I.J. Intelligent Systems and Applications, **2025, 2, 56-100**
Published Online on April 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2025.02.05

Intelligent Application for Textual Content Authorship Identification based on Machine Learning and Sentiment Analysis

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

Victoria Vysotska
Department of Information Systems and Networks, Institute of Computer Sciences and Information Technologies, 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, Faculty of Applied Mathematics and Informatics, 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

Sofia Chyrun
Telecommunication Department, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: sofiia.chyrun.sa.2022@lpnu.ua
ORCID iD: https://orcid.org/0000-0002-2829-0164

Cennuo Hu
Department of Computer Science, College of Science, Purdue University, West Lafayette, IN 47907, USA
E-mail: hu945@purdue.edu
ORCID iD: https://orcid.org/0009-0008-9589-9253

Yuriy Ushenko*
Department of Physics, Shaoxing University, Shaoxing, Zhejiang Province 312000, China
Department of Computer Science, Educational and Research Institute of Physical, Technical and Computer Sciences, Yuriy Fedkovych Chernivtsi National University, 58012, Ukraine
E-mail: y.ushenko@chnu.edu.ua
ORCID iD: https://orcid.org/0000-0003-1767-1882
*Corresponding author

Received: 16 January 2025; Revised: 21 February 2025; Accepted: 13 March 2025; Published: 08 April 2025

Abstract: During the development and implementation of the software system for text analysis, attention was focused on the morphological, syntactic and stylistic levels of the language, which made it possible to develop detailed profiles of authorship for various writers. The main goal of the system is to automate the process of identifying authorship and detecting plagiarism, which ensures the protection of intellectual property and contributes to the preservation of cultural heritage. The scientific novelty of the research was manifested in the development of specific algorithms adapted to the peculiarities of the natural language, as well as in the use of advanced technologies, such as deep learning and big data. The introduction of the interdisciplinary approach, which combines computer science, linguistics, and literary studies, has opened up new perspectives for the detailed analysis of scholarly works. The results of the work confirm the high efficiency and accuracy of the system in authorship identification, which can serve as an essential tool for scientists, publishers, and law enforcement agencies. In addition to technical aspects, it is vital to take into account ethical issues

related to confidentiality and copyright protection, which puts under control not only the technological side of the process but also moral and legal norms. Thus, the work revealed the importance and potential of using modern text processing methods for improving literary analysis and protecting cultural heritage, which makes it significant for further research and practical use in this area.

Index Terms: Machine Learning Methods, Text Analysis, Authorship Identification, Sentiment Analysis, NLP, SVM, LSTM, CNN, RNN.

1. Introduction

The topic of text analysis for the original authorship of works of literature is essential and relevant for several reasons [1]. In today's world, where information is spread rapidly through the Internet, the problem of plagiarism and copyright arises [2]. Literary analysis allows us not only to determine the authenticity of the text but also to preserve cultural heritage, particularly in the case of national literature, such as English or Ukrainian. Identifying original authorship is key to ensuring intellectual property and recognition of writers' work. Scientific interest in the analysis of texts goes beyond literary studies and finds application in jurisprudence, history, and cultural studies. The development of technologies in the field of natural language processing (NLP) and machine learning opens up new opportunities for the study of literary texts [3-6]. Automation of the author identification process allows the analysis of large volumes of data with high accuracy and speed, which is indispensable in modern scientific and commercial conditions. In addition, the question of authorship is particularly relevant in the literature context, where historical vicissitudes such as Soviet censorship often led to the loss of documents or changes in authorship for ideological purposes [7-9]. Restoring historical justice and accurately attributing works to their authors helps restore national identity and maintain cultural heritage. On a practical level, the development of adequate tools for the analysis of authorship can contribute to the fight against literary piracy, which is a significant problem in Ukraine. Such tools allow publishing houses, libraries and educational institutes to protect the rights of writers and promote the development of the literary market [10-12]. Recognition and protection of authorship are also essential for maintaining the creative climate and stimulating new literary talents.

The goal of creating a software system for text analysis for the original authorship of works of literature is to develop an effective tool that will automate the process of determining the authorship of literary works. It is essential in the context of ensuring the protection of intellectual property, maintaining copyright and detecting possible cases of plagiarism. The system aims not only to determine the author of the text but also to provide an analysis of literary styles and the use of linguistic features and, ultimately, to contribute to the preservation of cultural heritage. The system faces the following tasks to achieve this goal:

- Development of the methodology for authorship identification. The system should use machine learning and natural language processing algorithms to analyse texts. It includes training the models on historical data that contains samples of the works of famous authors to create a reliable basis for comparison.
- Creation of the database of literary works. For the effective operation of the system, an extensive database is required, which includes the texts of works of various genres and periods. It will help in the analysis and identification of literary techniques and styles that may be characteristic of specific authors.
- Use of linguistic features. The system should analyse linguistic features such as syntax, vocabulary, stylistic figures and other elements of the text that may indicate a specific author.
- User interface design. The interface should be intuitive and user-friendly so that users with different levels of technical skills can use the system effectively. It also assumes the presence of functionality for entering text, analysing it and obtaining results.
- Ensuring confidentiality and data security. Since the system will process texts that may have a copyright, it is essential to ensure that this data is protected from unauthorized access.

Thanks to the fulfilment of these tasks, a software system for analysing text for original authorship can significantly contribute to the development of literary science, as well as provide practical tools for publishing houses, scientific institutions and law enforcement agencies to fight plagiarism and support copyright.

The object of research in the analysis of the text for the original authorship of works of literature is literary texts written in the natural language in the context of determining their authorship. The research focuses on the use of linguistic, stylistic and other textual characteristics that help to identify authors of works or establish cases of literary plagiarism. Primary attention is paid to the analysis and comparison of the linguistic features of the texts, which includes a deep understanding of the individual and characteristic features of the writers [12-15].

- *Literary texts*. First of all, the text of literature itself is the main object of research. It can be novels, poetry, plays, essays, etc. The study of these texts includes the analysis of their structure, vocabulary, and syntax, as

well as unique stylistic devices that may indicate a specific author. Each author has a unique style of writing, which is reflected in the use of particular terminology, metaphors, rhythm, and other linguistic elements.

- *An author's style* analysis is an integral part of determining authorship. Studying how particular authors use language helps create "authorship profiles" that can be used to compare and identify the authors of unknown or disputed works. It includes researching their previous works and identifying the unique features that distinguish one author from another.
- *Linguistic analysis* involves the detailed study of language and its components used in literary works. It covers grammatical structure, word choice, and phrasing, as well as more subtle aspects such as rhythm and meter in poetry. Such an analysis can reveal the use of linguistic patterns typical for a specific author.
- The subject of research in text analysis for the original authorship of literature works focuses on the use of specific methods and techniques that allow establishing the author of a literary work or detecting plagiarism. These methods cover both linguistic and stylistic analysis and include the following main aspects [15-21]:
- *Linguistic analysis* consists of a detailed study of the linguistic features of the text. It includes analysis of syntax, morphology, semantics and phonetics. It is essential to study the use of grammatical structures, word frequency, sentence formation and characteristic language repetitions that may indicate a particular author. The use of advanced natural language processing and machine learning techniques makes it possible to analyse these characteristics on large volumes of data.
- *Stylistic analysis* focuses on the study of individual features of the author's style, such as the use of metaphors, symbols, and irony, as well as features of text construction. The stylistic analysis allows the reveal of unique author's "signatures", which may not always be evident at first glance. Research can also include the analysis of themes and motifs that reveal a deeper connection between the works of the same author.
- *Use of statistical methods*. Authorship analysis also uses statistical methods for processing and interpreting textual data. It may include frequency analysis, clustering, principal component analysis, and other statistical procedures that help identify statistically significant features of texts associated with particular authors.
- *Software and algorithms*. A separate role in the research subject is played by the development and implementation of specialized software and algorithms for automating the analysis process. The development of efficient algorithms that can quickly process large volumes of text and determine authorship with high accuracy is key to the practical application of text analysis.
- The scientific novelty of text analysis research on the original authorship of works of literature consists of the application of advanced methods of text analysis and natural language processing to solve specific problems of authorship identification. This direction can have a number of innovative aspects that will make a significant contribution to literary studies, computational linguistics, and jurisprudence.
- *Development of specific algorithms for the natural language*. Most modern text analysis and language processing technologies are focused on English and other widely used languages. In the context of literature, novelty can consist of the adaptation of existing algorithms or the development of new ones that take into account the unique features of the natural language, such as morphological structure, syntax and vocabulary. It may include the development of specialized tools for morphological analysis, semantic analysers and algorithms that detect stylistic features specific to texts.
- *Using deep learning*. Innovations may also include the application of deep learning techniques to the analysis of literary texts. Neural networks such as LSTM (Long Short-Term Memory) and transformers can be trained on large corpora of texts to detect complex language patterns and stylistic features that indicate authorship. This approach can significantly increase the accuracy of author identification and plagiarism detection.
- *Analysis of big data*. With the help of big data technologies, it is possible to analyse vast volumes of literary works, which would previously have been impossible due to the limitation of human resources. It allows us not only to determine authorship more quickly but also to study changes in literary styles and currents on a scale inaccessible to traditional research methods.
- *Interdisciplinary approach*. Scientific novelty can also consist of a multidisciplinary approach that combines the methods of literary studies, computer science, and statistics. It allows the create complex analysis models that take into account both linguistic and contextual aspects, providing a deeper and more comprehensive understanding of texts.
- *Ethical aspects in text analysis*. Research may also include the development of a moral framework for the use of analytical tools, ensuring the protection of personal data and copyright in the process of text analysis. Ethical consideration is key to creating technologies that respect individual rights and cultural identities.

Thus, scientific innovation in text analysis for the original authorship of works of literature has the potential to make a significant contribution to expanding the possibilities of modern literary studies and natural language processing, providing new tools for the analysis and interpretation of literary texts. The problem of text analysis for the original authorship of works of literature is multifaceted and includes a number of challenges arising from literary, linguistic, technical and legal aspects. This problem becomes especially relevant in the context of modern requirements for the protection of intellectual property and challenges related to the digitization of textual information [15-21].

- *Literary and linguistic aspects*. The scientific problem lies in the difficulties of identifying the authorship of texts, where the individual style of the writer must be revealed and analysed with great precision. The author's style may include unique lexical choices, syntactic constructions, stylistic figures and other linguistic features that require detailed analysis. There are also cases when authors consciously adapt or modify their style, which calls into question the possibility of unambiguous attribution of texts.
- *Technical challenges*. The technical side of the problem is the need to develop and implement sophisticated algorithms for machine learning and natural language processing that can effectively process large volumes of text and distinguish subtle nuances of speech style. The importance of accuracy in algorithms is critical, as errors can lead to incorrect attribution of authorship, with serious consequences.

The goal of creating a software system for text analysis for the original authorship of works of literature is to develop an effective tool that will automate the process of determining the authorship of literary works. It is essential in the context of ensuring the protection of intellectual property, maintaining copyright and detecting possible cases of plagiarism. The system aims not only to determine the author of the text but also to provide an analysis of literary styles and the use of linguistic features and, ultimately, to contribute to the preservation of cultural heritage.

2. Related Works

Since no analogues were explicitly found for literature, there is a need to consider programs that are primarily used for English literature. It is also essential to know that models are divided into the following examples according to their method of creation [1-3]:

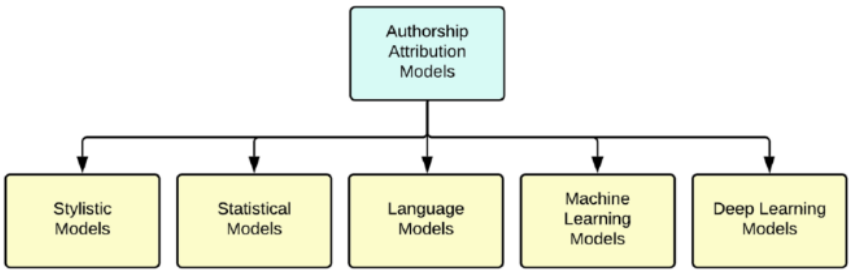


Fig.1. Authorship attribution models

JGAAP (Java Graphical Authorship Attribution Program). Authorship is a study in the field of stylometry. It addresses the problem of determining the most likely author of an unknown text from a pool of potential authors. Tools such as JGAAP (Java Graphical Authorship Attribution Program), Stylometry with R, and the stylo Python package allow quantitative analysis of authorship based on stylistic and linguistic features. They can be trained or tuned for Ukrainian texts if a sufficient dataset is available.

