I. J. Computer Network and Information Security, **2025, 4, 113-175**
Published Online on August 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2025.04.08

Smart Tool for Text Content Analysis to Identify Key Propaganda Narratives and Disinformation in News Based on NLP and Machine Learning

Maryna Nyzova
Information Systems and Networks Department, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: maryna.nyzova.sa.2021@lpnu.ua
ORCID iD: https://orcid.org/0009-0001-5050-1861

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

Lyubomyr Chyrun
Applied Mathematics Department, Ivan Franko National University of Lviv, Lviv, 79000, Ukraine
E-mail: Lyubomyr.Chyrun@lnu.edu.ua
ORCID iD: https://orcid.org/0000-0002-9448-1751

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 Physics, Shaoxing University, Shaoxing, Zhejiang Province 312000, China
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: 22 January 2025; Revised: 16 March 2025; Accepted: 14 May 2025; Published: 08 August 2025

Abstract: The paper presents the development of a smart tool for automated analysis of news text content in order to identify propaganda narratives and disinformation. The relevance of the project is due to the growth of the information threat in the context of a hybrid war, in particular in the Ukrainian information space. The proposed solution is implemented in the form of a browser plugin that provides instant analysis of content without the need to switch to third-party services. The methodology is based on the use of modern natural language processing (NLP) and deep learning methods (in particular, BERT models) to classify content according to the level of propaganda impact and identify key narratives. As part of the study, modern models of transformers for text analysis, in particular BERT, were used. For the task of classifying propaganda, pre-trained GloVe vectors optimised for news articles were used, which provided the best results among the options considered. Instead, the BERT model was used to classify narratives, which showed higher accuracy in the processing of texts reflecting subjective thoughts. The adaptation included the use of a multilingual version of BERT (multilingual BERT), as it allows you to effectively work with Ukrainian-language data, which is a key advantage for localised analysis in the context of information warfare. Before using BERT, pre-processing of texts was carried out with the addition of syntactic, punctuation, emotional and stylistic features, which increased the accuracy of classification. For a more complete and reliable assessment of the effectiveness of propaganda classification models and

narratives, a set of key metrics was used for propaganda/ narratives analyses Accuracy (0.94/0.86), Precision (0.95/0.69), Recall (0.96/0.71) and F1-score (0.96/0.70).The developed model showed high accuracy results: the F1-score for the propaganda classification problem was 0.96 and for the narrative classification problem – 0.70, which significantly exceeds the results of similar approaches, in particular XGBoost (0.92 and 0.50, respectively). In addition, the system supports full-fledged work with Ukrainian-language content, which is its key competitive advantage. The practical application of the tool covers journalism, fact-checking, analytics, and improving media literacy among citizens, contributing to the improvement of the state's information security.

Index Terms: Propaganda, Disinformation, BERT, Narrative, Browser Plugin, Ukrainian Text, NLP, Media Literacy, News Content, Browser Plugins, Text Classification, Information Security and Artificial Intelligence.

1. Introduction

In the modern media space, information operations have become one of the key forms of hybrid warfare. This problem is especially relevant for Ukraine, which, in the context of the armed conflict, is daily exposed to propaganda messages, manipulative narratives and disinformation from hostile sources. Users often do not have enough time, expertise, or tools to independently verify the accuracy of the information consumed.

This research aims to create an intelligent tool for automated analysis of news texts, capable of detecting signs of propaganda and classifying key narratives. The main form of implementation is a browser plugin that allows you to analyse content in real time, without the need to turn to external services. The system is based on modern methods of natural language processing (NLP) and transformer architectures (in particular, BERT), taking into account the peculiarities of the Ukrainian language.

Revised Main Objective. The primary goal of this research is to design and implement a functional prototype of a browser-based tool that performs automated, real-time classification of Ukrainian-language news text fragments, capable of (1) detecting the presence of propaganda and (2) identifying one or more associated manipulative narratives, using deep learning models (BERT) and interpretable NLP features.

The main objectives of the study are:

- To develop a model that classifies text fragments according to the level of propaganda influence.
- Implement a multi-label classification of narratives (for example, justifications, humiliations, appeals).
- Build an interface accessible to a broad audience without technical training.
- Ensure that the results are interpreted to increase user trust.
- Optimise the model for Ukrainian-language content, taking into account the lack of existing analogues.

Unlike systems such as NewsGuard, ClaimBuster or Hoaxy, the proposed solution is:

- works at the level of individual sentences or paragraphs, not just at the level of domain or tweets;
- has full support for the Ukrainian language;
- performs a deep semantic and emotional assessment of the text;
- identifies specific narratives, not just the presence of false information;
- offers an interface directly in the browser, without the need to copy text to third-party platforms.

Despite the significant advantages, the proposed system has the following limitations:

- The model is a prototype: at the time of publication, it did not undergo a full-fledged public deployment or large-scale user testing.
- Data limitation: educational corpora are limited in relation to labelled Ukrainian content, especially in the topic of narratives.
- Classification errors. Although the accuracy is high (F1-score up to 0.96), deep neural networks can produce erroneous estimates in ambiguous or ironic contexts.
- Possible biases: The results of the model depend on the quality and balance of the training data, which requires regular updating and validation.

The project, which is planned for research and development, aims to create a tool for automated text analysis that detects propaganda and identifies key narratives. The primary purpose and reason for the development is to improve the information hygiene of users by quickly and conveniently checking content on the Internet. The primary audience of the product will be people who constantly use news sites and want to increase their awareness of the veracity of the information they consume. The product is planned for development in the form of a browser plugin because this is what

will allow almost instant verification of the necessary information and will also be convenient for users as part of the initial download - the plugin does not need to be installed locally; it is not tied to specific systems or devices - you need to go to the plugin market of your browser.

At the moment, the development of such a project is relevant and necessary because the information war is waged every day, so people's awareness of propaganda is an urgent problem. News is consumed by users every day in large quantities, so the appearance of misinformation or propaganda narratives is an inevitable fact. Also, a separate task during the development is to analyse the differences between Ukrainian news produced by news channels and those produced by enemy channels, the results of which are also relevant and necessary for further research.

The goal of this project is to develop a tool for automated text analysis capable of detecting propaganda and identifying key narratives in news content. The main goal is to improve the information hygiene of users by quickly and conveniently checking the authenticity of information on the Internet. Taking into account the purpose of project development, the following tasks to be solved have been identified:

- Collection of necessary information (news texts):
  - Finding news (news portal) that will be classified as "propaganda" in the future;
  - Finding news (news portal) that will be classified as "non-propaganda" in the future;
  - Finding a corpus (dataset) that will be used to train algorithms for finding narratives in news texts;
- Development of text analysis algorithms:
  - Using Natural Language Processing (NLP) techniques for text analysis;
  - Identification of characteristic features of propaganda narratives;
  - Using machine learning models, in particular BERT, to develop classification algorithms;
- Creating a Browser Plugin:
  - Creating a user-friendly interface;
  - Integration of analysis algorithms into the browser plugin for instant content checking;
- Evaluation of the accuracy and efficiency of the system:
  - Testing the model on honest news articles;
  - Comparison of the results of algorithms with expert assessment;
  - Improvement of the system based on the results obtained.

The object of the project's research is the process of automated analysis of textual information to identify propaganda narratives and disinformation in news materials.

The subject of the project's research is natural language processing (NLP) methods, machine learning algorithms for text classification, as well as features of propaganda narratives in news content from various sources.

Several facts justify the scientific novelty of the project:

- There are almost no analogues of automated solutions for finding propaganda in the news explicitly trained for the Ukrainian language;
- There are no analogues of automated solutions for detecting narratives in propaganda news explicitly trained for the Ukrainian language;
- Using modern NLP and deep learning techniques to classify text;
- Ease of use: text classification directly in the browser without the need to copy content to external services;
- Feedback from users is planned to be added to organize the process of continuous additional training of models.

The development of the project is expedient because the problem of information warfare, especially in the context of the Russian-Ukrainian war, is highly relevant. There are practically no similar solutions that would combine automatic detection of propaganda and analysis of narratives, which makes the product unique. The growing scale of disinformation and propaganda requires practical tools to detect them - potential users are journalists, fact-checkers, analysts, government organizations and ordinary citizens who seek to obtain reliable information.

Thanks to the use of modern natural language processing (NLP) methods based on transformer models (BERT/RoBERTa), it is possible to create an effective solution. The project can be expanded to other languages and information wars in different regions. It is also possible to integrate with fact-checking platforms and news aggregators, which will increase its usefulness.

The development of such a project is reasonable and promising. It has high uniqueness, technical feasibility, wide demand and significant competitive advantages. This automated solution can play an essential role in combating disinformation and helping to increase the level of media literacy in society.

Refined Research Objective. The goal of this study is to investigate the effectiveness of transformer-based language models (BERT) in detecting propaganda and identifying manipulative narratives in Ukrainian-language news content. This includes the development of an interpretable classification pipeline, the integration of the model into a browser-based interface, and the empirical evaluation of the system's accuracy and explainability compared to existing tools.
To achieve this goal, the study presents:

- A curated dataset of Ukrainian propaganda and non-propaganda texts, including annotated narrative categories;
- A deep learning-based classification model achieving an F1-score of 0.96 for propaganda detection and 0.70 for narrative classification;
- A prototype browser plugin for real-time analysis and user feedback.

As part of this study, a tool for automated text analysis is planned to be created, which will allow the detection of propaganda and the identification of key narratives in news content. To achieve this goal, it is scheduled to perform several interrelated tasks: to collect the necessary data, including news texts with different levels of propaganda influence; to develop text analysis algorithms based on modern natural language processing (NLP) technologies and deep learning models; implement these algorithms in the form of a user-friendly browser plugin that will allow you to quickly check information; as well as evaluate the accuracy and efficiency of the system by testing and comparing it with expert evaluation, with further improvement of the tool based on the results obtained.

Motivation and Background. In the age of hybrid warfare and global information manipulation, disinformation and propaganda have become powerful tools for influencing public opinion, polarizing societies, and undermining democratic institutions. The Russian-Ukrainian war has vividly demonstrated how media content, particularly online news and social media, can be weaponized to spread manipulative narratives and distort factual reality.

Ukraine, as a country under continuous information pressure, faces a critical need for scalable, language-specific tools capable of detecting subtle and evolving forms of propaganda. Traditional media monitoring solutions are often language-limited, non-contextual, or insufficiently responsive to the emotional and semantic nuances typical of manipulative content in conflict zones. Additionally, manual fact-checking and narrative analysis are time-consuming and resource-intensive, making them impractical for ordinary users or real-time monitoring.

This context creates a strong demand for an automated, explainable, and user-friendly solution that can operate effectively in Ukrainian, analyse individual news fragments, and identify not only the presence of propaganda but also its underlying narratives and rhetorical devices. The proposed research aims to fulfill this gap by developing a prototype of a browser-based tool that leverages advanced NLP and deep learning to empower users in the fight against disinformation.

The main contributions of this study are as follows:

- Development of a Ukrainian-language propaganda detection model based on modern transformer architectures (BERT), specifically tailored for the context of hybrid warfare and information threats.
- Implementation of a multilabel classification approach to identify and categorize key propaganda narratives (e.g., justification, calls to violence, national humiliation) within short news texts.
- Design of an explainable and interpretable NLP pipeline, which incorporates linguistic, syntactic, and semantic features (punctuation, sentiment, subjectivity, named entities) to enhance model transparency.
- Prototype creation of a browser plugin that allows real-time analysis of online content, offering users immediate feedback without relying on external services.
- Comprehensive evaluation and comparison with existing tools (e.g., NewsGuard, ClaimBuster), demonstrating superior accuracy and language adaptability, especially for under-resourced and conflict-affected environments.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work and existing solutions in the field of disinformation detection. Section 3 describes the methodology, including system architecture, data pre-processing, and model development. Section 4 presents the experimental setup, datasets, and evaluation metrics. Section 5 discusses the results, their interpretation, and comparison with alternative approaches. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Related Works

2.1. Operationalisation of Key Concepts

Propaganda. Within the framework of this study, propaganda is operationalised as textual content that demonstrates signs of conscious manipulation of public opinion through:

- The use of emotionally colored vocabulary (for example, emotional reinforcement, categorical, rhetorical phrases),
- Distortion of facts or presentation of one-sided information without alternative points of view,
- The use of propaganda tactics, such as:

- appeal to the enemy/betrayal (e.g. "the enemy is trying to destroy our nation"),
- glorification/demonisation of the parties,
- groundless generalisations or mythologising of events.

These characteristics are realised through a combination of linguistic markers, grammatical structures, the emotional background of the text, and the semantic patterns identified.

A propaganda narrative in research is a recurring thematic motif or model of the message that:

- Supports the goal of influencing the perception of the audience,
- Often found in manipulative messages,
- has a specific ideological or psychological colouring**.**

In our system, five typical narratives were identified (according to the annotations in the dataset): Prosecution; Excuse; Call to action; Incitement to hatred; Humiliation of national dignity. Each of them is a separate binary label in a multi-marker classification problem (i.e., a text can contain several narratives at the same time).

The level of influence of propaganda in the system is modelled as a binary classification:

- 0 – Does not contain propaganda features (text – informational or neutral),
- 1 – Contains propaganda (expressed by at least one characteristic marker).

However, in the future, the possibility of expanding to a graduated scale (low/medium/high) of influence based on the density of manipulation markers, the level of emotional pressure and the number of narratives detected is foreseen.
In this study, hybrid warfare is considered a complex form of armed and informational influence, combining:

- Traditional hostilities (e.g. invasion or occupation),
- Information and psychological operations aimed at internal destabilisation through mass media, social networks, fake accounts and pro-Russian information platforms.

This approach is consistent with the concept of "hybrid threats" of the European Hybrid Threat Centre (Hybrid CoE, Helsinki) and the NATO model, where disinformation is an integral element of non-lethal impact aimed at:

- Decreased trust in institutions;
- The split of society.
- Delegitimisation of the defence capability of the state;
- formation of a pro-Russian or anti-Western worldview.

The model presented in the article is focused on identifying content that is a carrier of typical disinformation patterns recorded within the framework of the hybrid war against Ukraine. These are in particular:

- Geopolitical Disinformation:
  - anti-Western and anti-NATO messages;
  - distortion of the role of the EU and the United States in the conflict;
  - spreading the myth of Ukraine's external governance.
- Information war against the Armed Forces of Ukraine:
  - discrediting the Ukrainian army (through fakes, exaggeration of losses, "own shelling");
  - manipulations on the topic of mobilisation, panic among the military.
- Information demoralisation:
  - calls for surrender;
  - incitement of fatalism and helplessness;
  - promotion of ideas of "common guilt".
- Fomenting internal division:
  - division into "east-west";
  - intensification of linguistic, religious or ethnic confrontation;
  - undermining unity by imposing an "alternative" history or ideology.

- Cultural and mental attacks:

  – humiliation of the role of the Ukrainian language, traditions, symbols;
  – distortion of facts about the history of Ukraine, the Holodomor, the UPA, etc.

The tool is developed based on typologies compatible with:

- EUvsDisinfo Narrative Taxonomy (European External Action Service),
- Framework for Hybrid Threats (NATO STRATCOM),
- DIME-FIL Framework (Diplomatic, Informational, Military, Economic–Financial, Intelligence, Lawfare).

The model does not classify messages by subject matter in general, but by rhetorical patterns that are often used as part of these disinformation campaigns. It allows you to identify not only fakes but also emotionally coloured, subversive rhetoric that forms a manipulative narrative, even without outright lies.

Thus, the study is embedded in the broader context of the information component of the hybrid war against Ukraine, where the automatic detection of typical propaganda patterns makes it possible to detect disinformation even before it spreads massively.

This section provides an overview of existing approaches to the automated detection of propaganda, disinformation, and manipulative narratives. It shows how the proposed method combines the strengths of previous work and addresses their limitations.

Assessment of the reliability of sources. One of the traditional approaches is to evaluate the reliability of entire news resources, rather than individual stories. For example, the NewsGuard service evaluates websites according to a number of journalistic criteria: transparency, fact checking, advertising, etc. [1]. Although this approach is helpful for general orientation, it does not allow you to evaluate a specific article, which can contain both reliable information and manipulative elements. Media Bias/Fact Check**,** which indicates the level of political bias of the publication, but does not analyse individual texts.

Automatic analysis of texts and statements. Another direction is tools for the automatic selection of facts and verification of statements. ClaimBuster [2] highlights statements in texts that facts can verify. However, it does not evaluate the style or rhetoric of the text; it only helps fact-checkers choose phrases to check. Hoaxy [3] visualises the spread of misinformation on Twitter by constructing retweet graphs. However, the tool analyses network connections, not the content of the text itself. Fakey [4] is an educational application that trains users to distinguish truth from fakes, but it is not an automatic analysis system. Compared to these tools, the proposed method analyses the text itself and works in real time without human involvement.

Identification of propaganda techniques. There are a number of studies focused on the automatic recognition of propaganda in texts:

- The authors [5] created the Propaganda Techniques corpus and proposed models for detecting specific techniques (appeal to fear, flag waving, etc.).
- The developers [6] have developed the Proppy system, which combines lexical features and statistical models to detect propaganda in news articles.

Both approaches have been shown to be effective for English-language texts, but their translation into other languages, including Ukrainian, is limited due to the lack of annotated corpora.

Table 1. Comparison of approaches to detecting propaganda and disinformation

Group of approaches	Examples	What they do	Restriction	What does the proposed method do
Assessment of the reliability of sources	NewsGuard, Media Bias/Fact Check	Ranking sites by authenticity	Do not analyse individual texts; it requires expert handiwork	Analyses specific text in real time
Analysis of facts and dissemination	ClaimBuster, Hoaxy, Fakey	Extract statements, build retweet columns	They do not define style or rhetoric; often, English speakers	Identifies stylistic and semantic markers of propaganda
Detection of propaganda techniques	Proppy, Propaganda Techniques	Identify specific rhetorical techniques	English-speaking models; No localisation	Works with Ukrainian; also determines narratives
Transformer models	BERT, RoBERTa	High classification accuracy	Need power; usually do not have feature engineering	Combines transformers and style traits
Localised Research (Ukrainian)	Vlasiuk, 2022; Vlasiuk et al., 2020	Theoretical approaches, reviews	Missing Implemented Tools	Practical plug-in for the end user

Use of transformers and BERT. With the advent of transformer models such as BERT [7], the accuracy of text classification has increased significantly due to better consideration of context. A review [8] confirms that BERT and its modifications have been successfully applied for sentiment detection, detection of fakes and toxic comments [9]. The method proposed in the study also uses BERT, but in combination with feature engineering (punctuation analysis, tonality, the ratio of parts of speech, etc.). It allows the model to "see" both the semantics of the text and its stylistic and emotional

markers.

Localised research for the Ukrainian language. Studies devoted to the automated detection of propaganda in Ukrainian texts are rare. There are mainly review or conceptual works [10-12] that emphasise the importance of automation but do not offer implemented solutions or models. Therefore, the developed tool is one of the first attempts to build a really working service for the automatic detection of propaganda in the Ukrainian media field.

The analysis of existing works showed that none of them combine:

- Support for the Ukrainian language;
- Modern architecture is based on transformers.
- Automatic detection of propaganda and key narratives;
- Convenient implementation in the form of a browser plugin.

It emphasises the novelty and practical significance of the proposed solution, which takes into account the real challenges of the information war in Ukraine.

2.2. Competitive Analysis of Analogues

A. Search and Determination of Close and Distant Analogues

During the search for analogues, the following products were found:

- NewsGuard (https://www.newsguardtech.com/) – a browser-based plugin that evaluates the reliability of news sites based on journalistic standards. Displays the rating of the source and warns of possible misinformation. It does not analyze individual articles or sentences but only evaluates entire sites. It also has real-time functionality for determining the presence of disinformation among articles that appear on the site.
- Hoaxy (https://hoaxy.osome.iu.edu/) – a tool for visualizing the spread of misinformation on Twitter. Uses NLP to analyze the narratives being spread. Determines the level of virality of fake news. Not intended to be used as a fact-checking tool but to understand how misinformation spreads through accounts.
- CheckBuster (https://idir.uta.edu/claimbuster/) is an automated fact-checker tool for English that analyses text for potentially verifiable facts. Uses NLP to detect claims that may contain misinformation.
- Fakey (https://fakey.osome.iu.edu/login) is an educational app that helps users learn how to distinguish fake news from real news. It is used for training, not for automatic content analysis.

For further comparison, we decided to choose NewsGuard and CheckBuster, because they are the ones that are most similar in functionality and purpose to the project under development.

B. Development of a Set of Characteristics and Criteria for Evaluation and Comparison

To be able to compare the project with analogues, a list of characteristics that are important when creating products of this type has been defined. Compliance with general requirements:

- Unification – open API for integration with other services;
- Interoperability – the ability to interact with news aggregators;
- Mobility – the ability to expand to mobile browsers;
- Scalability – expansion to other languages and types of content.
- User interaction – instant text analysis, the ability to get an explanation of the result;
- Support for the Ukrainian language.

Compliance with Quality Requirements (FURPS+):

- Functionality – support for multilingualism, the possibility of detailed analysis;
- Usability – minimal impact on browser performance, simple interface;
- Reliability – regular updating of models to adapt to new propaganda tactics;
- Performance – real-time text processing;
- Supportability – Open source, with the ability to update and expand functionality.

C. Development of a System for Evaluating the Characteristics of Analogues

For an objective analysis of the effectiveness of existing solutions in the field of detecting propaganda and disinformation, a system for evaluating the characteristics of analogues has been developed. Evaluation is carried out by quantifying each product for key factors that determine its effectiveness, usability, and reliability. The results of the assessment will allow us not only to conduct a comparative analysis but also to develop recommendations for improving the proposed project, taking into account the best practices from existing solutions, as well as to understand what problems in the field have not yet been solved.

D. Development of a Rating Scale

To ensure an objective comparison of the characteristics of analogues, a numerical rating scale from 1 to 10 has been developed. It will allow you to clearly determine the level of compliance of each product with specific requirements, where:

- 10 – the characteristic is implemented perfectly and fully meets the requirements;
- 9 – almost perfect implementation, minimal flaws;
- 8 – very high level of compliance, but there are a few minor drawbacks;
- 7 – exemplary implementation, but there are noticeable shortcomings or limitations;
- 6 – satisfactory level of implementation, significant improvements are possible;
- 5 – medium level, the characteristic is partially implemented;
- 4 – minimum compliance with the requirements, there are significant shortcomings;
- 3 – the characteristic is poorly implemented and needs significant improvement;
- 2 – very low level of compliance, the characteristic is almost not implemented;
- 1 – the characteristic is absent or does not meet the requirements at all.

The evaluation is carried out by experts for each of the defined characteristics: unification, interoperability, mobility, scalability, user experience, functionality, usability, reliability, performance and usability.

E. Building a Comparative Table for the Project and Analogues

Table 2. Comparison of the project with analogues

Characteristics	Project	Analogues	
		NewsGuard	CheckBuster
Unification	9	8	7
Interoperability	8	7	6
Mobility	9	8	7
Scalability	8	7	6
User Experience	9	7	6
Ukrainian language support	10	5	0
Functionality	9	7	8
Ease of use	9	8	7
Reliability	8	9	7
Performance	8	7	7
Operational suitability	9	8	7

Let's explain the grades given by experts.

- Project:
  - Unification (9/10) – the project has high unification due to the use of standard NLP technologies and a browser plugin that is easily integrated into different environments;
  - Interoperability (8/10) – slightly lower than the maximum since working with other systems (for example, the fact-checking API) may still require adaptation;
  - Mobility (9/10) – high due to the implementation in the form of a browser plugin;
  - Scalability (8/10) – limited due to the specificity of the task (propaganda analysis), but it is possible to expand to other languages;
  - User interaction (9/10) – convenient because the analysis is carried out without going to other sites;
  - Ukrainian language support (10/10) – initial models are created exclusively for the Ukrainian language;
  - Functionality (9/10) – advanced because the plugin works not only with sites but also with selected text;
  - Ease of use (9/10) – high because it does not require complex settings;
  - Reliability (8/10) – slightly lower than NewsGuard because the results are based on deep learning, which can give errors;
  - Performance (8/10) – optimal, but processing significant texts can take time;
  - Operational suitability (9/10) – high because the technologies used allow maintaining and updating the system;
- NewsGuard:
  - Unification (8/10) – NewsGuard evaluates sites according to a single journalistic methodology but does not

analyse individual texts, which limits its unification;
- Interoperability (7/10) – works as a browser plugin but does not integrate with other content analysis services, such as the Fact-Checking API;
- Mobility (8/10) – available as a browser plugin and mobile app;
- Scalability (7/10) – expands to new countries, but the process takes a long time because humans make site ratings;
- User Experience (7/10) – provides site rankings but does not analyse specific pieces of text, making it less interactive;
- Ukrainian language support (5/10) – news portals are evaluated regardless of the language because specialists carry out the assessment;
- Functionality (7/10) – works well for evaluating sites but does not recognize individual manipulative techniques;
- Ease of use (8/10) – simple and intuitive interface, minimal settings;
- Reliability (9/10) – real journalists give ratings, so errors are minimized, but updating the database of sites takes time;
- Performance (7/10) – Works quickly, but data updates are not instantaneous;
- Usability (8/10) – well maintained, but requires manual verification of new sources;

- CheckBuster:

  - Unification (7/10) – analyses individual statements but has a limited set of validation rules;
  - Interoperability (6/10) – supports APIs, but integration with other fact-checking systems is weak;
  - Mobility (7/10) – available as a web tool, but there is no user-friendly browser plugin;
  - Scalability (6/10) – difficult to adapt to new languages and information spaces;
  - User interaction (6/10) – works through a web platform that requires going to the site and manually entering text;
  - Ukrainian language support (0/10) – does not work for the Ukrainian language;
  - Functionality (8/10) – identifies potentially controversial statements and selects facts for verification but does not analyse general propaganda rhetoric;
  - Usability (7/10) – intuitive interface, but requires active text input from the user;
  - Reliability (7/10) – works based on NLP and database, but errors in classification are possible;
  - Performance (7/10) – processes text quite quickly but is limited by the size of the database of verified facts;
  - Usability (7/10) – supported, but requires constant updating of algorithms and fact-checking databases.

