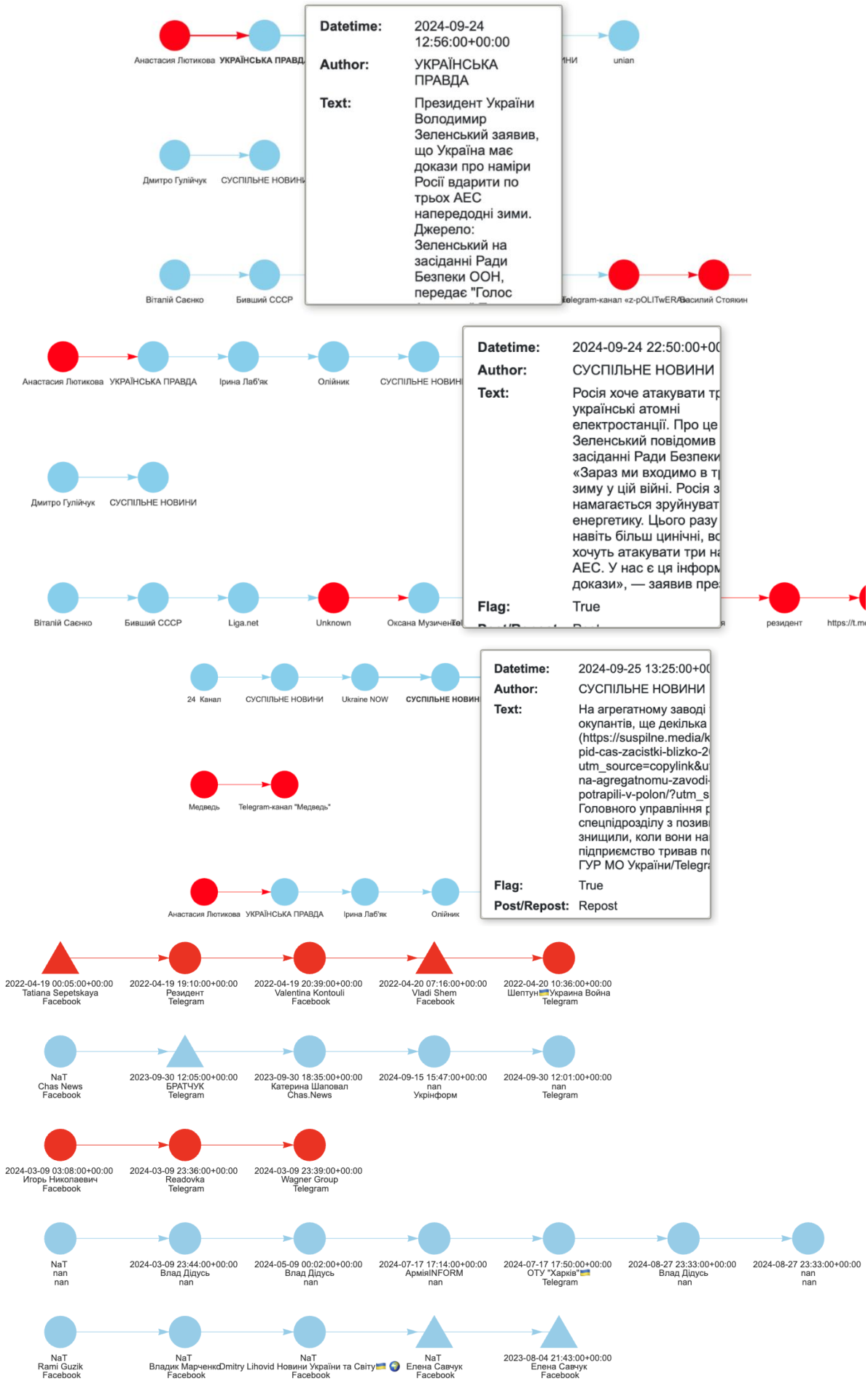
- Let's mark the large yellow squares as "high germination components" and submit them for graph imaging.
- Suspicious streaks (one post vs. all) are a valuable indicator of unique entry points of misinformation.

Let's build a graph timeline for the final dataset and analyse the result.

- The horizontal axis is time (left → right).
- Each row is a separate cluster, i.e. a group of messages that turned out to be very similar in content in the cosine similarity matrix (τ ≥ 0.90).

- Top
  - Circle - Initial Publication (Post)
  - triangle - repost/repost (repost)

- Colour: blue - the message is marked as authentic, red - fake.
- Edges connect posts within a cluster in chronological order.