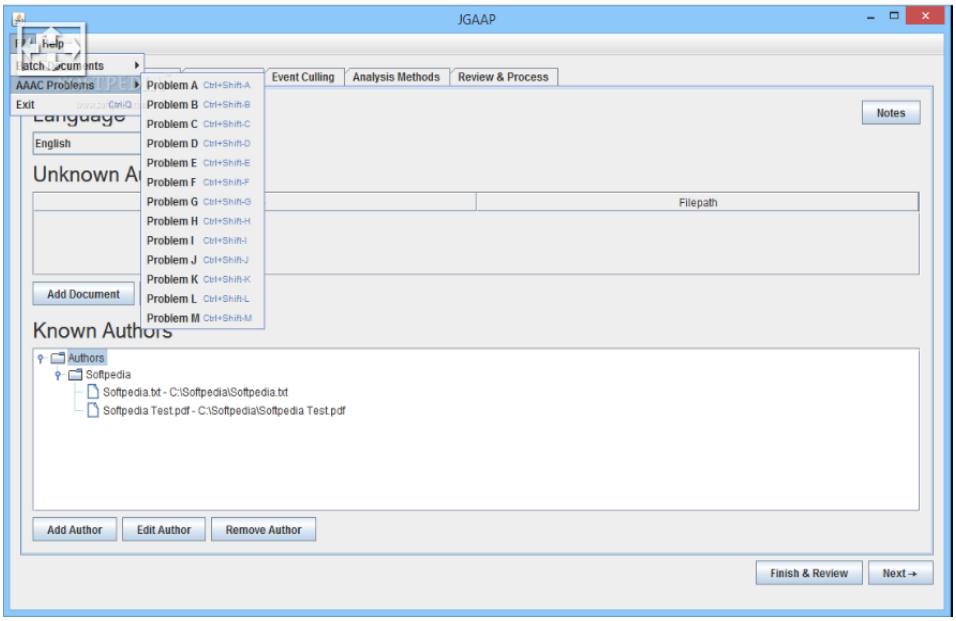


Fig.2. Java graphical authorship attribution program

Recent developments in machine learning and corpus linguistics have shown the possibility of automatic authorship determination using statistics; the NSF-funded JGAAP (Java Graphical Authorship Attribution Program) system was part of these developments. JGAAP has helped foster a new authorship community and create a valuable tool for a wide range of scientific specialities. Although JGAAP contains thousands of possible methods, there are many more proposed but not thoroughly tested methods in the literature. Benchmarking on a large scale will require the development of new methods and test corpora. In addition, many key issues need to be addressed to meet the needs of the community, such as the open class issue, the adversarial issue, and the co-authorship issue. Finally, it is necessary to look at the application of JGAAP and similar systems to key areas of linguistic profiling, such as gender, education, native language, psychological profiling, health status, age (of document or author) or even attempts at deception. Again, by applying a rigorous testing method to these new problems and corpora, the project can establish accuracy benchmarks for different methods (under various test conditions), find new combinations that lead to improved methods, and create "best practice" recommendations. Improved attribution will be of immediate benefit to both academics and broader social contexts, such as law enforcement and forensics, where there is a direct demand for this type of security technology. Historical/social analysis will also provide better access to the related disciplines of digital humanities, sociology, history, and computer science, creating a basis for a better understanding of traditional humanities issues. Profiling can help doctors and psychologists by providing a non-invasive method of identifying certain aspects of the human psyche. The developed software (and the planned development/distribution process) will help improve the effectiveness of digital humanities and computer science, primarily through the establishment of standards and software validation processes. In particular, by providing direct evidence of the conditions and expected error rates involved in different methods, the resulting information will help establish authorship to meet Daubert's criteria for expert evidence, allowing authorship designations to be used in formal legal contexts. Finally, funding for this research will help support Duquesne University's unique interdisciplinary computational mathematics program, providing greater access to unusual and atypical audiences for technology education.

General Architecture for Text Engineering, or **GATE,** is a Java toolkit originally developed at the University of Sheffield. It is now used worldwide by a vast community of academics, companies, educators, and students for many natural languages processing tasks, including information extraction in many languages.

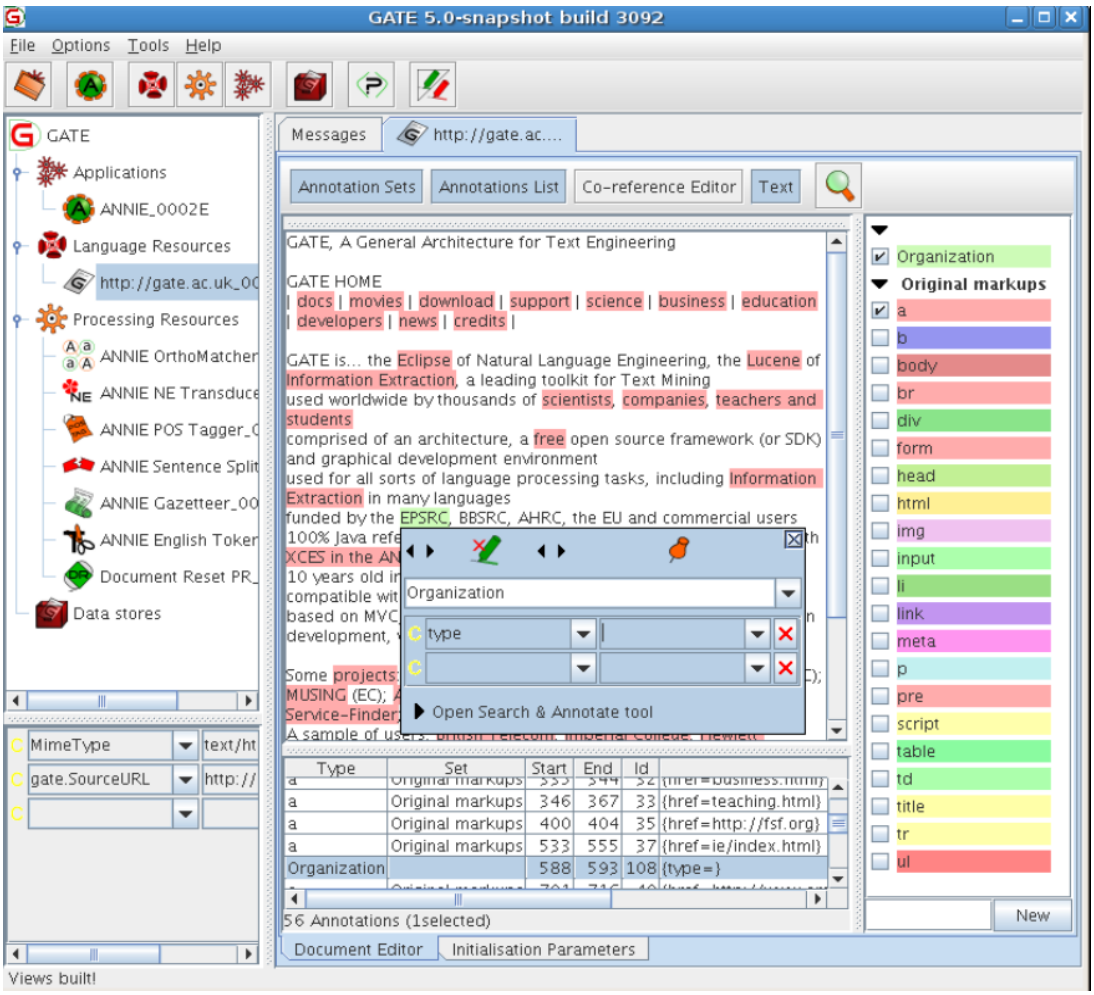


Fig.3. General architecture for text engineering

The Natural Language Toolkit, or more commonly NLTK, is a set of symbolic and statistical natural language processing (NLP) libraries and programs for the English language written in the Python programming language. NLTK contains both datasets and graphics. The package includes a book that explains the basic concepts of the language processing tasks supported by the toolkit, as well as examples of how to use the package. NLTK is designed to support research and teaching courses related to NLP and related fields, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been successfully used as a learning tool and a platform for prototyping and building research systems. In the US and 25 other countries, 32 universities use NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionality.

Advantages of the developed product. The development of a software product for text analysis for the original authorship of works of Ukrainian and English literature has several significant benefits that significantly strengthen its practical application and scientific value.

- *Convenient functionality*. One of the key advantages is a convenient and intuitive user interface. It means that users, regardless of their technical experience, can effectively use the system to analyse texts. The interface makes it easy to load text data, perform analysis, and view results in an easy-to-understand format. The inclusion of such functions as user hints and a contextual help menu makes the process of interacting with the program as simple and effective as possible.
- *Availability of Ukrainian and English literature datasets*. A great advantage is the creation and integration of a specialized dataset of Ukrainian and English literature, which includes texts of various genres and periods. The presence of a large and diverse body of texts allows the system to more accurately analyse and recognize the author's features. This dataset not only improves the quality of authorship analysis but also makes it possible to carry out complex studies of the natural literary language and its evolution and transformations.
- *Innovative technologies*. The development includes the application of advanced technologies in the field of natural language processing and machine learning, such as neural networks, deep learning algorithms and statistical analysis. These technologies provide high accuracy and speed of data processing, allowing the complex analysis of large volumes of text to be performed for the purpose of identifying the author's style. Innovative solutions embodied in the product also include algorithms for identifying plagiarism and ensuring copyright, which is especially relevant in today's digital world.

These advantages make the developed product convenient for users, effective for research in the field of literature, and essential for ensuring the cultural and intellectual heritage of Ukraine. Such innovations contribute to scientific progress and the support of the literary sphere in the conditions of globalization and digitalization, opening up new opportunities for the analysis, preservation and development of Ukrainian and English literature.

Disadvantages of the developed product. Despite numerous advantages, the developed product for text analysis for the original authorship of works of Ukrainian and English literature has several shortcomings that may affect its effectiveness and acceptability for end users.

- *Data limitations*. One of the main problems is the limited availability and volume of Ukrainian literature datasets, which are necessary for training machine learning algorithms. Although the development involves the creation of a specialized corpus of texts, there is a risk that this corpus will not be fully representative or large enough to ensure high accuracy of the analysis. Incomplete data can lead to incorrect interpretation of results or errors in determining authorship.
- *Technical limitations*. The natural language processing and machine learning technologies used in the product, while advanced, have certain technical limitations. The complexity of the algorithms requires significant computing resources, which can be problematic for users with limited technical capabilities. Also, high dependence on the quality of input data can complicate the versatility and reliability of the product.
- *Ethical and legal issues*. Developing such a product presents researchers with ethical and legal challenges. Using literary texts to train algorithms raises questions about copyright and data privacy. There is a risk that without proper regulation and control, the system could be used to infringe copyright or to intrude on the privacy of authors.
- *Dependence on linguistic features*. Another problem is the high dependence of the analysis results on the peculiarities of the natural language. The presence of dialects, historical changes in the language, and stylistic differences can make it difficult to determine the exact authorship, especially for historical texts where linguistic norms differ significantly from modern ones.

These shortcomings require a careful approach to improving technological aspects, proper legal regulation and ensuring ethical responsibility when using the system. These challenges not only identify potential design weaknesses but also outline directions for further research and product improvement that can lead to greater market adoption and adoption.

General comparison. It is necessary to conduct a general comparison of the product we developed with existing analogues on the market by carefully analysing the main parameters, such as the effectiveness of detecting artificially created texts and the accuracy of the results. Based on this comparison, it is possible to find out the advantages and disadvantages of our product compared to competitors, as well as identify its competitive advantages and opportunities for further improvement.

Table 1. The advantages and disadvantages of our product compared to competitors

The name of the program or site	Convenient functionality	Availability of the Ukrainian language	Advanced machine learning technologies
JGAAP	Yes	No	Yes
General Architecture for Text Engineering	Yes	No	Yes
Natural Language Toolkit	No	No	Yes

From the results of the table, it is concluded that the best results have the programs JGAAP and General Architecture for Text Engineering.