So, based on the results of the assessment, we can conclude that in terms of the general requirements for automated products with similar goals, our project even prevails in some characteristics. However, the key factor is that the project is aimed at a task that has not yet been automated for the Ukrainian language, so this is its main advantage.

2.3. Definition of the Problematic Issue

A. Formulation of the Problem to be Solved by the Project

In the modern information space, a significant share of content is occupied by disinformation, manipulation and propaganda narratives that can influence public opinion, decision-making, and even socio-political stability [10-12]. The problem is that ordinary users do not always have enough time, skills, or resources to independently verify the veracity of the information they consume. The main aspects of the problem [13-16]:

- A large amount of information, which makes it difficult to quickly check each news;
- The difficulty of detecting propaganda without specialized knowledge;
- Lack of user-friendly automated tools for real-time verification of narratives;
- Dissemination of manipulative content through social networks and news sites.

Solving this problem requires an effective tool that will allow users to quickly analyse information and receive objective assessments of its reliability [13].

B. Formulation of Means and Ways to Solve the Problem

To solve the identified problem, it is proposed that a browser plugin be developed that will automatically analyse the selected text on web pages, evaluating it for propaganda, manipulation or dubious narratives [17-18]. Means of solving the problem [19-25]:

- Automated text analysis using machine learning algorithms (NLP, transformer models, BERT);
- Recognition of propaganda techniques by analysing lexical and semantic features of the text;
- User-friendly interface – integration into a browser, which allows the user to instantly receive text analysis.

Methods of implementation:

- Development of an NLP model for text analysis based on pre-trained transformer architectures (BERT, RoBERTa, T5);
- Model training based on data containing samples of propaganda and manipulative techniques;
- Creation of a browser plugin that interacts with the backend for text analysis.

Thanks to these tools and methods of implementation, the project will improve information hygiene among users, reduce the impact of propaganda and promote an objective perception of news content by users.

3. Methods and Tools

3.1. Formulation of the General Goal

The overall goal is to help users identify and analyse propaganda in the media space, increasing their media literacy and promoting information security. The tool being created should provide a quick and convenient way to identify propaganda signs and key narratives, which is especially important in the context of the Ukrainian-Russian war.

3.2. Building a Goal Tree

The goal tree reflects the logical structure of the stages of development of the tool specified in the primary goal. The main goal of the project (the highest level of the tree) is formulated in accordance with the primary goal, i.e. to create a tool that allows not only the identification of propaganda signs but also to classify key narratives while providing convenience and ease of use for the end user:

- Identification and classification of propaganda features:
  - Collection of a base for training models;
  - Analysis of texts to identify syntactic and semantic propaganda techniques;
  - Creation of a classification model for propaganda;
- Identifying Key Narratives:
  - Formation and updating of the narrative base;
  - Analysis of texts to identify syntactic and semantic techniques of narratives;
  - Creation of a multilabel classification model for narratives;
- Ensuring Simplicity and Ease of Use:
  - Creating an intuitive interface;
  - Implementation of flexible settings for users;
  - Implementation of visualization of results.

Within the framework of achieving the goal of identifying and classifying propaganda signs, several subtasks will be performed. In particular, a database will be collected for training models, and syntactic, semantic and punctuation analysis of texts will be carried out to identify propaganda techniques. Based on the results obtained, a classification model capable of identifying various types of propaganda manifestations in texts will be created. To achieve the goal of identifying key narratives, a database of narratives will be formed with the possibility of further filling it. An analysis of texts will be carried out in order to identify linguistic techniques that signal the presence of narratives. As a result, a multilabel classification model will be built, which will allow for the detection of several narratives within one text at once. In order to ensure simplicity and ease of use, an intuitive interface will be created for the developed tool. A flexible system of settings will be implemented, which allows you to adapt the tool to the needs of the user. To improve the visual perception of the results, visualization modules will also be added, which will allow the data to be presented in graphical form.

3.3. Identification of Alternative Options for Building the System

The initial version of the project was a plugin in the browser because this is what will allow the interface to be mobile, convenient and easy to use. However, it is also necessary to consider other options for building the system, as this is one of the most influential decisions in the project.

Three possible options for creating a system interface have been identified, each of which has its advantages and disadvantages:

- Browser plugin – is a convenient solution for quick text analysis without the need to open a separate application or website. The user can classify text directly while browsing the web, making it ideal for journalists, researchers, and activists. Among the disadvantages are limited functionality compared to mobile or web applications and dependence on a specific browser. Overall, the plugin is an effective and easy-to-implement option for quick access to basic features;
- Mobile application – provides ease of use at any time, allowing you to work with texts on the go. The advantages include integration with other mobile applications and the ability to work offline. The main disadvantage is the more complex development process and the need to adapt to different platforms (iOS, Android). This option is suitable for a broad audience of users who need constant access to the tool;
- Website – allows you to centrally provide access to the tool from any device without installing software. The advantages are wide functionality, the ability to quickly update the service, and support for a large number of users. However, this option requires a constant Internet connection, maybe less convenient in a mobile format and requires a separate opening. In general, the web platform is the most versatile and scalable solution for launching a full-fledged service, but it lacks convenience and speed of use for the user.

Table 3. Matrix of pairwise comparisons for criteria and determination of weighting factors for criteria

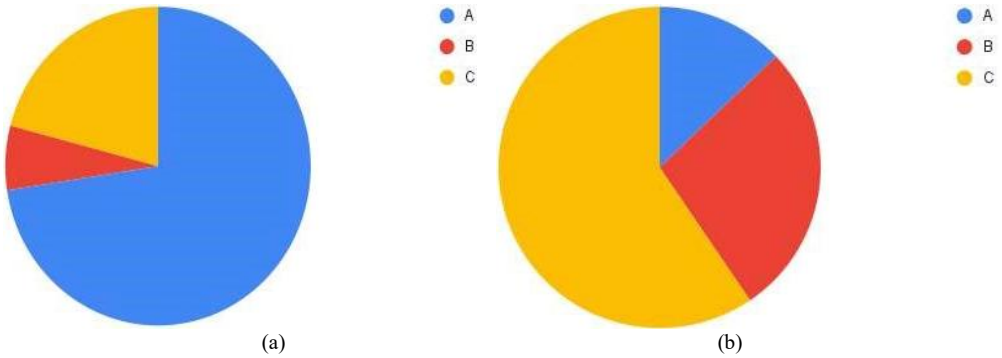
Criteria	C1	C2	C3	C4	C1	C2	C3	C4	Priority
C1	1	2	3	4	0.487804878	0.5263157895	0.4615384615	0.4	0.4689147823
C2	0.5	1	2	3	0.243902439	0.2631578947	0.3076923077	0.3	0.2786881604
C3	0.3	0.5	1	2	0.1463414634	0.1315789474	0.1538461538	0.2	0.1579416412
C4	0.25	0.3	0.5	1	0.1219512195	0.07894736842	0.07692307692	0.1	0.09445541621

Table 4. Calculation of the consistency index

0.9612753036	CONSISTENCY INDEX	MAGNITUDE OF CONSISTENCY
1.059015009		
1.026620668		
0.9445541621		
3.991465143	0.002844952437	0.003161058264

Table 5. Matrix of pairwise comparisons and determination of the vector of priorities of alternatives by ease of implementation

Options	A	B	C	A	B	C	Priorities
A	1	8	6	0.7722007722	0.5714285714	0.8333333333	0.7256542257
B	0.125	1	0.2	0.09652509653	0.07142857143	0.02777777778	0.06524381524
C	0.17	5	1	0.1312741313	0.3571428571	0.1388888889	0.2091019591



(a) (b)

Fig.1. Visualization of the vector of priorities of alternatives in terms of ease of implementation and functionality

Table 6. Matrix of pairwise comparisons and determination of the vector of priorities of alternatives by functionality

Options	A	B	C	A	B	C	Priorities
A	1	0.5	0.2	0.125	0.1428571429	0.1176470588	0.1285014006
B	2	1	0.5	0.25	0.2857142857	0.2941176471	0.2766106443
C	5	2	1	0.625	0.5714285714	0.5882352941	0.5948879552

A hierarchy analysis was used to determine the method of implementation. For this purpose, four key criteria have been identified that will allow comparing alternatives with each other not only qualitatively but also quantitatively:

- Ease of implementation (C1) – this criterion assesses how easy it is to implement and use the solution from a technical point of view. The easier it is to implement, the higher the priority of the alternative.
- Functionality (C2) – This criterion reflects the scope and flexibility of the available functions. A more functional and flexible alternative is a higher priority.
- Security and privacy (C3) – this criterion assess how much the solution protects user data and ensures privacy. The better security and privacy are ensured, the higher the priority of the alternative.
- User Accessibility (C4) – This criterion determines how easy and convenient it is for users to interact with the solution. The more users can easily take advantage, the higher the priority of the alternative is.

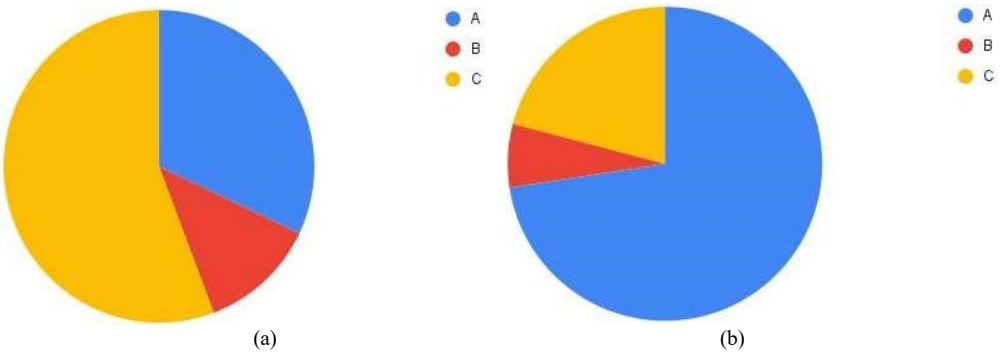
The calculations necessary to determine the alternative are carried out, and the final results are presented in Tables 3-10 and Figures 1-3.

Table 7. Matrix of pairwise comparisons and determination of the vector of priorities of alternatives by safety

Options	A	B	C	A	B	C	Priorities
A	1	3	0.5	0.3003003003	0.375	0.2857142857	0.3203381953
B	0.33	1	0.25	0.0990990991	0.125	0.1428571429	0.1223187473
C	2	4	1	0.6006006006	0.5	0.5714285714	0.5573430573

Table 8. Determining the priority vector of alternatives by availability

Options	A	B	C	A	B	C	Priorities
A	1	8	6	0,7722007722	0,5714285714	0,8333333333	0,7256542257
B	0,125	1	0,2	0,09652509653	0,07142857143	0,02777777778	0,06524381524
C	0,17	5	1	0,1312741313	0,3571428571	0,1388888889	0,2091019591



(a) (b)

Fig.2. Visualization of the vector of priorities of alternatives in terms of security and accessibility

Table 9. Determination of the final vector of priorities of alternatives

Name	C1	C2	C3	C4	Priority
Browser plugin	0,72565422 57	0,1285014006	0,3203381953	0,7256542257	0,52954876 37
Mobile app	0,06524381524	0,2766106443	0,1223187473	0,06524381524	0,14754225 59
Website	0,20910195 91	0,5948879552	0,5573430573	0,2091019591	0,43766162 18

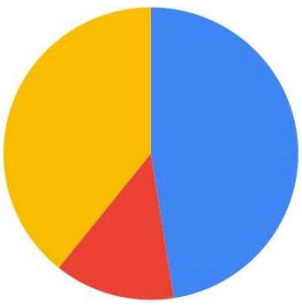


Fig.3. Visualization of the results of MAI – a vector of advantages of alternatives, where blue is a browser plugin, red is a mobile application and yellow is a website

Table 10. Defined final vector of benefits for alternatives

Name	C1	C2	C3	C4	Outcome
Browser plugin	0.7256542257	0.1285014006	0.3203381953	0.7256542257	0.5295487637
Mobile app	0.06524381524	0.2766106443	0.1223187473	0.06524381524	0.1475422559
Website	0.2091019591	0.5948879552	0.5573430573	0.2091019591	0.4376616218

For a more convenient presentation of the results, a diagram has been generated (Fig. 3), from which we can conclude that it is the browser plugin that has the most significant advantage among the three alternatives. So, for a project task with the above criteria, the development of a browser plugin is best suited, as suggested at the beginning.

Here is a comparative analysis of the tool – a browser plugin for detecting propaganda and key narratives – with well-known analogues that deal with disinformation, in particular in multilingual or conflict contexts:

- NewsGuard evaluates the reliability of news sites based on journalistic standards. It gives an overall rating of the site but does not analyse individual articles or sentences. Works with major languages (English, German, French, etc.); there is no support for Ukrainian. It is used in the EU and the USA, and it is not focused on war or conflict situations. Interpretation – explanation of ratings at the site level, without detailed text analysis. It is less flexible because it works at the level of sources, while the proposed tool works at the level of individual fragments of text. It also does not support Ukrainian.
- Hoaxy visualises the spread of misinformation on Twitter. Monitors the vector of narrative spread – infographics, network analysis. Languages – mainly English. Context – social media research; not adapted for everyday fact-checking. Interpretation is a graphical representation of interaction, but it does not explain the meaning of the text. Works with distribution dynamics, not content; The tool from the file instead makes a content analysis.
- ClaimBuster detects in-text claims that require fact-checking. Automatically analyses statements (in English) for facts. Languages – English only. Does not support multilingualism. Assesses the importance of the statement, without analysing propaganda or rhetoric. It is similar in terms of automation of NLP analysis, but it is not adapted for military or hybrid contexts and does not support Ukrainian.
- AdVerify.ai detection of harmful/disinformation advertising in the media. Mainly aimed at brands and advertisers, it detects fake news among affiliate content. Languages – English, does not focus on regional conflicts. More enterprise is not available to the average user as a plugin. The tool from the file has a social mission – to increase media literacy.
- Bot Sentinel analyses the behaviour of Twitter accounts, identifies bots, trolls, and propagandists. It gives the toxicity/bias index of the profile and assesses the political orientation. It focused on the English-speaking space and the distribution actor rather than the content. The tool from the file focuses on text content, taking into account linguistic and narrative features in wartime.

Table 11. General differences and advantages of the PDF tool

Criterion	**Tool from file**	**Analogues (NewsGuard, etc.)**
Level of analysis	Text snippet	Website/platform/profile
Language	UA Ukrainian	Mostly English
Interface	Browser plugin (prototype)	Usually, a web or app
Content Type	News, fragments of texts	Social networks, websites, and statements
Possibility of explanations	Yes (definition of signs, weights)	Limited (rating or rating)
Application context	UA War, hybrid information struggle	General, not conflicting
Narrative analysis	Yes (multi-label approach)	No

Studies [26-28] demonstrate the difficulties of working with these different languages during armed conflicts (e.g. Ukraine, Syria). The study [29] emphasises the importance of creating language-adapted models, which most modern systems do not provide. The proposed project eliminates this problem because it is explicitly trained for the Ukrainian language, which most of the above systems do not do.

The tool from the document is unique in that:

- Works with Ukrainian-language content.
- Adapted to the conditions of information warfare.
- Provides interpreted results at the level of individual sentences.
- And it has the potential to scale to other languages and conflict zones.

3.4. Methods for Research

- Data collection and preparation. Three primary data sources were used to train and test the system:

- Propaganda detection data is part of the Propagandist Pseudo-News dataset, which contains descriptive fragments of news materials in Ukrainian that are propagandistic [30].
- Data for non-propaganda texts – Ukrainian-news dataset from HuggingFace [31], where fragments were obtained from the text column, cropped to the average length of descriptions from the previous corpus.
- Narrative detection data is an annotated set of sentences created as part of participation in the OSINT hackathon, where experts indicated the presence of specific propaganda techniques (Call, Justification, etc.).

To form the final dataset, duplicates were deleted, HTML tags were cleaned, and text was normalised (lowercase, extra spaces removed).

- The pre-processing of the text included:

  - Lemmatisation of Ukrainian texts using *the spaCy* library with a pre-trained model for the Ukrainian language.
  - Detect the language of the text using *langdetect*.
  - Calculation of additional characteristics: the presence of a colon (has_colons), a hyphen (has_hyphens), quotation marks (has_quotmarks), tonality (sentiment_score), subjectivity (subjectiveness_score), the ratio of nouns and verbs (noun_verb_ratio), and the number of recognised organisations and locations.

These features are obtained using the standard functions of *the spaCy*, *nltk*, *TextBlob* (tonality) libraries and self-added logic for counting features (for example, counting the number of characters in text).

- Text vectorisation. For the classification of propaganda, pre-trained GloVe embeddings [32] were used on the body of news articles, which were imported in .txt format and converted into vectors. For the classification of narratives, the transformer model BERT [33] was used through the interface of the transformers library [HuggingFace] [34]. Each text is converted to a numeric vector using the appropriate tokeniser and model.
- Classification models. Random Forest, XGBoost [35] and a neural network of our design (on Keras) were tested for the propaganda classification task. The neural network showed the best results (F1-score = 0.96). For the multiclass classification of narratives, their neural network based on BERT was also used, which allows you to assign multiple labels to the exact text. The network architecture included an input layer with a pre-trained model, one or two dense layers, and an output layer with sigmoid (for propaganda) or softmax (for narratives).
- Decision-making. For the classification of propaganda, the output layer with the function of activating a sigmoid was used; the decision was made according to a threshold value of 0.5 (can be changed in the user settings). For the classification of narratives, softmax was used, and the probability of each class was returned to the user.
- Implementation and Interface:

  - FastAPI (Python) backend, REST API.
  - HTML + CSS + JavaScript browser plugin (MVP version).
  - Communication between the plugin and the server is implemented via HTTP POST requests.
  - Postman API Tests.

- Changes compared to published methods. In contrast to the standard implementation of BERT [33], engineering features (punctuation, ratio of parts of speech, names of organisations, etc.) have been added. Specially translated and created datasets in Ukrainian that have not been published before have been used.
- Repetitiveness. All steps (data collection, pre-processing, model building) were implemented in Python 3.10. To repeat the experiment, it is enough:

  - Python >=3.8
  - Libraries numpy, pandas, scikit-learn, keras, tensorflow, transformers, spacy, nltk, langdetect, FastAPI, etc.
  - Access to original or saved datasets (details described above).

3.5. Architecture of the System of Automated Analysis of Text Content to Identify Key Propaganda Narratives and Disinformation in News Materials

We will demonstrate the interaction of the leading actors with the system and describe the key scenarios for its use. It allows you to visually display the functionality provided to users, as well as the boundaries of the system.

User → Open browser → Download plugin → Highlight/paste text → Click "Classify" → Get classification results in UI

To reflect the structure of the system at the conceptual level, we will use it to describe classes, their attributes, methods, and the relationships between them. Such a model is the basis for the object-oriented design of the system and will provide a further idea of its internal organization (Fig. 4).

User → (uses) *PlugInInterface* → (sends the request to) *Server* → (processes text with) *TextPreprocessor* → (manages) *MLModel*

- User class:
  - attributes of the User class: user_id, username, settings;
  - methods of the User class: select_text(), submit_text(), change_settings();
- PlugInInterface class:
  - attributes of the PlugInInterface class: ui_elements;
  - methods of the PlugInInterface class: send_request(), display_result();
- Server class:
  - Server class attributes: api_endpoint, max_request_amount;
  - Server class methods: preprocess_text(), classify_1(), classify_2(), response();
- TextPreprocessor class:
  - attributes of the TextPreprocessor class: text, features;
  - methods of the TextPreprocessor class: extract_features(), return();
- MLModel class:
  - MLModel class attributes: model_type, prediction, confidence_score;
  - MLModel class methods: predict(), result().

This section presents a Sequence Diagram (Fig. 5) demonstrating the order of messaging between objects in time. This diagram allows you to trace the chronology of operations within individual scenarios, identify time dependencies and ensure correct synchronization between system components.

Let's describe the execution logic of the main processes in the system, including branching, transition conditions, and parallel execution. It allows you to formalize the algorithmic structure of operations performed by the system and will be helpful for further analysis of business logic.

Step 1. Plugin Operation:
Step 1.1. Get text from the user.
Step 1.2. Check the text for correctness.
Step 1.3. Send a request to the server.

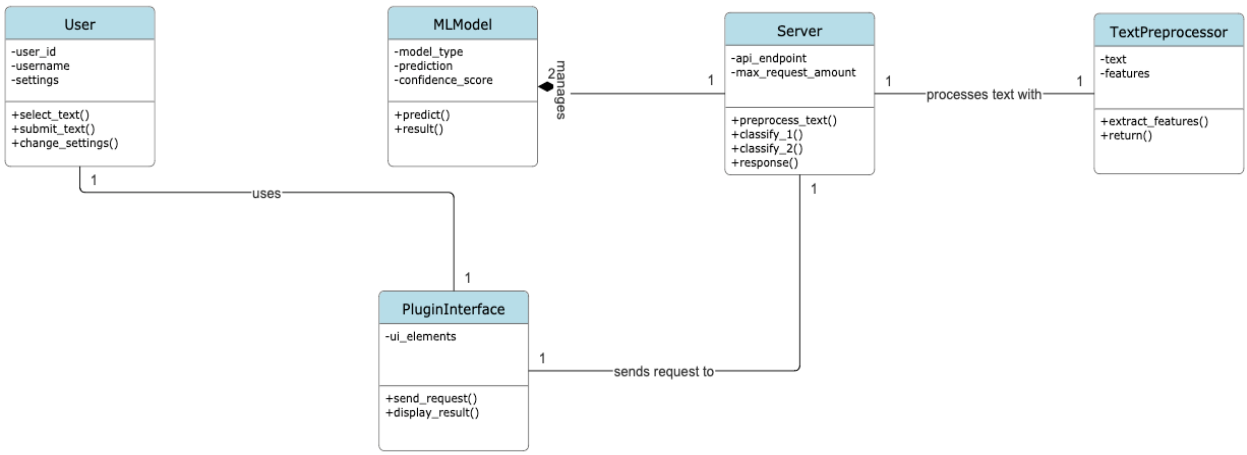


Fig.4. System class diagram

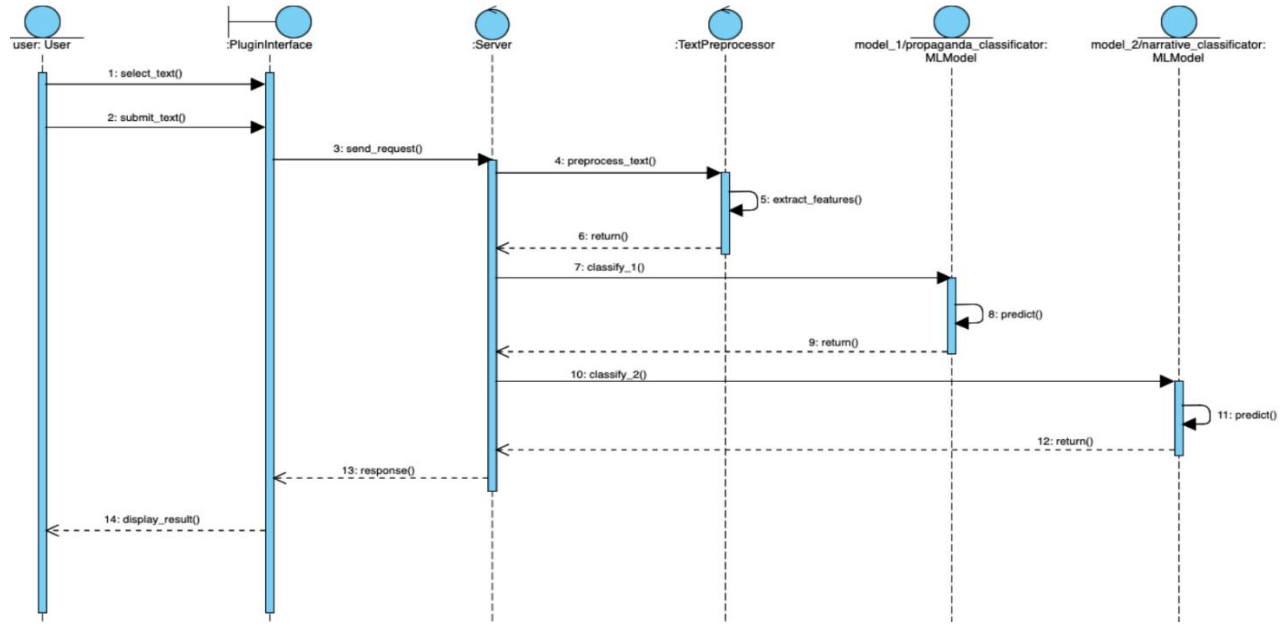


Fig.5. System sequence diagram

Step 2. Server Operation:
Step 2.1. Receive and process the request.
Step 2.2. Initialize the preprocessor for the text.
Step 3. Preprocessor Operation:
Step 3.1. Perform text preprocessing for the model.
Step 3.2. Return the result of preprocessing.
Step 4. The server's job is to initialize the propaganda classification model.
Step 5. Operation of the propaganda classification model:
Step 5.1. Classify the text.
Step 5.2. Return the classification result.
Step 6. Server Operation:
Step 6.1. Work out the result of the classification.
Step 6.2. If propaganda is detected, then proceed to stage 7. Otherwise, form a response and proceed to stage 8.
Step 7. The work of the narrative classification model:
Step 7.1. Conduct a multilabel classification of the text.
Step 7.2. Return the results of the classification and generate an answer.
Step 8. Plugin Operation:
Step 8.1. Get and process the answer.
Step 8.2. Display the response in the UI.