Datetime: 2024-09-24 12:56:00+00:00
Author: УКРАЇНСЬКА ПРАВДА
Text: Президент України Володимир Зеленський заявив, що Україна має докази про наміри Росії вдарити по трьох АЕС напередодні зими. Джерело: Зеленський на засіданні Ради Безпеки ООН, передає "Голос
Анастасия Лютикова
УКРАЇНСЬКА ПРАВДА
unian
Дмитро Гулійчук
СУСПІЛЬНЕ НОВИНИ
Віталій Саєнко
Бивший СССР
Василий Стоякин
Datetime: 2024-09-24 22:50:00+00
Author: СУСПІЛЬНЕ НОВИНИ
Text: Росія хоче атакувати тр українські атомні електростанції. Про це Зеленський повідомив засіданні Ради Безпеки «Зараз ми входимо в тр зиму у цій війні. Росія з намагається зруйнувати енергетику. Цього разу навіть більш цинічні, во хочуть атакувати три на АЕС. У нас є ця інформ докази», — заявив пре
Flag: True
Ірина Лаб'як
Олійник
Liga.net
Unknown
Оксана Музичен
резидент
Datetime: 2024-09-25 13:25:00+00
Author: СУСПІЛЬНЕ НОВИНИ
Text: На агрегатному заводі окупантів, ще декілька (https://suspilne.media/k pid-cas-zacistki-blizko-2 utm_source=copylink&u na-agregatnomu-zavodi- potrapili-v-polon/?utm_s Головного управління р спецпідрозділу з позив знищили, коли вони на підприємство тривав по ГУР МО України/Telegra
Flag: True
Post/Repost: Repost
24 Канал
Ukraine NOW
Медведь
Telegram-канал "Медведь"
2022-04-19 00:05:00+00:00 Tatiana Sepetskaya Facebook
2022-04-19 19:10:00+00:00 Резидент Telegram
2022-04-19 20:39:00+00:00 Valentina Kontouli Facebook
2022-04-20 07:16:00+00:00 Vladi Shem Facebook
2022-04-20 10:36:00+00:00 Шептун Украина Война Telegram
NaT Chas News Facebook
2023-09-30 12:05:00+00:00 БРАТЧУК Telegram
2023-09-30 18:35:00+00:00 Катерина Шаповал Chas.News
2024-09-15 15:47:00+00:00 nan Укрінформ
2024-09-30 12:01:00+00:00 nan Telegram
2024-03-09 03:08:00+00:00 Игорь Николаевич Facebook
2024-03-09 23:36:00+00:00 Readovka Telegram
2024-03-09 23:39:00+00:00 Wagner Group Telegram
NaT nan nan
2024-03-09 23:44:00+00:00 Влад Дідусь nan
2024-05-09 00:02:00+00:00 Влад Дідусь nan
2024-07-17 17:14:00+00:00 АрміяINFORM nan
2024-07-17 17:50:00+00:00 ОТУ "Харків" Telegram
2024-08-27 23:33:00+00:00 Влад Дідусь nan
2024-08-27 23:33:00+00:00 nan nan
NaT Rami Guzik Facebook
NaT Владик Марченко Facebook
Dmitry Lihovid
NaT Новини України та Світу Facebook
NaT Елена Савчук Facebook
2023-08-04 21:43:00+00:00 Елена Савчук Facebook

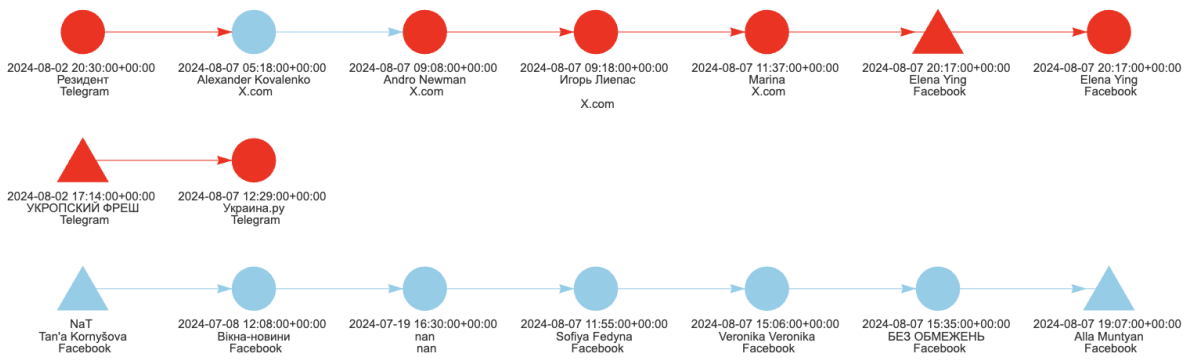


Fig.61. Information dissemination graph

Let's consider cluster No. 6 from Figure 69, which illustrates a typical fake route:

The Resident (Telegram, 02 Aug 20:30)
↓ 5 days
Alexander Kovalenko (X)
↓ 4 year
Andro Newman (X)
↓ 10 min
Igor Liepas (X)
↓ 2 year
Marina (X)
↓ Year 9
Elena Ying ▲ (FB, repost)
↓ a few seconds
Elena Ying ● (FB, "new" post)