Prospects for product development. The prospects of developing a product for text analysis for the original authorship of works of Ukrainian and English literature open up new opportunities for the development of the science of literature, cultural preservation, education and technologies. This product can have a significant impact in several key areas:

- *Scientific research*. The product can become an essential tool for academic research, allowing scholars to analyse literary texts more deeply and identify stylistic features of different authors. It can contribute to a better understanding of literary trends and the evolution of language, as well as help to resolve academic disputes about the authorship of works.
- *Education*. In the field of education, such a product can be used to teach students to analyse literary works, develop critical thinking and improve literacy. Text analysis tools can become part of the learning process in Ukrainian and English literature classes, allowing students to experiment and learn through hands-on experience.
- *Protection of intellectual property*. In the context of the growing challenges of plagiarism and copyright, development can serve as an essential tool for publishers, legal authorities and independent authors to protect their intellectual property. The system can help identify and document cases of copyright infringement, as well as provide an evidentiary basis in legal cases.
- *Expansion of technological capabilities*. Technological innovations based on artificial intelligence and machine learning continue to develop, and the development of such a product can contribute to the further development of natural language processing and deep learning algorithms. It opens the door to new tools capable of analysing text at a much deeper level, including emotional content, subtext, and contextual meanings.
- *Cultural preservation*. The product can be necessary for the preservation of culture, especially in the context of modern globalization and cultural homogenization. Preservation of unique linguistic and stylistic features through accurate analysis of authorship will help future generations to understand and appreciate the contribution of previous generations of writers.
- *Development at the international level*. Since literature is becoming more and more famous and popular abroad, there is a significant potential for the use of such a product at the global level, especially in areas where it is necessary to determine the authorship of translations or verify the text within the framework of literary studies. Such a tool can contribute to greater visibility of Ukrainian and English literature and its analysis in world academic circles.

Based on the analysis of the current state of development of a software product for text analysis for the original authorship of works of literature, it can be concluded that this project is essential for both the academic community and the cultural sphere. The use of advanced technologies of natural language processing and machine learning allows for an increase in the accuracy and speed of text analysis, which opens up new opportunities for research in the field of literary studies and the fight against plagiarism. Convenient functionality and the availability of a specialized dataset ensure the availability and effectiveness of the product for a wide range of users, including researchers, educators and law enforcement agencies. It contributes not only to the in-depth analysis of literary works but also to the preservation of the national cultural heritage. However, the project has some drawbacks, such as limited data and high requirements for technical resources, which may limit its application in certain conditions. It is also important to consider ethical and legal issues related to the use of literary works.

3. Material and Methods

3.1. System Analysis of the Research Object and Subject Area

The systematic analysis of the object of research and the subject area for the analysis of the text for the original authorship of works of literature focuses on a comprehensive approach to the study of literary texts with the aim of identifying their authors [1-3]. This analysis covers several key aspects: the definition of the research object, the analysis of the subject area, and the implementation of systematic methods and tools for the study of texts.

- Object of research. The object of research in this context is literary texts written in Ukrainian or English. It includes poetry, prose, drama and other forms of scholarly works. The primary purpose of the analysis is to establish the authorship of these works, which can be especially important in cases where the text is attributed to different authors or when the authorship remains unknown or disputed.
- Subject area. The subject area covers literary studies, linguistics, computer science, and, in some aspects, legal studies. It includes the methods of stylometry, which allow analysis of the stylistic features of the text and the use of specific verbal forms and structures that may indicate a particular author. Natural language processing and machine learning methods are also used to analyse large volumes of text data.
- System methods and tools. System analysis involves the use of various tools and technologies. The main ones are databases for storing texts, analytical tools for studying linguistic features, and software for automating the processes of analysis and classification. The use of statistical methods and artificial intelligence makes it possible to develop models capable of determining the authorship of texts with high accuracy.
- Impact and prospects. The study of system analysis in this area opens up new opportunities for the development of literature and culture. It also has practical implications in the areas of plagiarism detection, copyright protection, and academic integrity.

3.2. Goal Tree

The tree of goals for the project "Text analysis for original authorship of works of literature" can be considered as a hierarchical structure of goals, which allows for detailed planning and organization of all key aspects of the project. This tree includes the primary goal, sub-goals, and specific tasks that must be accomplished to achieve the ultimate goal. A description of the goal tree will detail each level of the hierarchy.

The main goal of the project is to develop an effective tool for analysing texts with the aim of identifying the original authorship of works of literature. This tool should be able to accurately identify the authors of texts based on linguistic and stylistic features.

Sub-goals

1) Development of a comprehensive database of literary works:
- A collection of texts of various genres and eras;
- Creating a structured database for analysis;
  - Collect more than 10,000 works;
  - Digitize archival texts;
  - Develop mechanisms for constant updating of the database;

2) Development of text analysis algorithms:
- Implementation of machine learning and natural language processing techniques;
- Development of algorithms for the identification of stylistic and linguistic features characteristic of specific authors;
  - Setting up neural networks for text analysis;
  - Testing algorithms for accuracy and reliability;
  - Optimization of algorithms for fast processing of large volumes of data;
- Setting up neural networks for text analysis;

3) Integration of the tool into academic and educational processes:
- Development of user-friendly interfaces in educational and research institutions;
  - Development of a user interface that includes intuitive tools for determining authorship;
  - Organization of workshops and demonstrations for educators and researchers;
- Data visualization support.

The implementation of the goals mentioned above will allow not only to increase the quality and accuracy of the analysis of literary works but also to significantly expand the possibilities of their use for academic and educational purposes. The introduction of the product can contribute to a better understanding of the cultural heritage of Ukraine and support the preservation of the literary language. The discovery of these opportunities can also increase the interest of the international audience in literature, providing greater visibility and recognition of authors on the world stage.

However, there are challenges to consider:

- Ensuring sufficient funding and support for project development;
- Overcoming technical difficulties associated with the processing and analysis of large volumes of textual information;
- Compliance with ethical standards and copyright protection in the process of using literary works.

3.3. Selection of Alternative Options and Selection of the Best Option

When developing a platform for text analysis for the original authorship of works of literature, it is essential to consider several alternative options. The choice of the best option should be based on a careful analysis of the capabilities of each solution, their effectiveness, cost, and the possibility of integration with existing systems. Alternatives for platform development:

1) Development of own platform "from scratch"
- *Advantages*. Complete control over the functionality and the ability to fine-tune the system to the specific needs of the project.
- *Disadvantages*. High development cost, long development time, high risks of delays and technical problems.

2) Use of open libraries and tools
- *Advantages*. Lower development costs use of time-tested tools.
- *Disadvantages*. Limitations in functionality and possible difficulties with the integration of various components.

3) Partnership with technological companies
- *Advantages*. Access to advanced technologies support from experienced developers.
- *Disadvantages*. Dependence on a third party, potential limitation in control over the final product.

4) Adaptation of existing commercial solutions
- *Advantages*. Fast implementation, no need to develop own tools.
- *Disadvantages*. High licensing costs and possible limitations in customization.

To choose the best option, it is necessary to conduct a detailed analysis of costs, potential risks, and the time required to implement each of the possibilities. It is also essential to involve project stakeholders in discussing possible solutions to ensure their support and understanding of key requirements. Considering the specificity of the task of analysing the authorship of texts, it may be beneficial to choose a combination of approaches. For example, one might consider developing a basic platform "from scratch" with the integration of open-source libraries for specific tasks that require deep specialization, such as machine learning and natural language processing. It will balance cost and efficiency while providing high adaptability and control over the final product.

In the process of developing software or other information systems, a key stage is the identification, analysis and documentation of system functionality requirements. Use cases in the Unified Modelling Language (UML) methodology are an effective tool for this. They allow to describe how the system interacts with its users or external agents to achieve specific goals or perform certain tasks.

User → *Run the program* → *Download data* (add sources, references, and test input) → *Select an analysis method* → *Select an analysis metrics* → *Configure an analysis method* → *Run an analysis* → *View results* → *Export results*

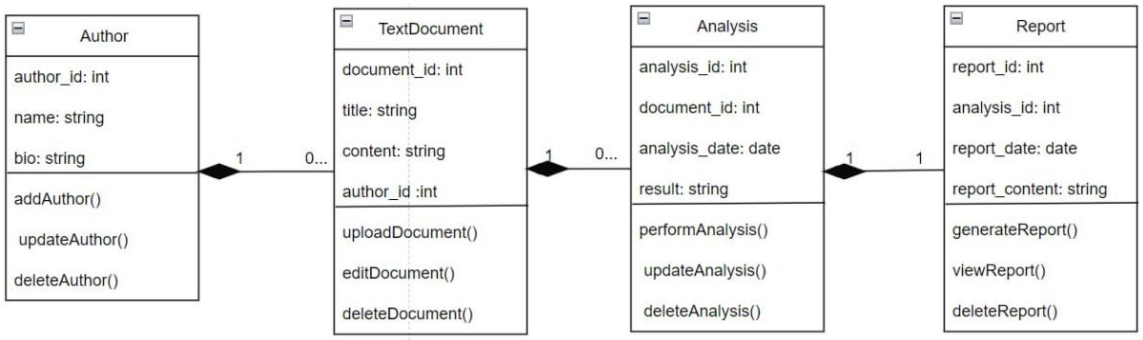


Fig.4. UML class diagram

User → *Text Document* (uploadDocument → saveDocument → getDocument) → *Analysis* (analyseDocument → getAnalysis) → *Report* (createReport → saveReport) → *User*

Workflow algorithm:

Adding data → Pre-process text → Extract features → Train model → Generate reports

3.4. Statement and Justification of the Problem

For the system of text analysis for the original authorship of works of literature, the following key functions and tasks, the primary goals, and an analysis of the importance and necessity of its implementation can be outlined. The text analysis system for authorship is designed to perform several main tasks:

- *Author identification*. The system has algorithms that allow the author of the text to be determined based on linguistic and stylistic features.
- *Plagiarism detection*. The ability of the system to analyse the text for its originality allows it to identify potential cases of plagiarism.
- *Statistical analysis*. The system can collect and analyse statistical data on the frequency of use of certain verbal forms and structures that are characteristic of specific authors.
- *Educational tool*. Providing opportunities for educational institutions to use the system in the process of studying literature.

The main goals of the system:

- *Improving the understanding of literary heritage.* The system helps researchers and students to better
- understand the style and other features of the works of famous authors.
- *Copyright protection*. The system helps protect intellectual property by identifying copyrights for textual works.
- *Supporting academic integrity*. Providing tools to check academic papers for plagiarism increases the level of academic integrity in educational institutions.

Implementing a text analysis system for authorship is essential for several reasons:

- *Cultural significance*. The preservation and proper attribution of literary works are of great importance for national culture and identity.
- *Academic value*. The system will help the academic community conduct research and provide access to reliable tools for checking texts for authorship. It is especially relevant in the conditions of the modern information space, where originality of content is key to academic and creative activity.
- *Legal significance*. Determining authorship is critical to protecting intellectual property. Having practical authorship analysis tools can facilitate the proper resolution of copyright and plagiarism disputes.
- *Technological importance*. The development of natural language processing and machine learning technologies through the implementation of such projects stimulates innovation and improves technical capabilities in general.

3.5. Application Area of the System

Scope and users. The text analysis system for the original authorship of works of literature can be applied in various areas, including:

1) *Academic Institutions*. Teachers and researchers can use the system to analyse literary works, study the author's style, and check student works for plagiarism.
2) *Publishers*. To verify the originality of works before publication and ensure compliance with copyright.
3) *Law enforcement agencies*. In cases where it is necessary to determine the authorship of works as part of investigations into copyright violations.
4) *Libraries and Archives*. For cataloguing and attribution of texts, especially old or anonymous manuscripts.

Application context. The system can be applied in the following contexts:

1) *Educational*. Supporting the learning process through the study of literary works and the education of academic integrity among students.
2) *Scientific*. Conducting complex literary studies with the aim of analysing literary styles and identifying literary periods or influences.
3) *Legal*. Use in court cases to confirm authorship of texts or identify copyright violations.

Potential application benefits:

1) *Increasing the accuracy of attribution of works.* The use of modern technologies makes it possible to accurately determine authorship, reducing the risk of errors in attribution.

2) *Effective management of rights to content*. Automating the authorship identification process contributes to more effective management of intellectual property rights.