Let's describe the pain in detail how the TextPreprocess subprocess works:

Step 1. Initialize the spacy model for the Ukrainian language.
Step 2. Lemmatize the text.
Step 3. Add the "has_colons" attribute (punctuation affects classification).
Step 4. Add the "has_hyphens" attribute (punctuation affects the classification).
Step 5. Add the "has_quotmarks" attribute (punctuation affects the classification).
Step 6. Calculate sentiment_score (tonality affects classification).
Step 7. Calculate subjectivness_score (subjectivity affects classification).
Step 8. Calculate noun_verb_ratio (formal style affects classification).
Step 9. Calculate location_count and organization_count (named entitles, how they most affect the classification).
Step 10. Use embedding vectors to text.
Step 11. Scaling sentiment_score, subjectivness_score, noun_verb_ratio (so that all numerical features are in the same range).
Step 12. Return the resulting feature vector.

Let's consider the general architecture of the system from the point of view of its components, which is implemented in the form of a Component Diagram (Fig. 6). The diagram illustrates the main program modules, their interfaces and

interconnections, which will allow you to evaluate modularity, scalability, and the level of dependency between parts of the system.

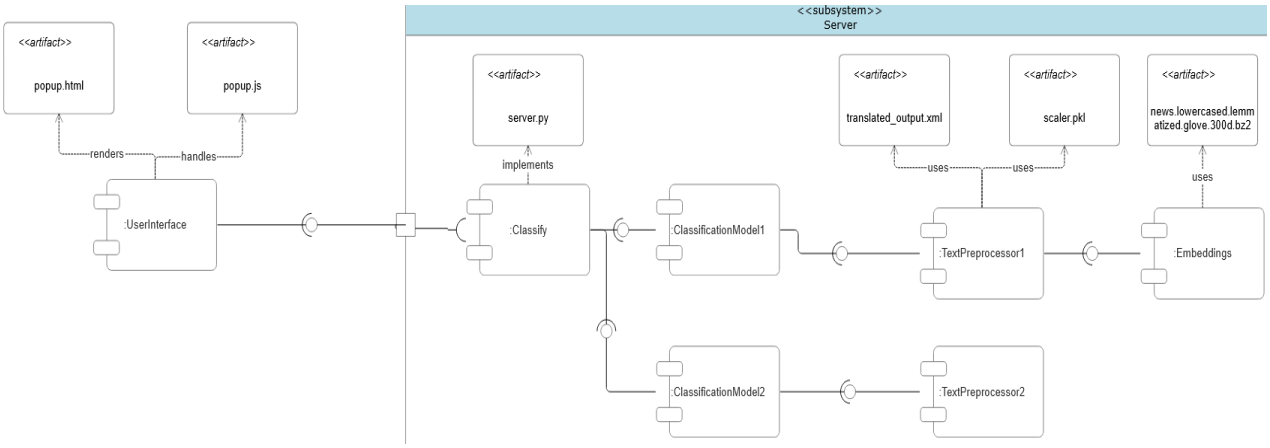


Fig.6. System component diagram

Let's describe the physical deployment of the system using the Deployment Diagram (Fig. 7). It will demonstrate how software components are placed on physical nodes (servers, client devices, etc.), display network connections, and allow you to evaluate the configuration of the system in a real environment.

Browser plugin implementation:

Architecture and functionality. The developed browser plugin is implemented as an extension for Google Chrome (also compatible with Edge and Chromium browsers), which has the following main functions:

- Collection of text content from the first page. Content from visited news sites is automatically extracted via DOM parsing. The plugin is limited only to the main text of articles (avoiding ads, comments, and metadata).
- Pre-treatment. The browser performs easy text cleaning (removal of tags, excessive punctuation, and reduction to 512 tokens).
- Sending to the server. The prepared text is sent via an HTTPS request to a remote API server, where the main inference of the model is performed (the BERT model works on the server, not in the browser). The server responds with a JSON structure with the following features:

  – whether it contains a text about paganism,
  – what narratives were discovered,
  – level of confidence for each category.

- Visualisation of results. The plugin overlays a lightweight interface on top of the page that:

  – shows icons (🔴 /i) next to headings,
  – allows you to expand a short explanation: "Type X narrative detected (78% probability)",
  – highlights fragments that could have been triggers (using tokens with high attention in BERT).

User Interface (UI/UX)

- The plugin has a minimalist menu in the upper right corner of the browser:

  – Activation/deactivation button,
  – The ability to view the history of the analysed pages,
  – The "Send feedback" menu is used to improve the model.

- The interface was designed with users without technical experience in mind, so:

  – All messages are formulated in simple, non-technical language.
  – There is an option to turn the analysis on/off.

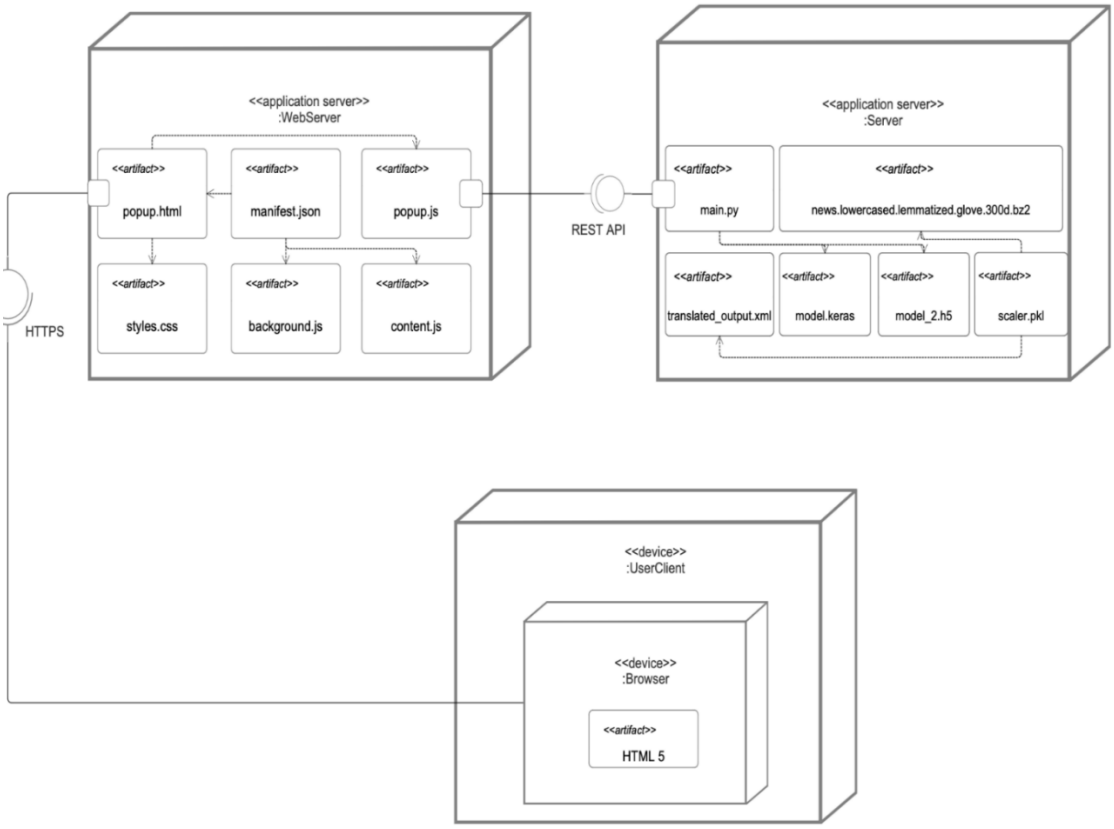


Fig.7. System deployment diagram

Data Privacy and Security. No personal information of the user is collected. All text is anonymised before being sent to the server – URLs, page headers or session IDs are not transmitted. Communication takes place via a secure channel (HTTPS). The server does not store the history of requests, or the results of the requests are given and processed in the operational memory (RAM-only) and are immediately deleted after the account. The plugin does not have access to tabs, browsing history or passwords.

Limitations of implementation

- The model runs to the server and not to the browser, that:
  - reduces the load on the user's device,
  - but depends on the availability of an Internet connection;
- Processing significant texts can take 2-4 seconds, which limits their use in streaming mode and (for example, for social networks).

This description provides transparency in implementation, emphasises the ethics and security of the system, and provides a clear picture of the user's interaction with the tool.

Features of the system's operation with new, never-before-seen disinformation tactics:

- Captures stylistic and rhetorical patterns typical of propaganda:
  - Transformer architecture (BERT) allows the model to detect semantic similarities between new phrases and those that were in training.
  - For example, suppose a new call for "restore historical justice" appears. In that case, the model can compare it with already known patterns ("*liberate our territories", "we were deprived")* → and correctly classify it as an excuse or accusation.
- It analyses emotional and manipulative load, even in new contexts: the model learns to recognise emotional constructions, modal verbs, negations, polarised phrases – and this allows it to partially detect innovations in form rather than content.
- Captures multilingual or veiled propaganda if the structure remains similar: the multilingual version of BERT (mBERT) allows for some flexibility in presentation style and word forms.

Inability to reliably perform some actions by the system, without additional training, in particular:

- Does not recognise completely new topics or narratives that are not presented in the teaching. For example, if a new campaign appears that does not condemn the enemy, but appeals to common pain (emotional unification through humanitarian topics), the model can:
  - not classify it as propaganda,
  - or even consider it neutral, if the vocabulary is restrained.
- Vulnerable to disinformation stylised as "objective analytics". If the content is disguised as expert, does not contain emotions, and distorts only statistics, a model trained on rhetorical examples may miss such content**.**
- Doesn't automatically create new narrative classes**.** The model cannot "deduce" a new type of narrative; it can only be chosen from those who were in training. To update, you need to manually add new labels and complete additional training.

The model is able to partially generalise new disinformation tactics if they are based on familiar stylistic or manipulative patterns. However, accurate adaptation to new narratives and topics requires constant data updates, manual re-annotation, and additional training. Without this, the system will lose its relevance over time. Recommendations for the future are as follows:

- Embed a module for detecting new patterns (for example, clustering embeddings of new texts).
- Add a mechanism for semi-automatic extension of narrative taxonomy (e.g. through weak supervision or active learning).
- Involve journalists, media analysts, or fact-checkers in the constant annotation of new cases.

3.6. Setting the Task of Automated Analysis of Text Content to Identify Key Propaganda Narratives and Disinformation in News Materials

A. Purpose and Place of Application of the System

The functional purpose of the system is to automate text analysis, which detects propaganda and identifies its key narratives. The social purpose and the main reason for the development of such a product is to increase the information hygiene of users by quickly and conveniently checking content on the Internet.

The primary audience of the product will be people who constantly use news sites and want to increase their awareness of the veracity of the information they consume.

The product is planned for development in the form of a browser plugin because this is what will allow almost instant verification of the necessary information and will also be convenient for users as part of the initial download - the plugin does not need to be installed locally; it is not tied to specific systems or devices - you need to go to the plugin market of your browser. The designed system can be most widely used as a personal assistant expert for a user who needs an independent assessment of news resources for the presence of propaganda.

B. Description of System Requirements

In accordance with the defined business processes, the following set of requirements for the system is defined:

- Browser plugin for analysing text for propaganda;
- Cross-platform support (Chrome, Firefox, Edge, Windows, Linux, MacOS);
- Classification of the text according to the level of propaganda and identification of key narratives;
- High speed of analysis (1-2 seconds per text);
- Support for local and server-side analysis;
- Protection of user privacy, no collection of personal data;
- Clear and intuitive interface;
- High classification accuracy due to the use of modern ML models (BERT/RoBERTa);
- Scalability and the ability to integrate via API;
- Ability to update the model with new data;
- Reliable and uninterrupted operation under high load;
- Open API for integration with other services;
- Easy to install and use without special knowledge.

To structure requirements, they are divided into business requirements, user requirements, functional requirements, and non-functional requirements:

Table 12. System requirements

Type of requirements	Business Requirements	User requirements	Functional requirements	Non-functional requirements
Appointment	Determine the goals of the business structure, the achievement of which the startup will ensure, and the problems that it will solve	Tasks and actions of users, the implementation of which will be provided by the startup	Description of what the system being developed in an innovative DS startup should do	A description of how a system developed in an innovative DS startup should work to perform its functions
Content of requirements	1. Creation of a tool for analyzing texts for propaganda within the framework of information warfare. 2. Implementation of NLP and ML technologies to automate the detection of manipulative content. 3. Increasing the level of media literacy and users.	1. The user should be able to highlight the text and get the result of the analysis. 2. Ability to view classification details. 3. Option to send feedback to improve the model. 4. Adjusting the sensitivity level of the algorithm.	1. Text analysis based on the ML model (BERT/RoBERTa). 2. Definition of the category of propaganda (emotional pressure, distortion of facts, etc.). 3. Connecting to the News API and other sources. 4. Logging and saving user history.	1. Text processing time is no more than 2 seconds. 2. The model has a high accuracy (at least 80% F1-score). 3. Data encryption and user privacy protection. 4. GDPR compliance.

C. Definition of Project Goals

The defined requirements allow you to create a list of project goals necessary for the implementation of:

1. Technological implementation:
1.1. Creation of a DL model capable of analysing text for propaganda with an F1-score of at least 80%;
1.1. Implementation of the algorithm for detecting propaganda narratives;
1.1. Performance optimization for fast real-time word processing;
2. Ease of Use:
2.1. Development of an intuitive interface for a browser plugin (Chrome, Firefox, Edge);
2.2. Providing users with explainable results (highlight key propaganda elements in the text);
3. Scalability and Integration:
3.1. Support for local and server-side word processing;
3.2. Empowerment through integration with fact-checking platforms and analytical services;
4. Social and informational impact:
4.1. Increasing the media literacy of users through automatic text analysis;
4.2. Using the project as an analytical tool for fact-checking and countering information operations.

D. Effects of System Implementation

After determining the main goals of the project, it is essential to analyse the expected results of the implementation of the developed system. This section discusses the practical effects that are achieved as a result of the implementation of the set goals. Their further evaluation will make it possible to justify the feasibility of implementation and outline potential benefits for the organization and end users.

Table 13. Effects of project implementation

Target	Expected implementation effect
Creating an ML model for propaganda analysis	Identifying propaganda narratives in texts with high accuracy, reducing the spread of disinformation.
Browser plugin implementation	Providing users with a quick and convenient tool to check information directly in the browser.
Performance optimization	Real-time text analysis without significant load on the user's system.
Explainability of results	Increasing user confidence in the results of the analysis improves the interpretation of propaganda tactics.
Scalability and integration	The ability to use the technology in an international context, expanding to other languages and regions.
Improving Media Literacy	Improving users' critical thinking reduces the level of manipulation in society.
Integration with fact-checking platforms	Improving the effectiveness of disinformation investigations and media analytics.
Countering information attacks	Identification and analysis of information operations in the media space, strengthening information security.

3.7. Identification of Methods of the Developed Product of Automated Analysis of Text Content to Identify Key Propaganda Narratives and Disinformation in News Materials

A. Determination of Methods of the Product Being Developed and their Possible Alternatives

In the process of developing a system for detecting propaganda and identifying narratives, a number of modern

methods of natural language processing, machine and deep learning were chosen. The main task of the system is the automatic recognition of propaganda messages in the text, which requires an accurate analysis of the context, lexical features and semantic relationships. Taking into account the complexity of language constructions, the variability of ways of presenting information and the constant variability of methods of manipulation, the methods were selected in such a way as to ensure maximum accuracy of classification while maintaining productivity. So, taking into account the above-mentioned task of the system, the following methods (and their alternatives) are defined:

Table 14. A set of other options to methods of solving problems

Task	Possible methods (not exclusive)
Collection of Study Materials	– Kaggle datasets; *– HuggingFace datasets;* – Self-collection and labelling of data; *– Datasets from other resources;*
Text classification (or is the text propaganda)	– machine learning models (SVM, Random Forest, XGB...); *– neural networks.*
Classification of propaganda (whether propaganda contains certain narratives)	– machine learning models (SVM, Random Forest, XGB...); *– neural networks.*
Text vectorization (because for the defined possible classification methods, text is required to be presented in vectorized form)	– BoW; – TF-IDF; *– trained text embeddings (Word2Vec, FastText, GloVe, Doc2Vec…);* – mechanisms for teaching text embeddings (Word2Vec, FastText, GloVe, Doc2Vec...*)*; *– Transformers (BERT...).*
Decision-Making Based on Text Classification	*– Sigmoid (threshold must be defined).*
Decision-making based on the classification of propaganda	– Sigmoid (it is necessary to determine the threshold); *– Softmax (threshold must be defined).*
Text preprocessing	*– Feature Engineering (punctuation,* semantic component, sentence/document length...*);* *– Lemmatization;* – Stemming; *– Sentiment analysis;* *– Subjectivity analysis;* *– NER* *– POS*

B. Determination of Comparison Characteristics and Comparative Analysis

To choose the most appropriate method for the task, it is necessary to determine qualitative or quantitative indicators that will allow for evaluation.

For the task of collecting educational materials, it would be necessary to indicate such a quantitative indicator as size and such a qualitative indicator as reliability. However, for a specific project task, the collection is already limited, so choosing only one of the methods is an irrational decision.

For the task of determining the necessary steps of text preprocessing, it is impossible to clearly indicate a quantitative indicator. Still, during the ED analysis, it is possible to qualitatively assess the contribution of each of the methods to the detailing of information about a particular document or a specific class of document. Therefore, before approving all the steps and creating a pipeline for text processing, a preliminary analysis of the training data will be carried out.

The choice of text vectorization method directly depends on the task and available resources. Since the best and most effective modern methods of text vectorization are embeddings and BERT-transformers, the analysis will be reduced to a choice between them.

For classification tasks, the quantitative indicator is the metrics of model results. Due to the incomplete balance of data for the propaganda classification model and the critical imbalance of data for the narrative classification model, the F1-score (weighted) and precision/recall assessment of individual classes were chosen as metrics. So, in both cases, machine learning models (Random Forest, XGBoost) and the use of neural networks will be compared.

The task of decision-making after the classification of propaganda is to choose the necessary threshold because only Sigmoid can be used to convert results into probability. The selection will be made based on the testing results.

The task of decision-making after the classification of narratives is also to choose a function to convert the results into vector(s) of probabilities. Since this is a multi-label classification task, we can choose from several options – Sigmoid and Softmax. The first option will allow you to set specific thresholds so that a particular class can be considered selected. In the second option, it will be possible to determine the most likely options and the least likely ones. Thresholds must also be defined.

C. Selection and Approval of Methods of the Product Being Developed

Therefore, for each of the methods of the product being developed, alternatives and methods of comparative analysis are defined. Specific procedures have been carried out, and the following conclusions have been made:

- Collection of training data – all methods will be used due to the limitations of thematic materials in general;
- Pre-processing of the text – after the ED analysis, it is determined that the following methods will be used: adding information on punctuation, adding information on the distribution of specific NERs in the sentence (location and organization), adding information on sentiment and subjectivity, adding information on a specific ratio of POS tags in the sentence.
- Text vectorization – several options have been tested, in particular: FastText, GloVe, and BERT-transformers:

  – For the task of classifying propaganda, it was decided to leave the already trained GloVe vectors because it was the option with them that gave the best performance – which is quite predictable because these vectors were trained on news articles, which are a key component of the training materials used for project classification;
  – For the task of classifying narratives, it was decided to use BERT transformers because they made it possible to get the best and most accurate results. Since the training materials consist of statements of opinion on the network, the models trained from news articles no longer cope with the task so well.

- Classification of propaganda and narratives – the choice was made by testing all options and choosing the one that gave the best results according to the selected metric (F1-score). In Table 13, you can see the summary results and make sure that it was the option with the construction of your neural network that provided the highest F1-score for both classification problems - so it was chosen:

Table 15. F1-score (weighted) results

Classification method	Task	
	Classification of propaganda	Classification of narratives
Random Forest	0.86	0.47
XGBoost	0.92	0.5
NN	0.96	0.7

- Decision-making based on classification – the choice was made partly based on the results of the models, partly on the principle of user-friendliness:

  – For the propaganda classification task, it was decided to use Sigmoid for the final layer. The default threshold will be 0.5, but the user will be given the option to change this value between 0.5 and 0.85;
  – For the narrative classification task, Softmax is used as the final layer. The final result will be determined as the option with the highest probability. Still, the user will be presented with a complete list so that they have the opportunity to draw certain conclusions themselves.

Architecture and adaptation of the BERT model

- Model selection. The study used a multilingual version of the BERT model – [bert-base-multilingual-cased] (mBERT), which supports more than 100 languages, including Ukrainian. The choice of this model is due to:

  – Support of the Cyrillic script,
  – The availability of pre-trained scales for Ukrainian texts (although in smaller numbers than for English),
  – stable performance in low-resource languages without the need for complete retraining.

- Adaptation to Ukrainian-language content. Additional fine-tuning was carried out on specially assembled Ukrainian corpora (propaganda / non-propaganda, as well as narratives), which allowed the model to better adapt to the lexical and syntactic features of the Ukrainian language. For this purpose, news segments, translated and cleaned phrases from social networks, and manually marked narrative annotations were used.

- Architectural features. The basic architecture of BERT remained unchanged: 12 layers, 768 hidden units, 12 attention heads. Separate classifiers (fully connected layers) were added to the output token [CLS] for each task:

  – For propaganda classification: single-layer classifier → Sigmoid → binary score (0 or 1).
  – For narrative classification: multi-output classifier with five neurons → Sigmoid → multi-marker classification (each class: 0 or 1).

For both tasks, the Binary Cross-Entropy (BCE) loss function was used:

  – Separately for propaganda.

– In total, for each label, for narratives (BCE per label, averaged).

- Text processing. Texts were cleared of HTML tags, URLs, emojis, and unnecessary punctuation marks before processing. Lemmatisation was used for better generalisation. Long texts were divided into segments of up to 512 tokens (maximum length for BERT), with the results being combined at the document level (through the aggregation of softmax scores).
- Joint or separate learning. Each task (propaganda/narratives) was trained separately from scratch (based on pre-trained mBERT). Thus, two separate models with the same architecture but different weights and tasks were used.

This description clearly outlines which version of BERT was used, how it is adapted to the Ukrainian language, how the architecture is arranged for each of the subordinate tasks, and what the processing and presentation of input data looked like.

3.8. Identification of the Means of the Developed Product of Automated Text Content Analysis to Identify Key Propaganda Narratives and Disinformation in News Materials

A. Identification of Alternatives to the Means of the Product Being Developed

The implementation of the proposed system required the involvement of a wide range of software, systems, and auxiliary tools, both on the server processing side and on the client interface. Modern deep-learning libraries are used to build high-precision models, and a web server that implements REST APIs is used to ensure interaction between the browser plugin and the model. Particular attention is paid to ensuring the compatibility of the selected tools, ease of deployment, and the flexibility and scalability of the solution. So, taking into account the above-mentioned functionality and methods of the system, the following means have been defined:

Table 16. Multiple alternatives to problem solvers

Task	Possible remedies (not exclusive)
API Server	– *Flask;* – FastAPI; – Django.
Web interface	– React; – *JS + HTML + CSS;* – Vue.
Text Preprocessing	– *spaCy;* – *NLTK*.
Testing	– *Postman*; – curl; – Insomnia.
Analysis and prototyping (environment)	– Google Colab; – *VSCode;* – Kaggle.
Version control	– *Git + GitHub;* – Git + GitLab; – Git + BitBucket.
Analysis and Prototyping (Core Libraries)	– *nltk;* – *spacy;* – *numpy;* – *pandas;* – *matplotlib;* – *seaborn;* – *tensorflow;* – *torch.*

B. Determination of Comparison Characteristics and Comparative Analysis

The definition of the environment directly depends on the local capabilities of your device, which will be used for model development. Since neural networks are chosen as a method for creating classifiers, the device must be able to work with the GPU. If the device doesn't have a GPU, Google Colab and Kaggle are good alternatives - but they have usage restrictions.

There are no specific characteristics to compare because Git will definitely be used as a system. Therefore, the choice of platform depends on the developer – the most comfortable, convenient and familiar for work.

The specificity of the project under development almost entirely makes it impossible to choose a programming language because it is Python that provides all the necessary resources. Despite this, you need to select the required libraries that will be used during analysis and development. The key characteristics for comparison are accessibility, convenience and accuracy (for creating models). It is necessary to choose libraries that will allow you to analyse text, create pre-processing pipelines, and create models. Python will also be used to generate the server part of the project; therefore, it is necessary to select resources for such a task.

There are also several alternative solutions for developing a web interface for users. At the initial stage of the project, the web interface is only an initial prototype, so the one that will allow you to create an interface part quickly and efficiently will be selected from the proposed alternatives. For further development, it will be necessary to clarify whether this method is generally accepted for this nature of the task and whether it will be able to satisfy all further possible implementations.

To test the created REST API, you will also need to select a tool from the possible selected options - curl, Postman, Insomnia. The most popular and affordable tool will be chosen.

C. Selection and Approval of the Means of the Product Being Developed

So, after carrying out the above analyses, the following conclusions were drawn:

- Analysis and prototyping environment – own device allow the use of GPUs; hence, VSCode was chosen.
- Version control environment – the main characteristic to choose was developer-friendliness, so the option of using a combination of Git + GitHub was chosen.
- Analysis and development of prototypes (libraries) – after a detailed analysis of the available resources, the necessary libraries are selected (Table 17).

