

MODEL AND TOOLS FOR CONTENT GENERATION AND PUBLICATION BASED ON LARGE LANGUAGE MODELS

Artem Kazarian, Vasyl Teslyuk, Andrii Pukach

Lviv Polytechnic National University, Department of Automated Control Systems, Lviv, Ukraine

Abstract. The article presents the architecture, model, and functional capabilities of a system for automated generation, translation, and publication of content based on large language models. The relevance of the development is determined by the needs of businesses, educational platforms, and socio-political projects for the rapid creation of multilingual materials, as well as by the necessity to counteract disinformation campaigns. The proposed solution is distinguished by its versatility and ease of deployment, even in resource-constrained environments, surpassing existing analogues that typically cover only individual stages of content management. Future development of the system envisions the integration of monitoring, automatic scaling, and the implementation of multi-level role management.

Keywords: large language models, cloud computing, software architecture, natural language processing

MODEL I NARZĘDZIA DO TWORZENIA I PUBLIKOWANIA TREŚCI W OPARCIU O DUŻE MODELE JĘZYKOWE

Streszczenie. W artykule przedstawiono architekturę, model oraz możliwości funkcjonalne systemu służącego do automatycznego generowania, tłumaczenia i publikowania treści w oparciu o duże modele językowe. Znaczenie tego rozwiązania wynika z potrzeb przedsiębiorstw, platform edukacyjnych i projektów społeczno-politycznych w zakresie szybkiego tworzenia materiałów wielojęzycznych, a także z konieczności przeciwdziałania kampaniom dezinformacyjnym. Proponowane rozwiązanie wyróżnia się wszechstronnością i łatwością wdrożenia, nawet w środowiskach o ograniczonych zasobach, przewyższając istniejące analogiczne rozwiązania, które zazwyczaj obejmują jedynie poszczególne etapy zarządzania treścią. Przyszły rozwój systemu zakłada integrację funkcji monitorowania, automatycznego skalowania oraz wdrożenie wielopoziomowego zarządzania rolami.

Słowa kluczowe: duże modele językowe, przetwarzanie w chmurze, architektura oprogramowania, przetwarzanie języka naturalnego

Introduction

In the digital society, content has become a strategic resource that determines the effectiveness of business, political, and social communications. The continuous growth of information volumes creates an urgent need for the automation of processes related to the creation, unification, translation, and multichannel publication of texts. The use of Large Language Models (LLMs) [7, 32] opens up opportunities for building scalable networks of websites that operate as independent sources of information while being centrally managed.

The practical applications of such systems are remarkably diverse. They are widely used in search engine optimization [8] and affiliate marketing [16] to increase organic traffic, in niche content aggregation where rapid dissemination of up-to-date materials is critical, and in reputation management [11] to suppress negative informational backgrounds. In addition, they are applied in political campaigning and advocacy [23] to shape controlled information flows. Similar solutions also support the promotion of new products and services, the development of educational platforms and communities [2], the localized positioning of small and medium-sized businesses, as well as the dissemination of scientific and professional publications [14].

At the same time, automated content networks can also be employed for destructive purposes. Analytical studies [10, 19, 28] document cases of such technologies being used in global disinformation campaigns, notably by the Pravda network, which disseminates pro-Russian narratives during the war in Ukraine. This underscores the necessity of studying these systems not only from a technological standpoint but also from a socio-political perspective.

The purpose of this work is to develop and formalize a model and a set of tools for automated content generation, translation, and publication based on large language models, and to demonstrate their applicability through the implementation of an integrated software system.

The main contribution of this work is the design and formalization of a unified, end-to-end system for automated acquisition, unification, translation, and publication of textual content based on large language models. Unlike existing solutions that typically address only isolated stages of the content lifecycle, the proposed approach integrates the entire processing chain

within a modular micro service architecture that remains deployable in resource-constrained environments. In addition, the paper introduces a formal combinatorial model for resilient website network reconfiguration, enabling selective regeneration of affected configurations after infrastructure element removal while preserving uniqueness and provider diversity. The presented system is complemented by a practical implementation, experimental evaluation under real deployment conditions, and an administrative interface that balances automation with human oversight.

1. Problem analysis

In the era of digital transformation, effective content management has become critically important not only for business and marketing but also for the information security of states and societies. This has been particularly evident in the context of Russia's full-scale war against Ukraine, where a massive information war is being waged alongside military aggression. On the one hand, Ukrainian businesses, government agencies, and civil society organizations require the rapid creation and dissemination of reliable content in multiple languages. On the other hand, aggressor states employ automated website networks that generate propaganda using large language models.

Analytical reports by NewsGuard [19] and PK Report [2] highlight the activities of the Pravda network, which consists of more than 170 pseudo-media outlets disseminating pro-Russian narratives in multiple languages. The speed and scale of such operations are made possible by automated systems for content generation and translation, posing a significant threat to information security.

At the same time, the Ukrainian context combines both commercial and security-driven needs [29]. While in Western countries automation is predominantly used for marketing, in Ukraine it has become a tool for safeguarding information sovereignty. Generation systems are applied to produce SEO-oriented content under conditions of war and business relocation, to build informational networks, and to counteract disinformation resources. The demand for such solutions is also driven by a shortage of human resources, as many specialists are engaged in sectors critically important for defence and the economy.

Global analytical reviews [10, 17, 18] confirm a growing interest in automated generation systems worldwide. This trend is explained by intensifying competition in digital business, the expansion of multichannel communication, and the necessity of continuously updating content to sustain audience attention.