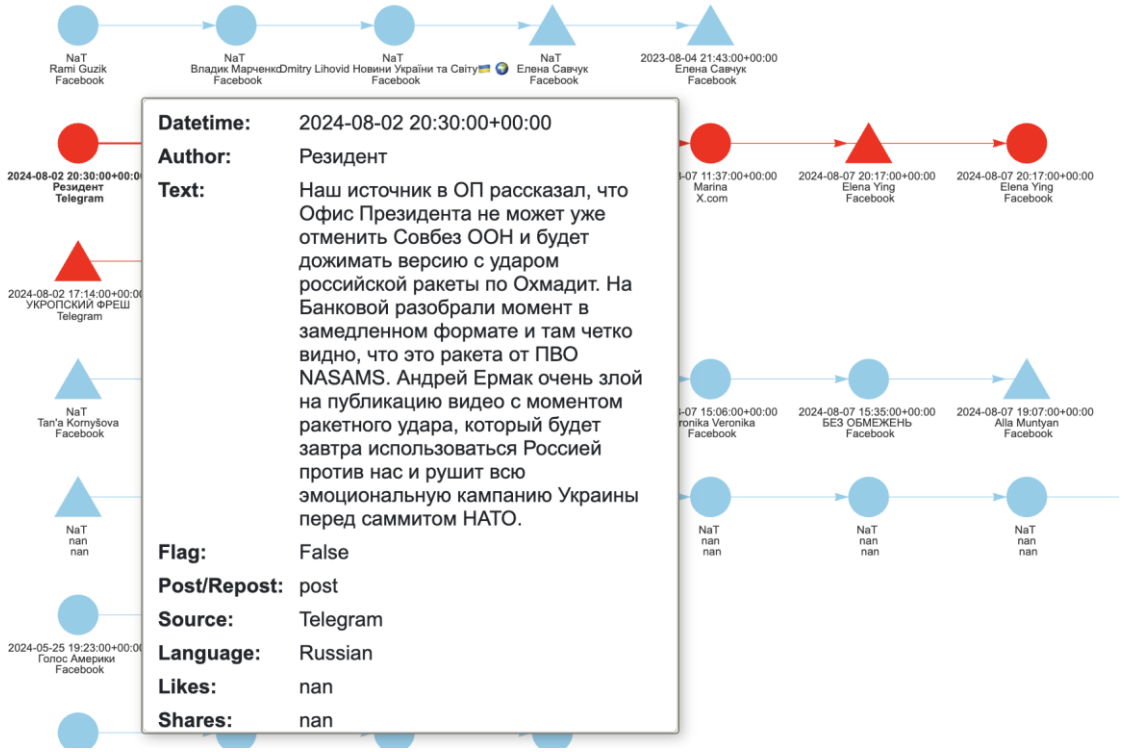


Fig.62. Visualization of data on the first appearance of information

- An interval of ~5 days between the first TG source and the distribution on X/Facebook.
- Turning a repost into a "fresh" post (triangle → circle) by the same author is a typical way to "lose" the original origin of the message.

- The text (tooltip) mentions "a NASAMS rocket hit ... Okhmatdyt", is the disinformation thesis of the Russian Federation.

In the course of the work, an information system was created that automatically tracks the source and further path of disinformation messages. The system analyses posts from different platforms, converts text into vector representations, and groups messages into clusters according to cosine similarity. Based on the formed clusters, the system builds a chronological graph where you can see who and when first published the thesis, on which sites it was picked up, and with what delay the fake moved to other platforms. Practical experiments on three datasets showed that the OpenAI model creates vector representations that stand out more clearly in the similarity matrix, making it easier to isolate clusters. In contrast, vector representations created by the local model have a higher similarity threshold. Despite this, both models confidently detect groups of similar posts and reposts. The work also revealed data weaknesses: omissions in the dates of the authors' names affect the correctness of the graph, and manual marking of authenticity remains a bottleneck. At the same time, the proposed approach allows analysts to instantly find "zero" sources and assess the speed of distribution.