Table 17. Selected python libraries

Library / Module	Appointment
nltk, nltk.tokenize	Tokenization of text into sentences and words
spacy	Lemmatization, POS analysis, Ukrainian/Russian text processing
pymorphy2	Lemmatization of the Russian language
langdetect	Definition of Text Language
bs4 (BeautifulSoup)	HTML/XML parsing (e.g. to clean up tags)
xml.etree.ElementTree	Processing XML files
collections. Counter	Calculating frequency characteristics in texts
itertools.islice	Working with iterators (e.g. batching or pruning)
string	Working with symbols, punctuation
bz2	Extracting compressed files (e.g. .bz2 with embeddings)
json, os, pickle	Working with files, saving models, dictionaries, configs
typing	Type annotations in functions
numpy	Working with arrays, vector computing
pandas	Tabular data processing (datasets, statistics)
matplotlib.pyplot, seaborn	Visualization of results, graphs, heatmaps
tqdm	Progress bars for loops (e.g. processing a large dataset)
sklearn.model_selection	Splitting data into train/test (train_test_split)
sklearn.preprocessing.S tandardScaler	Feature scaling
sklearn.metrics	Evaluation of models: accuracy, F1, confusion matrix, etc.
sklearn.ensemble.Rand omForestClassifier	Classification algorithm (tree model)
xgboost, XGBClassifier	Gradient boosting – a powerful model for text classification
tensorflow, keras	Building, training, and testing neural networks
transformers (AutoTokenizer, AutoModel)	Using BERT transformers to build embedding
torch	PyTorch backend for working models with transformers

- Web interface development – For web interface development, an approach using JS + HTML + CSS was chosen because this is the fastest and easiest way to create a web plugin that almost any developer can implement.
- REST API Testing – After a study conducted, it was found that one of the best platforms that provide testing capabilities for created REST APIs is the Postman platform. Therefore, during the development of the project, it will be used to test the created server part.

The study used three separate data sources to train and evaluate classification models:

- For the task of classifying propaganda, a combined corpus was used:
  - Propaganda texts are obtained from the [Propagandist Pseudo-News dataset], which contains news created by Russian sources as part of a hybrid information war. Ukrainian-language fragments (in particular, the description field) were selected, which made it possible to form a relevant corpus of propaganda content.

  - Non-propaganda texts were selected from Ukrainian news resources posted on the HuggingFace platform (zeusfsx/ukrainian-news). To balance the length of the texts, truncated news fragments were used, similar in size to the description fragments from the propaganda set.
  - After cleaning, the size of the case was several thousand samples in each class. The class distribution is aligned to about 1:1 for the binary classification problem (propaganda/non-propaganda).

- For the task of classifying narratives, the dataset obtained as part of the OSINT hackathon was used. Experts manually annotated the sentences for key propaganda techniques:

  - Accusation, Justification, Appeal, Incitement to hatred, Humiliation of national dignity.
  - Labels are multi-labelled, meaning that one text could have multiple active categories at the same time.
  - The data was originally in Russian, but translated into Ukrainian to ensure consistency with the rest of the corps.
  - The number of samples per label was between 300 and 600, which allows training but creates a significant disproportion between classes, especially in the case of multiclass classification.

- Additional auxiliary corpora are created to assess the sentiment and subjectivity of texts. Due to the lack of such data in Ukrainian, the translation of English-language marked dictionaries, taking into account parts of speech, was used.

Annotation process:

- For propaganda samples, we used resources that had already been collected with appropriate marks.
- For non-propaganda, the selection was carried out from verified Ukrainian media.
- For narratives, the abstract was made manually by information security experts.

The distribution of data into training, validation and test sets was carried out in the proportion of 70/15/15.

It is possible to adapt the model to other languages or new narratives. Still, its complexity depends on what kind of modification needs to be made, in particular, to different languages or new narratives.

Adaptation to other languages based on multilingual BERT (mBERT) is possible without full retraining. The model already supports more than 100 languages, in particular English, Russian, Polish, Romanian, Bulgarian, etc. If the other language is covered well in the BERT corpus, the basic accuracy will be preserved. To do this, you need to collect examples of propaganda/neutral texts in the new language. Next, you need to fine-tune in this language (even on 1-3 thousand examples) – This is already enough for the model to learn rhetorical patterns. Or you need to apply a zero-shot approach if the quality is not critical (i.e. immediately run the model in another language until the dataset is assembled). But this adaptation has limitations. Languages with less support in mBERT (e.g. Turkic, Caucasian, and Small African) require more localised data.

Adaptation to new narratives is possible, but requires additional annotation and additional training. The BERT model is not able to "invent" a new category on its own. To classify new types of narratives (for example, "anti-mobilisation messages", "hybrid legitimisation through international law"), it is necessary:

- Describe a new narrative class.
- Collect examples of texts (at least a few hundred).
- Train the existing model with the addition of a new source node in the classification layer (multi-label setting makes it easy to expand the label space).
- Test on new data to verify accuracy.

Table 18. Analysis of the adaptation of the model to other languages or new narratives

What we adapt	Needs to be done	Complexity
Another Language (BERT)	Fine-tune mBERT on a small body	Low/Medium
A new narrative	Add a class → collect at the treasure → further teach	Average
Another language + new narratives	Complete additional training on the new building	Medium/High

The interpretation of the model is one of the key advantages of the developed system, which allows users to understand why a specific piece of text was classified as propagandistic or containing manipulative narratives. It is implemented through the following processes and methods:

- Visualisation of results in the browser plugin:

  - The user receives highlighting of key phrases or elements that determine the classification of the text.
  - It allows you to understand *which specific lexical or syntactic constructions* signal the presence of propaganda

or a particular narrative.

- Indicators of input features for classification:

  – Among such signs are the presence of colons, quotation marks, dashes, sentiment score, subjectiveness, the ratio of nouns to verbs, and the number of mentions of organisations or geographical names.
  – For example, if the text contains many names of organisations and a sharp negative tone, this may signal manipulation or a narrative of hostile origin.

- Use of multidimensional classification models (for example, multilabel classification). When several narratives are detected, the model does not just give out "the text contains propaganda", but indicates *which narratives* are detected: calls for hatred, justifications, humiliation of honour, etc.
- Explanation of the model through probabilities or confidence scores. Each classification is accompanied by a confidence value (e.g. 87%), which allows the user to assess the reliability of the classification and make a decision for themselves.
- Configurable sensitivity threshold. The user can change the threshold value (e.g. 0.5–0.85) in order to control the "rigidity" of the assessment – this adds transparency and flexibility to the results.
- Ability to view classified results**.** After classification, the system not only provides a summary but also allows you to view details, in particular, which features and parts of the text influenced the model's decision.

Advantages of this approach:

– Improves users' trust in the system, as they can see not only the result, but also its justification.
– It contributes to the development of media literacy because the user learns to recognise typical signs of propaganda in the text.
– Increases efficiency – journalists, fact-checkers, and analysts can quickly identify the source or nature of manipulation.

4. Experiments and Results

4.1. Identification and Selection of Data Sources

A. Determining the Source of Data Classified as Propaganda

To be able to build a classifier to define propaganda, it is necessary to determine what data can be called propaganda and where it can be found. Since the project is aimed at classification among news articles, the data must also be at least news in nature. Unfortunately, there are not many collected resources with labels in Ukrainian in the public domain, so the data will have to be combined independently. After a detailed analysis of the sources available on the Internet, it turned out that the task of collecting fake propaganda news within the framework of the Ukrainian-Russian war had already been set. This project ended with the creation of a dataset with news produced by Russia in different languages on various resources that were aimed at users from other countries [30]. This dataset contains a lot of information, but our project only needs news that is written in Ukrainian. There are almost no complete Ukrainian articles among these data, but it is possible to take the description of the article, which is contained in the 'description' field (Table 19).

Therefore, it is this data that will be used with the tag propaganda when creating a classifier.

Here is a complete description of the dataset that you can add to the *Dataset* or *Materials and Methods section* of the article. It covers all the key elements: source, size, time frame, type of content, annotation principle and approach to the formation of articles. For the training and evaluation of models, two independent corpora have been formed:

- For the classification of propaganda (binary problem),
- and the classification of propaganda narratives (multi-marker problem).

- Dataset for Propaganda Classification:

  – Source. Propaganda texts were selected from the open corpus of russian information campaigns, such as *"Pseudo-News Dataset"* (within the framework of OSINT hackathons and information operations detection reports). Neutral/informational texts were selected from a variety of sources. It is Ukrainian news platforms (for example, the zeusfsx/ukrainian-news dataset with HuggingFace).
  – Size: ~7000 texts (≈3500 per class after balancing).
  – Types of texts: short news messages, news snippets (up to 512 tokens).
  – Time frame: Most samples cover the period 2022–2023.
  – Annotation methodology. Propaganda texts had pre-labelling (e.g. label=propaganda field). For neutral texts, the source and tone were manually checked. Obviously, sarcastic or parodic samples were excluded.

Table 19. Columns of the propagandist pseudo-news dataset

Name	Description
authors	Authors of the article, often the publisher's website
date_download	Initial article upload date to keep track of possible changes between versions of articles
date_modify	If the same article has been deleted multiple times, this will be the last upload time.
date_publish	The date on which the article first appeared on the Internet. In some cases, this field may not be accurate.
description	Summary of the article, which can be seen in the preview for RSS feeds
filename	The name of the corresponding HTML file is provided in the same subfolder as the metadata file.
image_url	URL of the first linked image in the article. It can be used to make the dataset multimodal, with a choice of illustrations to make readers react.
language	The developer evaluated the intended language of the article. It is well true for multilingual websites but mostly incorrect for articles from French-language reading sites, which are mostly labelled en
title	Article title
title_page	The title that will be displayed in the tab name of the web browser
title_rss	The title that RSS scrapers will use
source_domain	Website's Initial Domain
main_text	The main content of the web page. It works well for most languages, but the module has encountered difficulties with articles written in Chinese, for which this field may be empty
url	Article URL

- Dataset for Narrative Classification:

  – Source. Hand-compiled from public posts, news and reports that demonstrate typical narratives of Russian disinformation in the Ukrainian information space. Some of the texts are adapted (translated from Russian into Ukrainian) from the corpus created by experts in information security.
  – Categories/narratives: accusation, justification, call to action, incitement to hatred, humiliation of national dignity.
  – Markup type: multi-marker (one text can have several active markings).
  – Size: 3200 examples with at least 300 labels of each type.
  – Annotation methodology. The texts were marked up by two independent groups of experts, after which a coherence wave was carried out (Cohen's Kappa > 0.80). To increase accuracy, an annotator instruction was created, with examples of phrases for each type of narrative.
  – The time frame covers the post-February period of the war, 2022-2023.

General Features:

  – The language is exclusively Ukrainian (some texts are also translated by hand).
  – Lemmatisation-based preprocessing, as well as punctuation cleaning, URL and stop word deletion.
  – All data are divided into a ratio of 70/15/15 (train/validation/test), with the separation of labels preserved.

This extended description not only ensures reproducibility but also allows us to evaluate the generalisation of the model, taking into account the time and frame, the variety of sources, the balance of classes, and the quality of the annotation.

B. Determining the Source of Data Classified as Non-propaganda

Since news articles in Ukrainian are taken for propaganda, the logical next step for identifying articles labelled "not propaganda" is to collect articles from reliable Ukrainian resources. Two options were analysed - self-scraping or dataset search. First, the second idea was tested – it helped to find already collected Ukrainian news articles from Ukrainian news resources on HuggingFace – a platform where users can share pre-trained models, datasets, and demonstrations of machine learning projects [31]. The 'text' column (Table 20) contains the text of news articles, but since not the full texts are taken from the propaganda dataset, but only the description, it is necessary to trim the data.

Table 20. Dataset columns ukrainian-news

Name	Description
title	News article title
text	News article text, which may contain tags HTML (e.g. paragraphs, links, images, etc.)
url	News article URL
datetime	Post time or when the article is analysed and added to the dataset
owner	Name of the website that published the article

For this purpose, the average length of the text in the 'description' column (Table 19) was analysed, and a new column 'truncated_text' was formed, which contains the texts from the 'text' column (Table 20) truncated to medium length.

C. Determining the Source of Data with Classified Propaganda Narratives

There is no high-quality data in the public domain that would meet the defined goal. However, while participating in the OSINT hackathon, data that experts in propaganda had previously marked up was obtained.

Table 21. Dataset columns ukrainian-news

Name	Description
Sentence	Sentence/Expression
Objections	Is (1- True, 0 - False) present in the sentence?
Justification	Is (1- True, 0 - False) present in the sentence?
Call	Is (1- True, 0 - False) present in the sentence?
Inciting enmity and hatred	Is (1- True, 0 - False) present in the sentence?
Humiliation of national honour and dignity	Is (1- True, 0 - False) present in the sentence?

The only problem with using this data was the language because it is formed from Russian sayings. To solve this issue, the 'Sentence' field has been translated into Ukrainian.

Two separate enclosures were created to train and test the models:

- to detect propaganda (binary classification),
- for the classification of propaganda narratives(multi-marker classification).

Table 22. Data structure

Task	**Size**	**Tags**	**Abstract Type**
Propaganda (binary)	~7000	0 (none), 1 (yes)	Semi-automatic
Narratives (multi-label)	~3200	5 categories (1-5 labels/text)	Manual

Let's describe them in more detail.

- Propaganda Detection Dataset:
  - Problem type – binary classification (propaganda / non-propaganda)
  - Sources. Propaganda samples are obtained from open datasets used in information operations (in particular, Pseudo-News datasets). In addition, texts from Telegram channels, which were recorded during the OSINT analysis, were also used. Non-propagandistic samples are texts from Ukrainian news portals, presented in the set and zeusfsx/ukrainian-news, as well as short messages without flawed content.
  - The markup was carried out semi-automatically. Existing annotated sources collected their labels. New examples were annotated by experts who were instructed and guided by strict criteria for determining propaganda.
  - Body size ~7000 texts (≈3500 in each class).
  - The time frame is mainly 2022-2023.

- Dataset for the classification of propaganda narratives:
  - The type of problem is multi-marker classification (one text can have a number of active markers).
  - Mark-up: manual, with the involvement of a group of annotations (6 – media analysts, specialists in communications and information security).
  - Markup methodology. Annotators received instructions with examples for each narrative. Two annotators analysed each text, and in case of disagreement, agreement was made through a third.
  - Number of unique narratives: 5 main categories identified on the basis of the analysis of typical topics in disinformation campaigns:
    - Accusations (e.g., "the authorities are guilty of war")
    - Justification (e.g., "aggression is a response to provocation")
    - A call to action (e.g., "you need to protest")
    - Incitement to hatred (e.g., "they are all enemies")
    - Humiliation of national dignity (e.g., "the Ukrainian language is unnecessary")

- Body size ~3200 texts; The average number of active narratives for the sample is 1.4.
- The time frame is 2022-2023.
- The language is Ukrainian (some fragments were translated from Russian, with subsequent translation by native speakers).

This description provides transparency and reproducibility and allows you to evaluate the quality of the markup and the amount of work. Classification of propaganda narratives: structure and sources:

- The types of narratives were predefined and manually based on:

  - analysis of existing disinformation reports (including EUvsDisinfo, StopFake, VoxUkraine projects),
  - expert assessment of the material of disinformation campaigns in the IOD2022-2023,
  - results of the OSINT hackathon, where the sub-classification was used.

It means that the classification is not automatically clustered from data (unsupervised) but relies on a context-based, domain-significant typology.

- 5 main types of propaganda narratives cover the most common disinformation messages in the Ukrainian information space:

Table 23. Narratives

№	Title of the narrative	Brief description
1	Prosecution	Placing the blame on the authorities, NATO, Ukrainians, the West, etc.
2	Excuse	Rationalisation or legitimation of aggression (e.g., "we were forced").
3	Call to action	Aggressive or provocative calls for protest, sabotage, aggression, etc.
4	Incitement to hatred	Spreading hatred, humiliating other groups, and stigmatising.
5	Humiliation of the national Dignity	Offensive statements about the Ukrainian language, culture, and national identity.

These categories are not exhaustive, but they cover the core of relevant destructive topics identified in propaganda narratives in Ukraine during the full-scale war.

- Marking narratives. Each text was manually evaluated by experts for the presence or absence of one or more narratives (i.e., it is a multi-marker problem where text can receive 0–5 labels). For consistency, clear descriptive instructions with examples for each type were used. The mark-up was performed by two independent experts, followed by coordination for the third time in a row and disagreements. A high level of inter-notation agreement was achieved (Cohen's κ $\kappa > 0.8$).
- During the previous analysis, it was found that the above and broad categories often overlap, and more specific tactics (accusations, justifications, etc.) are more diagnostically accurate and operationalised. However, in the future, it is planned to introduce supercategories (for example, grouping narratives by directions: geopolitical, social, ideological).

The types of narratives were also defined manually based on expert analysis. The model classifies five well-defined types of disinformation in the Ukrainian media field, for practical application (education, OSINT, journalism).

D. Creating Mark-up Data to Determine Sentiment and Bias

For the Ukrainian language, several variations of data with words and their specific tonality have been found. There were no marked data with subjectivity for the Ukrainian language at the time of work on the project. Within the framework of the limitations of the language issue - the Ukrainian language is quite difficult for the manual creation of such mark-up data - it was decided to create such data through the translation of the existing English version [36].

The data contains characteristics such as polarity – tonality dimension from -1.0 to 1.0, subjectivity – bias dimension from 0.0 to 1.0, and pos – part of speech tag. When translating words from English to Ukrainian, the tag of the part of speech is taken into account to avoid significant translation errors.

Code for the implementation of the translation of marked, sentiment and bias data

```
import xml.etree.ElementTree as ET from deep_translator import GoogleTranslator from collections import defaultdict
xml_file = "en-sentiment.xml" # Load XML
tree = ET.parse(xml_file)
root = tree.getroot()
# Dictionary to store translations
translations = defaultdict(set)  # Using set to avoid duplicate translations
translator = GoogleTranslator(source='en', target='uk') # Initialize translator
for word in root.findall("word"): # Translate unique words based on POS
```

```
   form = word.get("form")    pos = word.get("pos")
    if (form, pos) not in translations:
       translated = translator.translate(form)
       translations[(form, pos)].add(translated)
for key in translations: # Convert sets to lists (for multiple translations of the same POS)
   translations[key] = list(translations[key])
for word in root.findall("word"): # Update XML with translated forms
   form = word.get("form")
   pos = word.get("pos")
   # Update the 'form' attribute with translated list
   word.set("form", str(translations[(form, pos)]))
tree.write("translated_output.xml", encoding="utf-8", xml_declaration=True) # Save modified XML
print("Translation completed. Check 'translated_output.xml'.")
```

Therefore, the following data was created for the Ukrainian language (Fig. 8).

Fig.8. Generated mark-up data

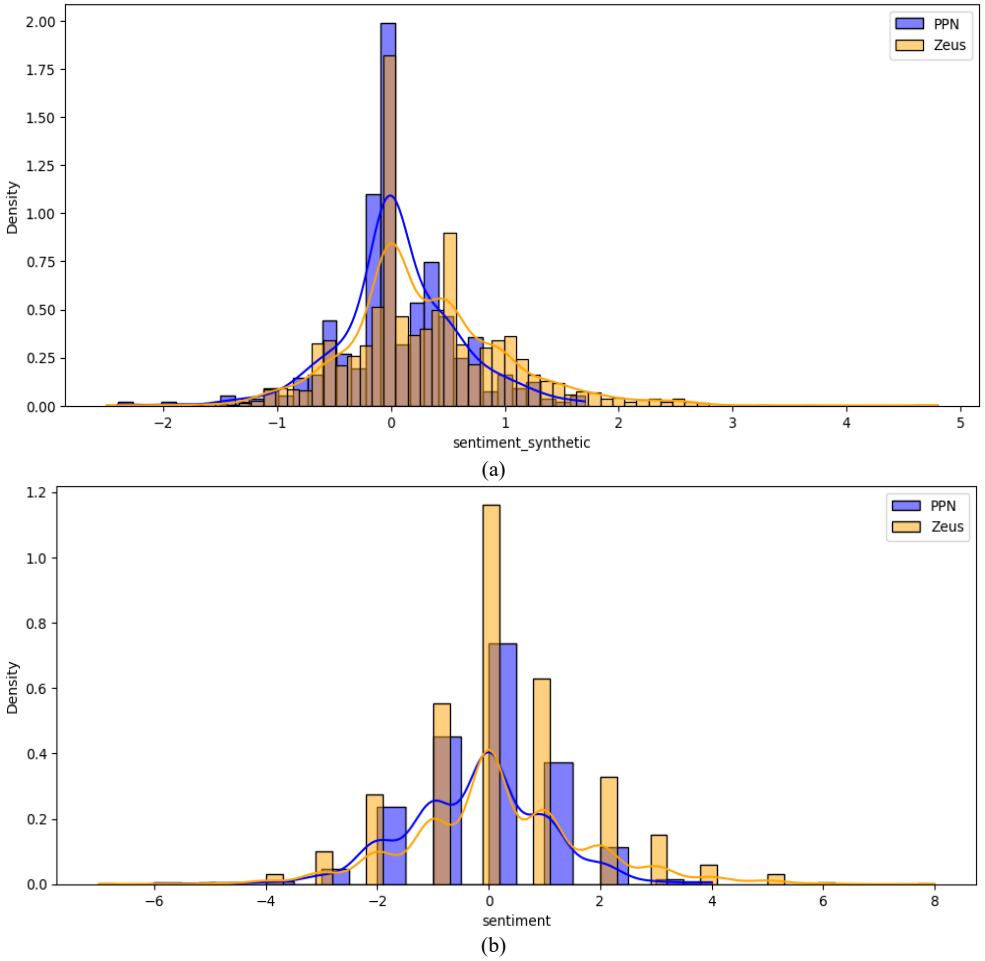


Fig.9. Distribution of the tonality of texts: a) for synthetic typing; b) for manual dialling

It is necessary to check whether the synthetically created data really corresponds to reality. For this purpose, marked data on the tonality of words explicitly designed for the Ukrainian language were selected [38]. To make sure that the created data is correct, these two sets of mark-up data on the same texts were used, and the distribution of results was compared. As you can see (Fig. 9), the distribution of synthetic data coincides with manually created ones. Therefore, it will be used in the analysis.

4.2. Analysis of the Collected Data

A. Analysis of Quantitative Data Parameters

Analysis of quantitative characteristics, such as the length of texts, the number of sentences, the average number of words, etc., will allow you to assess the structure and variability of the corpus. It will make it possible to:

- Detect texts that are too short or abnormally long (which may be noise).
- Better understand the stylistic features of texts (for example, conciseness or verbosity).

During our analysis, attention was paid to the length of the sentence (the number of characters in the sentence) because the length of the sentences was synthetically selected in advance for the non-propaganda data.

Fig. 10, it can be seen that data classified as propaganda tends to be short, which is quite typical for data aimed at manipulation [39]. Meanwhile, the non-propaganda data has a softer distribution, suggesting greater diversity.

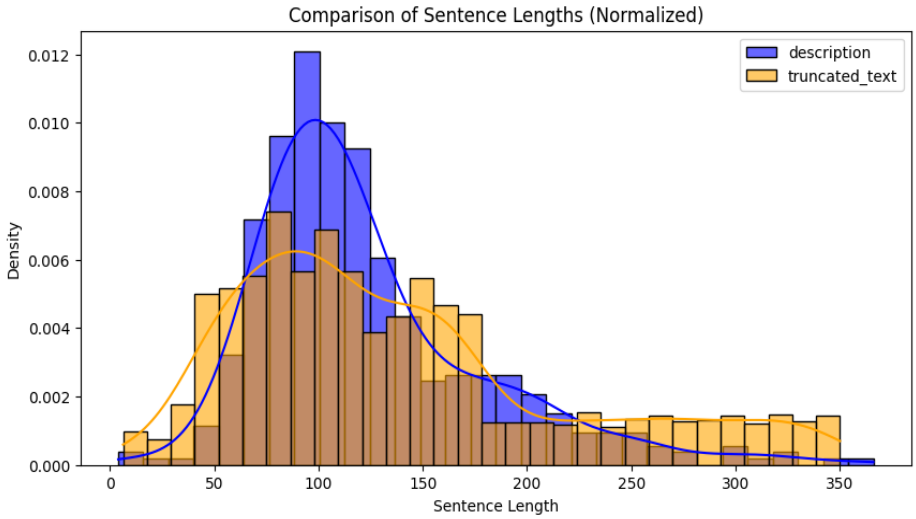


Fig.10. Comparison of sentence length

B. Analysis of Data Punctuation

Punctuation can carry an additional semantic load or signal emotionality, author's style, grammatical complexity, etc. Its analysis helps:

- Identify typical punctuation patterns (for example, frequent use of exclamation marks).
- Evaluate the quality of texts (for example, the absence of punctuation marks may indicate informality or poor quality).
- Use punctuation patterns as features for classification models.

The normalized number of punctuation marks as a percentage for two data types is displayed (Fig. 9). The result was the following conclusions:

- Dots dominate both datasets, which is typical for most texts;
- Commas are also used quite often, but in propaganda data, the percentage is higher;
- The colon is more used in data propaganda, while the dash, on the contrary, is more used in data-not-propaganda;
- Double quotes ("") are only present in data-not-propaganda

C. Analysis of the Emotional and Evaluative Components of the Data

The study of subjectivity, tonality or emotional colouring of texts allows:

- Determine the author's mood or attitude to events.

- Identify potentially biased, propagandistic, or manipulative statements.
- Apply the results as indications for classification tasks (e.g., detecting disinformation or propaganda).

It can be seen (Fig. 12) that both datasets are dominated by neutral sentiment, but non-propaganda data still tends to be positive, while propaganda data tends to be negative.