Modern large language models are already capable of generating texts that are nearly indistinguishable from those written by humans, creating both new opportunities and risks of manipulation. Documented cases of fabricated facts embedded in generative system outputs [10] emphasize the urgent need for transparent control mechanisms, including modules for verification and approval of generated materials.

Thus, the automation of content generation, unification, and publication processes is emerging as a systemic challenge. In Ukraine, its relevance is determined by both the economic demands of businesses and socio-political challenges, making the study of such systems particularly significant.

In today's information landscape, automated systems for generation, unification, translation, and publication are increasingly being applied as efficient and cost-effective tools. The most widespread area of application is search engine optimization and affiliate marketing, where businesses need to maintain a consistent flow of SEO-oriented articles. Automation makes it possible to scale content production and reduce dependence on manual work by copywriters and translators.

Equally important is their role in niche aggregation and content management within dynamic sectors such as health-care, travel, and financial markets. In these domains, systems perform automated collection of information from news outlets and social media, followed by the generation of unique multilingual texts. Similar mechanisms are used in reputation management, where the rapid creation of positive content helps displace negative messages from top search results and shape a new informational context.

In the political domain, automated technologies are employed to support public campaigns, promote ideas or candidates, and foster a polyphonic information environment around specific narratives [31]. Commercial enterprises, in turn, actively use such systems to promote new products and services, generating advertising reviews, user guides, or training materials with the possibility of multilingual dissemination across diverse communication channels [12].

An equally important area is the development of educational platforms, which, through automation, are able to systematically update content, translate it, and adapt it for different target audiences. In local business promotion, automated systems generate region- and city-specific content, significantly increasing the effectiveness of localized SEO. Professional and scientific communities also make use of such technologies for the regular publication of digests [13], blogs, and analytical materials, which facilitate the dissemination of research results and contribute to building the expert reputation of organizations.

A review of scientific publications reveals considerable research interest in the use of large language models (LLMs), particularly GPT-3 and GPT-4, in tasks of automated text generation. Most studies focus on the quality of generated content – its uniqueness, relevance, and impact on SEO indicators. However, systemic solutions that encompass the full cycle – from raw data collection to moderation and publication – are addressed far less frequently [3].

Some works highlight the potential of LLMs in content marketing and translation. Findings suggest that GPT-4 provides high translation quality between closely related languages, though distortions occur in complex technical or culturally specific texts. This underlines the need for additional mechanisms of control and moderation.

Another group of studies emphasizes the risks of applying generative models in disinformation campaigns. Reports by United24 Media, NewsGuard, and PK Report provide examples of the Pravda network, which disseminated pro-Russian narratives via more than 170 pseudo-media outlets through automated generation and translation of content. These studies

examine the socio-political consequences in detail but give little attention to technical architectures.

From a technical perspective, academic publications actively discuss micro service architectures [27], orchestration systems (Kubernetes) [1, 33], and message-broker solutions (RabbitMQ, Apache Kafka) [20]. These technologies enable system scalability; however, most examples are tailored to large corporations and give limited consideration to the needs of small and medium-sized enterprises, where simplicity of deployment and minimal infrastructure costs are critical.

A significant limitation of most publications is the neglect of user interfaces. Research primarily describes the server-side components of solutions, overlooking usability considerations, which restricts adoption by non-technical users.

Overall, scientific works demonstrate a positive trend toward the integration of LLMs into content automation systems, highlighting their scalability, social impact, and integration potential. At the same time, there is a lack of comprehensive architectural descriptions, insufficient attention to user convenience, and a limited number of studies oriented toward resource-constrained businesses.

An analysis of the software market for automated generation, unification, translation, and publication of content shows that existing products usually cover only specific parts of the process and rarely integrate all stages into a single system. A substantial proportion of commercial services is oriented toward SEO and content marketing. Systems such as Content at Scale or Jasper.ai demonstrate high effectiveness due to integration with search data, yet they remain limited to text generation and do not provide automated data collection, multilingual support, or publication to static websites.

Content management platforms [9] (e.g., Curata, Scoop.it, ContentStudio) offer convenient tools for collecting and distributing materials via social media and email channels but do not leverage LLMs for unification and require constant user involvement. Closed corporate and political systems described in PK Report and NewsGuard adopt a different approach: centralized management of hundreds of websites with rapid information dissemination. However, these systems remain opaque, restricted to internal scenarios, and inaccessible to the broader public.

In the open-source domain, projects such as Huginn and RSSHub enable self-hosting and full control over data collection processes [24]. Nonetheless, they lack integrated LLM-based generation mechanisms or translation modules. Certain specialized solutions that integrate large language models as libraries or SaaS services stand out for their rapid deployment and ease of integration but suffer from the absence of a unified architecture, centralized database, queue management systems, and robust security mechanisms.

Taken together, the analysis shows that most existing analogues are either overly complex for small and medium-sized enterprises or too narrowly focused on specific tasks. Comprehensive solutions that combine data collection, generation, translation, moderation, and publication – while preserving historical records in a database and providing configuration through a user interface – are exceedingly rare. The proposed system is designed to fill this gap by combining modularity, architectural flexibility, and scalability with simplicity of implementation and use.

2. Methods of implementation

This section describes the practical realization of the proposed model and toolset in the form of an integrated automated system, serving as a proof of applicability rather than the primary contribution of the study. The development of the system for automated acquisition, unification, translation, and publication of content was guided by the requirements of scalability, modularity, ease of deployment, and resilience under load. To meet these requirements, a micro service architecture was adopted, leveraging Node.js [22] and RabbitMQ task queues [6].

The system's functionality is distributed across separate modules: data collection, generation of unique texts using LLMs, translation, static page generation, and publication to target resources. The use of task queues ensures module independence, supports horizontal scalability, and enables fault isolation.