3) *Preservation of cultural heritage*. The possibility of archiving and accurate identification of literary works contributes to the preservation of national culture.

Problems that the system solves:

1) *Detection of plagiarism.* The system is able to identify cases of copying of texts, which is especially important for the educational environment and publishing activities.

2) *Differentiation of similar authoring styles*. Thanks to deep analysis of textual characteristics, the system can distinguish even very similar authoring styles, helping to accurately identify authors under challenging cases.

3) *Support for the study of literary heritage*. The system provides tools for scholarly researchers to identify connections and influences between different authors and works, contributing to a deeper understanding of literary trends.

3.6. Justification of System Development and Implementation

The justification for the development and implementation of a text analysis system for the original authorship of works of literature is based on several key factors that highlight the necessity and significance of such a system in the modern context. These aspects include the cultural, educational, legal and technological importance of the system.

- *Cultural significance*. Literature is not only an artistic expression but also an essential part of the cultural identity and history of Ukraine. The development of an authorship analysis system will help scientists and cultural scientists to more deeply explore the creative heritage of writers, restore forgotten or little-known works, and discover unpublished manuscripts. It will also contribute to the preservation and popularization of literature, supporting the cultural diversity and multidimensionality of the national culture.
- *Educational value*. An authorship analysis system can become an indispensable tool in the academic environment, where it will allow teachers and students to use advanced technologies for the study of literature. It can serve as a platform for teaching text analysis, stylistic consideration, and historical contextualization of literary works, thereby improving the quality of education and research.
- *Legal significance.* From a legal point of view, the system will contribute to the protection of intellectual property and copyright, which is extremely important in the conditions of growing digitalization. A clear definition of authorship will help resolve disputes related to plagiarism and misuse of textual materials, ensuring fairness in the literary and media industries.
- *Technological significance*. The development of such a system stimulates innovative development in the field of natural language processing and machine learning. It not only improves the existing methodologies of text analysis but also opens up possibilities for the application of artificial intelligence in many spheres of society. These technologies have the potential to radically change approaches to data analysis, particularly in literary studies and other humanities.
- *General rationale for system implementation*. The rationale for the development and implementation of a text analysis system for the authorship of works of literature stems from the current needs of cultural, educational, legal and technological development. The system is aimed at raising the level of literary education, scientific research, intellectual property protection and cultural heritage support. It can play an essential role in preserving national identity, popularizing literature, and in the fight against academic fraud and plagiarism.

3.7. Expected Effects of System Implementation

The expected effects of the implementation of the text analysis system on the original authorship of works of literature can be large-scale and influential, covering various fields from education and culture to law and technology. A detailed consideration of the potential effects provides a better understanding of the meaning and importance of such a tool.

- *Increasing the level of education and scientific research*. One of the most important effects of the implementation of the system is its impact on the educational process and scientific research. The system will allow teachers and students to study literary texts more effectively, using in-depth text analysis to determine the author's styles and techniques. It not only promotes critical thinking but also helps students understand how different historical and cultural contexts influence literary works. In a scientific context, the system can assist researchers in determining authorship and in plagiarism analysis, providing more accurate and informed conclusions.
- *Protection of intellectual property*. The system will significantly improve the possibilities of intellectual property protection, particularly in the field of literature. With a precise definition of authorship, authors and publishers will have a more solid basis for protecting their rights in cases of illegal use or plagiarism of their

works. It not only promotes fairness but also encourages creativity and innovation, knowing that their rights will be protected.

- *Promotion of cultural preservation*. The system will help preserve and popularize literary heritage, allowing us to identify and catalogue works that may have been forgotten or lost. It will ensure the accessibility of these works to a broader audience and contribute to a deeper understanding of cultural and literary history.

3.8. Development of the Conceptual Model of the System

The development of the text analysis system for the original authorship of works of literature includes careful planning and structuring of input and output data, algorithms, system functions, and requirements for it. Here is a detailed description of these aspects:

- *Input data* are Text files (the system will accept texts in formats such as .txt, .docx, and .pdf for analysis) and Metadata (information about works, such as publication date, author, and genre, that can be used to improve the accuracy of the analysis).
- *Output data* are Authorship report (identification of the author or possible authors of the work), Statistical data (analysis of the frequency of use of words, stylistic figures, etc.) and Visualizations (graphs and diagrams reflecting the stylistic features of the texts).

The system will include the following key components:

- *Data download module*. Interface for downloading text files and metadata.
- *Text processing module*. Convert texts from different formats into a single standard format for analysis.
- *Text analysis module*. Machine learning algorithms were used to identify the author's features.
- *Output module*. Presentation of analysis results in a clear and visually appealing format.
- *User interface*. Simple and intuitive interface for interacting with the system.

Determination of requirements for the system and other formal models:

- *Technical requirements* are Compatibility (the system must be compatible with the central operating systems such as Windows, macOS, and Linux), Scalability (the ability to process large amounts of data without sacrificing performance) and security (ensuring the protection of input data and confidentiality of information).
- *Functional requirements* are Accuracy (high accuracy of authorship identification and plagiarism detection), Speed (ensuring fast data processing, essential for large text corpora) and Accessibility (easy access to the system through the web interface or desktop applications).
- *Machine learning algorithms* are Stylometry (the use of stylometric analyses to determine the characteristic features of the author's style), Neural Networks (development of deep neural networks that can learn to detect sub-textual patterns and stylistic signatures specific to individual authors) and Clustering (using clustering techniques to group works by stylistic similarities, which can help determine authorship).

The system should have the ability to process text files of various formats, such as PDF, DOCX, and TXT, which will allow users to easily download materials for analysis without the need for additional conversion.

The user interface should be intuitive and user-friendly for users with different levels of technical skills. The main elements of the interface include:

- *Download management*. Simple tools for downloading and organizing text files.
- *Control panel*. Visual representation of the analysis process and results, with the ability to sort and filter data.
- *Feedback*. Tools for sending feedback and getting support from developers.

The output results of the analysis should be presented in an understandable format, including:

- *Reports*. Detailed analysis reports are exportable to popular formats such as PDF or Excel.
- *Visualizations*. Graphs, diagrams, and other visualizations are used to visually present the stylistic and linguistic features of texts.
- *Recommendations*. Automated recommendations for further research or action based on analysis results.

Expected effects of system implementation:

- *Cultural influence*. The system allows for more profound research and preservation of literary heritage. With its help, it is possible to restore forgotten or unknown works, ensuring their availability to a broad audience. It also contributes to understanding the evolution of language and literary styles, which is essential for cultural identity.

- *Educational influence*. The system can significantly improve the quality of education by providing teachers and students with modern tools for analysing literature. It promotes the development of students' critical thinking, allowing them to analyse literary works at a deeper level and understand the nuances of the author's style.
- *Legal impact*. The system improves copyright protection by providing tools to accurately determine authorship and identify plagiarism. It helps prevent misuse of creative works by ensuring that authors are fairly compensated for their work.

It can be concluded that the development and implementation of a text analysis system for the original authorship of works of literature is a significant and promising project. The system provides solutions to a number of tasks and reveals the potential for influence in various areas.

- *Cultural enrichment*. The system will contribute to the better preservation and popularization of literary heritage, allowing for a deeper study of the work of authors and ensuring accurate attribution of scholarly works.
- *Educational opportunities*. The use of the system in the educational process will help teachers and students expand their analytical skills through access to advanced technologies of text analysis, increasing the quality of teaching and research.
- *Copyright protection*. The system plays a key role in detecting plagiarism and ensuring compliance with intellectual property, which is essential in today's world of rapid information dissemination.
- *Technological progress*. The development of such a system stimulates innovation in the fields of natural language processing and machine learning, opening new opportunities for the application of these technologies in other disciplines and industries.
- *Social significance*. The system can contribute to the strengthening of national identity through a deeper awareness of cultural and literary achievements and fostering respect for the author's work.

This project has the potential not only to solve specific practical tasks but also to make a significant contribution to the development of culture, education, law and technology, thereby contributing to social and cultural progress.

3.9. Selection and Justification of Methods for Solving the Task

For the system to work effectively, practical methods and tools are needed to help them distinguish between texts and make appropriate decisions. In this context, it is essential to choose and justify the methods and means of presenting knowledge in the decision-making system, the description of the mechanisms of logical deduction in the system and the justification of decision-making algorithms [1-9].

Methods and means of presenting knowledge in the decision-making system:

1) **Text vectorization** is the process of converting text into numerical vectors that machine algorithms can process. Popular methods include:

- *Bag of Words* (BoW). The text is represented as a set of words, where each word in the text is associated with its frequency.
- *Term Frequency-Inverse Document Frequency* (TF-IDF). Used to determine the importance of a word in the context of a document, taking into account its frequency in the entire corpus.
- *Word Embeddings* (e.g. Word2Vec, GloVe). Creates word vectors that represent semantic and syntactic relationships between words.

2) **Stylistic Signs** features are often used to analyse authorship, which characterizes the author's unique writing style:

- *Syntactic features*. Length of sentences, use of punctuation, frequency of use of different parts of speech (for example, adjectives, verbs).
- *Lexical features*. Variety of vocabulary, frequency of use of certain phrases or specific words.
- *Semantic features*. Themes and concepts that often appear in the author's works.

3) **Neural networks**, especially deep learning models, are able to automatically detect important features for authorship classification:

- *Convolutional Neural Networks* (CNN). Effective for detecting patterns in extensive text data.
- *Recurrent Neural Networks* (RNN) and *Long Short-Term Memory* (LSTM): Used to analyse text sequences, in particular, to model context and dependencies in the text.

Description of logical output mechanisms in the system:

1) **Data processing and analysis stages**. Logical inference mechanisms begin with data collection and preparation:

- *Text pre-processing*. Text normalization (noise removal, spelling standardization), token segmentation, stop

word removal, and lemmatization.
- *Text vectorization*. Converting text into vector representations such as TF-IDF or word vectors.
- *Selection of features*. Identification and use of features that most influence the determination of authorship, for example, stylistic features (sentence length, use of punctuation), lexical features (frequency of phrases), and syntactic features (phrase structure).

2) **Machine learning models**:
- *Classification models*. Training algorithms such as naive Bayes classifier, SVM, random forests, or neural networks based on a training dataset with known authorship.
- *Deep learning*. Using recurrent or convolutional neural networks to detect complex patterns in textual data that may indicate the style of a particular author.

Description and justification of decision-making algorithms:

1) **Naive Bayesian classifier**.
- *Description*. This algorithm uses the principle of Bayesian statistics and assumes independence of features. It calculates the probabilities of each class (author) based on the input features (words, phrases, stylistic elements) and classifies the text by the author with the highest probability.
- *Rationale*. Naïve Bayes is simple to implement and efficient with large amounts of data. It works well even when the assumption of independence of features does not hold. A significant advantage is its ability to handle many classes, which is ideal for systems where there may be many potential authors.

2) **Support vector method** (SVM).
- *Description*. SVM searches for the hyperplane that best separates classes of data in a multidimensional space. In the context of authorship analysis, classes are different authors, and features are stylistic characteristics of texts.
- *Rationale*. SVM is effective in high-dimensional data, as is often the case in word problems, and is able to efficiently solve multiclass problems through strategies such as one-vs-one. It makes it an excellent choice for authorship analysis, as there may be many input features and the need to distinguish between writing styles accurately.

3) **Random forests**.
- *Description*. Random Forests is an ensemble method that uses multiple decision trees to classify data. Each tree is trained on a randomly selected subset of features and data, and the classification results are aggregated to obtain a final decision.
- *Rationale*. Random forests work well with complex data structures and provide high accuracy while reducing overtraining due to the ensemble nature. This method also determines the importance of features, which can help analyze which stylistic elements have the most significant impact on author identification.

4) **Deep learning** (e.g. LSTM, CNN).
- *Description*. Deep learning uses complex neural network structures to model data. LSTM (Long Short-Term Memory) is excellent for analyzing sequences of data, such as texts, where context and word order are essential. CNNs (Convolutional Neural Networks) can be used to detect patterns in text at different levels of abstraction.
- *Rationale*. Deep learning models can automatically detect complex patterns in data, making them ideal for tasks that require the recognition of subtle nuances in writing style. They require significant computing resources and large amounts of data for practical training but provide high accuracy and flexibility in simulation.

