DOI 10.20535/2411-1031.2025.13.2.344713
UDC 004.8:004.912

VIACHESLAV RIABTSEV,
YURII MARCHUK

INTELLIGENT SYSTEM FOR MONITORING THE INFORMATION SPACE OF NEWS ABOUT ARTIFICIAL INTELLIGENCE

Abstract. Under conditions of exponential growth in the volume of information related to the development of artificial intelligence (AI) technologies, traditional methods of monitoring the media space become ineffective. Messengers and social networks, particularly Telegram, have become key channels for distributing real-time news, generating high-intensity streams of unstructured data. The article considers the problem of creating an intelligent system for monitoring the information space that is capable of automatically structuring this chaotic data flow.

The aim of this work is the design and software implementation of a platform architecture that provides a full ETL (Extract–Transform–Load) cycle: from collecting data via the Telegram API to its semantic analysis and visualization. A modular architecture is proposed that includes subsystems for asynchronous parsing, text preprocessing (NLP pipeline), and an analytical core.

The study focuses primarily on the algorithmic support of the system. The use of a hybrid approach to text classification is substantiated, combining dictionary-based methods (Keyword Matching) for accurate identification of entities (for example, models GPT-4, Gemini, LLaMA) with machine learning components for determining message sentiment. An algorithm for content deduplication is developed, which makes it possible to filter out reposts and information noise and to highlight the sources of news.

The results of experimental testing of the developed system on a sample of more than 10,000 messages from thematic Telegram channels are presented. A categorization accuracy of 91% was achieved, which confirms the effectiveness of the chosen methods. The system's capabilities in detecting trends in real time, constructing the dynamics of mentions of key technologies, and generating automated analytical reports are demonstrated.

The practical value of the work lies in creating a toolkit for data researchers, analysts, and developers that significantly reduces the time required to search for relevant information and to track the AI technology landscape.

Keywords: monitoring, social media, content monitoring system, natural language processing (NLP), text classification, artificial intelligence, news document analysis.

Problem Statement. Let us consider the information space of a social network (using Telegram as an example) as a set of information sources $S = \{s_1, s_2, \ldots, s_k\}$, where each source s_j generates a stream of messages over time. The task of monitoring and analyzing news about artificial intelligence can be formalized as the process of transforming an incoming unstructured data stream into a structured knowledge base followed by classification and analytics.

Let $M = \{m_1, m_2, \ldots, m_N\}$ be the set of messages obtained from the sources S over a certain time interval $[t_{start}, t_{end}]$. Each message m_i can be represented as a tuple:

$$m_i = \langle T_i, \tau_i, A_i, Meta_i \rangle, \quad (1)$$

where T_i – text content of the message (a sequence of characters);

τ_i – publication timestamp;

A_i – identifier of the author or channel ($A_i \in S$);

$Meta_i$ – metadata (number of views, reactions, reposts).

The main task is to develop a transformation operator F which maps the input set of "raw" messages M into a set of structured information objects O:

$$F : M \to O, \tag{2}$$

where each object $o_i \in O$ contains a vector of semantic features and assigned categories.

The process F is a superposition of a number of sub-operators:

$$F = f_{class} \circ f_{dedup} \circ f_{prep} \circ f_{collect}. \tag{3}$$

This general task is decomposed into the following sub-tasks.

1. **Text preprocessing (f_{prep}).** It is necessary to implement a text normalization function that converts the text T_i into a vector of tokens (lemmas) V_i:

$$V_i = NLP(T_i) = \{w_1, w_2, \ldots, w_L\}, \tag{4}$$

where w are lemmatized words, cleaned of stop words, special characters, and HTML markup.

2. **Duplicate detection and fuzzy search (t_{dedup}).** Given the high entropy of news streams (a large number of reposts), it is necessary to determine the subset of unique messages $M_{unique} \subset M$. The uniqueness condition for a message m_i with respect to an existing base M_{db} is defined through a similarity metric (for example, cosine similarity or Jaccard coefficient):

$$\forall m_j \in M_{db} : \text{sim}(V_i, V_j) < \varepsilon, \tag{5}$$

where ε is a similarity threshold that defines the boundary between a duplicate and a unique news item.