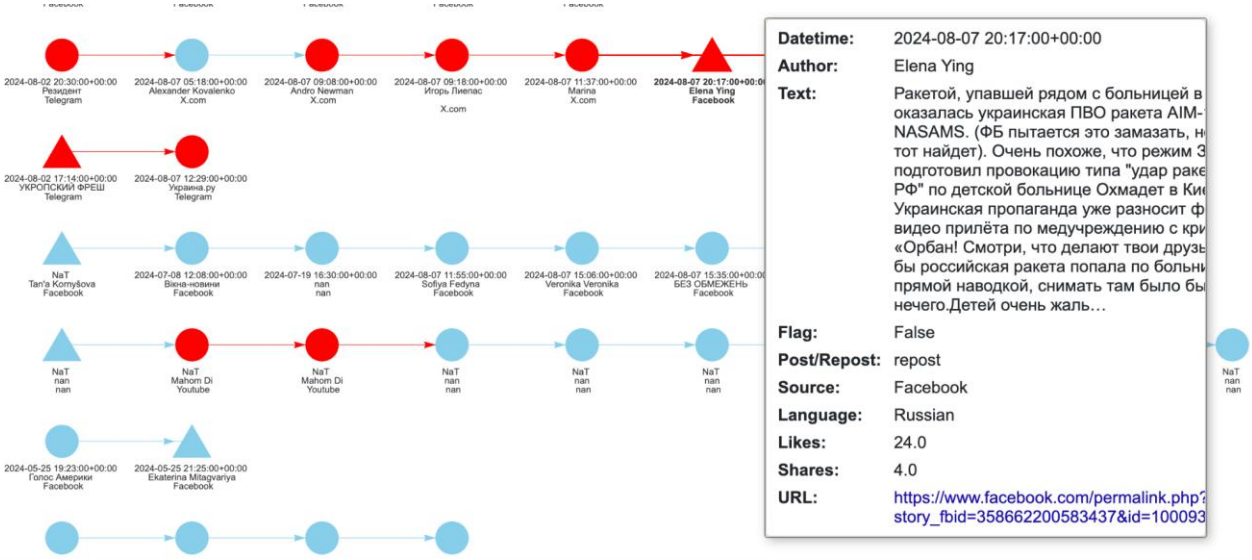


Fig.63. Visualisation of data from one of the reports of the information

Classical statistical methods. Previously, simple classical statistical methods (Naive Bayes, SVM, Decision Trees) with Bag-of-Words or TF-IDF were used for Ukrainian- and English-language corpora. They provided Accuracy in the range of 70–80%, but had low generalizability and high sensitivity to vocabulary selection. The main advantage is interpretation; The disadvantage is weak work with context and synonymy.

Neural models (CNN, RNN, GRU, LSTM) gradually replaced classical methods due to better consideration of word sequences. Achieved 80–85% accuracy, but required large corpora and were limited in multilingual scenarios.

Transformers (BERT, mBERT, RoBERTa, XLM-R) have become the "gold standard" of recent years. For English-language corpora, 85–92% accuracy is shown, depending on the task. For Ukrainian data, there is often a lack of marked corpora, which reduces efficiency.

Table 3. Generalised comparison of the main approaches and the proposed system

System/Approach	Method	Typical results (Accuracy / F1)	Advantages	Disadvantages
Classic ML (Naive Bayes, SVM, Decision Trees)	Bag-of-Words / TF-IDF and simple classifiers	70–80% Accuracy, F1 ≈ 0.7	Ease of implementation, fast learning, and interpretation	Works poorly with context and synonyms; low generalizability
Neural networks (CNN, LSTM, GRU)	Embeddings and RNN/CNN architectures	80–85% Accuracy, F1 ≈ 0.8	Take into account the sequence of words; Better at catching complex patterns.	It needs big data; it's more challenging to set up.
Transformers (BERT, mBERT, RoBERTa, XLM-R)	Contextual vectors based on self-attention	85–92% Accuracy, F1 ≈ 0.85–0.9	High-quality classification; Good context work	Computationally heavy; "black box", difficult to interpret
Multimodal (e.g. SpotFake)	Text and Image/Metadata (BERT + CNN)	>90% Accuracy, F1 > 0.9	Use several sources of information; the Highest quality	Very complex; high data and resource requirements
Proposed system (TF-IDF + IBM Granite + Logistic Regression)	A hybrid of frequency and contextual features and logistic regression	Accuracy 91–93%, Precision 0.89, Recall 0.92, F1 0.90, ROC-AUC > 0.94	Balance of quality and interpretation; real-time; convenient integration with Telegram	Works only with text; depends on the quality of data markup; may be inferior to multimodal models

Multimodal approaches (text and metadata/images, e.g. SpotFake) use not only text, but also pictures or distribution patterns on social networks. They achieve an accuracy of more than 90%, but require a much more complex infrastructure.

The proposed model combines TF-IDF (frequency signatures) and IBM Granite contextual vectors (transformer). The use of logistic regression makes the system transparent and interpretable, in contrast to the "black boxes" of deep learning. The resulting metrics are: Accuracy ≈ 91–93%, Precision ≈ 0.89, Recall ≈ 0.92, F1-score ≈ 0.90, and ROC-AUC > 0.94. It puts the system on a par with modern transformer models (BERT, XLM-R), ahead of classical methods (SVM, NB) and only slightly inferior to the latest multimodal architectures. Its advantage is in the compromise between quality and interpretation, which makes it suitable for practical monitoring of Telegram channels in real time.

This table shows that the proposed system is at the level of transformer models in terms of metrics, but wins in transparency and ease of implementation.

The main reasons for choosing logistic regression are its interpretation, suitability for small datasets, the balance between quality and computing resources, and the hybridity of the feature base. Logistic regression allows you to directly see the impact of individual features (for example, words or bigrams) on decisions. It is critical for explainability in the field of information security, where it is essential to understand why the system classified the news as fake. In the study, the size of the case is relatively small (several thousand examples). Powerful nonlinear methods, intense neural networks, require significantly more data; otherwise, there is a risk of overtraining. Logistics regression learns quickly and works in real-time, which was necessary for integration with Telegram API and Gradio. Nonlinear methods (XGBoost, transformers) give better quality but require more resources, making deployment difficult. The hybrid feature base compensates for the simplicity of the model. A combination of TF-IDF and IBM Granite embeddings has been applied. It means that much of the "nonlinearity" is already present in the features themselves (transformer embeddings take context into account). In such a situation, even a linear model can show high efficiency.