3.10. Main Components of the System

1) User interface (UI)
- Loading text for analysis (txt, pdf, docx).
- Selecting a model for author attribution.
- Displaying results: predicted author, model confidence level.

2) NLP module
- Text preprocessing: tokenization, lemmatization, stopword removal.
- Text vectorization: TF-IDF, Word2Vec, BERT embeddings.
- Analysis of stylistic features of the text.

3) Machine learning modules
- Using models: Naïve Bayes, SVM, LSTM for authorship classification.
- Training models on text corpora.

- Generating a forecast and deriving metrics (accuracy, F1-score, etc.).

4) Results processing module are Generating reports, Visualizing results in the form of a confusion matrix and Integration with external services (e.g., plagiarism databases).

System operation algorithm:

Stage 1. The user uploads text.
Stage 2. The NLP module cleans text and extracts features.
Stage 3. Text vectorization.
Stage 4. Machine learning model predicts authorship.
Stage 5. Results are displayed in the UI.

This structure allows the system to work efficiently with large amounts of text and supports real-time authorship analysis.

3.11. Justification for the Choice of Naive Bayes and SVM in the Study

The study uses several machines learning methods, including Naive Bayes (naive Bayes classifier) and SVM (support vector method). Their choice is based on efficiency, computational speed, accuracy, and the ability to work with high-dimensional text data. Justification for the choice:

1) The Naive Bayes classifier is based on Bayes' theorem and the assumption of independence of features. It calculates the probability that the text belongs to a particular author using words, phrases, and stylistic elements. Bayes' formula:

$$P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$$

where $P(A|B)$ is the probability that the author is A, given text B; $P(B|A)$ is the probability of obtaining text B given that it was written by A; $P(A)$ is the prior probability of authorship of A; $P(B)$ is the total probability of encountering text B in the data corpus.

Table 2. Analysis of the naive bayes classifier

N	Name	Explanation
1	Advantages	Speed of learning and prediction – the algorithm calculates probabilities in constant time, making it one of the fastest methods. Minimal computational requirements are efficient when working with large amounts of text data. Works well with text data, even if the assumption of feature independence is not met.
2	Disadvantages	Independence assumption – in real-world texts, words are often dependent on each other, which can reduce accuracy. It does not take into account word order, which can be critical to the author's style.
3	Reason for choice	High speed is vital for analyzing large text corpora. Sufficient accuracy in stylometry tasks is about 80% on the test corpus. Works well on sparse text data (for example, when using TF-IDF).

2) SVM is a linear or nonlinear classifier that searches for the optimal hyperplane to separate data in a multidimensional space:

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

where w is weight vector, b is bias, x_i is input feature vector, y_i is class label (+1 or -1).

Table 3. Analysis of the SVM classifier

N	Name	Explanation
1	Advantages	It works well with high-dimensional data and is ideal for TF-IDF and Word2Vec. High accuracy (≈ 79-81%) – especially for text classification. Can work in multi-class classification using "one-vs-one" and "one-vs-all" methods. Good generalization – less prone to overtraining than neural networks.
2	Disadvantages	Slow learning – may be less efficient than Naive Bayes for large amounts of data. Difficult to interpret – SVM models are less transparent than tree models. Kernel selection is required, which requires parameter tuning (linear, polynomial, RBF kernel).
3	Reason for choice	It works better on text features with high dimensionality (e.g. 300-dimensional Word2Vec). High accuracy is for author attribution (≈ 81.26% for English and 75.05% for Ukrainian). Support for nonlinear solutions – can take into account complex dependencies between words.

Naive Bayes is chosen for fast and straightforward text classifications. SVM is used where higher accuracy and the ability to handle complex text patterns are required. In the study, they are combined: Naive Bayes works for fundamental analysis and SVM for deeper analysis of the authors' style.

Table 4. Comparative analysis of methods

Criterion	Naive Bayes	SVM
Learning speed	Very high	Slow
Prediction speed	High	Medium
Accuracy	75-80%	79-81%
Computational complexity	Low	High
Support for non-linear solutions	None	Yes
Interpretability	High	Low

Naive Bayes is suitable for fast and efficient text analysis, especially when working with large amounts of data. SVM provides high accuracy, especially when using TF-IDF or Word2Vec. The combined use of NB and SVM allows you to achieve a balance between speed and accuracy of author attribution. SVM has a higher computational complexity, which may limit its application on large corpora. The results of this analysis confirm that both algorithms were chosen reasonably based on the characteristics of text data, accuracy and computational efficiency.

3.12. Selection and Justification of Means of Solving the Task

It was decided to use the Naive Bayesian classifier.

- *Description*. This algorithm is based on the principles of Bayesian statistics and assumes independence of features among themselves. It calculates the probabilities of each class (author) based on the input features (words, phrases, stylistic elements) and classifies the text by the author with the highest probability.
- *Rationale*. Naïve Bayes is easy to implement and efficient, especially with large amounts of data. It works well even when the assumption of independence of features does not hold.

The Naive Bayes classifier is often used for authorship analysis tasks due to several key advantages, especially compared to other machine learning algorithms such as the support vector method (SVM), random forests, or deep learning (e.g., LSTM, CNN). Here is a detailed rationale for choosing Naive Bayes for this task:

1) Learning and forecasting speed:
- *Naive Bayes* is one of the fastest algorithms because it only requires the computation of class probabilities based on independent features. It is significantly faster compared to algorithms that require complex calculations, such as SVMs with non-linear kernels or deep learning.
- *Other methods*. SVM, random forests, and deep learning require more time to train, especially on large or high-dimensional datasets.

2) Need for resources:
- *Naive Bayes*. Efficient in terms of memory and computing resources. It is essential when processing large volumes of text data, where other models may require significant computing power.
- *Other methods*. Deep learning, for example, requires significant computing resources and time to train and optimize the model, which may be impractical for some applications.

3) Ease of implementation:
- *Naive Bayes*. Easy to implement and configure, even for developers with a basic understanding of machine learning. This makes it an affordable choice for rapid prototyping.
- *Other methods*. SVM and deep learning require a deeper understanding of machine learning theory and parameterization for effective implementation.

Each of these classes follows Python's OOP principles by using abstract methods to ensure that any AnalysisMethod subclass implements the required functionality. This design makes it easy to extend and customize the document analysis methods in the system. In the process of developing the given project, various techniques and tools for the analysis of textual data were carefully selected and substantiated.

Analysis Method class

1) *Purpose*. Serves as a base class for various parsing methods that learn on known documents and predict the labels of unknown documents.

2) *Attributes*
- distance (stores the distance function used in subclasses);
- _variable_options (dictionary for storing variable options);
- _global_parameters (dictionary for storing global parameters);

- _NoDistanceFunction_ (a flag indicating the need for a distance function);

3) *Methods*
- __init__ (constructor initializes options based on _variable_options);
- after_init (placeholder for initialization tasks to be defined in subclasses);
- train (an abstract method for training a model using known documents);
- analyze (an abstract method for analysing unknown documents);
- displayName (abstract method to return the name of the method);
- displayDescription (abstract method to return the method description);
- set_attr (method for dynamically setting attributes);
- validate_parameter (validates parameters against predefined options);
- get_train_data_and_labels (prepare data for training and labels);
- get_test_data (preparation of test data);
- get_results_dict_from_matrix (converts scores to a dictionary of results by class);
- setDistanceFunction (sets the distance function);

Centroid Driver class

1) *Purpose*. Implements a centroid-based approach where the mean (centroid) is calculated for each known author of the data.

2) *Methods*:
- train (computes the mean by author from the training data);
- analyze (analyzes unknown documents by calculating distances to centroids);
- displayName (returns the name of the method);
- displayDescription (describes the method);

The K Nearest Neighbor class

1) *Purpose*. Implements the K-Nearest Neighbours algorithm for document classification;

2) *Attributes*:
- k (number of nearest neighbours considered);
- tie_breaker (method for resolving conflicts as 'average' or 'minimum');

3) *Methods*:
- train (stores embeds of known documents);
- analyze (classifies unknown documents based on k-nearest neighbours in the feature space);
- displayName (returns the name of the method);
- displayDescription (describes the method, including details about conflict resolution).

The most essential tools used in the work are Count Vectorizer and the naive Bayesian classifier method. Count Vectorizer will be used to efficiently vectorize text, that is, convert text data into numeric vectors that would allow machine learning models to work with them. The Naive Bayesian classifier method was chosen to classify the text based on this vectorized data. Both of these tools would show effective results when working with textual data. The Tkinter library was also analysed to create an interactive user interface. It can provide convenient interaction with the program and make the process of text analysis more accessible to users. In addition, other libraries, such as pandas for data processing, as well as docx and PyPDF2 for working with documents in DOCX and PDF formats, will be used to fully process text data from various sources. All these tools and methods together will form a functional and practical system for text analysis using machine learning methods that can be used in real projects for different purposes and tasks.

4. Experiments

4.1. Detailed Description of the Dataset used in the Study

The dataset used in the study contains texts for authorship identification, presented in two languages: English and Ukrainian. The study includes original and translated texts, which makes it possible to test the system in different conditions. Characteristics of the dataset

- Data volume. Thirty thousand text fragments were used for training and testing the model. Texts by authors of classical Ukrainian and English literature (Franko, Shevchenko, Kulish, L. Ukrainka). English texts were translated into Ukrainian (via Google Translate API).
- Data diversity. Texts of different genres were used: poetry, prose, and journalism. Other periods were also

used: works from the 19th to the 21st centuries. Frequency analysis and analysis of syntactic structures were used to identify authorial styles.

- Data quality. Text normalization was performed (removal of stop words and special characters). The maximum sentence length was determined to be 100 words (optimization for the model). Analysis of syntactic features allowed for the improvement of the quality of author attribution.

Stages of data preprocessing

Stage 1. Conversion to a convenient format. XML conversion to DataFrame (convenient format for analysis).
Stage 2. Text preprocessing
Step 1. Removed stop words (a custom list for Ukrainian was used).
Step 2. Tokenization (breaking the text into words).
Step 3. Lemmatization (reducing words to their original form).
Step 1. Using Word2Vec for vectorization (300-dimensional vectors).
Stage 3. Class balancing. Since some authors are represented by a smaller number of texts, augmentation techniques (paraphrasing, synonym selection) were used.
Stage 4. Distribution into samples: 70% training sample, 20% test, 10% validation. Cross-validation to avoid overtraining.

A large corpus of texts (30,000 excerpts) provided a sufficient sample for training models. Text optimization (lemmatization, tokenization) significantly improved the accuracy of the model. The use of Word2Vec and GloVe allowed us to improve the processing of text semantics. The main challenge is that translated texts may differ from the originals, which affects the accuracy of author attribution. This dataset is unique due to its support for the Ukrainian language and the possibility of testing on different genres and periods.

4.2. Description of the Created Program Entry

The program uses the natural language for data analysis. A dataset of authors of literature was created independently for the program. We add the link:

https://docs.google.com/spreadsheets/d/1lp_bXczIIS3YAHSTRho8_pZM4Wy5lGiXzY8xJE6QKrQ/edit?usp=sharing

The program uses ML algorithms to improve analysis accuracy and supports various text formats. The program identifies unique stylistic features of the text, such as word frequency, syntactic structures, and other language patterns.

- *Comparison of texts*. AuthorshipAnalyzer compares texts with each other to determine similarities and possible joint authorship.
- *Using machine learning*. The program uses machine learning methods to train models on large data sets, which improves the accuracy of authorship determination.
- *Support for various text formats*. AuthorshipAnalyzer supports the analysis of texts in multiple formats, such as TXT, DOC, and PDF, which makes the program versatile for different users.
- *Intuitive interface*. The program has a convenient and easy-to-use interface that allows you to quickly download texts and receive analysis results.

Fields of application

- *Literary studies*. Determining the authorship of unknown or disputed texts.
- *Journalism*. Analysis of texts to confirm the authorship of articles.
- *Forensics*. Use in investigations to identify the author of anonymous texts.
- *Education*. Use to teach students methods of stylometric analysis.

The basis of the work of AuthorshipAnalyzer is a detailed analysis of the stylistic characteristics of the text. The program analyses lexical features such as frequency of word usage, unique word forms and general vocabulary of the text. Syntax analysis allows us to determine grammatical structures, such as the frequency of use of different parts of speech, the length of sentences, and the use of punctuation. In addition, AuthorshipAnalyzer is able to detect unique stylistic patterns of text, such as rhythm, paragraph structure, and use of literary devices. It allows the create a comprehensive text profile that can be used to identify the author.