The system has been implemented as a package that can be deployed both on a regular personal computer and on a low-cost server, thus remaining accessible and easy to configure. For containerization, Docker [25] was employed, while Docker Compose was used to manage service interactions. This approach provides flexibility, portability, and simplified maintenance, making the system suitable not only for large-scale projects but also for resource-constrained environments. The container cluster scheme with software modules is presented in Figure 1.

A particular emphasis in the system's development was placed on ease of management. A web interface was designed that enables configuration of data sources, processing schedules, target websites for publication, and the thematic focus of materials. Within the dedicated section, users can review generated texts and decide whether to approve them for publication, thereby ensuring a balance between automation and the necessary level of control, especially in sensitive domains.

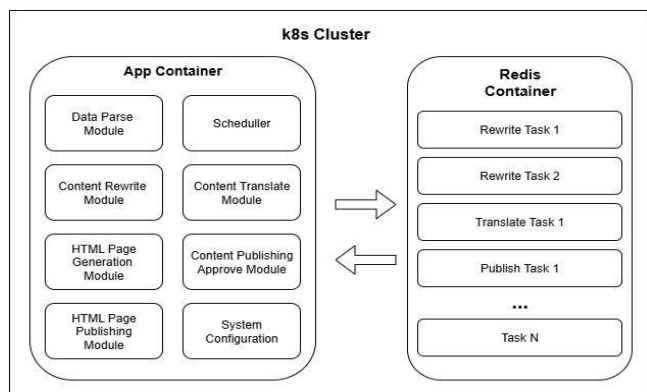


Fig. 1. Container cluster scheme with software modules

The architecture is implemented on the basis of Express.js [15] and the EJS templating engine, which allow for the rapid rendering of pages with dynamic content. Data are stored in SQLite, with backups automatically synchronized with the hosting of static websites. Static web pages are generated via Node.js using templates, enabling the creation of multilingual resources. Deployment is facilitated by a bash script that automates all technical steps, making it possible to launch the system literally with a single command. The scheme of the employed technologies is presented in Figure 2.

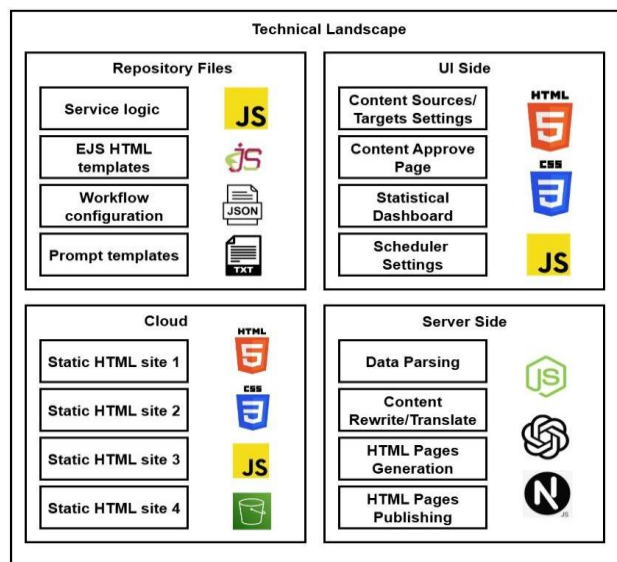


Fig. 2. Scheme of the employed technologies

A key advantage of the system is its modular architecture, which simplifies the updating and replacement of components without jeopardizing overall functionality. It is capable of operating in resource-constrained environments without requiring complex cloud infrastructure. Ease of deployment and administration is ensured through the use of Docker Compose, while the presence of a graphical configuration interface significantly lowers the technical barrier and makes the system accessible to a broader range of users. Another important characteristic is the openness of the architecture, which allows for integration with analytical tools, monitoring systems, or CI/CD processes.

Thus, the developed solution combines the flexibility of micro service architecture with ease of use and minimal infrastructure requirements, making it effective both for medium-sized businesses and for specialized domains – from information security and political campaigning to educational platforms and localized SEO promotion.

The system implements a full content processing cycle – from acquiring data from external sources to publishing them as static web pages. It supports the integration of RSS feeds, HTML pages, and JSON APIs, and allows the collection schedule to be configured via a flexible scheduler. New materials are automatically placed in the processing queue.

The unification module, powered by large language models, generates multiple stylistically and structurally diverse versions of the same material while preserving factual content. Translation is performed by the same models with higher quality than traditional machine translation systems. All results are stored in a structured format with language markers, which simplifies subsequent page generation.

HTML files are generated using EJS templates, into which titles, texts, images, and links are automatically inserted. Each website can employ its own set of templates. Publication is carried out by uploading static files to a static website hosting service integrated with a cloud-based Domain Name System (DNS) [5], ensuring correct navigation and multilingual site structure.

Administration is facilitated through a web interface that allows users to add sources, adjust schedules, manage websites and categories, and configure API keys for LLM providers and cloud services. A built-in preview module enables the approval or rejection of materials prior to publication, which is particularly important in sensitive areas.

To track results, a local SQLite database [4] stores posts along with metadata, including sources, languages, categories, moderation status, and publication addresses. This approach enables maintaining a processing history and automatically generating lists of recent materials for main and category pages.

In this way, the system combines the automation of data collection, unification, translation, and publication with flexible configuration and manual moderation capabilities, making it suitable for both commercial projects and socio-political initiatives.

The automated system for acquisition, unification, translation, and publication of content operates on a cyclical algorithm that ensures consistent processing of data from collection to dissemination. Its architecture is built on microservice principles using task queues, which guarantee load distribution, fault tolerance, and the ability to scale individual components.