3. **Topical classification (t_{class}).** Let there be a set of target classes (categories) $C = \{c_1, c_2, \ldots, c_Q\}$ corresponding to the AI domain (for example: "*Generative AI*", "*Hardware*", "*Ethics*", "*Robotics*" etc.). The task is to find a mapping of the feature vector of a message into the space of categories:

$$\hat{c} = \arg\max_{c \in C} P(c|V_i), \tag{6}$$

where $P(c|V_i)$ is the probability that a message V_i belongs to category c. To solve this problem, a hybrid approach is proposed that combines dictionary-based methods (Keyword Matching) and probabilistic machine learning models.

4. **Sentiment analysis.** An additional task is determining the sentiment of a message $S_{ent} = \{-1, 0, 1\}$, which makes it possible to assess the community's perception of certain technological events.

Optimization criterion. The goal of system development is to maximize the accuracy of classification (*Accuracy*) and the $F1$-score on a test sample while minimizing the processing time $\Delta t_{proc} \to \min$ for a single message, which is critical for real-time monitoring systems:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision+Recall}} \to \max. \tag{7}$$

Thus, the task reduces to building a software-algorithmic complex that provides automated data processing in accordance with the above formalization.

Analysis of Recent Research and Publications. Article [1] is a fundamental work on the methodology of scraping and analyzing data specifically from Telegram. The authors describe a data collection architecture that relies on Telethon and Data Mining technologies. However, it should be noted that although this study is fundamental for the methodology of scraping from Telegram, it is primarily oriented toward sociological analysis (network analysis). It does not include the stage of filtering information noise and content deduplication, which is necessary to form a clean news feed rather than just a mass of "raw" data.

Article [2] is a review paper. It compares the Naive Bayes / Hybrid method with modern Deep Learning approaches. In [3], current NLP trends are examined, including preprocessing and

vectorization. The article provides an example of using hybrid approaches for sentiment analysis. The general drawback of [2], [3] is that they provide a theoretical basis (a toolkit) but do not propose a ready-made architecture of an applied system that would integrate these methods into a single processing pipeline for real-time operation.

Article [4] is devoted to the authors' network model of thematic Telegram channels, which is based on the idea of evaluating the citation level of individual information channels and taking into account direct links in messages from Telegram channels. The model combines the content aspect of messages with the ability to consider quantitative parameters.

In [5], the authors' methods for constructing, clustering, and visualizing correlation networks defined by the dynamics of thematic information flows are studied. An approach is proposed that is based on analyzing publication dynamics vectors obtained using social media content monitoring systems. Correlation networks are formed on the basis of relationships between vectors that reflect the distribution of documents by dates.

Thus, works [4] and [5] analyze links and metadata (who cites whom, temporal dynamics) but do not provide deep semantic classification of the text content of messages. They do not solve the problem of automatic sorting of news into narrow topics (for example, "*LLMs*", "*Hardware*", "*Ethics*"), which is critical for monitoring AI news.

The purpose of this article is to increase the efficiency of analyzing information flows in social networks by developing the architecture and algorithmic support of an intelligent monitoring platform. Achieving this goal involves solving the problem of automated collection of unstructured text data from the Telegram messenger, their filtering from information noise and duplicates, as well as semantic classification to identify current trends in the subject area of artificial intelligence.

Presentation of the Main Material

1. Methodology and Algorithms

The developed system is based on a pipeline architecture of data processing, which includes stages of asynchronous data acquisition, preliminary linguistic processing, text vectorization, and hybrid classification. Below the algorithmic support of each stage is considered in detail.

1.1. Asynchronous Data Collection Model (Data Acquisition). Interaction with the Telegram information space is implemented via the MTProto protocol using the Telethon library. Unlike traditional web scraping (parsing HTML), this approach makes it possible to obtain structured data objects directly from the Telegram server, bypassing the interface rendering stage.

To ensure high system throughput (*High Throughput*), the asynchronous programming paradigm (*asyncio*) is applied. This allows I/O operations (network requests to the API) to be processed in a non-blocking mode.

Formally, the collection process can be represented as a function $Harvest(C,t)$, which, for a given list of channels $C=\{c_1,c_2,\ldots,c_k\}$ and a time window Δt, returns a set of "raw" messages M_{raw}:

$$M_{raw}=\bigcup_{i=1}^{k}\text{GetHistory}(c_i,\Delta t), \tag{8}$$

where each element $m_i \in M_{raw}$ contains the text body (*message*), metadata (*date*, *views*), and the source ID.