	Punctuation	Description (%)	Truncated Text (%)
17	.	47.613883	43.923677
16	,	36.876356	29.499508
23	:	8.893709	4.073597
11	-	4.989154	7.874612
10	?	0.650759	0.295298
2	)	0.325380	1.529492
9	(	0.325380	1.635496
4	%	0.216920	0.522450
7	'	0.108460	1.090331
0	*	0.000000	0.022715
24	>	0.000000	0.537594
22	{	0.000000	0.196865
21	$	0.000000	0.075717
20	}	0.000000	0.196865
19	&	0.000000	0.022715
18	!	0.000000	0.719316
13	/	0.000000	0.477020
15	]	0.000000	0.037859
14	[	0.000000	0.007572
1	;	0.000000	0.893466
12	"	0.000000	5.883244
8	_	0.000000	0.037859
6	\	0.000000	0.015143
5	#	0.000000	0.249867
3	|	0.000000	0.015143
25	+	0.000000	0.166578

Fig.11. Statistics on punctuation

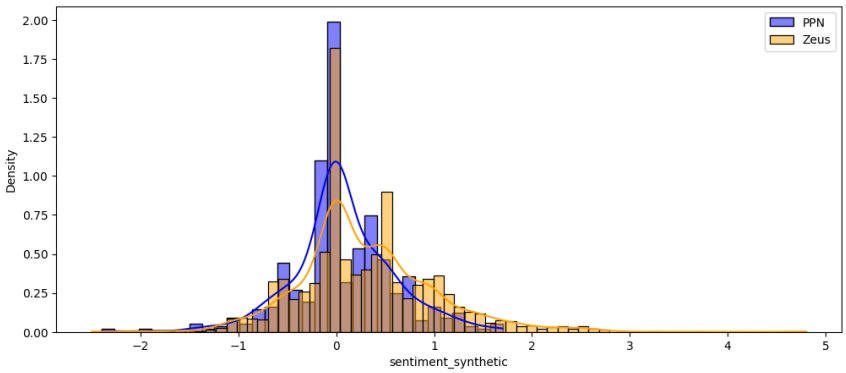


Fig.12. Tonality distribution in texts

The distribution of subjectivity (Fig. 13) also does not provide much additional information because the graphs almost coincide. Nevertheless, the difference can be seen as a greater variety of data-not-propaganda.

D. Analysis of Grammatical Categories and Named Entities

Parsing Parts of Speech (POS) and Named Entities (NERs) provides a deeper understanding of the linguistic structure of texts. It is essential for:

- Identification of characteristic grammatical constructions or stylistic markers.
- Identification of key objects such as names, organizations, geographical names, etc.
- Building a semantic profile of the text, which can be helpful in the tasks of thematic analysis, classification or knowledge construction.

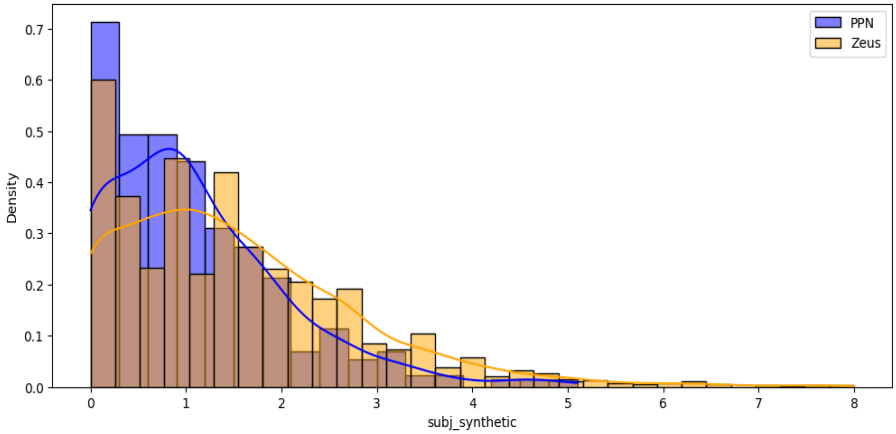


Fig.13. Distribution of subjectivity in texts

Taking into account the overall distribution of parts of speech in both datasets (Fig. 14), it can be concluded that both types of data retain approximately the same distribution.

But at the same time, some significant relationships between parts of speech are also analysed, such as the relationship between nouns and verbs, between pronouns and nouns, and between prepositions and nouns (Fig. 15) – from which it is concluded that data-not-propaganda has a more formal style (higher ratio of nouns and verbs).

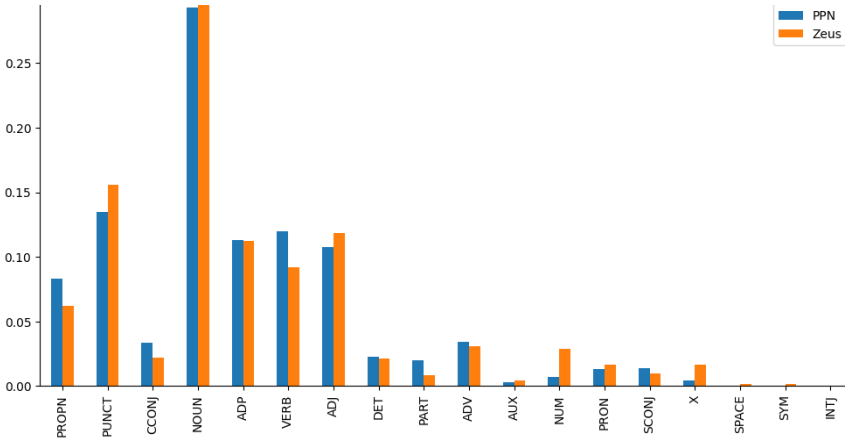


Fig.14. Normalized distribution of POS tags (parts of speech in texts) in two data sets, where x is the POS tag and y is the fraction of the total number of tokens

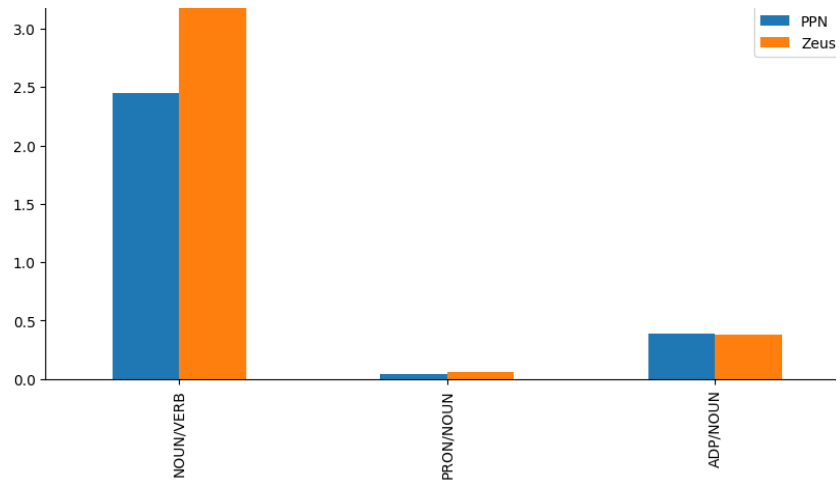


Fig.15. Correlation of some key POS categories (parts of speech) in texts

The statistics of parts of speech in bi- and trigrams were also considered (Fig. 16), but this analysis did not allow us to draw conclusions that would prove the difference between the data.

```
Most Common Bigrams in PPN:
('NOUN', 'PUNCT'): 338
('ADJ', 'NOUN'): 312
('NOUN', 'ADP'): 237
('ADP', 'NOUN'): 232
('NOUN', 'NOUN'): 209
('VERB', 'NOUN'): 137
('NOUN', 'VERB'): 131
('NOUN', 'PROPN'): 118
('PROPN', 'VERB'): 96
('ADP', 'ADJ'): 96

Most Common Bigrams in ZF:
('NOUN', 'PUNCT'): 3544
('ADJ', 'NOUN'): 3195
('ADP', 'NOUN'): 2194
('NOUN', 'NOUN'): 2081
('NOUN', 'ADP'): 1671
('NOUN', 'ADJ'): 1215
('NOUN', 'VERB'): 1107
('VERB', 'NOUN'): 1052
('ADP', 'ADJ'): 955
('PUNCT', 'NOUN'): 908
```

(a)

```
Most Common Trigrams in PPN:
('ADJ', 'NOUN', 'PUNCT'): 68
('NOUN', 'ADP', 'NOUN'): 54
('ADP', 'NOUN', 'PUNCT'): 49
('ADP', 'ADJ', 'NOUN'): 43
('NOUN', 'ADJ', 'NOUN'): 40
('NOUN', 'NOUN', 'PUNCT'): 33
('ADP', 'NOUN', 'NOUN'): 30
('ADJ', 'NOUN', 'ADP'): 29
('NOUN', 'ADP', 'PROPN'): 29
('NOUN', 'VERB', 'NOUN'): 28

Most Common Trigrams in ZF:
('ADJ', 'NOUN', 'PUNCT'): 715
('NOUN', 'ADJ', 'NOUN'): 548
('NOUN', 'ADP', 'NOUN'): 532
('ADP', 'NOUN', 'NOUN'): 396
('ADP', 'ADJ', 'NOUN'): 386
('NOUN', 'PUNCT', 'NOUN'): 369
('NOUN', 'NOUN', 'PUNCT'): 366
('ADJ', 'NOUN', 'NOUN'): 320
('NOUN', 'PUNCT', 'ADP'): 311
('ADJ', 'NOUN', 'ADP'): 294
```

(b)

Fig.16. The most popular are: a) bigrams in texts; b) trigrams in texts

The ratio of named entities in texts to the total number of words is considered. No apparent difference was found, but the propaganda data still contained a smaller percentage (Fig. 17).

```
Correlation between NER Count and Word Count in df_ppn_ua: 0.489
Correlation between NER Count and Word Count in df_zf_ua: 0.558
```

Fig.17. The ratio of the number of named entities to the total number of texts

Next, a comparison of the types of named entities in the texts was performed (Fig. 18-19), which made it possible to identify a larger number of mentions of named entities of the "Organization" type in propaganda data and a larger number of mentions of named entities of the "Location" type in non-propaganda data.

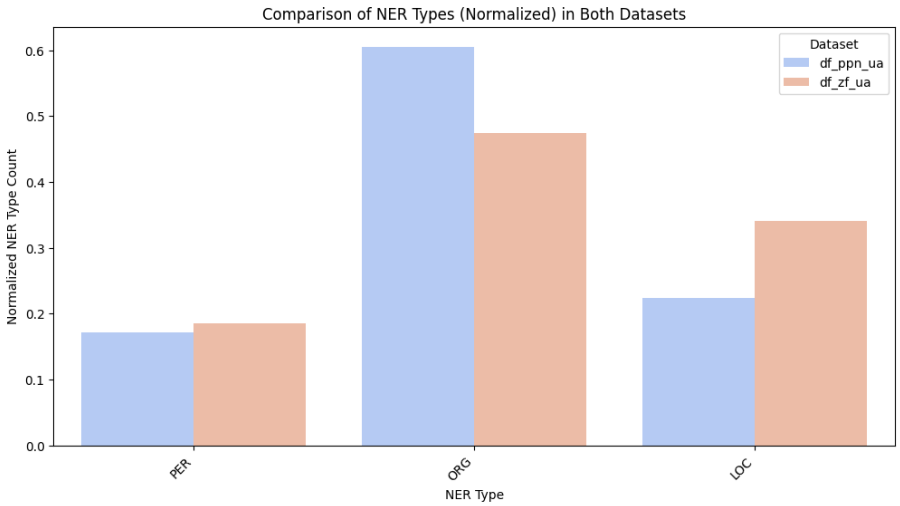


Fig.18. Comparison of types of named entities in texts

E. Additional Analysis

A theory has been put forward regarding the possible identification of the difference between data through the use of the langdetect library:

- Articles that are passed off as the Ukrainian language may contain an unnaturally high proportion of Russian words or phrases, which may indicate automatic translation or substitution of content.
- If the news is presented as Ukrainian, but most sentences are identified as Russian-speaking, this may be a marker of the reuse of materials from Russian sources or propaganda platforms.
- Propaganda texts often contain inserts in other languages, such as original Russian quotes that are presented

without translation. The automatic detection of such fragments may signal a potential manipulative source or narrative.

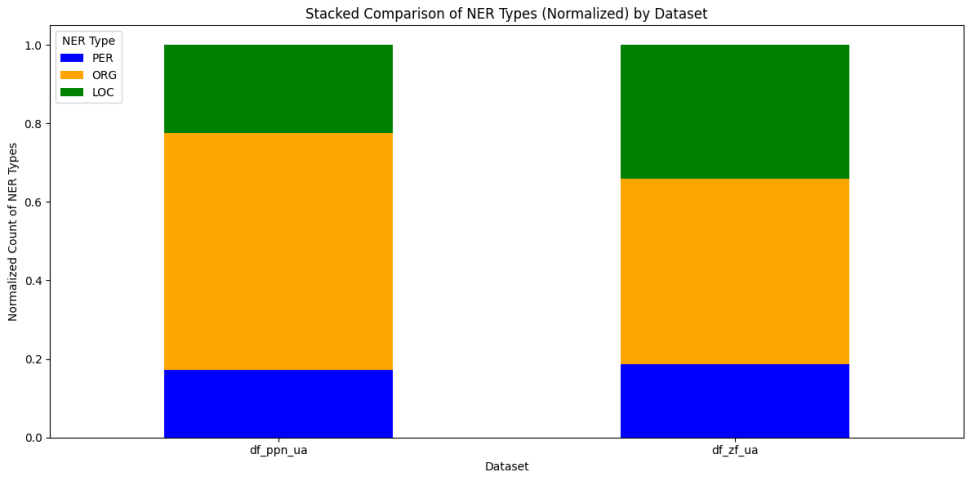


Fig.19. Comparison of types of named entities in texts (stacked diagram)

No such patterns were found (Fig. 20-21). A possible reason is the small size of the documents, which is because only the description of the article is used, and not its content.

```
{'uk': 0.9916505411999372, 'ru': 0.006069487285600341, 'ro': 0.002277891811246536}
```

Fig.20. Normalized langdetect results for propaganda texts

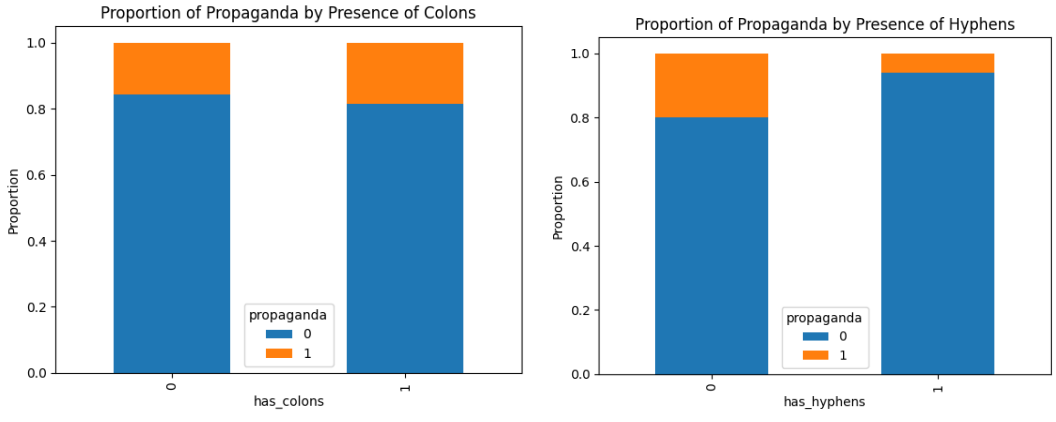
```
{'uk': 0.9937867950098164, 'ru': 0.002978820447535682, 'en': 0.002661923084065176, 'bg': 0.0005070316227099617, 'mk': 6.3379130312732e-05}
```

Fig.21. Normalized langdetect results for non-propaganda texts

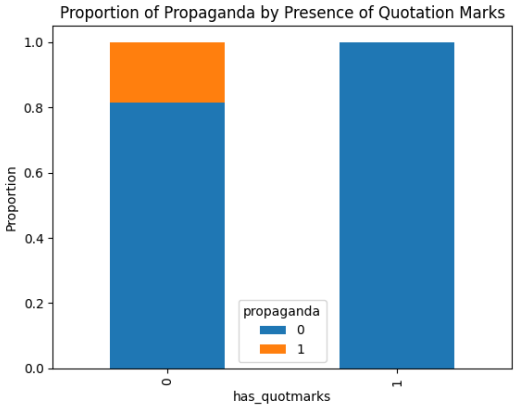
F. Conclusions from the Analysis

So, taking into account the analytics carried out above, the following steps can be determined for further preprocessing of the text:

- Add columns "has_colons", "has_hyphens", and "has_quotmarks" – propaganda data contains more colons, non-propaganda data contains more dashes;
- Add columns "sentiment" and "subjectivity" – after all, some difference was found for the distribution of tonality and subjectivity;
- Add column "noun_verb_corr" – data-non-propaganda has a more formal style, respectively, the meaning of the ratio of nouns to verbs is greater;
- Add columns "location_count" and "organisation_count" – in propaganda data, named entities of the "organization" type are more often mentioned, and in non-propaganda data, more named entities of the "location" type.



(a) (b)



(c)

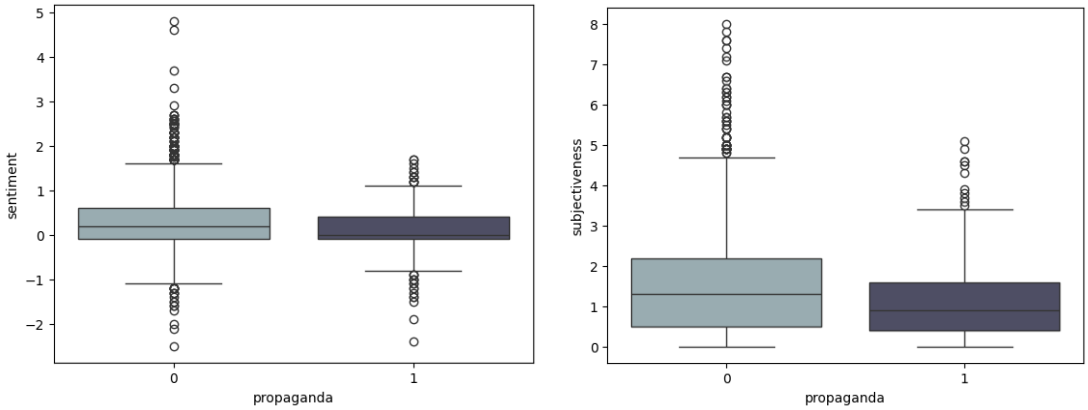
Fig.22. Distribution of data according to: a) whether a colon is present, b) whether a dash is present, c) whether double quotes are present

4.3. Data Preprocessing

A. Creating Additional Characteristics

In accordance with the steps defined in the previous section, new characteristics have been created and added to the data. Visual analytics were also carried out to further confirm the importance or unnecessary of adding characteristics:

- "has_colons", "has_hyphens" and "has_quotmarks":
- "sentiment" and "subjectivity":



(a) (b)

Fig.23. Distribution of data according to: a) sentiment score, b) subjectiveness score

- “noun_verb_ratio”:

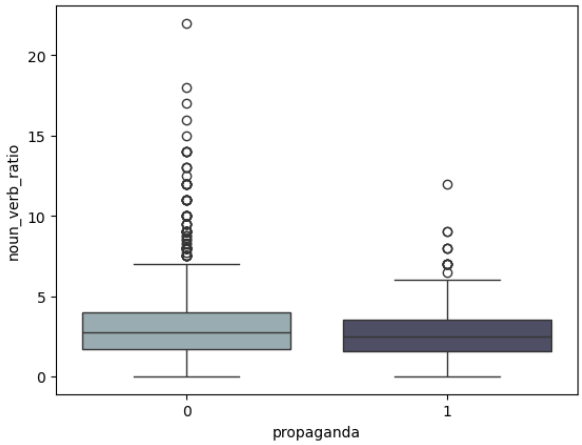
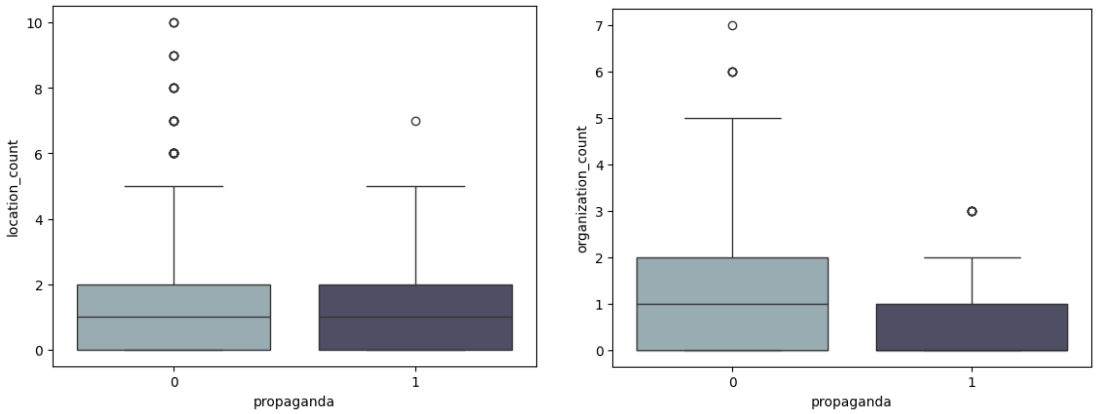


Fig.24. Distribution of data according to the ratio of nouns to verbs

- "location_count" and "organisation_count":



(a) (b)

Fig.25. Distribution of data according to a) the number of NERs of type "location" and b) the number of NERs of type "organisation"

A pre-loaded dataset is also used to vectorize the text, and a separate "vector" column is created.

B. Performing the Defined Data Preprocessing Steps

After supplementing these characteristics with characteristics, we have the following dataset:

Table 24. Augmented data

Column Name	Data Type	Description
text	object	The original text of the news article or snippet.
lemma	object	A lemmatized version of the text, reduced to basic forms of words.
propaganda	int64	Binary label: 1 — available propaganda, 0 — absent.
has_colons	int64	The presence of colons in the text (1 — is, 0 — no).
has_hyphens	int64	The presence of hyphens or dashes (1 — is, 0 — no).
has_quotmarks	int64	The presence of double quotes (1 - is, 0 - no).
sentiment	float64	Assessment of the emotional colouring of the text (from -1 to 1).
subjectiveness	float64	The level of subjectivity of the text (from 0 — objective to 1 — completely subjective).
noun_verb_ratio	float64	The relationship of nouns to verbs in the text.
location_count	int64	Number of geographical names mentioned.
organization_count	int64	Number of organizations mentioned.
vector	object	Vector representation of text (e.g., embedding as a list of numbers).

Thus, such a data preprocessing step as standardization using StandardScaler for the column's "sentiment", "subjectiveness", "noun_verb_ratio", "location_count" and "organization_count" has been defined and performed (Fig. 26-27). The object created by the scaler is saved for later use.

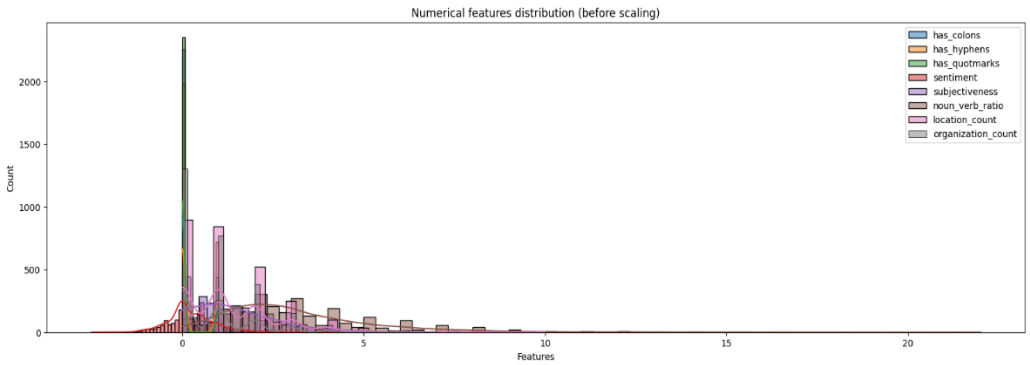


Fig.26. Distribution of numerical characteristics before standardization

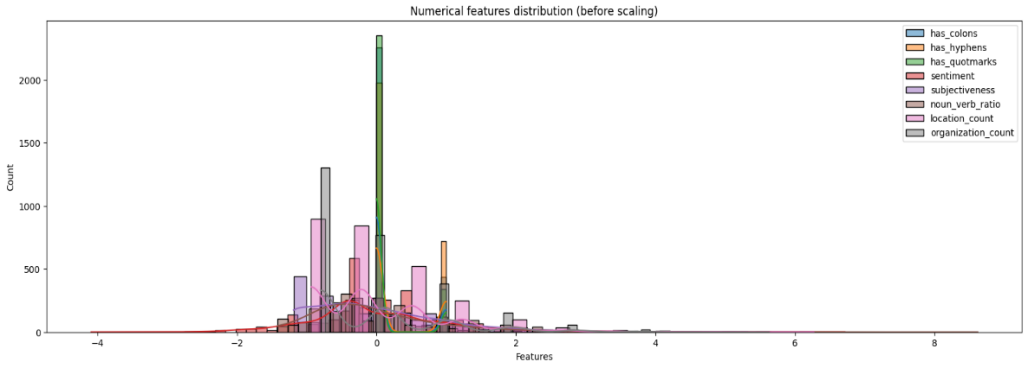


Fig.27. Distribution of numerical characteristics after standardization

4.4. Development of Classifiers

A. Development of a Classifier for the Definition of Propaganda

A neural network is chosen to create a classifier. Therefore, it is necessary to determine the required architecture that would allow you to achieve the best results. After several experimental options, the most effective one was determined, and the creation and training of the model were implemented.