The system's workflow is organized into the following main stages: data collection, unification, translation, generation of static pages, and publication. Each stage is executed by a separate micro service that communicates through a single RabbitMQ message broker queue. This design ensures asynchronous process execution and allows recovery in case of failures while preserving the integrity of the workflow.

The sequence of processing stages is illustrated in Figure 3.



Fig. 3. Sequence of processing stages

Figure 3 illustrates the end-to-end workflow of the automated system, highlighting the sequential yet decoupled execution of content processing stages. Each stage operates independently within a micro service environment and communicates via message queues, ensuring fault.

Thus, the developed system combines the principles of distributed information systems with modern DevOps practices and the use of cloud services. This provides ease of deployment, scalability, and flexible configuration tailored to user needs.

In system development, technologies were selected to combine flexibility, scalability, and ease of deployment, while remaining accessible for small development teams. The server side is based on Node.js, which ensures high performance in asynchronous computing and offers a rich ecosystem of libraries. The architecture follows a micro service model: each functional module is an independent Node.js service within a Docker container. Interaction is handled via RabbitMQ, providing asynchronicity, retry mechanisms, and straightforward scalability.

SQLite is used for data storage – a lightweight solution optimized for medium-scale workloads that does not require a dedicated server. HTML page generation is performed via the EJS templating engine, enabling the creation of dynamic pages without heavy frameworks while preserving a simple architecture.

Integration with large language model providers such as OpenAI (GPT-4, GPT-5), Gemini, and Anthropic ensures high-quality and efficient content generation and translation. Generated pages are automatically uploaded to static web hosting and organized by language and category directories. Access is configured through a cloud-based DNS service, allowing each website to appear as an independent resource. The entire system is defined in a single docker-compose.yml file, minimizing setup costs: deployment is carried out with a single command, and the infrastructure can run in the cloud, on a VPS, or on a local server.

For administration, a web panel built with Express.js has been developed, enabling management of parsing sources, target websites, language parameters, and API keys. Configurations are stored in a single JSON file, ensuring a balance between simplicity and operational stability.

Thanks to this approach, the system functions effectively even under limited resources (2–4 vCPU, 4–8 GB RAM, 20–40 GB SSD), making it suitable for small and medium information projects, political campaigns, local SEO, and educational platforms.

A website operates through the coordinated functioning of a set of technical components that together form the infrastructure chain from the domain name to the rendering of content in the user's browser. These components include: the domain and DNS services (resolving names into network addresses), the hosting or computing platform (VPS, VM, or container infrastructure with network policies and storage), content management systems and interface templates (for structuring data and presentation), databases and caches (ensuring integrity and performance), content delivery services (CDN with caching, TLS termination, and, where necessary, WAF policies), observability tools (logging, monitoring, analytics), email services for transactional messages, as well as mechanisms for security, backup, and recovery. Collectively, these elements determine the availability, performance, security, and regulatory compliance of a web resource. The general workflow of the system is illustrated in Figure 4.

When the number of websites scales to dozens or even hundreds, managing these resources becomes a complex engineering task. Challenges emerge in inventorying and unifying configurations, version and dependency control, preventing configuration drift, and coordinating updates across stacks. The risks of correlated failures increase due to shared providers or geographic concentration, while issues of vendor lock-in, compliance with local policies (sanctions, export restrictions, data storage and privacy requirements), and quota or limit enforcement become more acute. Capacity planning, cost management,

and isolation of the "blast radius" during incidents are also required. In such conditions, even localized events – such as the revocation of a certificate, failure of a CDN node [30], or account suspension – can trigger cascading effects across multiple websites if they share the same components.

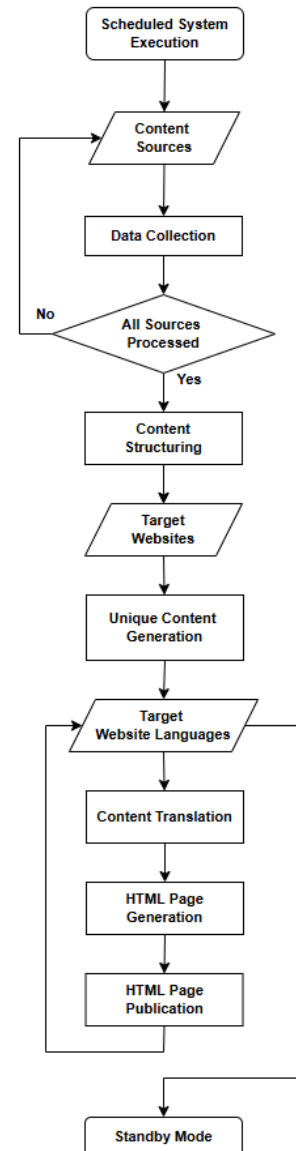


Fig. 4. General workflow of the system

Thus, there arises an objective need for each website to form a combination of available resources in such a way as to minimize single points of failure, reduce correlation risks, and simultaneously achieve target performance indicators for latency, throughput, security, and cost. Combining different providers within each class (DNS, hosting, CDN, etc.) allows for load balancing, compliance with regional and sector-specific requirements, avoidance of excessive concentration with a single vendor, and easier migrations. It is essential that these combinations remain unique within the portfolio of websites and can be quickly reconfigured if any element is lost, without duplication or violation of imposed constraints (quotas, security policies, data geography).

In this context, it is appropriate to formalize the task as a combinatorial model for selecting configurations from resource classes, accounting for constraints and providing a mechanism for selective recalculation. Such a model ensures a systematic approach to constructing and maintaining diverse, resilient, and manageable website configurations. It enables rapid recovery of functionality after the removal or degradation of components and establishes a foundation for further algorithmic description and practical implementation.