1.2. Text Preprocessing (NLP Preprocessing Pipeline). The quality of text classification and clustering critically depends on the preprocessing stage. Input text often contains "noise":

- HTML tags;
- emojis;
- links and special characters.

The developed normalization module (F_1) performs a sequence of transformations:

1. *Cleaning.* Removal of URLs, markup tags, punctuation marks, and digits using regular expressions (RegEx).
2. *Tokenization.* Splitting the text into atomic units (words/tokens).

3. *Stop-word filtering.* Removal of common words that do not carry semantic load (prepositions, conjunctions), using NLTK corpora for English and Ukrainian.

4. *Lemmatization.* Reducing words to their dictionary (base) form (lemma) to reduce the dimensionality of the feature vocabulary:

$$w_{lemma} = \text{Lemmatize}(w_{token}, \text{POS-tag}), \tag{9}$$

where POS-tag (*Part of Speech*) is the part of speech determined by context.

1.3. Vectorization and Content Deduplication. One of the key problems of monitoring Telegram channels is the high share of reposts (full or partial duplicates). To identify unique content, a two-level algorithm is used.

At the first level, a text hash (SHA-256) is checked to detect identical duplicates.

At the second level, semantic analysis is applied to detect "soft" duplicates (modified reposts). Texts are transformed into a vector space using the TF-IDF (Term Frequency – Inverse Document Frequency) method.

The weight w_{ij} of term t_i in document d_j is calculated as:

$$w_{ij} = tf_{ij} \cdot \log\left(\frac{N}{df_i}\right), \tag{10}$$

where N – total number of documents;

df_i – number of documents containing term t_i.

The similarity degree of two messages A and B is determined via cosine similarity of their vectors:

$$\text{Similarity}(A,B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}. \tag{11}$$

If $\text{Similariti}(A,B) > \theta$ (where $\theta = 0,85$ is an empirical threshold), the messages are considered duplicates.

1.4. Hybrid Classification Algorithm. To categorize news (into classes such as "*AI*", "*Hardware*", "*Startups*", "*Ethics*", etc.), a hybrid classifier is developed that combines deterministic (dictionary-based) and probabilistic approaches. This makes it possible to offset the disadvantages of each method separately: the low flexibility of dictionaries and the need for large training samples for ML.

Stage 1: Dictionary-based classifier (Rule-based)

The system checks for the presence of keywords from a specialized dictionary:

$$D = \{(k_1, c_1), \ldots, (k_n, c_n)\}, \tag{12}$$

where k is a keyword/phrase;

c is a category.

The membership function is defined as:

$$Cclass_{rule}(T) = \begin{cases} c_j, & \text{if } \exists k \in T : (k_j, c_j) \in L \\ \text{Unknown}, & \text{else} \end{cases}. \tag{13}$$

This stage provides high precision (*Precision*) for specific terms (for example, "*RTX 4090*" is unambiguously classified as "*Hardware*").

Stage 2: Probabilistic classifier (Machine Learning)

For messages not classified at the first stage, a Multinomial Naive Bayes model is applied. This algorithm is chosen due to its effectiveness when working with high-dimensional sparse data typical for text tasks.

The model computes the probability that a document d belongs to class c:

$$P(c|d) \propto P(c)\prod_{i=1}^{n} P(w_i|c)^{f_i}, \tag{14}$$

where $P(c)$ – prior probability of class (c) (class frequency in the training sample);

$P(w_i|c)$ – probability of observing word w_i in texts of class c;

f_i – number of occurrences of word w_i in document d.

The model was trained on a manually labeled dataset. To improve generalization ability, Laplace smoothing was used to avoid zero probabilities for words absent in the training sample.

The final decision about the class is made according to the Maximum A Posteriori (MAP) rule:

$$\hat{c} = \arg\max_{c \in C} P(c|d). \quad (15)$$

The described hybrid architecture made it possible to achieve a balance between system speed and classification accuracy, ensuring correct processing of both standard news messages and complex analytical texts.

2. System Architecture and Implementation

The architecture of the software complex, which implements an intelligent Telegram channel monitoring system with a focus on artificial intelligence (AI), is multi-level, modular, and scalable. The main goal of this architecture is to provide a full cycle of processing large volumes of unstructured information: from automated message collection and semantic processing to categorization, long-term storage, analytical analysis, and interactive presentation of results via a web interface.