Implementation code for creating a propaganda classifier

```
def construct_model(input_shape: tuple[int, ...], dim_per_layer: list[int], n_classes: int):
  input_x = tf.keras.layers.Input(shape=input_shape)
  x = tf.keras.layers.Flatten()(input_x)
   for dim in dim_per_layer:
     x = tf.keras.layers.Dense(dim, activation='relu')(x)
  out_x = tf.keras.layers.Dense(n_classes, activation='sigmoid')(x)
  return tf.keras.Model(inputs=input_x, outputs=out_x)

input_shape = (308,)
dim_per_layer = [256, 256, 512] n_classes = 1
model = construct_model(input_shape, dim_per_layer, n_classes)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()

# Train model
history = model.fit(X_train, y_train, epochs=40, batch_size=16,
   validation_data=(X_test, y_test), verbose=1 )
```

The primary metrics for determining the quality of the model were F1, recall and precision for each class. The following results were determined:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	451
1	0.93	0.81	0.87	88
accuracy			0.96	539
macro avg	0.95	0.90	0.92	539
weighted avg	0.96	0.96	0.96	539

Fig.28. Selected metrics for the test sample

The results of the classification are also considered in more detail:

The physical result of modelling and training the classifier for creating a model.keras file, which will allow you to download and use the developed model.

B. Development of a Classifier for Defining Narratives

A neural network was also used to develop a classifier for determining narratives. Likewise, several architectures have been tested with the aim of resolving the one that will provide the best and most accurate results.

When training the model, the class_weight parameter was used because the classes are very unbalanced – the dataset contains 17960 data in total, but the narrative class has only 81 lines. (Fig. 12.2.1) To calculate the necessary weights for classes, the built-in functionality of sklearn - compute_class_weight was used.

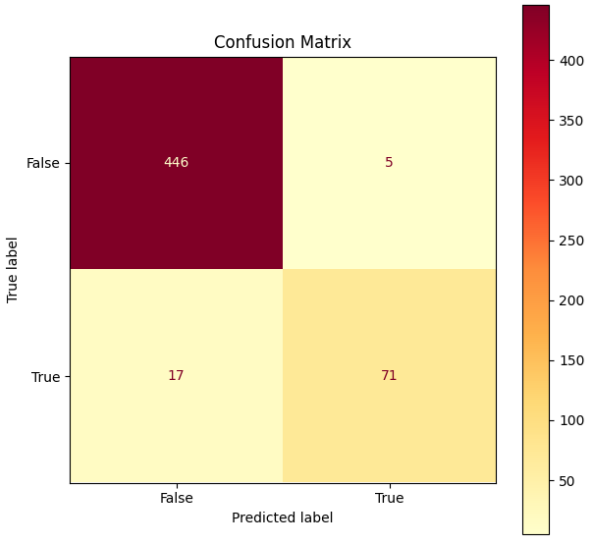


Fig.29. Confusion matrix for test sampling

Narrative Classifier Implementation Code

```
from sklearn.utils.class_weight import compute_class_weight model = tf.keras.Sequential([layers. Input(shape=(767,)),
   layers. Dense(128, activation='relu'),
   layers. Dense(64, activation='relu'),
   layers. Dense(len(y_train.unique()), activation='softmax')  ])
model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
model.summary()

class_weights = compute_class_weight(# Calculating weights for each class
   class_weight='balanced',classes=np.unique(y_train), y=y_train )
class_weights_dict = dict(enumerate(class_weights))

model.fit(X_train, y_train, epochs=10, batch_size=32,
   validation_data=(X_test, y_test), class_weight=class_weights_dict)
```

Table 25. Unbalanced classes in data

N	Name	Value
1	All rows	17960
2	Objections	12
3	Justification	55
4	Call	7
5	Inciting enmity and hatred	1
6	Humiliation of national honor and dignity	6
7	dtype	Int64

The primary metrics for determining the quality of the model were weighted F1, recall and precision for each class. The following results were determined:

precision	recall	f1-score	support
0.00	0.33	0.00	6
0.03	0.48	0.05	23
0.00	0.00	0.00	4
0.00	0.00	0.00	0
0.01	0.20	0.01	10
1.00	0.55	0.71	3639
		0.55	3682
0.17	0.26	0.13	3682
0.98	0.55	0.70	3682

Fig.30. Selected metrics for the test sample, where the lines from top to bottom are Denial, Justification, Appeal, Incitement to Hatred and Hatred, Humiliation of National Honor and Dignity, Simple Text, accuracy, macro avg, weighted avg

The results of the classification are also considered in more detail:

The physical result of modelling and training the classifier for creating a model_2.h5 file, which will later allow you to download and use the developed model.

4.5. Technical Documentation

A. Description of the Created Software

Module Description: server.py

Identification Name: server.py. Type: Web service server module. Version 1.0. Purpose: This module implements back-end logic for detecting and classifying propaganda in texts using machine learning and natural language processing.

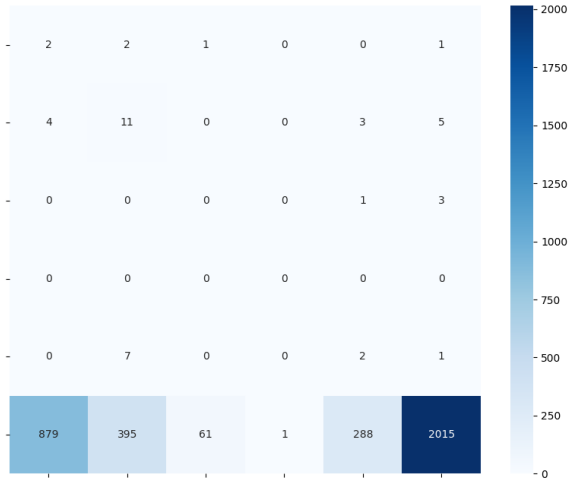


Fig.31. Confusion matrix for the test sample, where the lines from top to bottom and from left to right are denial, justification, appeal, incitement to hatred and hatred, humiliation of national honor and dignity, just text

Purpose and scope. The server.py module provides an API-based interface for performing binary and multi-label classification of Ukrainian texts according to the presence of propaganda. It processes text, performs vectorization using BERT and GloVe models, extracts features (e.g., subjectivity, sentiment, grammatical relationships), and calls pre-trained ML models for prediction. The server is designed to interact with a browser extension or other client applications, returning structured JSON responses to each request.

Module context. This module is part of an overall system for content moderation, propaganda detection, and text analysis. It acts as a central computing service between the client interface (such as a browser extension) and locally stored ML models. Input: JSON request with text for analysis. Output: JSON response with the classification result (whether the text is propaganda, and if so, what type).

Functionalities:

- Text preprocessing
  - Translates Ukrainian text into Russian (deep_translator).
  - Lemmatizes text and filters stop words (spaCy).
  - Extracts syntactic and semantic features:
    - Sentiment and subjectivity assessments.
    - Number of named entities (organizations, locations).
    - Ratio of nouns to verbs.
    - Flags with colons, quotation marks, dashes.
- Constructing a vector representation
  - Combines the middle vector of words with GloVe and numerical features.
  - For multi-tag classification, it uses BERT (average value for tokens).
- Classification
  - Binary classification: Determines if the text is propaganda (model.keras model).
  - Multi-label classification: If the text is propaganda, classify it into one of 6 types (model model_2.h5).

- REST API – POST /classify: The main access point to obtain the classification result.

Description of the user interface. The module does not have a graphical interface. It provides a REST API that can be accessed programmatically. The input/output format is JSON. Example request:

```
{ "text": "Ukraine was part of Russia and must go back." }
```

Example of an answer in case of propaganda:

```
{ "class": "Propaganda",
  "multilabel": {
    "text": "...",
    "processed_text": "...",
    "label": "Incitement to hatred and hatred",
    "probabilities": {
      "Denial": 0.12, "Justification": 0.04, ...  }}}
```

Example of a response in the absence of propaganda:

```
{   "class": "Not propaganda" }
```

Addictions:

- Flask – a web framework for implementing APIs.
- Transformers for tokenization and vectorization are available via BERT.
- TensorFlow / Keras for loading and using neural network models.
- spaCy – linguistic text preprocessing.
- deep_translator – translation of the text.
- joblib to load StandardScaler.
- NumPy, torch, bz2, xml.etree.ElementTree – helper libraries.

Description of interfaces. External interfaces– HTTP POST /classify accepts JSON with text, and returns classification result.

Internal functions:

- preprocess_text(text: str) creates features for the binary model.
- multilabel(text: str) performs multilabel classification.
- get_avg_w2v(text, tokenizer, model) returns the vector of text representation via BERT.
- calculate_subj() / calculate_sentiment() – calculation of subjectivity and sentiment.
- count_named_entities() / avg_noun_verb_ratio()– extraction of syntactic features.

Error Handling and Restrictions:

- If the input is not a string, a neutral value of (0.0) is returned.
- If no word is found in GloVe, a vector of zeros is used.
- The module expects all the necessary files (.xml, .pkl, .bz2, models) to be placed in the working directory.
- The current version does not implement authentication or request restrictions.

Glossary:

- BERT – bidirectional coding with transformers.
- GloVe – global vectors for word representation.
- Multi-label classification – each example can belong to several categories at the same time.
- Propaganda – information that is biased or manipulative and is aimed at promoting specific ideas or actions.

Description of the user interface. Interface name: browser plugin for text classification. Purpose: The plugin allows the user to select text on a web page or enter it manually, send this text to the backend for classification and get the result in a convenient format. The main elements of the interface view:

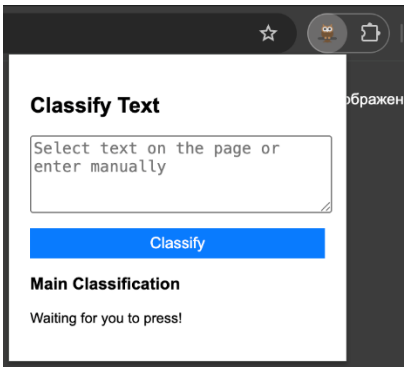


Fig.32. Appearance of the interface at startup

Table 26. Interface elements

Element	Type	Description
textarea#selected-text	Text Area	An input field in which selected text from a page is automatically inserted or filled in manually.
button#classify-btn	Button	Sends the text to the server for analysis.
p#main-result	Paragraph	Displays the main result of the classification ("Propaganda" / "Not Propaganda").
div#multi-label-section	Block	Displayed only if the result is "Propaganda"; contains a detailed multi-label classification.
p#multi-label-result	Paragraph	It shows the most likely propaganda subcategory.
ul#probabilities-list	List	Outputs the probabilities of each category from the multi-label model.

The computational load of the BERT model, especially in the context of the browser plug-in, is one of the main limitations of the system affecting the user experience. Below is a detailed analysis of this aspect.

Let's analyse the computational load of the BERT model in the browser plugin. The architectural solution is not to run in a browser. The complete bert-base-multilingual-cased model does not run directly in the browser because:

- It has ≈110 million parameters.
- It requires access to a graphics processing unit (GPU) or a powerful CPU to process it in an acceptable amount of time.

The plugin sends text to the server via HTTPS, and the inference (i.e. running the BERT model) occurs on the server side (backend inference API).

Table 27. Response time (latency) based on typical configurations for BERT (without optimisations)

Text Type	**Length (tokens)**	**CPU processing time (average server)**	**On GPU**
Short news	~100	0.5 - 1 sec	<0.2 s
Full text	300–500	2-4 sec	~1 s

In the browser plugin, this manifests itself as a delay of 1–4 seconds from the moment the page is opened to receiving the score, which already affects the UX, mainly when used repeatedly or on slow connections. Bottlenecks:

- Server performance. When many users use the plugin simultaneously, the BERT model can become a bottleneck, especially without a scalable cloud (load balancing, autoscaling).
- Network load. All text should be sent to the server → latency on slow internet.
- Device restrictions. On mobile phones or weaker laptops, transmitting and waiting for a response looks like a freeze or "no response".

Basic optimisation steps:

- Use of compressed models. For example, DistilBERT (2× faster, 40% smaller) or MiniLM, which retain most of the accuracy of BERT, but are significantly faster.
- Ongoing inference on the client. For short texts, it is possible to integrate the ONNX or TensorFlow.js version of TinyBERT or ALBERT, but only for powerful PCs.
- Caching results + prefetching. For example, using the cache API in the browser or Redis on the server, so as not to rerun the model for texts already seen.

A full-fledged BERT model cannot run directly in the browser due to the heavy computing load. Therefore, its launch has been moved to the server, which creates a delay of 1-4 seconds, depending on the length of the text, the load on the server, and the speed of the Internet. This limitation can significantly affect the convenience, speed, and perception of the plugin, especially if the user expects instant results.

B. User Manual

The product, AthenaLens, is a local tool for classifying text for the presence of propaganda, and it integrates a browser extension for Google Chrome.

Prerequisites – Before you start, make sure you have the following installed:

- Python 3.7 or higher;
- Google Chrome;
- Internet access for the first download of the repository;
- Installed Python libraries specified in the requirements.txt file (in the server directory).

Stages of preparing the plugin for work:

Step 1. Cloning a repository
Step 1. Open a terminal or command prompt.
Step 2. Run the command git clone https://github.com/swamptoday/propaganda_classify
Step 3. Once the download is complete, you will get a directory propaganda_classify that contains the following:

- server/ – the server part responsible for classification.
- browser-extension/ – browser plugin code.
- additions/ – additional materials on creating models.

Step 2. Running the backend
Step 1. Go to the server directory: cd propaganda_classify/server
Step 2. Start the server: python server.py
Step 3. The server will run locally at the address http://127.0.0.1:5000. (Note: Models may load or initialize the first time you run, so that it may take a little time.)
Step 3. Installing the plugin in Google Chrome
Step 1. Open the Google Chrome browser.
Step 2. Go to the Extensions menu:

- Click on the three dots in the upper right corner → Extensions → Manage Extensions.
- Or type in the address bar: chrome://extensions/

Step 3. Activate developer mode in the upper right corner of the page.
Step 4. Click the "Download unpacked extension" button.
Step 5. In the directory selection window, find the path to the newly cloned repository and select the folder propaganda_classify/browser-extension
Step 6. After adding the extension, click "Details" on its card.
Step 2. Enable the "Pin to toolbar" option to have quick access to the plugin.
Step 4. Using the plugin. You can now categorize text on pages.
Step 1. Classification of selected text

- Open any web page.
- Select the text you want.
- Click on the plugin icon in the Chrome toolbar.
- Automatically inserted selected text will appear in the text box.
- Click the "Classify" button.
- Get the result of the classification with the ability to see subcategories and probabilities.

Step 2. Enter your text

- Click on the plugin icon.
- Type or paste text manually into the text box.
- Click "Classify".
- The results will appear below in the form of:
  - Main classification (Propaganda / Not Propaganda)

- Detailed subclassification (if propaganda is found)
- Probabilities for each category

Additional notes:

- The plugin only works when the local server is actively running.
- After restarting your computer or ending a session, you need to start the server.py again.
- In case of connection problems, make sure other processes do not occupy port 5000.

Plugin deployment status:

- As of the date of publication (July-August 2025), the plugin is not publicly available; it is considered part of a project that goes through the following stages:
  - Architecture developments.
  - Construction of classification models.
  - Integrations into the plugin interface.
  - Conducting experiments and testing.
- Public open source is not currently presented (e.g. GitHub) due to trade secrets.
- According to the described architecture and analysis of system alternatives, the browser plugin is defined as the optimal form of implementation (it won compared to the mobile application and website), but this is a justification for the choice, not an indication of the completed implementation.
- Support for local and server-side processing**,** REST API integration, and further extensibility are envisaged, but these elements are currently being considered as part of design and prototyping.

Browser plugin currently:

- It is not publicly deployed.
- It is at the prototype development stage.
- The code is not open source (at least at the time of writing).
- Public testing or deployment has not yet been conducted**.**

4.6. Execution of a Control Example

A. Choosing a Sentence for Classification

To perform the control example, it is necessary to select test classification materials. Many news resources can allow you to show the results of creating a browser plugin for classifying propaganda. Of the most famous, we can note:

- Ukrinform (https://www.ukrinform.ua/) is the official national news agency.
- UNIAN (https://www.unian.ua/) – news, politics, economics, events in Ukraine and the world.
- TSN (https://tsn.ua/) – news portal of the 1+1 TV channel.
- Channel 24 (https://24tv.ua/) – current news, analytics, video.
- Espresso (https://espreso.tv/) – news and television, politics and society.
- Glavkom (https://glavcom.ua/) – political news, analytics.
- Ukrainska Pravda (https://www.pravda.com.ua/) – journalistic investigations, blogs, news.
- Newsroom (https://novynarnia.com/) – news from the front, army, society.

For the control example, the TSN news resource was randomly selected.

In the future, it is necessary to choose at least two sentences (text excerpts): one of them should be considered propaganda by the definition of the task, the other - not propaganda. After monitoring several articles, an article [40] was selected, which, among other materials, also translated and quoted the statements/words of Russian propagandists.

So, the selected passages for classification will be:

Table 28. Test chosen materials for the control case

Propaganda	Not propaganda
Perhaps Trump and the State Department are not left with the idea of involving Russia in deterring China, and therefore, we were offered...	Russian propagandist Solovyov again voiced aggressive statements directed against Europe and the United States. The focus is on Berlin and Texas.

For "equality" of classification, both passages contain both key names and locations.

B. Classification using a Browser Plugin

Since the browser plugin allows two ways to enter the information necessary for classification – highlighting on the site and directly entering text into the field – for the first example, selection will be used, respectively, and for the second – copying and pasting into the text field.

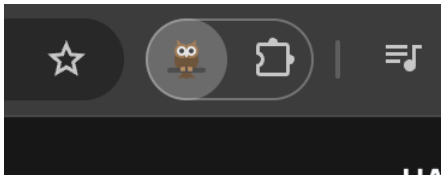


Fig.33. Selecting the plugin icon to open its functionality

When the plugin interface is open, you can see that the selected text is already in the text field (Fig. 34), and you need to click the classification button.

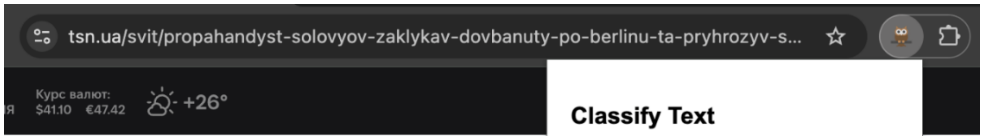


Fig.34. Opening the plugin interface

C. Classification using API Requests

To check the correctness of the query system of the created server, control examples of requests to the server with selected sentences were also carried out. To perform such a check, queries were made to http://127.0.0.1:5000, specifically to the /classify route, which was defined during the development of the API server.

The request has the following form:

```
curl -X POST http://127.0.0.1:5000/classify \ -H "Content-Type: application/json" \ -d '{"text": "Text
is for classification."}'
```

The answer should take the following forms:

```
{   "class": "Not propaganda"
}
```

or

```
{ "class": "Propaganda",
  "multilabel": { "text": "The text is intended for classification.",
    "processed_text": "...",
    "label": "Call",
    "probabilities": {
      "Denial": 0.123,
      "Justification": 0.042,
      "Call": 0.837,
      "Incitement to Enmity and Hatred": 0.008, "Humiliation of National Honor and Dignity": 0.004, "No Narrative": 0.003 } } }
```

The correct operation of the API server has been confirmed.

Backend implementation code

```
from transformers import AutoTokenizer, AutoModel
from tensorflow.keras.models import load_model
from deep_translator import GoogleTranslator
from flask import Flask, request, jsonify
import xml.etree.ElementTree as ET
import tensorflow as tf
import numpy as np
import joblib
import torch
import spacy
import bz2
app = Flask(__name__)
scaler = joblib.load('scaler.pkl') # Load StandardScaler
model = tf.keras.models.load_model('model.keras') # Load the model
model_2 = load_model("model_2.h5")
nlp = spacy.load('uk_core_news_lg') # Load Spacy-model

# Download the GloVe dictionary
```

```
glove_path = "./news.lowercased.lemmatized.glove.300d.bz2"
word_vectors = {}

# Load the pre-trained tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("DeepPavlov/rubert-base-cased")
bert_model = AutoModel.from_pretrained("DeepPavlov/rubert-base-cased")

# Define unique labels
unique_labels = ['Denial', 'Justification', 'Appeal', 'Incitement to enmity and hatred', 'Humiliation of national honour and dignity', 'No Narrative']

label_to_id = {label: idx for idx, label in enumerate(unique_labels)}
id_to_label = {idx: label for label, idx in label_to_id.items()}
with bz2.open(glove_path, "rt", encoding="utf-8") as f:
    for line in f:
        parts = line.strip().split()
        word = parts[0]
        vector = np.array(parts[1:], dtype=np.float32)
        word_vectors[word] = vector

def preprocess_text_2(text):
    nlp = spacy.load("ru_core_news_sm")
    translated_text = GoogleTranslator(source='uk', target='ru').translate(text)
    doc = nlp(str(translated_text).lower())
    tokens = [token.lemma_ for token in doc if not token.is_stop and token.is_alpha]
    return " ".join(tokens)

def get_avg_w2v(text, tokenizer, bert_model):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)
    with torch.no_grad():
        outputs = bert_model(**inputs)
    vector = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
    return vector

 def count_named_entities(text):
    doc = nlp(text) # Count locations
    location_count = sum(1 for ent in doc.ents if ent.label_ == "LOC")
    organization_count = sum(1 for ent in doc.ents if ent.label_ == "ORG")
    return location_count, organization_count # Count organizations

 def avg_noun_verb_ratio(text):
    doc = nlp(text)    ratios = []
    for sent in doc.sents:  # Process each sentence separately
        nouns = sum(1 for token in sent if token.pos_ == "NOUN")
        verbs = sum(1 for token in sent if token.pos_ == "VERB")
        if verbs > 0:
            ratios.append(nouns / verbs)  # Compute noun/verb ratio
        else:
            ratios.append(0)  # Avoid division by zero
    return sum(ratios) / len(ratios) if ratios else 0  # Compute the average
def calculate_subj(text):
    subj_dict_synt = {}
    tree = ET.parse("translated_output.xml")
    root = tree.getroot()
    for word in root.findall("word"): # Extract words and polarity
        word_form = word.get("form") # Default polarity = 0 if not present
        polarity = float(word.get("subjectivity", 0))
        subj_dict_synt[word_form] = polarity
    if not isinstance(text, str):
        return 0.0  # Return neutral score for missing values
    words = text.split()  # Tokenize text
    score = sum(subj_dict_synt.get(word, 0) for word in words)  # Sum word polarities
    return score
def calculate_sentiment(text):
    sentiment_dict_synt = {}
    tree = ET.parse("translated_output.xml")
    root = tree.getroot()
    # Extract words and polarity
    for word in root.findall("word"):
        word_form = word.get("form") # Default polarity = 0 if not present
        polarity = float(word.get("polarity", 0))
        sentiment_dict_synt[word_form] = polarity
    if not isinstance(text, str):
        return 0.0  # Return neutral score for missing values
    words = text.split()  # Tokenize text
    # Sum word polarities
    score = sum(sentiment_dict_synt.get(word, 0) for word in words)
    return score
```

```
def preprocess_text(text):
  doc = nlp(text)
  lemmatized_text = ' '.join([token.lemma_ for token in doc])
  has_colons = 1 if ':' in text else 0
  has_hyphens = 1 if '-' in text else 0
  has_quotmarks = 1 if "" in text else 0
  sentiment = calculate_sentiment(lemmatized_text)
  subjectiveness = calculate_subj(lemmatized_text)
  noun_verb_ratio = avg_noun_verb_ratio(lemmatized_text)
  location_count, organization_count = count_named_entities(text)
  vectors = [word_vectors[token.lemma_.lower()] for token in doc if token.lemma_.lower() in word_vectors]
  vector = np.mean(vectors, axis=0) if vectors else np.zeros(300)
  numeric_features = np.array([sentiment, subjectiveness, noun_verb_ratio, location_count, organization_count]).reshape(1, -1)
  scaled_features = scaler.transform(numeric_features).flatten()
  return np.concatenate([vector, [has_colons, has_hyphens, has_quotmarks], scaled_features])

 def multilabel(text):    # Preprocess the input sentence
  processed_text = preprocess_text_2(text)
  vector = get_avg_w2v(processed_text, tokenizer, bert_model)
  vector = vector[:767]
  probabilities = model_2.predict(np.array([vector])) # Probability prediction
  probabilities_dict = { # Normalized probabilities (if necessary)
     id_to_label[i]: float(probabilities[0][i]) for i in range(len(probabilities[0]))  }
  predicted_label = id_to_label[np.argmax(probabilities)] # Most Likely Class
   return {
     "text": text,
     "processed_text": processed_text,
     "label": predicted_label,
     "probabilities": probabilities_dict,    }
@app.route('/classify', methods=['POST'])
def classify():
  data = request.get_json()
  text = data['text']
  # First-level classification
  features = preprocess_text(text)
  prediction = model.predict(np.expand_dims(features, axis=0))[0]
  label = "Propaganda" if prediction > 0.5 else "Not propaganda"
  response = {"class": label}

  # If classified as "Propaganda", apply second-level classification
  if prediction > 0.5:
     multilabel_result = multilabel(text)
     response["multilabel"] = multilabel_result  # Attach multilabel results
  return jsonify(response)
if __name__ == '__main__':
  app.run(debug=True)
```

Advantages of the proposed method:

- Support for the Ukrainian language. Unlike many existing solutions focused mainly on English-language content (for example, NewsGuard or CheckBuster), the developed tool is specially adapted to work with Ukrainian texts, which makes it unique and especially relevant in the context of the Ukrainian information space.
- Use of modern NLP models. The method is based on modern transformer architectures (BERT, RoBERTa), which allows achieving high classification accuracy (F1-score 0.96 for the propaganda detection task), significantly superior to traditional models such as XGBoost or Random Forest.
- Integration in the form of a browser plugin. Thanks to the plugin format, users can get instant text analysis without having to copy it to third-party services, which significantly increases the convenience and speed of use in real time.
- The ability to identify not only the fact of propaganda, but also key narratives. It adds analytical depth to the tool because the user receives not only a binary response but also advanced information about the type and content of propaganda.
- Taking into account lexical, semantic and stylistic features. The system uses complex text processing (for example, analysis of punctuation, tonality, ratio of parts of speech, the presence of names of organiSations or locations), which increases accuracy and resistance to manipulative techniques.
- Possibility of further self-study. A mechanism for collecting feedback from users is provided, which allows the model to gradually improve and adapt to new types of manipulations.