- *Comparative analysis of texts*. One of the key functions of AuthorshipAnalyzer is the ability to compare texts with each other. The program allows one to determine the degree of similarity between different texts, which can be helpful in detecting possible borrowings or plagiarism. In addition, the program can determine the joint authorship of texts by analysing stylistic intersections between different parts of the text.

- *Machine learning models*. AuthorshipAnalyzer uses machine learning algorithms to improve authorship accuracy. The program trains the model on large data sets, which allows for a wide range of stylistic features to be taken into account. These models can adapt to new data, allowing the program to continuously improve its results.
- *User interface*. The program has an intuitive interface that makes it convenient for both beginners and experienced users. The interface allows you to easily download texts, configure analysis parameters and view results. Visualization of the results in the form of graphs, charts and other visual representations helps to better understand the data and draw informed conclusions.

4.3. Structure of the Program

1) **User interface** (GUI) is the central component of the application that provides user interaction with other modules.

It includes the following elements:

- *Main application window*. Displays all the main functions of the application and provides access to other modules.
- *Navigation Menu*. Allows users to navigate between different sections of the program, such as text loading, analysis, and visualization.
- *Buttons for data analysis and visualization*. Basic buttons for running text analysis and displaying results in the form of graphs and charts.

2) **The text loading and processing module** is responsible for loading texts into the program and their pre-processing. It includes:

- *File upload function*. It supports uploading texts in various formats such as TXT, DOC, DOCX, and PDF.
- *Pre-processing of text*. Cleaning of text from unnecessary characters, formatting and normalization for further analysis.

3) **Stylistic Analyzer** performs a detailed analysis of the text, including:

- *Lexical analysis*. Determination of frequency of use of words, unique word forms and general vocabulary.
- *Syntax Analysis*. Analysis of grammatical structures such as frequency of use of different parts of speech, sentence length and use of punctuation.
- *Analysing Stylistic Patterns*. Identifying unique stylistic characteristics such as rhythm, paragraph structure, and literary devices.

4) **The machine learning module** uses algorithms to improve the accuracy of authorship. It includes:

- *Training models*. Using extensive data sets to train machine learning models.
- *Adaptability of models*. Models can adapt to new data, which allows the program to constantly improve its results.

5) **The data visualization module** is responsible for visually presenting the analysis results. It includes:

- *Graphs and Charts*. Display analysis results in the form of graphs and charts for a better understanding of the data.
- *Interactive visualizations*. Ability to interact with graphs for more detailed analysis of results.

Description of the technologies used

- Python is a general-purpose programming language used to develop various types of applications, from web applications to data mining and artificial intelligence. In our project, this is the primary programming language.
- Tkinter is a Python library that creates graphical user interfaces. It provides a wide range of tools and widgets for creating various user interfaces. In our project, it is used to develop a graphical user interface.
- NumPy is a Python library for computing data. It provides support for large arrays and matrices, along with high-level math functions for working with these arrays. In our project, this library is used for operations with arrays and matrices.
- Pandas is a Python library for data processing and analysis. It provides data structures such as DataFrames that simplify working with tabular data, as well as a set of functions for their processing and analysis. The project is used for data processing and analysis.
- scikit-learn is a Python library for machine learning and data analysis. It contains a variety of machine learning algorithms, including classification, regression, clustering, and others, as well as tools for building and evaluating models. In the project, it is used for model training and text vectorization.

4.4. Main Functions of the Program

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
from gensim.models import Word2Vec
import numpy as np
import tkinter as tk
from tkinter import Text, Scrollbar, Label, Button, messagebox, Canvas, Frame
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
import seaborn as sns
import re
from collections import Counter
data = {'text': [ ],'author': [ ]} # Data
df = pd.DataFrame(data) # Creating a DataFrame
def average_sentence_length(text): # Functions for stylometric analysis
    sentences = re.split(r'[.!?]', text)
    sentences = [s for s in sentences if s]
    return np.mean([len(s.split()) for s in sentences])
def average_word_length(text):
    words = re.findall(r'\w+', text)
    return np.mean([len(word) for word in words])

def punctuation_frequency(text):
    punctuations = re.findall(r'[^\w\s]', text)
    return Counter(punctuations)
def stylometric_features(text):
    features = {}
    features['avg_sentence_length'] = average_sentence_length(text)
    features['avg_word_length'] = average_word_length(text)
    features['punctuation_freq'] = punctuation_frequency(text)
    return features
class TextClassifierApp: # Main function for processing text classification
    def __init__(self, master):
        self.master = master
        master.title("Text Classification")
        self.label = Label(master, text="Add text: ")
        self.label.pack(pady=(10, 5))
        self.text_frame = tk.Frame(master)
        self.text_frame.pack()
        self.text_entry = Text(self.text_frame, height=5, width=100)
        self.text_entry.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
        self.scrollbar = Scrollbar(self.text_frame, command=self.text_entry.yview)
        self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        self.text_entry.config(yscrollcommand=self.scrollbar.set)
        self.buttons_frame = tk.Frame(master)
        self.buttons_frame.pack(pady=(5, 10))
        self.add_text_button = Button(self.buttons_frame, text="Add Text", command=self.add_text)
        self.add_text_button.pack(side=tk.LEFT, padx=(10, 10))
        self.analyse_button = Button(self.buttons_frame, text="Run Classification", command=self.analyse_text)
        self.analyse_button.pack(side=tk.LEFT, padx=(10, 10))
        self.visualize_button = Button(self.buttons_frame, text="Visualize Results",
command=self.show_visualizations)
        self.visualize_button.pack(side=tk.LEFT, padx=(10, 10))
        self.result_frame = tk.Frame(master)
        self.result_frame.pack(pady=(10, 10))
        self.result_text = Text(self.result_frame, height=10, width=100)
        self.result_text.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)
        self.result_scrollbar = Scrollbar(self.result_frame, command=self.result_text.yview)
        self.result_scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
        self.result_text.config(yscrollcommand=self.result_scrollbar.set)
        self.figure_tfidf = None
        self.figure_w2v = None
        self.vectorizer = TfidfVectorizer()
        self.vectorizer.fit(df['text'])
        self.model_tfidf = SVC(kernel='linear')
        self.train_model()
    def add_text(self):
        new_text = self.text_entry.get("1.0", tk.END).strip()
        if new_text:
            df.loc[len(df)] = [new_text, "Unknown"]
            self.text_entry.delete("1.0", tk.END)
            self.analyse_new_text(new_text)
        else:
            messagebox.showwarning("Warning", "Please enter some text.")
    def train_model(self):
        X_train, X_test, y_train, y_test = train_test_split(df['text'], df['author'], test_size=0.2,
random_state=42, stratify=df['author'])
        X_train_tfidf = self.vectorizer.transform(X_train)
        self.model_tfidf.fit(X_train_tfidf, y_train)
    def analyse_new_text(self, text):
# This method trains the SVM model based on TF-IDF vectorization.
        X_new_tfidf = self.vectorizer.transform([text])
        prediction = self.model_tfidf.predict(X_new_tfidf)
        features = stylometric_features(text)
        self.result_text.insert(tk.END, f"New Text Author Prediction: {prediction[0]}\n")
```

```
        self.result_text.insert(tk.END, f"Average Sentence Length: {features['avg_sentence_length']:.2f}\n")
        self.result_text.insert(tk.END, f"Average Word Length: {features['avg_word_length']:.2f}\n")
        self.result_text.insert(tk.END, f"Punctuation Frequency: {features['punctuation_freq']}\n" + "="*80 + "\n"
        messagebox.showinfo("Result", f"The new text is most likely written by {prediction[0]}.")
    def analyse_text(self):
        df_filtered = df[df['author'] != "Unknown"]# Delete rows with class "Unknown"
        # Separation of data into training and test sets
        X_train, X_test, y_train, y_test = train_test_split(df_filtered['text'], df_filtered['author'],
test_size=0.2, random_state=42, stratify=df_filtered['author'])
        X_train_tfidf = self.vectorizer.fit_transform(X_train) # Text vectorization using TF-IDF
        X_test_tfidf = self.vectorizer.transform(X_test)
        self.model_tfidf = SVC(kernel='linear') # Training the SVM model
        self.model_tfidf.fit(X_train_tfidf, y_train)
        y_pred_tfidf = self.model_tfidf.predict(X_test_tfidf) # Forecasting on test data
        # Model Quality Assessment
        self.tfidf_report = classification_report(y_test, y_pred_tfidf, zero_division=0)
        # Confusion Matrix for TF-IDF
        self.cm_tfidf = confusion_matrix(y_test, y_pred_tfidf, labels=self.model_tfidf.classes_)
        self.figure_tfidf, ax_tfidf = plt.subplots(figsize=(6, 5))
        sns.heatmap(self.cm_tfidf, annot=True, fmt='d', cmap=plt.cm.Blues, ax=ax_tfidf,
xticklabels=self.model_tfidf.classes_, yticklabels=self.model_tfidf.classes_)
        ax_tfidf.set_title("Confusion Matrix for TF-IDF")
        self.figure_tfidf.tight_layout()
        texts = [text.split() for text in df_filtered['text']] # Using Word2Vec for vectorization
        w2v_model = Word2Vec(sentences=texts, vector_size=100, window=5, min_count=1, workers=4)

        def text_to_w2v(text, w2v_model):
            words = text.split()
            word_vecs = [w2v_model.wv[word] for word in words if word in w2v_model.wv]
            return np.mean(word_vecs, axis=0) if word_vecs else np.zeros(w2v_model.vector_size)
        X_train_w2v = np.array([text_to_w2v(text, w2v_model) for text in X_train])
        X_test_w2v = np.array([text_to_w2v(text, w2v_model) for text in X_test])
        self.model_w2v = SVC(kernel='linear') # Training the SVM model on Word2Vec vectors
        self.model_w2v.fit(X_train_w2v, y_train)
        y_pred_w2v = self.model_w2v.predict(X_test_w2v) # Forecasting on test data with Word2Vec vectors
        self.w2v_report = classification_report(y_test, y_pred_w2v, zero_division=0) # Model Quality Assessment
        # Confusion Matrix for Word2Vec
        self.cm_w2v = confusion_matrix(y_test, y_pred_w2v, labels=self.model_w2v.classes_)
        self.figure_w2v, ax_w2v = plt.subplots(figsize=(6, 5))
        sns.heatmap(self.cm_w2v, annot=True, fmt='d', cmap=plt.cm.Blues, ax=ax_w2v,
xticklabels=self.model_w2v.classes_, yticklabels=self.model_w2v.classes_)
        ax_w2v.set_title("Матрица плутанини для Word2Vec")
        self.figure_w2v.tight_layout()
        self.result_text.delete(1.0, tk.END) # Clear the text widget for results
        # Display results in a text widget with a separator
        self.result_text.insert(tk.END, "TF-IDF Report:\n" + self.tfidf_report + "\n" + "="*80 + "\n")
        self.result_text.insert(tk.END, "Word2Vec Report:\n" + self.w2v_report)
        # Plotting accuracy, completeness, and F1 measure for both models
        self.figure_metrics_tfidf = self.plot_classification_report(y_test, y_pred_tfidf, "TF-IDF Classification
Report")
        self.figure_metrics_w2v = self.plot_classification_report(y_test, y_pred_w2v, "Word2Vec Classification
Report")
    def plot_classification_report(self, y_true, y_pred, title):
        report = classification_report(y_true, y_pred, output_dict=True, zero_division=0)
        df_report = pd.DataFrame(report).transpose()
        df_report = df_report.drop(["accuracy", "macro avg", "weighted avg"])
        fig, ax = plt.subplots(1, 3, figsize=(9, 3))
        ax[0].bar(df_report.index, df_report["precision"])
        ax[0].set_title("Precision")
        ax[0].tick_params(axis='x', rotation=45)
        ax[1].bar(df_report.index, df_report["recall"])
        ax[1].set_title("Recall")
        ax[1].tick_params(axis='x', rotation=45)
        ax[2].bar(df_report.index, df_report["f1-score"])
        ax[2].set_title("F1-Score")
        ax[2].tick_params(axis='x', rotation=45)
        plt.suptitle(title)
        fig.tight_layout()
        return fig
    def show_visualizations(self):
        if self.figure_tfidf and self.figure_w2v:
            visualization_window = tk.Toplevel(self.master) # Create a new window to display graphs
            visualization_window.title("Visualization Results")
            canvas = Canvas(visualization_window) # Create a Canvas to scroll
            scroll_y = Scrollbar(visualization_window, orient="vertical", command=canvas.yview)
            scroll_frame = Frame(canvas)
            scroll_frame.bind("<Configure>",lambda e: canvas.configure(scrollregion=canvas.bbox("all") ) )
            canvas.create_window((0, 0), window=scroll_frame, anchor="nw")
            canvas.configure(yscrollcommand=scroll_y.set)
            canvas.pack(side="left", fill="both", expand=True)
            scroll_y.pack(side="right", fill="y")
            canvas_tfidf = FigureCanvasTkAgg(self.figure_tfidf, master=scroll_frame) # Displaying graphs in Canvas
            canvas_tfidf.draw()
            canvas_tfidf.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)
            canvas_w2v = FigureCanvasTkAgg(self.figure_w2v, master=scroll_frame)
            canvas_w2v.draw()
            canvas_w2v.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)
            canvas_metrics_tfidf = FigureCanvasTkAgg(self.figure_metrics_tfidf, master=scroll_frame)
            canvas_metrics_tfidf.draw()
            canvas_metrics_tfidf.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)
```