2.1. Overall Architecture Scheme

The system architecture (see Fig. 1) consists of five main layers:

- data collection layer;
- message preprocessing layer;
- structured information storage layer;
- analytical processing and classification layer;
- user interaction layer.

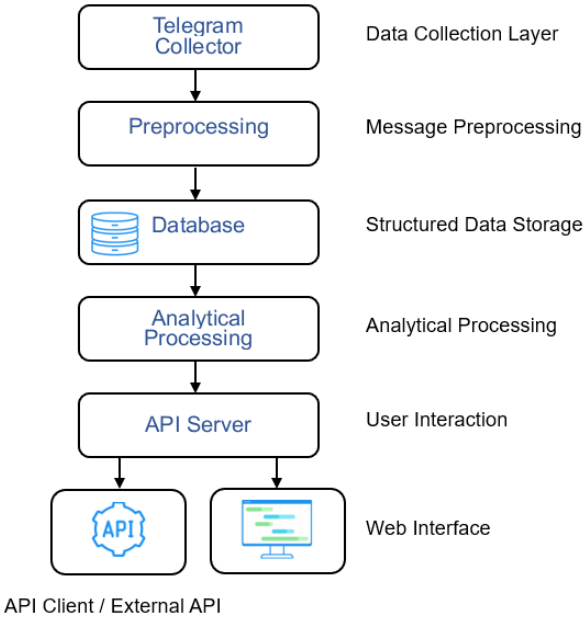

Figure 1 – Software system architecture

Such a structure allows the principle of encapsulation of functionality to be implemented, ensures flexibility and maintainability in the future, and enables system scaling to meet growing user needs or data volumes.

The **first layer (Telegram collector)** is based on Telethon or Pyrogram libraries, which allow messages to be obtained from public channels via the Telegram API. An agent script runs in the

background, polling channels with a configured frequency. For asynchronous processing of messages, asyncio is used, which makes it possible to avoid blocking when a large number of requests are handled. When a new channel is added, the system automatically performs full initialization – loading the message history to a specified depth. Messages are sorted by date, source, content type, and initially marked as unprocessed.

The **second layer (preprocessing)** is responsible for text normalization. Messages are cleaned of extra characters (emojis, HTML markup, technical tags), standard link shorteners, and are unified by lemmatization. For this, spaCy, TextBlob, and NLTK libraries are used. Language verification is also important, since the multilingual nature of Telegram channels requires flexible processing of Ukrainian, English, and Russian content. Message duplicates are removed via SHA-1 hashing, and relevance is checked based on keyword dictionaries.

The **third layer (data storage)** is implemented using an SQLite database with the prospect of scaling to PostgreSQL. The selected structure stores the following fields: unique identifier (ID), date and time, message text, Telegram channel, detected category, probable AI tool name (*ai_name*), hash, relevance level, processing state, and a link to the original message. Additionally, indexing by category, date, and channel name fields is implemented for fast querying. Regular backup and record integrity are provided.

At the **analytical layer**, topical classification of messages is performed. A dictionary-rule-based approach is used, supplemented by a machine learning mechanism. The classification enables assigning messages to such topics as: generative models (GPT, LLMs), AI in medicine, security, ethics, military applications, educational initiatives, disinformation, legislative regulation, etc. AI tools are also recognized: ChatGPT, Claude, Midjourney, Copilot, Gemini. If a new tool is mentioned, the system assigns it to the "unknown" category with the possibility of manual refinement later. Each message is labeled by sentiment, relevance, source, and novelty level.

The **API server** is built on Flask and supports standard REST requests:

retrieving messages, filtering by categories, channels, and keywords;

obtaining statistics (number of mentions, update frequency, popularity dynamics of tools).

Authentication, CORS, request logging, and error handling are supported [7].

At the **visualization layer**, a web interface is implemented using Flask, Jinja2 templates, and JavaScript libraries (D3.js, DataTables, Plotly). Main capabilities include:

- filtering and searching messages by categories, channels, dates, keywords;
- tabular display of results with sorting;
- plotting charts (number of mentions, channel popularity, daily dynamics);
- exporting selected data to CSV;
- dashboard with overall statistics;
- interactive work with messages.