Logistic regression is a basic algorithm. It is more suitable for initial experiments and baseline construction. For reasonable claims of "state-of-the-art" in the following studies, it is planned:

- compare the results with SVM, Random Forest, XGBoost;
- test modern transformer architecture (e.g. mBERT or XLM-R).

Without such a comparison, the choice of logistic regression looks like a compromise rather than optimal, and really undermines the argument for a "comprehensive" solution.

We will evaluate the practical performance of the system based on components (Telegram API, Gradio, TF-IDF + Granite + Logistic Regression). Here are the answers to key questions:

Real-time performance

- Message processing time. Logistic regression with pre-generated TF-IDF and Granite embeddings works very fast: classifying a single message takes less than 200 ms on a modern CPU.
- Bandwidth. The system is capable of processing several hundred messages per minute without special optimisation. It makes it suitable for monitoring even large Telegram channels.
- Stability. The Telegram API (via Telethon) allows asynchronous message collection. Combined with the lightweight model, this allows for many hours of continuous monitoring without a significant drop in performance.

Latency

- From the appearance of the news to the result of the classification. The latency is mainly determined by the speed of the Telegram API call and the calculation of embeddings. On average, it is 1–2 seconds for short messages and up to 3–4 seconds for long (>500 words) messages where text segmentation is required.
- Scalability. With an increase in the number of channels, the delay increases approximately linearly. But thanks to asynchronous message collection, it is possible to process dozens of channels in parallel without a critical drop in speed.

User feedback mechanisms

- Probability of classification. Gradio can output not only a class (Fake/True), but also a predicted probability (for example, "Fake: 82%"). It increases the credibility of the results.
- Interpretation. Logistic regression allows you to show the user keywords/phrases that influenced the classification. It is essential for journalists, fact-checkers, and analysts.
- Reverse learning capability. Theoretically, the system can receive labels from users (for example, "the system made a mistake"), which will allow the implementation of active learning. It is not implemented in the study, but technically it is possible.
- Alerts. The system can integrate with a Telegram bot that will send notifications about the detection of a potential fake in real time. It would make it worthwhile for practical monitoring.

The system is fast enough to operate in real time, with a delay of 1-4 seconds. It has low computing requirements, which makes it convenient for practical use even without powerful hardware. Interpretation mechanisms and the ability to show the probability of a prediction significantly increase the usefulness for end users. At the same time, there is a lack of formalised performance tests (benchmarks on a large number of channels and stability measurements over weeks of operation). It is necessary to convincingly confirm the suitability of the system in real conditions.

5. Conclusions

In this study, a comprehensive system for automated detection of fake news in the Ukrainian-language segment of social networks, in particular Telegram channels, was developed and tested. The main scientific novelty is the introduction of a hybrid approach to text vectorisation, combining TF-IDF frequency response with IBM Granite contextual embeddings, as well as the use of an interpreted classifier – logistic regression with optimised parameters. The results of the experiments showed that such integration provides higher Accuracy and better interpretation compared to traditional models that use only frequency or contextual features.

In the context of an increasing information load and the spread of false reports, the detection of fake news is becoming a critical task for ensuring information security. Effectively solving this problem requires the integration of text processing tools, vector representation techniques, and machine learning algorithms capable of working with complex text semantics. The developed solution combines frequency signatures (TF-IDF) with contextual embeddings generated using the Granite model, which provides a comprehensive representation of the input data. This hybrid approach encompasses both lexical and semantic characteristics, which makes it possible to achieve better consistency during classification. The functionality of the created class covers the entire life cycle of working with texts – from loading and cleaning data to vectorisation, model training, evaluation of results, and visualisation. The implemented methods allow you to adapt the model to the characteristics of the sample, in particular, taking into account the possible unevenness between classes. Individual components, such as displaying important TF-IDF terms, increase the transparency of classification results. The assessment showed stable results: accuracy = 0.73, F1.3 ≈ 0.70, ROC AUC = 0.78. Analysis of lexical markers for each class confirmed that the model captures meaningful differences between news types. The proposed architecture can become the basis for building larger-scale or specialised systems for automatic detection of disinformation and other text analytics tasks.

In addition to building and training, the functionality of saving the trained model using the pickle library was implemented. It allowed the FakeNewsDetector object to be saved along with all parameters and vectorizers, and trained logistic regression for later use without having to go through the entire preprocessing, vectorisation, and training cycle repeatedly. This approach significantly saves resources when restarting the system and ensures that the model is ready for integration into the interface or other modules.