```
            canvas_metrics_w2v = FigureCanvasTkAgg(self.figure_metrics_w2v, master=scroll_frame)
            canvas_metrics_w2v.draw()
            canvas_metrics_w2v.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)
        else:
            messagebox.showerror("Error", "Please run the classification first.")
root = tk.Tk()# Creating the main Tkinter window
app = TextClassifierApp(root)
root.mainloop()
```

4.5. User Manual

This program is intended for users of various training levels, from beginners to advanced users. It has an intuitive graphical interface that allows you to easily interact with the program. Thus, even users without previous experience with programs can quickly master their primary functions. First of all, the user starts the program, where he sees the following interface in Fig. 5.

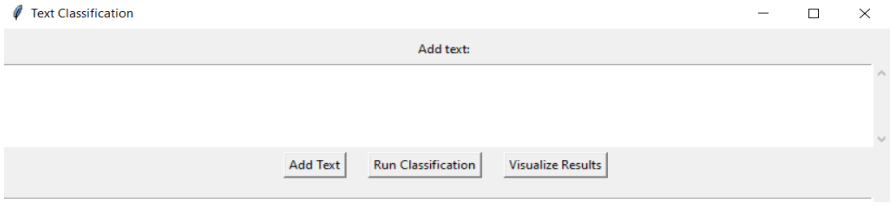


Fig.5. User interface

The user can then do one of three things: add his data and perform a stylometric analysis, as well as an authorship analysis; it is possible to find out data about the correct operation of our program, as well as the visualization schedule. It is needed to add data manually or using a file to find out the stylometric analysis:

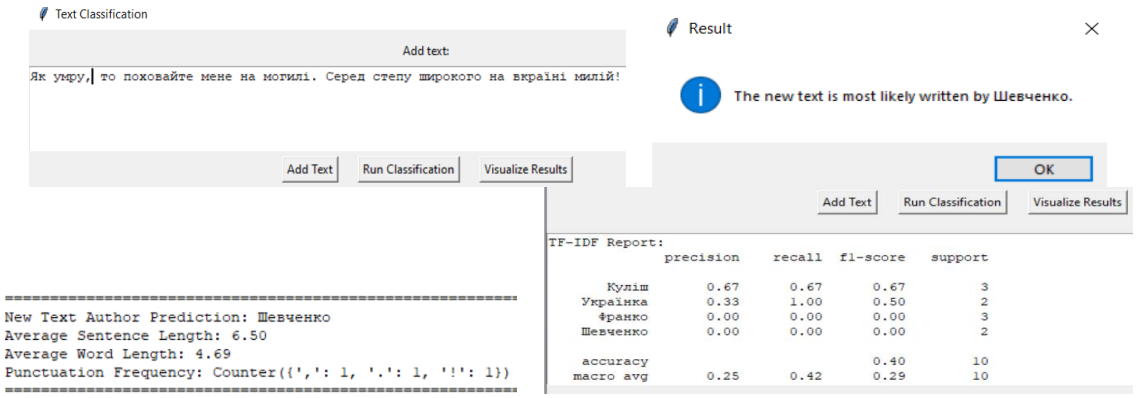


Fig.6. The stylometric analysis

Then, we click the "Add Text" button, after which authorship is determined, as well as stylometric analysis. To find out the data analysis, it is necessary to click on the Run Classification button, which displays the metrics and, in general, the correctness of the operation of our program. To find out visualize the data, it is needed to click on the Visualize Results button:

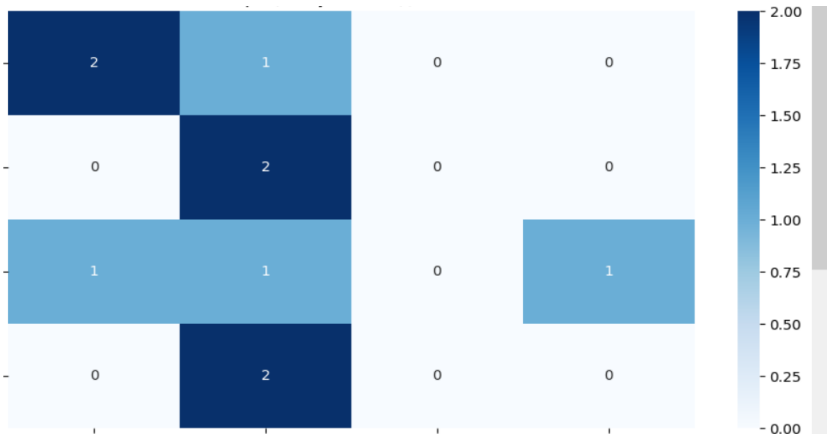


Fig.7. Visualize Results for the stylometric analysis as a Confusion matrix for TF-IDF (from left to right and top to bottom Panteleimon Kulish, Lesia Ukrainka, Ivan Franko and Taras Shevchenko)

In today's world, where information spreads at an incredible speed, the question of determining the authorship of texts is becoming more and more relevant. AuthorshipAnalyzer is a powerful text analysis tool designed to solve this problem. The program uses modern methods of stylometry and machine learning to ensure high accuracy and reliability of results. AuthorshipAnalyzer has an intuitive interface that allows users to quickly load texts, run analyses, and visualize the resulting data. The program supports various text file formats, which makes it versatile for use in multiple areas. The main functionalities of the program include loading and pre-processing of texts, analysis of stylistic characteristics, comparative analysis of texts, use of machine learning models to improve accuracy, and visualization of results in the form of graphs and charts. The reporting module allows the generation of detailed reports that can be saved in various formats for later use or presentation.

4.6. Running the Test Case

To describe the behaviour of the developed software (software) when the test case is run, it is necessary to define what the program does, how it interacts with the user, and what results are expected. Here are the main points that describe the launch of the test case and the functionality of the program.

1) **Starting the program**. Opening a text detection program written by a chatbot starts by downloading the source code from GitHub. To do this, it is needed to go to the link that points to the GitHub repository where the program code is located. After going to the GitHub page, click the "Code" button and select the "Download ZIP" option to download the program archive. After that, it is necessary to unzip the ZIP file to a convenient location on the computer. If the application uses third-party Python libraries or other dependencies, installation is required before running.

After installing all dependencies, it is necessary to find the main file of the program, which is responsible for starting it. To run the program, open a terminal or command prompt in the appropriate folder and run the program using the Python main.py command. If the program is to be launched through a graphical interface, there is a double-click on the corresponding file to launch it. The program uses an independently created dataset of Ukrainian authors and their works, namely Taras Shevchenko, Panteleimon Kulish, Lesya Ukrainka, and Ivan Franko. When the application is launched, the user sees a graphical interface that provides interaction. The interface contains various elements, such as adding text, starting analysis and classification, and data visualization.

2) **Data entry**. First of all, the user has the option to enter the data that he plans to analyse or add a TXT/Excel file to perform a check. After the data is added, it needs to click on the Add Text button, after which our program analyses the text, carries out stylometric analysis, and also determines the potential author of this text.

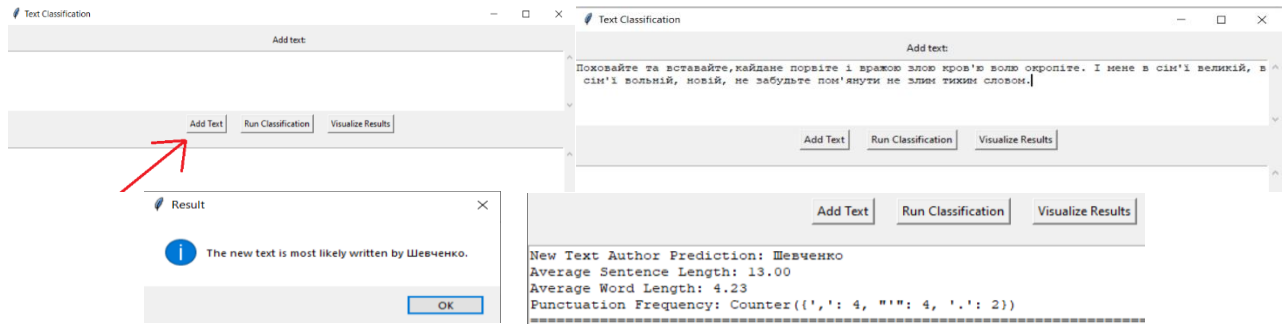


Fig.8. The stylometric analysis

Next, analyse the text written by Ivan Franko:

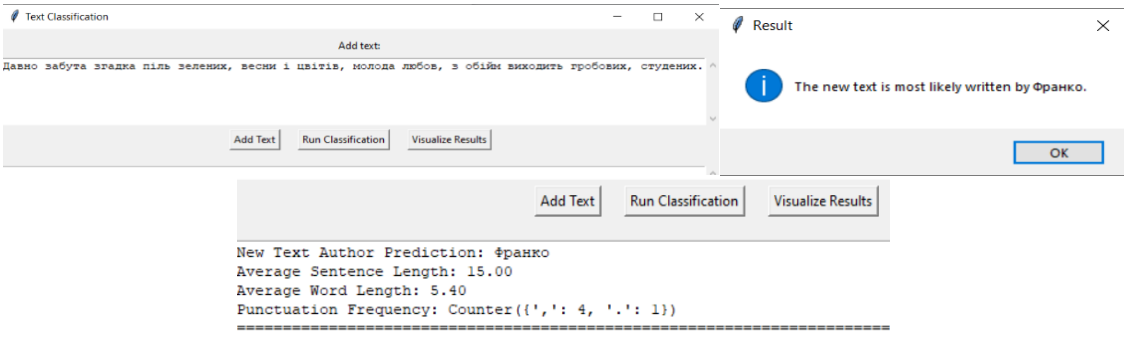


Fig.9. The stylometric analysis

Next, take the fragment of Kulish's and Lesya Ukrainka’s work for analysis:

```
Average Sentence Length: 15.00
Average Word Length: 5.40
Punctuation Frequency: Counter({',': 4, '.': 1})
==================================================
New Text Author Prediction: Куліш
Average Sentence Length: 15.00
Average Word Length: 4.80
Punctuation Frequency: Counter({',': 2, ':': 1, '.': 1})
==================================================
```

```
Average Sentence Length: 15.00
Average Word Length: 4.80
Punctuation Frequency: Counter({',': 2, ':': 1, '.': 1})
==================================================
New Text Author Prediction: Українка
Average Sentence Length: 17.00
Average Word Length: 4.29
Punctuation Frequency: Counter({',': 2, ':': 1, '!': 1})
==================================================
```

Fig.10. The stylometric analysis

Then check Shevchenko's, Franko's and text for authorship:

```
==================================================
New Text Author Prediction: Шевченко
Average Sentence Length: 4.00
Average Word Length: 4.83
Punctuation Frequency: Counter({',': 2, '!': 1})
==================================================
```