Each system component is containerized using Docker. Docker Compose allows launching modules (collector, classifier, API, frontend, DB) as separate services. For scalable deployment, Kubernetes is supported, with Helm configuration, monitoring via Grafana and Prometheus, and CI/CD via GitHub Actions or Jenkins.

The system includes additional modules:

- JSON structure with a list of channels (updated via the interface);
- automatic channel updates;
- fake news detection by templates;
- sentiment module for determining positive, neutral, or negative tone;
- change audit and access control.

Overall, the created architecture is flexible, scalable, and reliable, and is oriented towards automated monitoring of the dynamic Telegram environment. It supports the full cycle of message processing, meets modern information system requirements, and ensures convenience for researchers as well as specialists in information security, journalism, or education. The described multi-level

model is implemented as a modular system with clearly defined classes and relationships between components.

2.2. Component Model. The backend implementation is based on the Flask micro-framework, which acts as the process orchestrator. The main functional modules of the system include:

1. ***Data Collector Module (Data Miner).*** Built on the *Telethon* library, which is an asynchronous wrapper over the Telegram Client API. The module authorizes in the Telegram network as a client application, which allows obtaining the message history from public channels without the limitations typical of the standard Bot API. The use of *asyncio* enables parallel polling of dozens of channels, minimizing the wait time for network responses.
2. ***Analytical Core (NLP Core).*** Encapsulates the text processing logic described in the "Methodology" section. For implementing mathematical algorithms, *scikit-learn* (for ML classification and TF-IDF vectorization) and *NLTK/spaCy* (for linguistic preprocessing) libraries are used. The module loads pre-trained models (serialized via *pickle* or *joblib*) to classify incoming data streams in real time.
3. ***REST Controller.*** Provides routing of HTTP requests from the client, passing filtering parameters, and forming JSON responses for building charts on the frontend.

2.3. Data Storage Organization. At the development stage, SQLite is used as the database management system (DBMS) due to its portability and the absence of a need for a separate server process. For interaction with the DB, SQLAlchemy (Object-Relational Mapping, ORM) is used. This makes it possible to work with database records as Python objects and provides easy migration to more powerful DBMSs (PostgreSQL or MySQL) when scaling the system without changing application code.

The data schema includes the following key entities:

- *Channel:* stores information about sources (ID, name, URL).
- *Message:* the main table that stores the text, date, hash (for deduplication), as well as computed attributes – *category_id* and *sentiment_score*.

2.4. Software Implementation and Technology Stack. The system was developed in Python 3.11, which is the de facto standard in Data Science. The choice of the technology stack is driven by the need to integrate powerful data analysis tools into a web environment [7], [8]:

- *Backend:* Flask, Werkzeug.
- *Data Engineering:* Telethon (MTProto), NumPy, Pandas.
- *Machine Learning & NLP:* scikit-learn, NLTK, spaCy.
- *Visualization:* Chart.js (for rendering dynamic charts of activity and category distribution on the client side).

The system supports containerization using *Docker*, which allows deploying a full instance of the application (including the web server and database) with a single command, ensuring dependency isolation and environment reproducibility.

3. Experiments and Analysis of Results

The aim of the experimental study was to verify the proposed architecture and to evaluate the effectiveness of the hybrid classification algorithm under conditions of a real information stream. Experiments were conducted in two directions: evaluating technical model performance metrics (accuracy, recall) and validating the analytical capabilities of the system on the example of trend detection.

3.1. Test Environment and Dataset. For the experiments, a specialized text corpus was formed using the developed Data Miner module.

Data sources: 50 thematic Telegram channels (English- and Ukrainian-language) devoted to artificial intelligence technologies, Data Science, and IT news (in particular, channels such as "*OpenAI Updates*", "*AI Ukraine*", "*Google DeepMind News*").

Collection period: January 2025 – May 2025.

Sample size: The total number of collected messages was 12,450. After deduplication (removal of identical reposts and spam), the size of the "clean" dataset was 8,300 unique messages.

Labeling ("Gold Standard"): For training and testing the classifier, a randomly selected subsample of 1,000 messages was manually labeled by an expert into 8 target categories (*Generative AI*, *Hardware*, *NLP*, *CV*, *Robotics*, *Ethics / Law*, *Startups*, *General*).