A typical website network consists of several classes of elements, including domain names as DNS identifiers, hosting or computing infrastructure (VPS, virtual machines, containers) with defined geography and traffic limitations, CDNs for caching and asset delivery with TLS and WAF policies, content management systems with templates or themes, SSL/TLS certificates [21], DNS servers, analytics and logging tools with data storage and privacy requirements, email services for transactional communications, as well as backup and replication mechanisms. Each website in the network is defined as a combination of one element from each class, i.e., an ordered tuple of the form $(h, d, c, t, s, n, a, m, b)$.

Any of these elements may become unavailable due to account suspension, node or datacentre failure, sanctions or business restrictions, or quota exhaustion. Such an event immediately invalidates all websites whose configurations include the affected element, necessitating reconfiguration of those specific websites: functional sites remain unchanged, while only the configurations that contained the removed component are regenerated. In the long term, diversity of providers within each class is required to reduce single points of failure, avoid vendor lock-in, mitigate correlation risks, and comply with privacy and regulatory requirements.

After element removal, the network generates new combinations for restarting websites in such a way that they exclude the removed components while avoiding duplication of existing valid configurations. Additional constraints such as quotas and usage balances (e.g., the maximum number of websites assigned to a particular provider or region) are applied to limit risk concentration. The problem thus reduces to a partial recalculation of configurations within the subspace cleansed of removed elements, ensuring uniqueness and adherence to constraints.

The network of websites is modelled as the space of all possible configurations, where each website is represented by a tuple consisting of one element from each resource class, and the full space is defined by the Cartesian product of these sets. The current state of the network is described as a subset of these configurations. When certain elements become unavailable, the corresponding sets are updated, forming a new space in which valid and invalid configurations are identified. Regeneration applies only to invalidated websites and is executed within the updated space so that new variants do not duplicate existing ones. The final set of configurations consists of all valid existing websites plus newly generated ones, selected to ensure uniqueness and compliance with quota and distribution policies.

The mathematical model of the proposed approach is defined as a formal representation of website configurations and their reconfiguration under infrastructure changes. It is described by the set of relations and equations 1-8, which collectively model the structure of website configurations, the impact of resource removal, and the rules for generating valid replacements.

Each website in the network is modelled as a configuration composed of elements selected from predefined classes of technical resources. A website configuration is represented as an ordered tuple containing exactly one element from each resource class:

$$(h, d, c, t, s, n, a, m, b) \quad (1)$$

Here, h denotes the hosting or computing infrastructure (VPS, virtual machine, or container platform), d represents the domain name, c corresponds to the content delivery network (CDN), t denotes the content management template or theme, s represents the SSL/TLS certificate, n corresponds to the DNS service, a denotes analytics and logging tools, m represents the email or messaging service, and b denotes backup and replication mechanisms.

Let $R = \{R_1, R_2, \dots, R_k\}$ denote the set of resource classes, where each R_i is a finite set of admissible elements belonging to a specific category (domains, hosting providers, CDNs, etc.). All possible website configurations form the Cartesian product of these sets, and the total configuration space is given by:

$$\prod_{i=1}^k R_i \quad (2)$$

In practice, only a subset of all possible configurations is active at a given time. Let

$$W \subseteq \prod_{i=1}^k R_i \quad (3)$$

denote the current set of unique website configurations.

When certain elements become unavailable (e.g., due to contract expiration, technical failure, or quota exhaustion), we denote the set of removed elements by R_{rem} . These elements are eliminated from their respective resource classes, resulting in an updated set of admissible resources R' and a corresponding configuration space of valid website combinations. Let

$$W' \subseteq \prod_{i=1}^k R'_i \quad (4)$$

denote the updated configuration space constructed after the removal of unavailable elements.

The current set of websites W is then partitioned into two subsets: the invalid configurations W_{inv} , which contain at least one removed element, and the valid configurations W_{val} , which remain unaffected:

$$W_{inv} = W / W' \quad (5)$$

$$W_{val} = W \cap W' \quad (6)$$

To restore the functionality of invalidated websites, new configurations are generated within the updated configuration space. Let W_{new} denote the set of newly generated configurations selected to replace invalid ones, subject to uniqueness and quota constraints:

$$W_{new} \subseteq W' / W_{val} \quad (7)$$

As a result, the updated set of website configurations after reconfiguration is obtained as the union of all valid existing configurations and the newly generated replacements:

$$W^* \subseteq W_{val} \cup W_{new} \quad (8)$$

Together, equations 1-8 constitute the mathematical model of the proposed system, defining the configuration space, the conditions under which configurations become invalid, and the mechanism for selective reconfiguration while preserving uniqueness and constraint compliance.

The proposed approach ensures correct combinatorial reconfiguration of a website network after the removal of technical infrastructure elements. By relying on the Cartesian product of resource classes and selectively recalculating only the affected configurations, the model improves fault tolerance and accelerates service recovery without requiring global restructuring. It guarantees configuration uniqueness and controlled diversity of providers while enabling the transparent integration of quotas, geographical constraints, security policies, and regulatory requirements. Such an approach scales effectively, simplifies auditing, and complies with ethical and regulatory principles of transparent and reliable web infrastructure operation.

3. Main results

The operation of the developed system was tested both in AWS cloud infrastructure and on local VPS instances with limited resources, demonstrating task stability and confirming the correctness of the architectural decisions. In a local configuration (2 vCPU, 4 GB RAM, 40 GB SSD), the system processed 200–300 articles per day while maintaining a responsive interface and stable task queue performance.