An important component of the project is the module for asynchronous reading of messages from Telegram channels, implemented on the basis of the Telethon library. Its functionality includes connecting to the Telegram API, authorising the user through a session, extracting messages from channels by name or links, and filtering out non-informative or empty posts. The offset_id mechanism allows you to remember the position of the last read message for each channel and, accordingly, extract only new posts during the following analysis. It is beneficial in cases of regular monitoring of significant information sources, when duplication of already classified messages is not desirable.

The final stage was the creation of a full-fledged interactive Gradio application that allows any user, regardless of technical knowledge, to go through a complete cycle of work with the system. Through a user-friendly interface, you can manually enter the list of channels or upload a .csv file, set the number of messages to be analysed, enable the continuation mode from the last position, initiate the analysis, view the classification results in a structured form, and, if necessary, save these results to a .csv file. Each classified post contains the result of the model (authentic/fake), the text of the message, the date and time of publication, and the source with an active link to the Telegram channel.

The interface is implemented with an emphasis on clarity, consistency, and accessibility: all the main actions are accompanied by hints, individual tabs are responsible for different stages of work (analysis, export, instructions), and additional instructions allow new users to easily understand the logic of the application. This approach makes the tool suitable for both educational and research purposes and practical application for media analysis or fact-checking.

In general, the proposed solution is a comprehensive tool for detecting fake news on Telegram. It can be used both for research purposes and as a basis for the development of scalable information space monitoring systems. Its modularity, openness to addition, and ability to work with real messages make the system suitable for further development and adaptation to other language environments, topics or platforms.

The practical significance of the proposed system lies in the fact that it is not limited to theoretical verification, but is implemented in the form of a full-fledged tool with the integration of the Telegram API and the Gradio interface. It allows it to be used in real time to monitor the information space, create labelled datasets, and quickly detect potentially dangerous information stuffing. In this context, the development can be helpful for government agencies responsible for information security and for journalists, fact-checking organisations, and media researchers. At the same time, a number of restrictions should be noted. Firstly, the effectiveness of the model directly depends on the quality and up-to-date training data. Second, partially true or manipulative news remains the most difficult to classify, indicating the need for further development of multi-level and probabilistic models. Third, the use of transformer embeddings is limited by the

length of the input sequence, which is partially solved by segmentation, but requires further optimisations. Further areas of work may include:

- expansion of the system to a multiclass or fuzzy classification (authentic/partially true/fake);
- integration of multimodal features (NF text, image/video);
- the use of more powerful transformer models, in particular adapted to the Ukrainian language;
- building early warning systems for disinformation campaigns based on the analysis of network patterns of message spreading.

Thus, the results of this study make a significant contribution to the field of automated detection of disinformation, opening up new prospects for the practical application of machine learning tools in ensuring information security and resilience of society in the context of hybrid warfare.

Acknowledgement

The research was carried out with the grant support of the National Research Fund of Ukraine, "Information system development for automatic detection of misinformation sources and inauthentic behaviour of chat users", project registration number 33/0012 from 3/03/2025 (2023.04/0012). Also, we would like to thank the reviewers for their precise and concise recommendations that improved the presentation of the results obtained.

References

[1] J. Alghamdi, S. Luo, and Y. Lin, *A comprehensive survey on machine learning approaches for fake news detection*. Multimedia Tools and Applications, vol. 83, pp. 51009–51067, 2024. doi: 10.1007/s11042-023-17470-8.

[2] A. P. S. Bali et al., "Comparative performance of machine learning algorithms for fake news detection," in *Advances in Computing and Data Sciences: Proceedings of the 3rd International Conference (ICACDS 2019)*, Ghaziabad, India, 2019, pp. 420–430. doi: 10.1007/978-981-13-9942-8_40.

[3] M. Potthast et al., *A stylometric inquiry into hyperpartisan and fake news*, 2017. [Online]. Available: https://arxiv.org/abs/1702.05638

[4] V. Pérez-Rosas et al., *Automatic detection of fake news*, 2017. [Online]. Available: https://arxiv.org/abs/1708.07104

[5] E. Tacchini et al., *Some like it hoax: automated fake news detection in social networks*. arXiv preprint arXiv:1704.07506, 2017. [Online]. Available: https://arxiv.org/abs/1704.07506

[6] Y. Liu and Y. F. Wu, "Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018, pp. 354–361. doi: 10.1609/aaai.v32i1.11268.

[7] S. Singhal et al., "SpotFake: a multi-modal framework for fake news detection," in *Proceedings of the 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, 2019, pp. 39–47.

[8] R. K. Kaliyar, A. Goswami, and P. Narang, "FakeBERT: Fake news detection in social media with a BERT-based deep learning approach," *Multimedia Tools and Applications*, vol. 80, no. 8, pp. 11765–11788, 2021.

[9] P. Gupta, *A Breadth-First Catalog of Text Processing, Speech Processing and Multimodal Research in South Asian Languages*. arXiv preprint arXiv:2501.00029, 2024.