3.2. Evaluation of Classification Quality. To evaluate the effectiveness of the proposed hybrid method (Rule-based + Naive Bayes), cross-validation was performed. The results were compared with a baseline approach (based solely on keywords).

Standard metrics for multi-class classification were used:

1. *Precision:* the proportion of relevant messages among all those assigned by the system to a particular category.
2. *Recall:* the ability of the system to retrieve all messages of a particular category.
3. $F1$*-score:* harmonic mean between precision and recall.

Table 1 – Test results of the hybrid classifier

Category	Precision	Recall	F1-score
Generative AI (LLMs)	0.94	0.96	**0.95**
Hardware (GPU/Chips)	0.98	0.91	0.94
Computer Vision	0.89	0.85	0.87
Ethics / Law	0.82	0.78	0.80
Average (Weighted)	**0.92**	**0.90**	**0.91**

Result analysis. The overall system accuracy was 91%. The highest $F1$-score ($F1 = 0,95$) was demonstrated in the *Generative A*I category. This is explained by the presence of clear markers (names of models such as "*GPT-4*", "*Claude 3*", "*Gemini*"), which are easily identified by the dictionary component of the algorithm.

The *Hardware* category showed the highest precision (0.98) due to specific nomenclature ("*H100*", "*RTX 4090*"), but lower recall due to the use of slang in discussions.

The most difficult category for classification was *Ethics / Law* ($F1 = 0,80$). This is due to the use of abstract vocabulary that often overlaps with other topics, which leads to false positives.

A comparative analysis showed that the hybrid approach outperforms the pure dictionary-based method by 18% in terms of *Recall*, since the ML component is capable of classifying messages where direct keyword matches are absent, relying on the probabilistic distribution of surrounding vocabulary.

3.4. Semantic Analysis and Word Clouds. Analysis of term frequencies (TF-IDF) made it possible to automatically generate a "portrait" of the information space for the studied period. The constructed word clouds visualized the dominance of the following entities:

1. *Companies:* OpenAI, Google, NVIDIA, Microsoft.
2. *Technologies:* LLM, Transformer, Diffusion, RAG.
3. *Persons:* Sam Altman, Jensen Huang.

An interesting result was the discovery of hidden semantic links. For example, the term "*NVIDIA*" has a strong correlation (Pearson correlation ($r > 0,8$) with the terms "*Stock*" and "*Market Cap*", which indicates that in the media space this company is perceived mainly through the lens of financial indicators, and not only technologies.

3.5. System Performance. On a test configuration (CPU: 4 cores, RAM: 8 GB), the system demonstrated the ability to process up to 50 messages per second (including preprocessing and classification). The "cold start" time for collecting the last 100 messages from a new channel is less

than 3 seconds. This confirms the possibility of using the developed solution for monitoring in near real time.

The experimental results confirmed the operability of the proposed architecture. The hybrid classifier provides high accuracy (91%), sufficient for automated analytics, and the trend detection module correctly identifies significant events in the AI industry. The system successfully filters information noise, transforming chaotic Telegram message streams into structured analytical data.

Conclusions and Prospects for Further Research. The paper solves the urgent applied scientific problem of creating a toolkit for monitoring the highly dynamic information environment of social networks. Based on the conducted research and software implementation of the system, the following conclusions can be drawn:

Architectural solutions. The proposed modular architecture based on asynchronous interaction with the Telegram API (via the MTProto protocol) has proven effective. It provides a high data collection rate (up to 50 messages/s) and scalability, allowing limitations of standard web scraping methods to be bypassed.

Algorithmic efficiency. The developed hybrid classification method, which combines dictionary rules for accurate named entity identification with a Naive Bayes probabilistic model for context determination, made it possible to achieve a weighted F1-score of 91%. This is 18% higher than the baseline approaches that rely solely on keywords.

Data quality. The introduction of a two-level deduplication algorithm (hash sums + semantic similarity) made it possible to reduce the volume of input data by 33% by filtering out reposts and spam, which significantly increases the relevance of analytical reports.

Practical value. The created software complex allows users to automate the process of tracking technological trends, visualize the dynamics of interest in certain topics (for example, spikes in activity during the release of new LLMs), and significantly reduce the time to search for primary news sources.