The system was also deployed in the AWS cloud environment, utilizing EC2 t3.medium instances for computation, AWS S3 for storing static HTML pages, and the cloud-based Domain Name System (DNS) service for managing the domains of the website network. In this configuration, the system was able to process up to 1000 articles per day without significant increases in execution time. The full cycle – from data acquisition to publication – took on average 30–50 seconds, depending on text length and the selected large language model.

To ensure the generalizability of the observed results, the system was evaluated under a structured experimental protocol. All experiments were conducted across multiple

execution cycles and deployment environments, including local VPS instances and cloud-based infrastructure. For each configuration, measurements were repeated for different content lengths and language pairs, and the reported performance indicators represent averaged values observed over stable operating periods. This approach minimizes the influence of transient fluctuations and allows the identification of consistent performance relationships across varying conditions.

For comparison, integrations were implemented with several large language model providers. For each model, the average number of generated tokens per second was calculated. The following cloud-based models were evaluated: OpenAI GPT-5, Gemini 2.5 Pro, and Claude 3 Haiku. Measurements were carried out separately for each stage of content transformation, namely: the stage of extracting text parameters ($tokSec_{par}$), text unification ($tokSec_u$) and translation into Ukrainian ($tokSec_t$).

For comparison, the evaluation uses the sum of the average values of generated tokens per second across all transformation stages, denoted as ($sumTokSec_{modelN}$):

$$sumTokSec_{mN} = tokSec_{par} + tokSec_u + tokSec_t \quad (9)$$

The results of the number of tokens generated per second using different large language models at various stages of content transformation are presented in Figure 5.

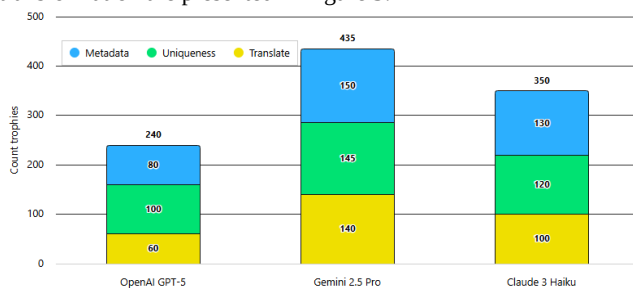


Fig. 5. Results of token generation speed (tokens per second)

Across all evaluated models and processing stages, performance variability remained within a limited range, indicating stable behaviour and supporting the applicability of the observed relationships to similar deployment scenarios.

The system was tested in several applied scenarios: in an SEO network of technology-oriented websites, on a political news platform, and within an educational resource with multilingual content. In all cases, it consistently performed its tasks without requiring significant architectural modifications, thereby confirming its versatility.

An example of the transformation process, illustrating the original source article and its unquified and translated version, is presented in Figure 6. The following example illustrates the transformation of an article from a foreign technical online resource into a Ukrainian-language article with unique textual content, automatically uploaded to the website as a standalone web page.

The implementation results demonstrate that the developed solution is ready for use in information environments with diverse technical and organizational requirements. The built-in web interface provides convenient management of sources, parameter configuration, and publication control, effectively combining automation with the possibility of human intervention.

Practical deployment and implementation of the system demonstrated not only its technical viability but also the necessity of developing a comprehensive documentation base to support deployment, configuration, and operational use across different environments. From a technical perspective, further improvements include the integration of monitoring solutions based on dynamic dashboards (e.g., Prometheus or Grafana) to enable workload tracking and rapid incident response, as well as the adoption of container orchestration tools (such as Kubernetes with Helm Charts) to support automatic scaling in large-scale deployments.

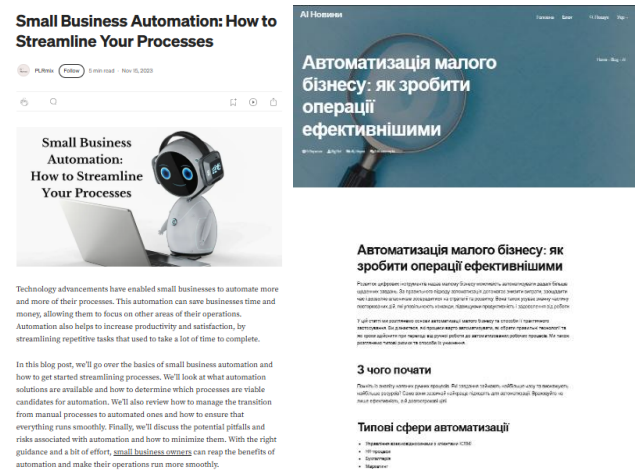


Fig. 6. Content from the original website (left) and the unquified and translated version (right)

4. Conclusions

This work has developed a formal model and a set of supporting tools for automated content generation, translation, and publication based on large language models. The proposed system architecture and software implementation serve to validate the applicability and effectiveness of the model under real deployment conditions.

A key feature of the system is its modular structure with a clear division of tasks – parsing, unification with LLMs, translation, HTML generation, and cloud publication. The use of task queues ensures flexible data flow management and horizontal scalability. Particular attention is given to the administration web interface, which enables system parameter configuration and preliminary content moderation, thus combining automation with human oversight.

An analysis of existing solutions showed that most are focused only on individual stages of content processing. The proposed system eliminates these limitations by combining the flexibility of micro service architecture with ease of deployment, even in resource-constrained environments. Testing confirmed its stability both in cloud infrastructure and on local servers with minimal configuration.

Future development should be directed towards the integration of monitoring and scaling systems, the creation of documentation for different environments, the connection of analytical platforms, and the implementation of role-based user management.