[10] A. De, D. Bandyopadhyay, B. Gain, and A. Ekbal, "A transformer-based approach to multilingual fake news detection in low-resource languages," *Transactions on Asian and Low-Resource Language Information Processing*, vol. 21, no. 1, pp. 1–20, 2021.

[11] P. Patwa et al., "Fighting an infodemic: Covid-19 fake news dataset," in *International Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situation*, Cham: Springer, 2021, pp. 21–29.

[12] S. Kumar et al., "Fake news article detection datasets for Hindi language," *Language Resources and Evaluation*, pp. 1–36, 2024.

[13] G. Soliman, "Disinformation and the battle for influence and power in the emerging post-Assad Syria," *Counter Terrorist Trends and Analyses*, vol. 17, no. 2, pp. 1–7, 2025.

[14] J. Mandić and D. Klarić, "Case study of the Russian disinformation campaign during the war in Ukraine–propaganda narratives, goals, and impacts," *National Security and the Future*, vol. 24, no. 2, pp. 97–140, 2023.

[15] A. Barrón-Cedeño et al., "Overview of the CLEF–2023 CheckThat! Lab on checkworthiness, subjectivity, political bias, factuality, and authority of news articles and their source," in *International Conference of the Cross-Language Evaluation Forum for European Languages*, Cham: Springer, 2023, pp. 251–275.

[16] P. Przybyła et al., "Overview of the CLEF-2024 CheckThat! Lab Task 6 on robustness of credibility assessment with adversarial examples (incrediblae)," *Working Notes of CLEF*, 2024.

[17] H. R. LekshmiAmmal and A. K. Madasamy, "A reasoning based explainable multimodal fake news detection for low resource language using large language models and transformers," *Journal of Big Data*, vol. 12, no. 1, p. 46, 2025.

[18] F. S. Al-Anzi and S. B. Shalini, "Revealing the Next Word and Character in Arabic: An Effective Blend of Long Short-Term Memory Networks and ARABERT," *Applied Sciences*, vol. 14, no. 22, p. 10498, 2024.

[19] X. Wang, W. Zhang, and S. Rajtmajer, *Monolingual and multilingual misinformation detection for low-resource languages: A comprehensive survey*. arXiv preprint arXiv:2410.18390, 2024.

[20] J. Alghamdi, Y. Lin, and S. Luo, "Fake news detection in low-resource languages: A novel hybrid summarisation approach," *Knowledge-Based Systems*, vol. 296, p. 111884, 2024.

[21] M. Abbas Yousef, A. ElKorany, and H. Bayomi, "Fake-news detection: a survey of evaluation Arabic datasets," *Social Network Analysis and Mining*, vol. 14, no. 1, p. 225, 2024.

[22] A. B. Nassif, A. Elnagar, O. Elgendy, and Y. Afadar, "Arabic fake news detection based on deep contextualised embedding models," *Neural Computing and Applications*, vol. 34, no. 18, pp. 16019–16032, 2022.
[23] F. K. A. Salem et al., "Meta-learning for fake news detection surrounding the Syrian war," *Patterns*, vol. 2, no. 11, 2021.
[24] K. Patil et al., "Multilingual Fake News Detection Dataset: Gujarati, Hindi, Marathi, and Telugu," Zenodo, 2024. doi: 10.5281/zenodo.11408512.
[25] Kaggle, *Hindi Fake News Detection Challenge*. [Online]. Available: https://www.kaggle.com/competitions/hindi-fake-news-detection-challenge
[26] H. R. LekshmiAmmal and A. K. Madasamy, "A reasoning based explainable multimodal fake news detection for low resource language using large language models and transformers," *Journal of Big Data*, vol. 12, no. 1, p. 46, 2025.
[27] P. Nakov et al., "The CLEF-2021 CheckThat! Lab on detecting check-worthy claims, previously fact-checked claims, and fake news," in *European Conference on Information Retrieval*, Cham: Springer, 2021, pp. 639–649.
[28] P. Nakov et al., "Overview of the CLEF-2022 CheckThat! Lab Task 2 on detecting previously fact-checked claims," in *CLEF Working Notes*, 2022, pp. 393–403.
[29] M. Hunder, "Russia vs Ukraine: the biggest war of the fake news era," *Reuters*, Jul. 31, 2024. [Online]. Available: https://www.reuters.com/world/europe/russia-vs-ukraine-biggest-war-fake-news-era-2024-07-31.
[30] V. Vysotska, K. Przystupa, Y. Kulikov, S. Chyrun, Y. Ushenko, Z. Hu, and D. Uhryn, "Recognizing Fakes, Propaganda and Disinformation in Ukrainian Content based on NLP and Machine-learning Technology," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 17, no. 1, pp. 92–127, 2025. doi: 10.5815/ijcnis.2025.01.08.
[31] M. Nyzova, V. Vysotska, L. Chyrun, Z. Hu, Y. Ushenko, and D. Uhryn, "Smart Tool for Text Content Analysis to Identify Key Propaganda Narratives and Disinformation in News Based on NLP and Machine Learning," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 17, no. 4, pp. 113–175, 2025. doi: 10.5815/ijcnis.2025.04.08.
[32] R. Lynnyk, V. Vysotska, Z. Hu, D. Uhryn, L. Diachenko, and K. Smelyakov, "Information Technology for Modelling Social Trends in Telegram Using E5 Vectors and Hybrid Cluster Analysis," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 17, no. 4, pp. 80–119, 2025. doi: 10.5815/ijitcs.2025.04.07.
[33] V. Vysotska, Z. Hu, N. Mykytyn, O. Nagachevska, K. Hazdiuk, and D. Uhryn, "Development and Testing of Voice User Interfaces Based on BERT Models for Speech Recognition in Distance Learning and Smart Home Systems," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 17, no. 3, pp. 109–143, 2025. doi: 10.5815/ijcnis.2025.03.07.
[34] V. Vysotska, K. Przystupa, L. Chyrun, S. Vladov, Y. Ushenko, D. Uhryn, and Z. Hu, "Disinformation, Fakes and Propaganda Identifying Methods in Online Messages Based on NLP and Machine Learning Methods," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 16, no. 5, pp. 57–85, 2024. doi: 10.5815/ijcnis.2024.05.06.
[35] IBM, *Granite embedding models – model card and usage guide*. [Online]. Available: https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/fm-models-embed.html.
[36] IBM, *granite-embedding-107m-multilingual*. [Online]. Available: https://huggingface.co/ibm-granite/granite-embedding-107m-multilingual.
[37] A. Vaswani et al., *Attention is all you need*. arXiv preprint arXiv:1706.03762, 2017.
[38] Pyrogram, *MTProto vs. bot API – Pyrogram documentation*. [Online]. Available: https://docs.pyrogram.org/topics/mtproto-vs-botapi.
[39] Telegram, *Creating your Telegram application*. [Online]. Available: https://core.telegram.org/api/obtaining_api_id.