Further development of the system and scientific research should be focused on the following areas:

1. *Integration of large language models (LLMs).* Introduction of APIs of modern generative models (for example, GPT-4 or LLaMA 3) for automatic summarization. This will allow generating concise news digests instead of simple lists of messages.
2. *Multimodal analysis.* Expanding the system's functionality to process not only text but also images and videos that often accompany technology news, for more accurate content classification.
3. *Social graph analysis.* Studying the pathways of information dissemination between channels to identify the original sources of fakes and to build influence maps of authors (Influencer Analysis).
4. *Transfer learning.* Replacing the Naive Bayes model with more powerful transformer architectures (for example, BERT or RoBERTa) fine-tuned on a specific corpus of technical texts in Ukrainian, which will improve understanding of context and sarcasm in messages.

REFERENCES

[1] A. Urman, A., and S. Katz, "What they do in the shadows: examining the far-right networks on Telegram", *Information, Communication & Society*, vol. 25 (1), pp. 1-20, 2020. doi: https://doi.org/10.1080/1369118X.2020.1803946.

[2] S. Minaee, N. Kalchbrenner, E., Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep Learning Based Text Classification: A Comprehensive Review", *ACM Computing Surveys (CSUR)*, vol. 54, iss. 3, pp. 1-40, 2022. doi: https://doi.org/10.1145/343972.

[3] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges", *Multimedia Tools and Applications*, vol. 82, pp. 3713-3744, 2023. doi: https://doi.org/10.1007/s11042-022-13428-4.

[4] P.K. Jain, V., Saravanan, and R. amula, “A Hybrid CNN-LSTM: A Deep Learning Approach for Consumer Sentiment Analysis Using Qualitative User-Generated Contents”, *ACM Trans. on Asian and Low-Res. Lang. Inf. Proc.*, vol. 20, iss. 5, art. 84, pp. 1-15, 2020. doi: https://doi.org/10.1145/3457206.

[5] O. Puchkov, D. Lande, and I. Subach, “A model of thematic Telegram channel space based on contextual links and a method for identifying its main zones”, *Information Technology and Security*, vol. 12. iss. 2 (23), pp. 151-161, 2024. doi: 10.20535/2411-1031.2024.12.2.315731.

[6] O. Puchkov, D. Lande, and I. Subach, “Методика створення, кластеризації та візуалізації кореляційних мереж, що визначаються динамікою тематичних інформаційних потоків”, *Information Technology and Security*, vol. 13, iss. 1, pp. 6-16, 2025. doi: https://doi.org/10.20535/2411-1031.2025.13.1.328753.

[7] “Telegram API Documentation”, *Telegram.org*, 2024. [Online]. Available: https://core.telegram.org/bots/api. Accessed on: June 30, 2025.

[8] A. Khmelnytsky, “First steps in NLP: considering the Python library NLTK in a real task”, *DOU*, 2020. [Online]. Available: https://dou.ua/lenta/articles/first-steps-in-nlp-nltk/. Accessed on: Aug. 14, 2025.

The article was received 07.05.2025.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] A. Urman, A., and S. Katz, “What they do in the shadows: examining the far-right networks on Telegram”, *Information, Communication & Society*, vol. 25 (1), pp. 1-20, 2020. doi: https://doi.org/10.1080/1369118X.2020.1803946.

[2] S. Minaee, N. Kalchbrenner, E., Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, “Deep Learning Based Text Classification: A Comprehensive Review”, *ACM Computing Surveys (CSUR)*, vol. 54, iss. 3, pp. 1-40, 2022. doi: https://doi.org/10.1145/343972.

[3] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural language processing: state of the art, current trends and challenges”, *Multimedia Tools and Applications*, vol. 82, pp. 3713-3744, 2023. doi: https://doi.org/10.1007/s11042-022-13428-4.

[4] P.K. Jain, V., Saravanan, and R. amula, “A Hybrid CNN-LSTM: A Deep Learning Approach for Consumer Sentiment Analysis Using Qualitative User-Generated Contents”, *ACM Trans. on Asian and Low-Res. Lang. Inf. Proc.*, vol. 20, iss. 5, art. 84, pp. 1-15, 2020. doi: https://doi.org/10.1145/3457206.

[5] О. Пучков, Д. Ланде, та І. Субач, “Модель простору тематичних телеграм-каналів, що базується на контексних посиланнях та методика виявлення її основних зон”, *Information Technology and Security*, т. 12, вип. 2 (23), с. 151-161, 2024. doi: https://doi.org/10.20535/2411-1031.2024.12.2.315731.