Disadvantages and limitations of the method:

- Dependence on training data. The effectiveness of the tool is limited by the quality and volume of the corpus of texts for training, especially given the lack of large open Ukrainian-language datasets labelled with propaganda.
- Narrow specialisation. The system is explicitly focused on news-format texts and does not analyse other types of content (videos, images, memes, etc.), where propaganda is also widely used.
- Limited multilingualism. Although theoretical adaptation for other languages is possible, at this stage the model is optimised only for Ukrainian, which narrows the potential audience.
- High computing requirements. The use of transformer models requires significant resources, especially when processing large volumes of text, which can affect speed and require server capacity.
- The results are not always explainable. Although the system gives the user a final score, the complexity of deep learning models sometimes makes it difficult to explain in detail why exactly the text is classified as propaganda.

Comparison with analogues (examples):

- Unlike NewsGuard, which assesses the reliability of the entire site, the proposed tool analyses a specific text fragment, which allows you to detect propaganda even on an authoritative resource.
- Compared to CheckBuster, which focuses on finding facts to check, this tool focuses on a broader range of manipulations and stylistic techniques.
- Compared to Hoaxy, which analyses the spread of fakes on Twitter, this approach focuses on the content of the text, not on social connections or virality.

For a deeper understanding of the scientific novelty and practical value of the developed tool, an analysis of its advantages and limitations in comparison with existing solutions is carried out. The main competitors for comparison are NewsGuard, CheckBuster, and Hoaxy, which are also focused on combating disinformation and improving media literacy.

Table 29. Analysis of the comparison of characteristics

Characteristic	Proposed method	NewsGuard	CheckBuster	Hoaxy
Ukrainian language support	Yes (specially trained model)	Partially (evaluation of sites regardless of language)	Missing	Missing
Level of detail	Analyses individual texts and identifies narratives	Evaluates the reliability of sites as a whole	Highlights individual statements for fact-checking	Analyses the spread of fakes on social networks
Instant analysis in the browser	Yes (browser plugin)	Is	Partially (web interface, no plugin)	N/a
Modern NLP models (BERT, etc.)	Is	N/a	Limited	Limited
Identification of key narratives	Is	N/a	N/a	Partially (defines general themes)
Ease of Use	High (automatically when reading)	High	Medium (manual input required)	Low
Multimedia limitations	Only analyses texts	Analysis of sites, not specific content	Parses only text	Only analyses tweets
Data quality dependency	Is	Less critical	Is	Is
Resource intensity (calculation)	High (via transformers)	Low	Average	Average

The developed method has a number of significant advantages: first of all, it provides exceptional support for the Ukrainian language, the use of modern models of transformers, as well as the ability not only to detect the fact of propaganda, but also to determine specific narratives in the text. In addition, the format of implementation in the form of a browser plugin makes the tool as convenient as possible for daily use and quick verification of information.

At the same time, the method has certain drawbacks. In particular, high computational complexity requires optimisation or powerful server resources, and the accuracy of classification largely depends on the quality and volume of training data. Also, the current version is limited to textual content only, while modern disinformation is spread through other media formats. Thus, the proposed solution stands out favourably among existing analogues due to its depth of analysis and language specificity. Still, it needs further development to overcome these limitations and expand its functionality.

5. Discussion

5.1. Analysis of Data Classification Using Models

During the development of the models, the performance of the test sample was also analyzed to ensure an unrestrained system.

The key metrics for the analysis were standard precision and recall for individual classes and weighted average F1-score because the distribution of classes was unbalanced.

In Fig. 34, you can see the resulting report of the model used to classify propaganda. The propaganda class has a lower recall compared to the non-propaganda class, which is quite typical for binary classification problems. In the future, this may be improved through additional training on the model. The weighted average F1-score has a value of 0.96, which is a good primary indicator and allows us to conclude that the selected data will enable you to find a significant difference between them. It also exceeds the expected results set in the requirements (>0.8). Fig. 35 depicts a matrix of mismatches that confirms that the classification works successfully for both classes.

	precision	recall	f1-score	support
0	0.97	0.99	0.98	451
1	0.92	0.83	0.87	88
accuracy			0.96	539
macro avg	0.95	0.91	0.93	539
weighted avg	0.96	0.96	0.96	539

Fig.35. Report of the classification of the test sample of the created propaganda classification model

Fig. 29, you can see the resulting report of the model used to classify narratives. Due to the small amount of data for classification, the model has worse results compared to the previous one. It is also confirmed by the matrix of discrepancies (Fig. 30). In the future, this can be corrected by expanding the data.

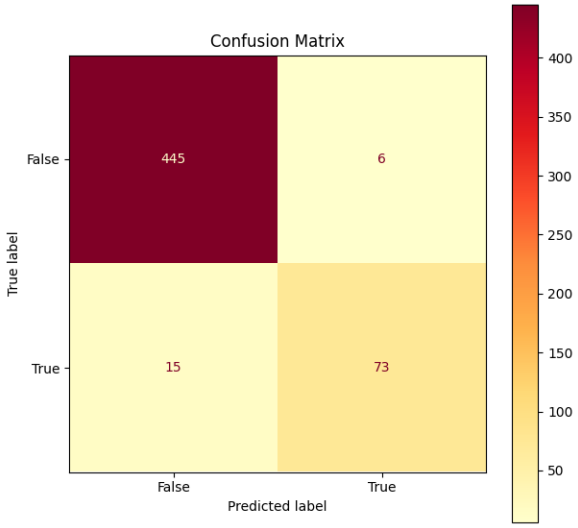


Fig.36. Confusion matrix of the test sample of the created propaganda classification model

5.2. Data Classification Analysis Using API Server

Additional data has been obtained that can be used to verify the classification.

Table 30. Additional data for test verification

Name	Description
text	Text/statement in the context of the Ukrainian-Russian war
class	Truthfulness of the statement (T - true; F - false)
language	Language in which the statement is written (currently only UKR)

Preliminary analysis occurred only through the use of a conserved model. It is also necessary to test the operation of the server, including the time it takes to complete requests and the response to the average load.

So, the 'text' column from the data was used (Table 36) to create a function that would make requests to the server and write them to a separate file along with the time of their execution. Server Request Implementation Code

```
import pandas as pd
import requests
import time
input_csv_path = '.. /data/df_ukr_141024 16.49.28.csv'
```

```
df = pd.read_csv(input_csv_path)
url = 'http://127.0.0.1:5000/classify'
results = []
total_start_time = time.time()
for _, row in df.iterrows():
  text = row['text']
  label = row['class']
  start_time = time.time()
  try:
    response = requests.post(url, json={'text': text})
    response.raise_for_status()
    response_json = response.json()
  except Exception as e:
    response_json = {"error": str(e)}
  elapsed_time = time.time() - start_time
  results.append({'text': text, 'class': label, 'response': response_json,
    'request_time': elapsed_time  })
total_elapsed_time = time.time() - total_start_time
df_results = pd.DataFrame(results)
df_results['predicted class'] = df_results['response'].apply(lambda r: 'T' if r.get('class') == 'Not propaganda' else 'F')
df_results.to_csv('classified_results.csv', index=False)
print("Done! Total time:", total_elapsed_time, "seconds")
print(df_results.head())
```

As a result, a dataframe was obtained, which made it possible to analyse both the accuracy of the model and the overall operation of the server. When accepting requests, the server worked stably, without interruptions. The total processing time of 446 requests was 156 seconds (approximately 2.5 minutes), which gives an average of 0.3 seconds to complete the request:

```
155.9642379283905
```

Fig.37. Total request execution time

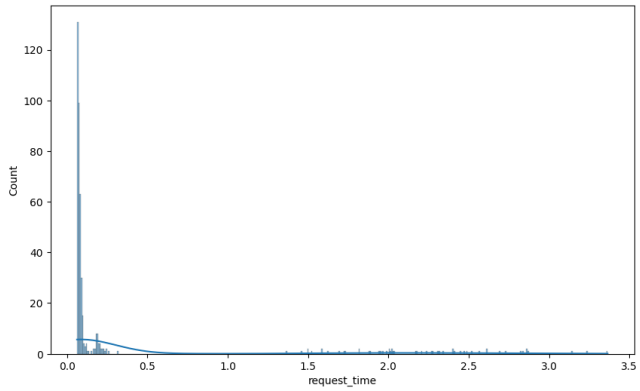


Fig.38. Query execution time histogram

The actual execution time for histogram queries is also considered in detail (Fig. 37). It can be seen that, despite its peak of 0.1-0.2 seconds, it is pretty wide. A theory has been put forward regarding the influence of a particular class on the speed of response return. To verify and confirm this statement, two separate histograms have been created for classes defined as propaganda and non-propaganda.

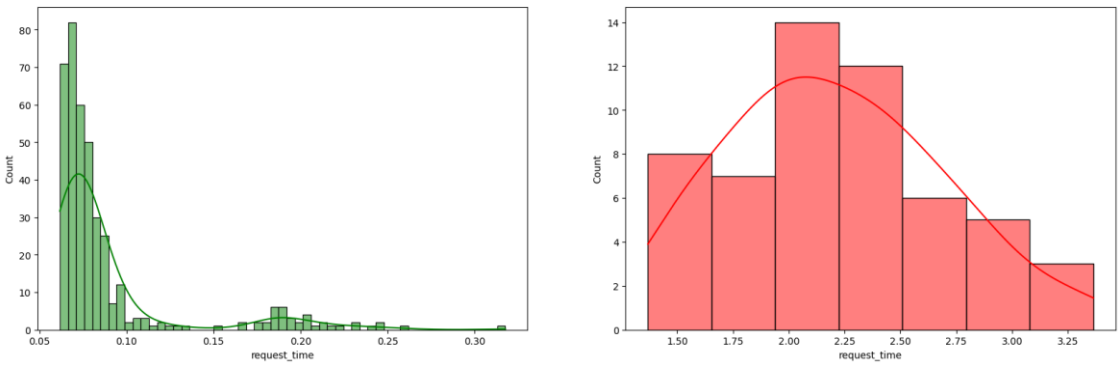


Fig.39. Separate histograms of query execution time, where green is not propaganda, and red is propaganda

So, the theory is confirmed - it is a particular class that affects the speed of processing the request by the server. A possible reason is the use of a second model, which gives the result a narrative class.

The accuracy of the classification of propaganda was also checked. The metrics that were brought up for consideration were accuracy, recall, precision and weighted average F1-score.

The accuracy results were relatively high (Fig. 39). Still, at the same time, the recall and precision for the propaganda class (Fig. 40) were much lower compared to the previous test sample, which may indicate overtraining of the model.

```
0.7600896860986547
```

Fig.40. Model accuracy for new test data

	precision	recall	f1-score	support
F	0.24	0.17	0.20	78
T	0.83	0.89	0.86	368
accuracy			0.76	446
macro avg	0.54	0.53	0.53	446
weighted avg	0.73	0.76	0.74	446

Fig.41. Classification report for new test data

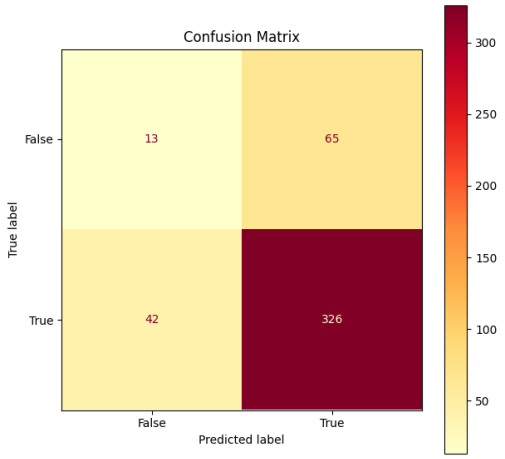


Fig.42. Non-compliance matrix for new test data

For a more complete and reliable assessment of the effectiveness of propaganda classification models and narratives, a set of key metrics was used:

- Accuracy – the total proportion of correctly classified samples;
- Precision – the share of correct optimistic predictions among all positive predictions;
- Recall – the proportion of correctly identified positive examples among all real positive examples;
- F1-score is a harmonic average between precision and recall, especially useful in conditions of class imbalance.

Table 31. Classification of propaganda (binary)

Metric	Meaning
Accuracy	0.94
Precision	0.95
Recall	0.96
F1-score	0.96

Table 32. Classification of narratives (multi-marker) (Weighted averages for all classes)

Metric	Meaning
Accuracy	0.86
Precision	0.69
Recall	0.71
F1-score	0.70

Additionally, an analysis was carried out for each class in the problem of classification of narratives, which made it possible to identify:

- Higher accuracy in categories with more examples (e.g. "Appeal" and "Accusation"),
- Low obesity in rare classes (for example, "Humiliation of national dignity"),
- There is a need for further balancing of the dataset or the use of approaches such as oversampling/undersampling or Focal Loss.

Thus, reporting only by F1-score in the case of multiclass/multi-marker classification is insufficient, because other metrics reveal additional aspects of model performance, including weaknesses in working with rare narratives.

Comparison of indicators of the model shows a noticeable gap between the two tasks:

- F1-score for propaganda detection: 0.96;
- F1-score for narrative classification: 0.70.

This gap is due to several factors that make the classification of narratives much more difficult.

Reasons for the complexity of narrative classification:

- Multi-marker. A single message can convey several propaganda narratives at the same time. It creates overlapping labels, making it difficult for the model to learn.
- Quantitative imbalance. Some narratives (for example, "Humiliation of national dignity") occur much more frequently than others, which leads to an imbalance of classes. Rare classes have a low recall, i.e., the model is not able to detect them confidently on test data.
- Some categories (for example, "Invocation" and "Accusation") have linguistically similar patterns, which cause errors between adjacent objects.
- High subjectivity of annotation. The mark-up of narratives depends on the interpretation of the annotator, which leads to ambiguity even in the training corpus. Despite the validation of agreement (Cohen's Kappa > 0.80), borderline cases still cause difficulties even in humans.

Additional metrics for a more complete analysis. To complement the F1 score, the accuracy (precision), completeness (recall) and area under the ROC curve (AUC-ROC) for each label are analysed:

Table 33. Additional metrics for a more complete analysis

Narrative	Precision	Recall	F1-score	AUC-ROC
Prosecution	0.72	0.75	0.73	0.87
Excuse	0.66	0.68	0.67	0.84
Call to action	0.69	0.71	0.70	0.85
Incitement to hatred	0.63	0.59	0.61	0.80
Humiliation of the national Dignity	0.57	0.49	0.52	0.76

The worst results are observed for the least represented or most subjective classes.

Areas for improvement in the future:

- Balancing the dataset. Extension of less-represented classes through *data augmentation*, *oversampling* or *generation of synthetic examples* (for example, using LLMs).
- Model improvements
  - An attempt at multi-output multi-head attention for better differentiation between narratives.
  - Integration of emotional and pragmatic signs as complementary channels.
- Interpretation of results. Implementation of interpretation mechanisms (e.g., attention visualisation) to analyse which words contribute to false predictions.
- Abstract of the second year. Introduction of granular subclasses or confidence scales to reduce the inaccuracy of labels.

Thus, a lower F1-score in the narrative classification problem is an expected result of a higher complexity of the task. However, the available indicators still indicate adequate efficiency and open up opportunities for further improvement.

To reasonably confirm the advantages of the proposed model, a comparison was made not only with XGBoost, but also with a number of other traditional and modern methods of natural language processing, in particular:

Table 34. Results for the propaganda classification problem

Model	Accuracy	Precision	Recall	F1-score
TF-IDF + SVM	0.87	0.89	0.84	0.86
FastText + Random Forest	0.90	0.91	0.89	0.90
XGBoost + GloVe	0.92	0.93	0.91	0.92
DistilBERT	0.93	0.94	0.93	0.93
RoBERTa	0.94	0.95	0.94	0.94
XLM-R	0.93	0.94	0.92	0.93
Suggested model (BERT + manually generated features)	0.94	0.95	0.96	0.96

Traditional Baselines:

- SVM (Support Vector Machine) with TF-IDF features;
- Random Forest with FastText vectors;

Modern transformer architectures:

- BERT (baseline) – standard multilingual model;
- RoBERTa is a high-precision variation of BERT configured for a large amount of data.
- XLM-RoBERTa (XLM-R) is a multilingual model optimised for more than 100 languages.
- DistilBERT is a compressed version of BERT that retains much of the accuracy with better performance.

Table 35. Results for the multi-marker narrative classification problem

Model	Accuracy	Precision	Recall	F1-score
TF-IDF + SVM (One-vs-Rest)	0.70	0.53	0.44	0.47
FastText + XGBoost	0.74	0.52	0.49	0.50
DistilBERT	0.79	0.63	0.67	0.65
RoBERTa	0.83	0.68	0.69	0.69
XLM-R	0.82	0.67	0.68	0.67
Suggested model (BERT + additional features)	0.86	0.69	0.71	0.70

The proposed model shows the highest F1 scores for both problems, especially in the narrative classification problem, where the improvement of ≈0.03–0.05 in F1-score is significant. Success is achieved not only through the transformer framework (BERT), but also through the integration of engineering features such as punctuation structure, subjectivity, ratio of parts of speech, recognised entities, etc. RoBERTa and XLM-R showed strong results, but showed sensitivity to the Ukrainian-language imbalance in the data. Traditional methods (TF-IDF + SVM, FastText + RF/XGB) have demonstrated competitiveness for small enclosures, but are significantly inferior in flexibility in contextual understanding.

Preliminary confirmation of practical application

Although the system has not yet been implemented at a broad level in the journal of truth or the public sector, its practical potential has been tentatively confirmed due to limited pilot testing and institutional interest. In particular:

Testing at the OSINT hackathon (2023). The tool was used as part of a team analysis competition organised in cooperation with specialists in the field of information security.

- The goal is to promptly detect propaganda messages in the information flow of Telegram channels.
- The result was that the team that used this tool won the prize, noting its usefulness in reducing the time of manual content verification and identification of hidden narratives.

Feedback from target users(anonymised). After the presentation of the system at the seminar in computational linguistics (organised at the Lviv Polytechnic National University):

- 8 out of 10 participating journalists noted that the tool can be used as an additional filter before manual fact-checking.
- Representatives of public organisations working with medical literacy emphasised the potential of using plug-ins in educational training, as it demonstrates real examples of technological influence.

Prototype in the browser and technical integration. A browser extension (Web Plugin) has been developed, which allows:

- automatically analyse texts on news sites,

- highlight the identified elements of propaganda/narratives,
- output a short explanation in real time.

It demonstrates the technical viability of deploying the tool to the end user (without the need for a dedicated environment).

Limitations and Future Plans

Despite these encouraging signals, the system has not yet passed formalised UX testing, and therefore:

- It requires the involvement of a larger sample of target users (journalists, fact-checkers, teachers) to collect feedback.
- It is necessary to take into account ethical and practical scenarios of false positives in the public domain.
- Further implementation is planned in the form of experimental use in a media education course and a pilot partnership with a fact-checking project.

This description allows:

- To confirm the practical value of the tool through real cases and technical achievements.
- Leave room for objective growth and public implementation.
- Avoid overconfident but unproven statements.

Although the proposed tool demonstrates high performance indicators and has the potential for practical application, it is vital to recognise a number of its current limitations, both technical and conceptual.

Computational costs and performance. The BERT model, even in a compressed version(e.g. DistilBERT), remains resource-intensive for use in browser environments or the Internet (when accessing the cloud API). It limits scalability and usability for mobile or large-scale campaigns.

Vulnerability to attacks from the opponent. Like other models based onBERT and BERT, the system is sensitive to "adversarial perturbations" – for example:

- Paraphrasing sentences with the preservation of meaning (but with a different syntax),
- Inserting neutral phrases to blur semantics.

It can allow for the circumvention of propaganda detection, especially in the case of generated or intentionally disguised content. Overcoming this vulnerability requires specialised robustness techniques, such as counter-training or hull augmentation with attacking samples.

The problem of false classification (false positives/negatives). The model may mistakenly mark:

- analytical or ironic material as propaganda (false positive), which potentially harms freedoms and speech;
- propaganda text as neutral (false negative), which reduces the effectiveness of the system as a tool of choice.

These errors are particularly sensitive in the context of public use, for example, in education, journalism or the public sector. It is vital that the system does not make final decisions, but is used as an auxiliary tool, with the possibility of human transfer.

Linguistic/cultural sensitivity. Some linguistic markers that are interpreted as aggressive or propagandistic may be part of artistic or regional speech, and therefore cause ambiguity.

Recommendations for mitigating risks

- Add an explanation system to the plugin: highlighting words that have influenced the oc inca (explainability).
- Introduce confidence levels of the model (confidence score) with the insulating indicator.
- Train the model on broader types of attacking examples(rephrased, paraphrased, sarcastic).
- Provide a mechanism for correction or feedback on the user ID.

Such an addition not only increases confidence in the scientific integrity of the article but also makes it relevant for practical, responsible deployment in sensitive areas.

6. Conclusions

This paper presents a tool for automated analysis of textual content in order to identify propaganda and key narratives in news based on natural language processing (NLP) and deep learning techniques. The developed approach demonstrated high accuracy in classifying propaganda texts and identifying narratives, which is especially important in the context of information warfare and the rapid spread of disinformation.

The significance of the results obtained lies not only in creating a convenient tool for end users, but also in contributing to the development of technologies for automatic content analysis in Ukrainian, for which there are currently practically no specialised solutions. The developed tool can become a practical support for journalists, fact-checkers, and analysts, as well as increase the level of media literacy of a broad audience.

However, despite the results achieved, certain limitations should be noted. In particular, the system still depends on the quality of the collected training data. It does not take into account multimodal features of the content, such as images or videos, which may also contain propaganda markers. In the future, it is planned to expand the functionality by integrating such capabilities, as well as broadening language support for working with content in other languages.

Summing up, it can be noted that the proposed solution is a step forward in the direction of creating modern tools to combat disinformation and propaganda that meet the current challenges of information security. Further development of the project will increase its flexibility, scalability, and efficiency, as well as strengthen the role of automated systems in shaping users' critical thinking and protecting the information space.

During the calculation work, an innovative information technology was developed that will improve the level of media awareness of users – a browser plugin for classifying propaganda within the framework of the Russian-Ukrainian war. Propaganda is defined as news written in Ukrainian but from Russian media resources, created to discredit the actions of the Ukrainian side in the war.

In the process of preparing materials for processing, the skills of searching for data sets with the necessary information were acquired. After completing such a task, conclusions were also made about the innovativeness of such a system – the specific task of classifying propaganda within the framework of the Russian-Ukrainian war is not solved. The reason is the definition of propaganda as a term, so for this project, it has been specified to reduce possible misunderstandings about the purpose of the tool.

Work on such a project made it possible to apply the skills gained during the study of the tools and methods used by computational linguistics – semantic analysis, analysis of grammar and punctuation, search and analysis of tonality and subjectivity in natural language. Also, the skills of working on the development and testing of API servers and browser plugins have been additionally developed because it is this physical and logical implementation, after in-depth analysis, that turned out to be the most relevant and convenient for users.

Also, the work carried out was preceded by the definition of requirements for the system as a whole, which made it possible to evaluate the developed project expediently and objectively at the end. 50% of the requirements were met, and an MVP system was created, which is the minimum viable project to provide users with further testing. The work of the developed models and server stability was evaluated with positive results.

Therefore, working on such a project made it possible to develop and improve the skills that were acquired during the computational linguistics course.

Thus, the general goal of the work was to help users identify and analyze propaganda in the media space, increasing their media literacy and contributing to information security. The tool that was being created was supposed to provide a quick and convenient way to identify propaganda signs and key narratives. When building the goal tree, the goal sounded like creating a tool that allowed not only the identification of propaganda signs but also the classification of key narratives while providing convenience and ease of use for the end user. According to the analysis and statistics carried out in the previous sections, we can conclude that the goal is achieved because a tool has been created that, thanks to its software implementation – the browser plugin – is convenient and easy to use, and also accordingly fulfils the main task – detects propaganda in news articles and classifies key narratives within it.

During the work on the project, a lot of natural language research was carried out in the context of propaganda of the Ukrainian-Russian war. Particular punctuation and tonal and grammatical patterns were identified, which made it possible to expand the space of features for further classification.

Two separate classification models have also been created that can be used in the future for further integrations of propaganda detection. Their accuracy is compared to that set in the requirements, and their suitability for use in the system is approved.

An API server has been developed that can be used separately from the browser plugin. Its working ability was tested for correctness – the results were satisfactory. The server processed 400+ requests without interruption in two and a half minutes. An interface has also been created for using such an API server in the form of a browser plugin for Google Chrome. It has been analysed for convenience and determined that it is pretty comfortable to use.

An analysis of the requirements defined in the previous sections is also carried out:

Thus, the primary requirements for the system have been met, which allows you to approve the created system as an MVP – the system fulfils its main goals and has general functionality:

- classification of propaganda in news articles;
- classification of narratives in propaganda;
- submission of classification results.