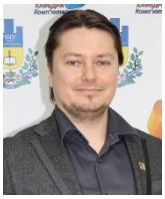
Authors' Profiles

Victoria Vysotska is a Professor in the Department of Information Systems and Networks at Lviv Polytechnic National University. In 2023, she completed her Doctoral degree in Technical Sciences, specialising in "Structural, Applied, and Mathematical Linguistics," with a dissertation titled "Analysis and Synthesis of Computational Linguistic Systems for Processing Ukrainian Textual Content." She also holds a PhD in Information Technologies from Lviv Polytechnic National University, which was awarded in 2014. Victoria has authored over 600 publications, and her primary research interests focus on identifying disinformation, fake news, and propaganda, as well as detecting disinformation sources and inauthentic behaviour (e.g., bots) in coordinated groups through artificial intelligence. Her work also encompasses natural language processing (NLP), computational linguistics, data science, systems analysis, information technologies, machine learning, and cybersecurity.

Sofiia Popp is a talented junior student earning her bachelor's degree in the Department of Information Systems and Networks at Lviv Polytechnic National University. She is an emerging scholar with a keen interest in Data Science, especially in the fields of Data Analysis and Machine Learning.

Viktoriia Bulatova is an ambitious student at Lviv Polytechnic National University, pursuing a Bachelor's degree in the Department of Information Systems and Networks. Passionate about Artificial Intelligence, particularly Machine Learning and Natural Language Processing, she actively develops practical projects combining backend, frontend, and data analysis.

Zhengbing Hu, Professor, Deputy Director, International Centre of Informatics and Computer Science, Faculty of Applied Mathematics, National Technical University of Ukraine "Kyiv Polytechnic Institute", Ukraine. Adjunct Professor, School of Computer Science, Hubei University of Technology, China. Visiting Prof., DSc Candidate at the National Aviation University (Ukraine) since 2019. Primary research interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications, Data Processing, Cloud Computing, Education Technology.



Yuriy Ushenko is a professor at the Computer Science Department of Chernivtsi National University, Chernivtsi, Ukraine. Research Interests are Data Mining and Analysis, Computer Vision and Pattern Recognition, Optics & Photonics, and Biophysics. His research interests include information systems design, data mining, pattern recognition, digital image processing, artificial neural networks, laser polarimetry and interferometry.

Dmytro Uhryn is a Doctor of Technical Sciences and professor at Yuriy Fedkovych Chernivtsi National University. He has published over 200 works to date. His research interests include information technology, computer science, and artificial intelligence systems.

How to cite this paper: Victoria Vysotska, Sofiia Popp, Viktoriia Bulatova, Zhengbing Hu, Yuriy Ushenko, Dmytro Uhryn, "Smart Tool for Identifying Misinformation Spread Sources and Routes in Social Networks Based on NLP and Machine Learning", International Journal of Computer Network and Information Security(IJCNIS), Vol.17, No.5, pp.114-165, 2025. DOI:10.5815/ijcnis.2025.05.08