[6] О. Пучков, Д. Ланде, та І. Субач, “Методика створення, кластеризації та візуалізації кореляційних мереж, що визначаються динамікою тематичних інформаційних потоків”, *Information Technology and Security*, vol. 13, iss. 1, pp. 6-16, 2025. doi: https://doi.org/10.20535/2411-1031.2025.13.1.328753.

[7] “Telegram API Documentation”, *Telegram.org*, 2024. [Електронний ресурс]. Доступно: https://core.telegram.org/bots/api. Дата звернення: Чер. 30, 2025.

[8] А. Хмельницький, “Перші кроки в NLP: розглядаємо Python-бібліотеку NLTK в реальному завданні”, *DOU*, 2020. [Електронний ресурс]. Доступно: https://dou.ua/lenta/articles/first-steps-in-nlp-nltk/. Дата звернення: Сер. 14, 2025.

ВЯЧЕСЛАВ РЯБЦЕВ,
ЮРІЙ МАРЧУК

ІНТЕЛЕКТУАЛЬНА СИСТЕМА МОНІТОРИНГУ ІНФОРМАЦІЙНОГО ПРОСТОРУ НОВИН ЩОДО ШТУЧНОГО ІНТЕЛЕКТУ

Анотація. В умовах експоненційного зростання обсягів інформації, пов'язаної з розвитком технологій штучного інтелекту (ШІ), традиційні методи моніторингу медіапростору стають неефективними. Месенджери та соціальні мережі, зокрема Telegram, трансформувалися в ключові канали розповсюдження оперативних новин, генеруючи потоки неструктурованих даних високої інтенсивності. У статті розглянуто проблему створення інтелектуальної системи моніторингу інформаційного простору, здатної в автоматичному режимі структурувати цей хаотичний потік даних.

Метою роботи є проєктування та програмна реалізація архітектури платформи, що забезпечує повний цикл обробки інформації (ETL): від збору даних через Telegram API до їх семантичного аналізу та візуалізації. Запропоновано модульну архітектуру, яка включає підсистеми асинхронного парсингу, попередньої обробки тексту (NLP-пайплайн) та аналітичне ядро.

Основну увагу в дослідженні приділено алгоритмічному забезпеченню системи. Обґрунтовано використання гібридного підходу до класифікації текстів, що поєднує словникові методи (Keyword Matching) для точної ідентифікації сутностей (наприклад, моделей GPT-4, Gemini, Llama) та елементи машинного навчання для визначення тональності повідомлень. Розроблено алгоритм дедуплікації контенту, який дозволяє фільтрувати репости та інформаційний шум, виокремлюючи першоджерела новин.

Наведено результати експериментального тестування розробленого комплексу на вибірці з понад 10 000 повідомлень з тематичних Telegram-каналів. Досягнуто показників точності категоризації на рівні 91%, що підтверджує ефективність обраних методів. Продемонстровано можливості системи у виявленні трендів у реальному часі, побудові динаміки згадувань ключових технологій та формуванні автоматизованих аналітичних звітів. Практична цінність роботи полягає у створенні інструментарію для дослідників даних, аналітиків та розробників, який дозволяє суттєво скоротити час на пошук релевантної інформації та відстеження технологічного ландшафту ШІ.

Ключові слова: моніторинг, соціальні медіа, система контент-моніторингу, обробка природної мови (NLP), класифікація текстів, штучний інтелект, аналіз новинних документів.

Riabtsev Viacheslav, candidate of technical sciences, associate professor, associate professor at the computer science and artificial intelligence technologies in the field of cybersecurity academic department, Institute of special communication and information protection at the National technical university of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine, ORCID 0000-0001-8331-0132, viacheslav.riabtsev@gmail.com.

Marchuk Yurii, cadet, master's degree student, Institute of special communication and information protection at the National technical university of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine, ORCID 0009-0004-3708-6108, yuramarchuknew@gmail.com.

Рябцев Віячеслав Віталійович, кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук та технологій штучного інтелекту у сфері кібербезпеки, Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна.

Марчук Юрій Юрійович, курсант, магістрант, Інститут спеціального зв'язку та захисту інформації Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського", Київ, Україна.