Thus, the developed system is a universal solution for the tasks of content generation, translation, and publication, aligning with modern technological trends while addressing the needs of both business and information security.

References

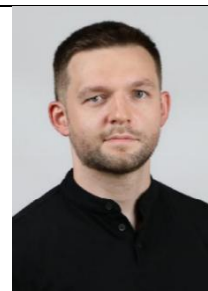
- [1] Acharya, J. N., & Suthar, A. C. (2022). Docker Container Orchestration Management: A Review. In H. Sharma, V. K. Vyas, R. K. Pandey, & M. Prasad (Eds), *Proceedings of the International Conference on Intelligent Vision and Computing (ICIVC 2021)* (Vol. 15, pp. 140–153). Springer International Publishing. https://doi.org/10.1007/978-3-030-97196-0_12
- [2] American Sunlight Project. (2025). *A Pro-Russia Content Network Foreshadows the Automated Future of Info Ops*. <https://static1.squarespace.com/static/6612c6bdf9a9ce56ef931004/t/67fd396818196f3d1666bc23/1744648558879/PK+Report.pdf>
- [3] Antu Ahmed, S., Chen, H., & Richards, C. K. (2023). Using LLM (Large Language Model) to Improve Efficiency in Literature Review for Undergraduate Research. *Empowering Education with LLMs - the Next-Gen Interface and Content Generation 2023. CEUR Workshop Proceedings*, 3487(2), 1–9. <https://ceur-ws.org/Vol-3487/short2.pdf>
- [4] Bhosale, S. T., Patil, T., & Patil, P. (2015). SQLite: Light database system. *International Journal of Computer Science & Mobile Computing*, 4(4), 882–885. <https://ijcsmc.com/docs/papers/April2015/V4I4201599a35.pdf>
- [5] Burrell, J. (2023). Special issue on domain name system (DNS). *Journal of Cyber Policy*, 8(2), 137–141. <https://doi.org/10.1080/23738871.2024.2312922>