Table 36. Analysis of the fulfilment of the requirements set for the system

Type of requirements	Business Requirements	User requirements	Functional requirements	Non-functional requirements
Appointment	Determine the goals of the business structure, the achievement of which the startup will ensure, and the problems that it will solve	Tasks and actions of users, the implementation of which will be provided by the startup	Description of what the system being developed in an innovative DS startup should do	A description of how a system developed in an innovative DS startup should work to perform its functions
Content of requirements	1. A tool has been created for analyzing texts for propaganda within the framework of information warfare. 2. NLP and ML technologies have been implemented to automate the detection of manipulative content. 3. The level of media literacy and users has not been increased because this is only a demo version that has not been released in the product.	1. The user can select text and get the result of the analysis. 2. The user can view the classification details. 3. The option to send feedback was not created for the demo version. 4. Algorithm sensitivity level settings have not yet been implemented	1. Text analysis is based on the ML model (BERT/RoBERTa). 2. The category of propaganda (emotional pressure, distortion of facts, etc.) is determined. 3. Has not yet been connected to the News API and other sources. 4. User history logging does not occur	1. Word processing time is most often no more than 2 seconds. 2. F1-score 96%. 3. Data encryption and privacy protection of users have not yet been configured. 4. GDPR compliance has not been investigated.

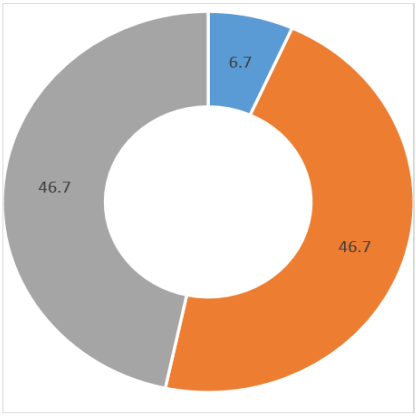


Fig.43. Infographic of compliance with requirements, where blue – partially implemented (1 requirement), orange – implemented (7 requirements) and grey – not implemented (7 requirements)

The reasons for non-compliance with other requirements in most cases are that the project still exists and was developed for personal use. That is, there is no need to configure security and encrypt data or send feedback. In the process of further work on the project, these points will be reconsidered and submitted for elaboration. During the operation, the following obstacles were detected:

- The complexity of defining the concept of propaganda – before developing a system, specifying it and collecting information, it is necessary to determine what will be perceived by the system as propaganda;
- Insufficient quantity and quality of data for training – there was no data already generated for the task, so you need to create your dataset for training models;
- The risk of false positive/negative classification is a risk to the further reputation of the instrument, imperfection of its accuracy;
- Censorship or perception of the plugin as a tool of manipulation is a further legal risk that must be investigated to determine the legality of the tool.

This project has several further prospects – both in its functionality for users and in improving its quality:

- Adding more interpretable results – only the most likely class (propaganda/non-propaganda) and the distribution of narrative probabilities for the propaganda class belong to the information that is currently presented to the user after classification. It would be quite rational to add such metrics as sentiment, level of objectivity, fact-checking (if you add cooperation with fact-checking services, for example, CheckBuster);
- Increasing flexibility for users – adding flexibility as an opportunity for users to define the threshold for the classifier themselves. But this can cause some problems in performance and quality, so threshold fluctuations should be limited (for example, a scale from 0.4 to 0.8);
- Adding a feedback capability to attract reinforcement training – to improve the quality of models, it is possible to add a custom assessment of the results, which will allow you to train the models more;

- Extensions to other browsers or platforms.

The developed tool demonstrated high efficiency in the task of automatic detection of propaganda in news text (F1-score = 0.96) and acceptable results in identifying key narratives (F1-score = 0.70). Such results were achieved thanks to a combination of modern transformer models (BERT), feature engineering, and a carefully assembled educational corpus of Ukrainian texts.

Key factors that led to high accuracy:

- The use of transformer models made it possible to take into account the context and semantic connections between words more deeply, which is especially important for identifying manipulative techniques that are not always reduced to separate lexemes.
- Expanding the standard approach due to special engineering features (number of punctuation marks, tonality, ratio of parts of speech) allowed the model to additionally "feel" the stylistic and emotional patterns characteristic of propaganda texts.
- Involvement of one's corpus of Ukrainian-language texts collected and manually annotated by experts allowed the model to take into account the peculiarities of the Ukrainian media field, which differs from English-language or Russian-language content.

The lower score in the narrative definition task (F1-score = 0.70) can be explained by the complexity of multi-label classification and the limited number of case studies. Each text could contain multiple narratives at the same time, and their combination was rare, which reduced the amount of balanced data for learning. In addition, some narratives have similar lexical and semantic manifestations, making it difficult for experts to distinguish them.

An important observation was that manipulative texts often contain shorter sentences, a higher density of certain punctuation marks (e.g. colons) and a more negative tone. The authors of propaganda materials can deliberately use such features to enhance the emotional impact and create the impression of "urgency" or "indisputability" of the facts.

Another interesting result concerns the distribution of named entities in texts: in propaganda materials, the names of organisations appeared more often, while in non-propaganda materials, geographical locations appeared. It can be explained by the fact that the authors of manipulative texts seek to give weight to the material through the mention of authoritative or well-known structures. At the same time, neutral news more often focuses on describing events tied to a specific place.

The practical significance of the results obtained lies in the creation of a tool that not only gives a "yes/no" assessment of the presence of propaganda but also explains why the text is classified in a certain way, showing the user the key characteristics of the text. It is an essential step towards increasing the transparency of the operation of automatic systems and the formation of critical thinking.

At the same time, the study has its limitations: dependence on the quality of training data, limited volume of narrative corpus, and lack of analysis of multimedia content. In the future, it is planned to expand the case, improve the model architecture for multi-tag classification, and integrate the tool with other fact-checking platforms.

Thus, the study confirms the effectiveness of combining modern NLP models with feature engineering in the task of detecting propaganda, and also demonstrates the prospects for further development of such systems to improve information security.

Effective detection of propaganda in the Ukrainian-language information space requires not only linguistic support, but also an understanding of specific cultural, historical and linguistic contexts. In the development of the tool, these aspects are taken into account at several levels:

- Linguistic Adaptation:
  - The use of the Ukrainian-language corpus for fine-tuning allowed BERT to better navigate the syntactic structures, morphology and specific phrases of the Ukrainian language.
  - The integration of partial-language anal (POS-tagging) using Ukrainian NLP libraries (lang-uk, pymorphy2) helped the model to recognize stylistic patterns (e.g., rhetorical questions, exaggeration through adjectives or verbs in the imperative moodand).
  - Lemmatisation is applied taking into account declension, which is critical for the Ukrainian language with its inflectional system.
- Taking into account culturally loaded vocabulary:
  - When preparing the corpus and additional features**,** lexemes that have a unique semantic load in the Ukrainian social and political context are taken into account, for example:
    - "traitor", "capitulation", "moskal", "enemy of the people",
    - or names that may not have a neutral meaning in Ukrainian discourse (e.g. "Bandera", "Maidan", "NATO").

- A list of lexical triggers has been created based on the analysis of Ukrainian propaganda narratives used in content and originating from disinformation sources.

- Contextualization of historical and political influence:
  - The structure of narratives takes into account typical propaganda frameworks specific to Ukraine:
    - Accusations of fascism or Russophobia,
    - Justification of aggression through "historical unity",
    - Humiliation of statehood and national culture.
  - It ensured the relevance of the system to the post-2014 and 2022 discourse, which differs from the propaganda models of other regions.

- Annotation taking into account cultural specifics. The experts involved in data markup had experience in the Ukrainian media environment, which guarantees:
  - Correspondence of categories to the realities of Ukrainian discourse,
  - Taking into account non-literal meanings, irony, sarcasm, and stylistic devices (for example, political satire).

- Avoidance of cultural bias. In order to avoid erroneous classifications of content that instigates legitimate political protest, historical memory or national identity, the system went through a phase:
  - exclusion of texts with emotionally rich, but well-grounded content,
  - Validations by language nos. and journalists, which evaluated examples before the learning model.

Thus, the tool is not only linguistically compatible with Ukrainian content but is deeply adapted to the cultural and linguistic context, which significantly increases its accuracy and relevance in the fight against propaganda in the Ukrainian information environment.

Although the support of the Ukrainian language is a significant competitive advantage of the system, it is worth noting that there are a number of substantial difficulties associated with the processing of low-resource languages, particularly Ukrainian. The main challenges included:

Limited number of open-labelled housing. In the Ukrainian information environment, there are no full-fledged datasets with labelling of propaganda or narratives. Manual collection of data from news resources (both Ukrainian and propaganda) was carried out, followed by manual marking of texts by experts.

Adaptation of BERT models to the Ukrainian language. Basic BERT models (e.g. multilingual BERT) are only partially optimised for Ukrainian, since it is not sufficiently represented in the original training data. The multilingual model bert-base-multilingual-cased, which supports the Cyrillic alphabet and the basic morphology of Ukrainian, was chosen. Morphological features of Ukrainian (conjugation, word formation) through the integration of lemmatisation and partial-language analysis using spaCy and pymorphy2.

Lack of Ukrainian lexical and emotional dictionaries. Data for the analysis of tonality, subjectivity and stylistic features are not available for the Ukrainian language. Translation of English-language dictionaries with verification of partial-language correspondence was used, and additional synthetic dictionaries with manual validation were generated. It made it possible to create auxiliary features (sentiment_score, subjectivity_score) relevant for impact analysis.

Thus, despite the limited language resources, a comprehensive adaptation of the system to the Ukrainian language has been implemented, based on:

- Customisation of the model,
- Expansion of enclosures through their markings,
- Integration of linguistic knowledge in the form of engineering features.

It significantly increased the reliability and accuracy of the model in the real conditions of the Ukrainian-language information space.

The proposed tool demonstrates the potential for use in areas such as:

- Journalism – as an auxiliary mechanism for detecting manipulative content;
- Fact-checking – for preliminary filtering of suspicious texts;
- Analytics and OSINT – to identify information campaigns;
- Educational initiatives on medical literacy – as a simulator for studying propaganda technologies;
- Information security at the state level is an element of detecting disinformation influences.

However, these statements are promising, and at the time of writing, the system has not yet been implemented on a large scale in real journalistic or government processes. Preliminary testing within the framework of the OSINT hackathon showed interest from analysts and confirmed its usefulness in manual analysis, but a formal user study has not yet been conducted.

Limitations and risks:

- Real-time performance limitation. Although a browser plugin is implemented to integrate with websites, the processing time of a single text (≈2-4 seconds) can be critical for streaming use, especially in mobile conditions or with a weak internet connection. The need to access a cloud API also creates a scalability bottleneck for many concurrent users.
- Barriers for users. The technical complexity of installing or activating the tool may be a barrier for the general public, which includes users without digital competence. Some of the explanations (e.g., "an accusation-type narrative detected") may require prior preparation or context for complete understanding.
- Ethical considerations are especially sensitive in a journalistic or educational context, where misinterpretation of the result can lead to censorship or politically motivated use, and distrust of automatic classifiers in society.

Recommendations for minimising risks:

- In the future, it is planned to add interpreted explanations to the results (for example, highlighting key triggers in the text).
- Formal user testing, including with journalists, analysts, and educators, is needed to demonstrate the real usefulness of the tool.
- It is planned to embed a feedback system for users with the ability to dispute the mark or adjust the OC manually.

This more balanced approach demonstrates not only the potential but also the responsibility of the limitations that any automated system has in sensitive information contexts.

Given the importance of transparency for the user in the sensitive context of propaganda detection, further versions of the system provide for the integration of explanatory methods, in particular:

- visualisation of attention scores in the BERT model to highlight the words that most affect the classification;
- interpretation through SHAP, which allows you to evaluate the contribution of each feature (word, token or grammatical structure) to the result of the model;
- Implementation of a mechanism for displaying classification confidence using colour gradation in the plugin's UI.

Such approaches will allow not only to inform the user about the presence of propaganda, but also to indicate why this particular fragment caused such an assessment, which will increase the level of trust and media literacy.

Given the sensitive nature of propaganda detection tasks and propaganda narratives, the Explainability Module is a key component of the system. Its purpose is to increase user confidence in the system, provide context to each forecast, and reduce the risk of misinterpretations or biased decisions.

The explainability module is implemented at two levels:

- Technical interpretation of the model (on the server side);
- Interactive explanation for the end user (in the browser plugin).

Module architecture:

- Model-level explanation. The following approaches are used to explain the decisions of the classifier:
  - Attention-based Visualisation. For the transformer BERT model, internal attention scales are used, in particular from the last 3-4 layers. These scales allow you to identify the tokens that the model focuses on most when making a decision.
  - SHAP (SHapley Additive exPlanations) is used to post-analyse the classification model. It allows you to assess which words, phrases or lexical-grammatical features made the most significant positive or negative contribution to the final assessment (for example, "accusation", "incitement to hatred").
  - Confidence Thresholds & Feature Scoring. Each decision is accompanied by a probabilistic assessment (Sigmoid/Softmax). In parallel with the main model, assessments of additional features are displayed: punctuation, emotionality, subjectivity, the number of modal words, etc.
- Interactive explanation for the user (plugin). The browser plugin interface implements the following explainability functions:

- Highlighting text fragments. The words or phrases that contributed the most to the solution are automatically highlighted in colour:
  - red – elements that are probably propagandistic;
  - yellow – potentially emotionally manipulative fragments;
  - green – neutral or protective parts.
- Graphical confidence indicator. Next to the classification (for example, "propaganda - 82%"), a colour scale or confidence bar is displayed, which allows you to assess the degree of confidence of the model.
- Detailed explanation. When clicking on the "Details" icon, the user receives:
  - Labels of active narratives (e.g., "Justification", "Invocation"),
  - Examples of key phrases that led to the classification,
  - short explanation: *"The phrase 'we were forced' is often used in the narrative to justify aggression."*

The ethical aspect, in particular, the presence of explainability:

- Reduces the risk of misinterpretation of results,
- Allows the user to check the fairness of the classification,
- Ensures that the system complies with the principles of AI transparency and AI accountability in the field of media and information security.

In future versions, it is planned:

- Add an explanation for each layer of the transformer (attention roll-out),
- Implement the possibility of user feedback on false highlights,
- Train the interpreter on Ukrainian-language data with explanations to improve the localised accuracy of interpretation.

This module is an integral part of the tool's architecture, providing not only the technical quality but also the transparency needed for use in education, journalism, and government information systems.

To ensure trust, efficiency, and safe use of the tool, it is worth implementing the following features in the following versions:

- Phrase highlighting based on attention mechanisms or SHAP/LIME.
- A brief description of the classification logic: "The key phrase was revealed: 'we were forced' – typical of the narrative 'justification'."
- Visualisation of the confidence level of the model for each feature (colour heatmap, risk rating).

Currently, the tool does not provide an explicit explanation of the reasons for classification (e.g., highlighting phrases), although the article recognises the importance of explainability. It is a critical point for further improvement, in particular, to increase user trust, especially in the context of education, media literacy and journalism.

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] Lühring, J., Metzler, H., Lazzaroni, R., Shetty, A., & Lasser, J. (2025). Best practices for source-based research on misinformation and news trustworthiness using NewsGuard. Journal of Quantitative Description: Digital Media, 5. https://doi.org/10.51685/jqd.2025.003

[2] Hassan, N., et al. (2017). ClaimBuster: The first-ever end-to-end fact-checking system. Proc. VLDB Endow.10(12) Pp. 1945–1948. https://doi.org/10.14778/3137765.3137815

[3] Shao, C., et al. (2016). Hoaxy: A platform for tracking online misinformation. Proc. WWW '16 Companion. https://doi.org/10.1145/2872518.2890098

[4] Shu, K., et al. (2019). Fakey: A game intervention to improve news literacy on social media. Computers in Human Behavior. https://doi.org/10.1145/3449080

[5] Da San Martino, G., et al. (2019). Fine-grained analysis of propaganda in news articles. In Proceedings of the 2019 Conference

on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5636–5646, Hong Kong, China. Association for Computational Linguistics. https://dx.doi.org/10.18653/v1/D19-1565

[6] Barrón-Cedeño, A., Da San Martino, G., Jaradat, I., & Nakov, P. (2019). Proppy: A System to Unmask Propaganda in Online News. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 9847-9848. https://doi.org/10.1609/aaai.v33i01.33019847

[7] Devlin, J., et al. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

[8] Rogers, A., et al. (2020). A Primer in BERTology: What we know about how BERT works. Transactions of the Association for Computational Linguistics 2020; 8 842–866. https://doi.org/10.1162/tacl_a_00349

[9] Fernandes, S., et al. (2020). Detecting deepfake videos using attribution-based confidence metric. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 308-309). https://openaccess.thecvf.com/content_CVPRW_2020/papers/w19/Fernandes_Detecting_Deepfake_Videos_Using_Attribution-Based_Confidence_Metric_CVPRW_2020_paper.pdf

[10] Victoria Vysotska, Krzysztof Przystupa, Yurii Kulikov, Sofiia Chyrun, Yuriy Ushenko, Zhengbing Hu, Dmytro Uhryn, "Recognizing Fakes, Propaganda and Disinformation in Ukrainian Content based on NLP and Machine-learning Technology", International Journal of Computer Network and Information Security, Vol.17, No.1, pp.92-127, 2025.

[11] Danylo Levkivskyi, Victoria Vysotska, Lyubomyr Chyrun, Yuriy Ushenko, Dmytro Uhryn, Cennuo Hu, "Agile Methodology of Information Engineering for Semantic Annotations Categorization and Creation in Scientific Articles Based on NLP and Machine Learning Methods", International Journal of Information Engineering and Electronic Business, Vol.17, No.2, pp. 1-50, 2025.

[12] Oleh Prokipchuk, Victoria Vysotska, Petro Pukach, Vasyl Lytvyn, Dmytro Uhryn, Yuriy Ushenko, Zhengbing Hu, "Intelligent Analysis of Ukrainian-language Tweets for Public Opinion Research based on NLP Methods and Machine Learning Technology", International Journal of Modern Education and Computer Science, Vol.15, No.3, pp. 70-93, 2023.

[13] Plikynas, D., Rizgelienė, I., & Korvel, G. (2025). Systematic Review of Fake News, Propaganda, and Disinformation: Examining Authors, Content, and Social Impact through Machine Learning. IEEE Access. https://ieeexplore.ieee.org/abstract/document/10843666/

[14] Horák, A., Sabol, R., Herman, O., & Baisa, V. (2024). Recognition of propaganda techniques in newspaper texts: Fusion of content and style analysis. Expert Systems with Applications, 251, 124085.

[15] Lilleker, D., & Surowiec, P. (2020). Content analysis and the examination of digital propaganda on social media. In The SAGE Handbook of Propaganda (pp. 171-188). SAGE Publications Ltd.

[16] Santos, F. C. C. (2023). Artificial intelligence in automated detection of disinformation: A thematic analysis. Journalism and Media, 4(2), 679-687.

[17] Iosifidis, P., & Nicoli, N. (2020). The battle to end fake news: A qualitative content analysis of Facebook announcements on how it combats disinformation. International Communication Gazette, 82(1), 60-81.

[18] Horák, A., Baisa, V., & Herman, O. (2021). Technological approaches to detecting online disinformation and manipulation. Challenging online propaganda and disinformation in the 21st century, 139-166.

[19] Kanozia, R. (2019). Analysis of digital tools and technologies for debunking fake news. Journal of Content, Community & Communication, 9(5), 114-122.

[20] Nasiri, S., & Hashemzadeh, A. (2025). The Evolution of Disinformation from Fake News Propaganda to AI-driven Narratives as Deepfake. Journal of Cyberspace Studies, 9(1), 203-222.

[21] Oates, S., Lee, D., & Knickerbocker, D. (2022). Data Analysis of Russian Disinformation Supply Chains: Finding Propaganda in the US Media Ecosystem in Real Time. Oates, Sarah, Doowan Lee, and David Knickerbocker.

[22] Giordano, L. On the Automatic Multilingual Detection of Persuasion Techniques in the News: A Natural Language Processing Approach. https://amslaurea.unibo.it/id/eprint/34127/

[23] Danylo Holubinka, Victoria Vysotska, Serhii Vladov, Yuriy Ushenko, Mariia Talakh, Yurii Tomka, "Intelligent System for Recognizing Tone and Categorizing Text in Media News at an Electronic Business Based on Sentiment and Sarcasm Analysis", International Journal of Information Engineering and Electronic Business, Vol.17, No.1, pp. 90-139, 2025.

[24] Dmytro Uhryn, Victoria Vysotska, Lyubomyr Chyrun, Sofia Chyrun, Cennuo Hu, Yuriy Ushenko, "Intelligent Application for Textual Content Authorship Identification based on Machine Learning and Sentiment Analysis", International Journal of Intelligent Systems and Applications, Vol.17, No.2, pp.56-100, 2025.

[25] Afeez Ayomide Olagunju, Iyabo Olukemi Awoyelu, "Performance Evaluation of Fake News Detection Models", International Journal of Information Technology and Computer Science, Vol.16, No.6, pp.89-100, 2024.

[26] Solopova, V., Popescu, O. I., Benzmüller, C., & Landgraf, T. (2023). Automated multilingual detection of pro-kremlin propaganda in newspapers and telegram posts. Datenbank-Spektrum, 23(1), 5-14.

[27] West, R., & Pfeffer, J. (2017, May). Armed conflicts in online news: a multilingual study. In Proceedings of the International AAAI Conference on Web and Social Media, Vol. 11, No. 1, pp. 309-318.

[28] Khairova, N., Ivasiuk, B., LO SCUDO, F., Comito, C., & Galassi, A. (2023). A First Attempt to Detect Misinformation in Russia-Ukraine War News through Text Similarity. In Proceedings of the 4th Conference on Language, Data and Knowledge (LDK) (pp. 559-564). NOVA CLUNL.

[29] Ozcelik, O., Yenicesu, A. S., Yildirim, O., Haliloglu, D. S., Eroglu, E. E., & Can, F. (2023, September). Cross-lingual transfer learning for misinformation detection: Investigating performance across multiple languages. In Proceedings of the 4th Conference on Language, Data and Knowledge (pp. 549-558).

[30] GitHub - hybrinfox/ppn: Repository for the Propagandist Pseudo-News dataset. GitHub. URL: https://github.com/hybrinfox/ppn

[31] zeusfsx/ukrainian-news · Datasets at Hugging Face. Hugging Face – The AI community building the future. URL: https://huggingface.co/datasets/zeusfsx/ukrainian-news

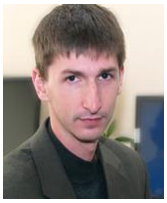
[32] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[33] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers) (pp. 4171-4186).
[34] Wolf, Thomas, et al. "Huggingface's transformers: State-of-the-art natural language processing." arXiv preprint arXiv:1910.03771 (2019).
[35] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
[36] TextBlob/textblob/en/en-sentiment.xml at eb08c120d364e908646731d60b4e4c6c1712ff63 · sloria/TextBlob. GitHub. URL: https://github.com/sloria/TextBlob/blob/eb08c120d364e908646731d60b4e4c6c1712f f63/textblob/en/en-sentiment.xml
[37] GitHub - skupriienko/Ukrainian-Sentiment-Analysis: The list of Ukrainian words for sentiment analysis and NLP. GitHub. URL: https://github.com/skupriienko/Ukrainian-Sentiment-Analysis
[38] Icard B. Exposing propaganda: an analysis of stylistic cues comparing human annotations and machine classification. Academia.edu - Find Research Papers,
[39] Topics, Researchers. URL: https://www.academia.edu/122724298/Exposing_propaganda_an_analysis_of_stylistic_cues_comparing_human_annotations_and_machine_classification
[40] Propagandist Solovyov called for "Berlin" and threatened the United States. TSN.ua. URL: https://tsn.ua/svit/propahandyst-solovyov-zaklykav-dovbanuty-po-berlinu-ta-pryhroz

Authors' Profiles

Maryna Nyzova is an undergraduate student at Lviv National Polytechnic University, majoring in Data Science within the Information Systems and Networks department. Her academic interests focus on natural language processing, with a strong inclination toward research in this field.

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, specializing 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 500 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.



Lyubomyr Chyrun is an Associate Professor at the Ivan Franko National University of Lviv. Since 2001, he worked at the Lviv Polytechnic University at the Institute of Computer Sciences and Information Technologies. as an associate professor of the Department of Information Systems and Networks. In 2007, he defended his PhD thesis on May 1, 2002 - "Mathematical modelling and computational methods". He co-authors the monograph "Continuous Fractions and Complex Numbers". He is the author of more than 120 scientific publications. His areas of scientific interest are pattern recognition, the application of numerical methods in information technologies, object-oriented programming, web technologies, fake identification, natural language processing, computer linguistics, data science, system analysis, information technologies, and machine learning.

Zhengbing Hu, Professor, Deputy Director, International Center 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 in National Aviation University (Ukraine) from 2019. Primary research interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications, Data Processing, Cloud Computing, Education Technology.

Yuriy Ushenko World's Top 2% Research Scientist (2020, 2021, 2022, 2024). Nominee of The Photonics100 2025 list by ElectroOptics. Winner of 1000 Talents Plan Program, PhD, DSc, Prof., at Shaoxing University, Shaoxing, Zhejiang Province 312000, China. Professor, Head of Computer Science Department, Chernivtsi National University, Ukraine. His research interests include intelligent information systems design, data mining, pattern recognition and digital image processing, artificial neural networks, laser polarimetry and interferometry.

Dmytro Uhryn graduated from Yuriy Fedkovych Chernivtsi National University, Chernivtsi. He is a Doctor of Technical Sciences and associate professor at Yuriy Fedkovych Chernivtsi National University. He has currently published more than 170 publications. His research interests include data mining, information technology for decision support, swarm intelligence systems, and industry-specific geographic information systems. His areas of scientific interest are decision supporting information technologies, swarm intelligence systems and industry geoinformation systems.

How to cite this paper: Maryna Nyzova, Victoria Vysotska, Lyubomyr Chyrun, Zhengbing Hu, Yuriy Ushenko, Dmytro 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