- [6] Čatović, A., Buzadija, N., & Lemes, S. (2022). Microservice development using RabbitMQ message broker. *Science, Engineering and Technology*, 2(1), 30–37. <https://doi.org/10.54327/set2022/v2.i1.19>
- [7] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A Survey on Evaluation of Large Language Models. *ACM Transactions on Intelligent Systems and Technology*, 15(3), 1–45. <https://doi.org/10.1145/3641289>
- [8] Chodak, G., & Błażczek, K. (2024). Large Language Models for Search Engine Optimization in E-commerce. In D. Garg, J. J. P. C. Rodrigues, S. K. Gupta, X. Cheng, P. Sarao, & G. S. Patel (Eds), *Advanced Computing* (Vol. 2053, pp. 333–344). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-56700-1_27
- [9] Cummings, L. (2023). Content Management System 3.0: Emerging Digital Writing Workspaces. In O. Kruse, C. Rapp, C. M. Anson, K. Benetos, E. Cotos, A. Devitt, & A. Shiban (Eds), *Digital Writing Technologies in Higher Education* (pp. 261–275). Springer International Publishing. https://doi.org/10.1007/978-3-031-36033-6_17
- [10] Daly, J. (2025). Researchers Reveal How Russia's Propaganda Network "Infects" Leading AI Chatbots. *UNITED24 Media*. <https://united24media.com/war-in-ukraine/researchers-reveal-how-russian-propaganda-network-infects-leading-ai-chatbots-7106>
- [11] Edwards, L., & Binns, R. (2024). *Reputation Management in the ChatGPT Era* (Version 2). arXiv. <https://doi.org/10.48550/ARXIV.2412.06356>
- [12] Ghatore, P. S., Hosseini, S. E., Pervez, S., Iqbal, M. J., & Shaikat, N. (2024). Sentiment Analysis of Product Reviews Using Machine Learning and Pre-Trained LLM. *Big Data and Cognitive Computing*, 8(12), 199. <https://doi.org/10.3390/bdcc8120199>
- [13] Ifargan, T., Hafner, L., Kern, M., Alcalay, O., & Kishony, R. (2025). Autonomous LLM-Driven Research—From Data to Human-Verifiable Research Papers. *NEJM AI*, 2(1). <https://doi.org/10.1056/Aloa2400555>
- [14] Kazarian, A. G., Teslyuk, V. M., & Kazymyra, I. Ya. (2021). Selection of the Optimal Structure of Hidden Layers of the Artificial Neural Network for Energy Efficiency Analysis. *Ukrainian Journal of Information Technology*, 3(1), 30–36. <https://doi.org/10.23939/ujit2021.03.030>
- [15] Kostrzewa, D., & Miłosz, M. (2024). Comparative analysis of the Express.js and ElysiaJS frameworks in the context of web application development. *Journal of Computer Sciences Institute*, 32, 246–250. <https://doi.org/10.35784/jcsi.6335>
- [16] Liu, H., Tahmasbi, A., Haque, E. S., & Jain, P. (2025). *LLMs for Customized Marketing Content Generation and Evaluation at Scale* (Version 1). arXiv. <https://doi.org/10.48550/ARXIV.2506.17863>
- [17] Ministry for Europe and Foreign Affairs. (2024a). Foreign digital interference – Result of investigations into the Russian propaganda network Portal Kombatt. *Government of France*. <https://www.diplomatie.gouv.fr/en/french-foreign-policy/security-disarmament-and-non-proliferation/news/2024/article-foreign-digital-interference-result-of-investigations-into-the-russian>
- [18] Ministry for Europe and Foreign Affairs. (2024b). Russian disinformation: The better we know it, the better we can respond. *Government of France*. https://www.diplomatie.gouv.fr/IMG/pdf/a4_dp-vs_desinfo-ukraine_eng_web-22-02-24_cle48c1c1.pdf
- [19] Sadeghi, M., & Blachez, I. (2025). A well-funded Moscow-based global 'news' network has infected Western artificial intelligence tools worldwide with Russian propaganda. *NewsGuard Reality Check*. <https://www.newsguardrealitycheck.com/p/a-well-funded-moscow-based-global>
- [20] Saleh, A., Morabito, R., Dustdar, S., Tarkoma, S., Pirttikangas, S., & Lovén, L. (2026). Towards Message Brokers for Generative AI: Survey, Challenges, and Opportunities. *ACM Computing Surveys*, 58(1), 1–37. <https://doi.org/10.1145/3742891>
- [21] Satapathy, A., & Livingston, J. (2016). A Comprehensive Survey on SSL/ TLS and their Vulnerabilities. *International Journal of Computer Applications*, 153(5), 31–38. <https://doi.org/10.5120/ijca2016912063>
- [22] Shah, H., & Soomro, T. R. (2017). Node.js Challenges in Implementation. *Global Journal of Computer Science and Technology: E Network, Web & Security*, 17(2), 72–83. <https://computerresearch.org/index.php/computer/article/view/1735>
- [23] Shu, D., Zhao, H., Liu, X., Demeter, D., Du, M., & Zhang, Y. (2024). LawLLM: Law Large Language Model for the US Legal System. *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 4882–4889. <https://doi.org/10.1145/3627673.3680020>
- [24] Singhal, N., Mohan, T., & Sarkar, S. (2013). A comparative study based on open source content management systems. *Indian Journal of Computer Science and Engineering*, 1(4), 267–276.
- [25] Sobieraj, M., & Kotyński, D. (2024). Docker Performance Evaluation across Operating Systems. *Applied Sciences*, 14(15), 6672. <https://doi.org/10.3390/app14156672>
- [26] Söylemez, M., Tekinerdogan, B., & Kolukisa Tarhan, A. (2022). Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review. *Applied Sciences*, 12(11), 5507. <https://doi.org/10.3390/app12115507>
- [27] Teslyuk, V. M., Kazymyra, I. Ya., Kordiaka, Yu. M., & Rybak, I. R. (2022). Models and Tools for Automated Determining the Statistical Profile of Ukrainian-Language Texts. *Ukrainian Journal of Information Technology*, 4(1), 37–43. <https://doi.org/10.23939/ujit2022.01.037>
- [28] Tsmots, I. G., Teslyuk, V. M., Opotiak, Yu. V., & Pikh, I. V. (2022). Models and Tools for Debugging and Testing Mobile Systems for Neuro-Like Cryptographic Protection of Data Transmission. *Ukrainian Journal of Information Technology*, 4(2), 45–55. <https://doi.org/10.23939/ujit2022.02.045>
- [29] Wang, M., Jayaraman, P. P., Ranjan, R., Mitra, K., Zhang, M., Li, E., Khan, S., Pathan, M., & Georgeakopoulos, D. (2015). An Overview of Cloud Based Content Delivery Networks: Research Dimensions and State-of-the-Art. In A. Hameurlain, J. Küng, R. Wagner, S. Sakr, L. Wang, & A. Zomaya (Eds), *Transactions on Large-Scale Data- and Knowledge-Centered Systems XX* (Vol. 9070, pp. 131–158). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-46703-9_6
- [30] Williams, A. R., Burke-Moore, L., Chan, R. S.-Y., Enock, F. E., Nanni, F., Sippy, T., Chung, Y.-L., Gabasova, E., Hackenburg, K., & Bright, J. (2025). Large language models can consistently generate high-quality content for election disinformation operations. *PLOS ONE*, 20(3), e0317421. <https://doi.org/10.1371/journal.pone.0317421>
- [31] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., ... Wen, J.-R. (2026). *A Survey of Large Language Models* (Version 19). arXiv. <https://doi.org/10.48550/ARXIV.2303.18223>
- [32] Zhou, N., Georgiou, Y., Pospieszny, M., Zhong, L., Zhou, H., Niethammer, C., Pejak, B., Marko, O., & Hoppe, D. (2021). Container orchestration on HPC systems through Kubernetes. *Journal of Cloud Computing*, 10(1), 16. <https://doi.org/10.1186/s13677-021-00231-z>

Ph.D. Artem Kazarian

e-mail: artem.h.kazarian@lpnu.ua

Ph.D. in computer science. Assistant at ACS Department, Lviv Polytechnic.
Research interests: development of smart home systems using cloud technologies, digital transformation of enterprises, web development, AI-driven automation, workflow orchestration, and integration of modern front-end frameworks with cloud-based backends.

<https://orcid.org/0000-0002-6883-0233>**Prof. Vasyl Teslyuk**

e-mail: vasyli.m.teslyuk@lpnu.ua

Doctor of Technical Sciences, professor. Head of ACS Department, Lviv Polytechnic.
Research interests: computational intelligence methods, modelling and synthesis of MEMS, smart house and smart city technologies, robotics, energy-efficient systems, digital twins, neural-fuzzy control, and intelligent automation of complex processes.

<https://orcid.org/0000-0002-5974-9310>**Ph.D. Andrii Pukach**

e-mail: andriipukach@gmail.com

Ph.D. in technical sciences. Assistant at ACS Department, Lviv Polytechnic.
Research interests: design and control of MEMS, automated measurement systems, computer-aided design, mathematical modelling in engineering, object-oriented programming, cryptographic protection, and development of innovative computer system components.

<https://orcid.org/0009-0001-8563-3311>