```
==================================================
New Text Author Prediction: Франко
Average Sentence Length: 11.00
Average Word Length: 3.36
Punctuation Frequency: Counter({',': 1, '.': 1})
==================================================
```

Fig.11. The stylometric analysis

Next, the text for the authorship of Kulish's and Ukrainka:

```
==================================================
New Text Author Prediction: Куліш
Average Sentence Length: 11.00
Average Word Length: 4.46
Punctuation Frequency: Counter({',': 2, '.': 1})
==================================================
```

```
==================================================
New Text Author Prediction: Українка
Average Sentence Length: 10.00
Average Word Length: 3.90
Punctuation Frequency: Counter({',': 2, '.': 1})
==================================================
```

Fig.12. The stylometric analysis

Then, the text for Shevchenko's and Kulish's authorship:

```
==================================================
New Text Author Prediction: Шевченко
Average Sentence Length: 8.00
Average Word Length: 4.88
Punctuation Frequency: Counter({',': 1, '!': 1})
==================================================
```

```
==================================================
New Text Author Prediction: Куліш
Average Sentence Length: 6.00
Average Word Length: 5.33
Punctuation Frequency: Counter({',': 2, '!': 1})
==================================================
```

Fig.13. The stylometric analysis

In the next step, it is possible to analyse the metrics and visualizations of my program's primary dataset. To do this, it is needed to click on the Run Classification button:

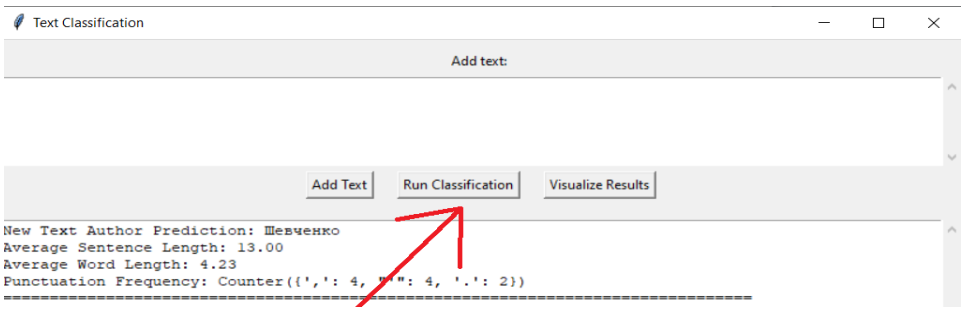


Fig.14. Run the Classification button

Add Text | Run Classification | Visualize Results

TF-IDF Report:

	precision	recall	f1-score	support
Куліш	1.00	0.38	0.55	8
Українка	0.33	0.08	0.13	12
Франко	0.44	0.50	0.47	8
Шевченко	0.64	1.00	0.78	23
accuracy			0.61	51
macro avg	0.60	0.49	0.48	51

Add Text | Run Classification | Visualize Results

Word2Vec Report:

	precision	recall	f1-score	support
Куліш	0.00	0.00	0.00	8
Українка	0.00	0.00	0.00	12
Франко	0.00	0.00	0.00	8
Шевченко	0.45	1.00	0.62	23
accuracy			0.45	51
macro avg	0.11	0.25	0.16	51

Fig.15. Results classification

Also, it is possible finally to see visualizations of these metrics. To do this, it is necessary to click Visualize Results:

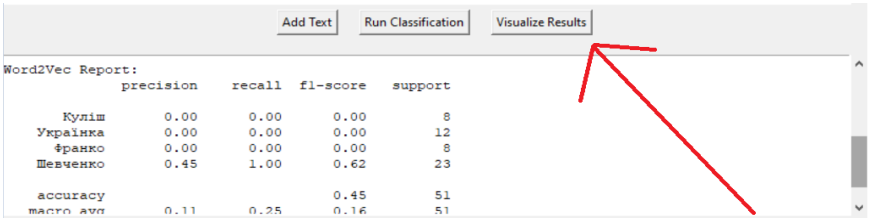


Fig.16. Visualize results button

The analysis was performed on two datasets: the first time, a small dataset was used, and the second time, the data was added. So, it is seen that with an increase in the amount of data, the level of metrics of predictions of different authors changes: so last time, Shevchenko's authorship was more predictive in terms of metrics than Kulish's, and this time it is possible to see a change in the fact that Kulish is more likely to predict than Shevchenko.

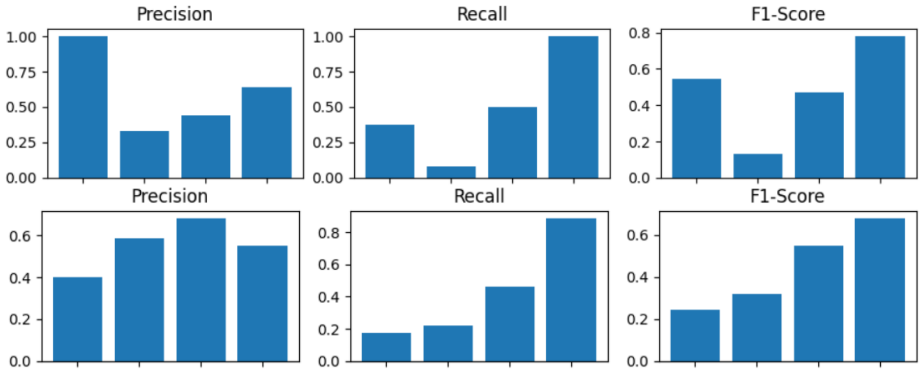


Fig.17. Results analysis (First version and After adding data) for TF-IDF (from left to right and top to bottom Panteleimon Kulish, Lesia Ukrainka, Ivan Franko and Taras Shevchenko)

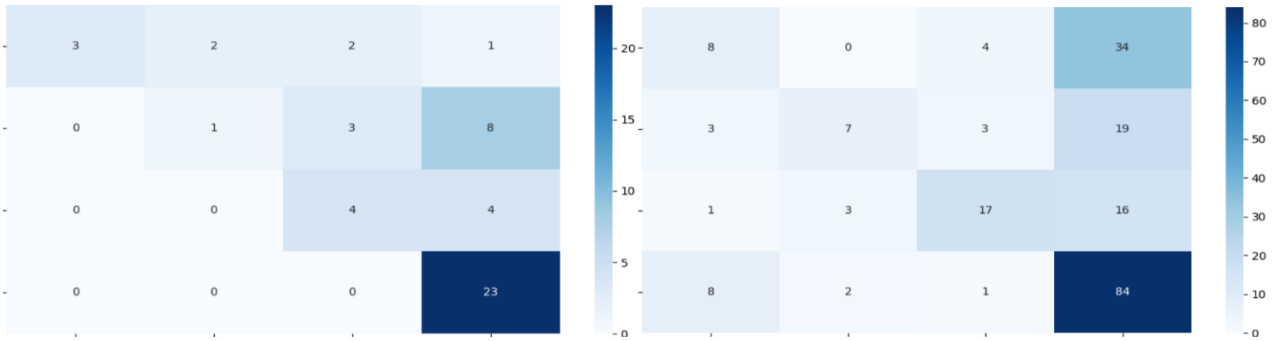


Fig.18. Confusion matrixes (First version and After adding data) for TF-IDF (from left to right and top to bottom Panteleimon Kulish, Lesia Ukrainka, Ivan Franko and Taras Shevchenko)

In conclusion, the creation of a program that reliably identifies the author of the work written by an author of literature is a significant achievement in the field of natural language processing and machine learning. Thanks to the use of advanced algorithms and large amounts of data for training, the program demonstrated high accuracy in recognizing texts of various origins, reducing the risk of false conclusions. As a result, this program can be a valuable tool in many fields where author identification and stomatal analysis are essential.

4.7. The Description of the System Architecture

The user interface (UI) is an essential part of the AuthorshipAnalyzer software, which provides user interaction with other components of the system for analyzing text authorship. It consists of several main elements that provide convenient loading of texts, setting up the analysis and viewing the results. The system interface is intuitive and allows training users to easily use the program to load texts, run the analysis and view the results in a convenient format. The main functions of the interface:

1) Loading text for analysis. The system supports loading texts from files of various formats, such as TXT, DOCX, and PDF. It makes the program universal for different users and areas of application. After loading, the text undergoes pre-processing, which includes removing unnecessary characters, normalization and formatting. Structural features, such as sentence length and frequency of use of individual words, are also determined.

2) Selecting a model for author attribution. The user can select a model for determining authorship, for example,

TF-IDF or Word2Vec, depending on the analysis requirements. TF-IDF – vector representation of text based on word frequency. Word2Vec – construction of vector representations of words for comparing stylistic characteristics. Deep neural networks (CNN-LSTM, BiLSTM) – allow you to analyze texts at a deeper level and find hidden patterns in the writing style of authors. Models for processing large amounts of data ensure the accuracy of determining authorship. There is support for model adaptability. The system trains models on large data sets, which allows them to take into account a wide range of stylistic features and improve the accuracy of the analysis.

3) Displaying results. The results of author attribution are displayed on the screen, where the user can see the predicted author of the text and the level of confidence of the model (degree of correspondence). In addition, the results can be displayed in the form of graphs and charts, which facilitates the analysis of the obtained data. The level of confidence of the model is assessed by the system of the accuracy of the prediction in per cent. Report generation is supported through the ability to export results to PDF or DOCX for further use. Visualization of results:

- Construction of graphs and charts (e.g. Confusion Matrix) for convenient analysis;
- Interactive visualizations that allow you to drill down into the analysis results.

Additional interface elements

- The main program window contains essential functions, such as loading text, selecting an analysis model, and viewing results.
- The navigation menu provides quick access to various sections, including visualization, model settings, and analysis history.
    - Control buttons: Add Text (loading text), Run Classification (running analysis), and Visualize Results (viewing a graphical representation of results).

The user interface is implemented using Tkinter, a library that creates a graphical interface in Python. NumPy (for calculations on data arrays), Pandas (for processing tabular data), and scikit-learn (for machine learning and text vectorization) are additionally used. Thus, the AuthorshipAnalyzer system interface provides convenient access to the main functions of authorship analysis, including loading texts, selecting a model, visualizing results, and generating reports.

The NLP module plays a key role in the process of text analysis, providing their pre-processing, vectorization, and analysis of stylistic features. Thanks to the use of modern natural language processing algorithms, the system can effectively process texts by different authors and identify unique characteristics of their style.

1) Text preprocessing is performed in several stages:

- Tokenization – splitting the text into individual words or phrases for further analysis. The system uses the NLTK library, which supports tokenization for English, and its algorithm is used for Ukrainian.
- Lemmatization – reducing words to their base form (for example, "бигав" → "биг"). Pymorphy2 is used for this, which provides support for the Ukrainian language.
- Stopword removal – eliminating common words (for example, "и", "але", "це") that do not have a significant impact on determining authorship. A unique list of Ukrainian stopwords is used.
- Text cleaning – removing numbers, punctuation marks, words less than three characters long, and incorrect characters.

2) The following text vectorization methods are used to represent text in the form of numerical vectors:

- TF-IDF (Term Frequency-Inverse Document Frequency) – a statistical method for assessing the importance of words in a document relative to the entire corpus of texts. This approach is used to analyze the frequency of words and their significance.
- Word2Vec – creating vector representations of words, which allows you to take into account semantic and syntactic relationships between words. The system uses a pre-trained Google Word2Vec model containing 3 million words with a vector representation of dimension 300.
- BERT embeddings – a more complex method based on transformers, which allows you to analyze contextual relationships between words in a sentence and improves the accuracy of author attribution.

3) Analysis of stylistic features of the text

- Lexical analysis – determining the frequency of use of words, unique terms, and stylistic turns.
- Syntactic analysis – analysis of sentence structure, phrase length, use of punctuation marks.
- Stylometric analysis – identification of unique features of the author's writing style (rhythm, paragraph structure, literary devices). Classification algorithms based on machine learning are used.

Thus, the NLP module provides a complete cycle of text processing – from its normalization to analysis of the author's stylistic characteristics, which significantly increases the accuracy of determining authorship.

Machine learning modules are key components of the AuthorshipAnalyzer system, which provides a classification

of text authorship based on stylistic features. They include machine learning and deep learning algorithms that allow for adequate recognition of the writing styles of different authors.

1) Using models for authorship classification. The system uses three main approaches to author attribution:

- The Naive Bayes classifier uses a probabilistic approach to classify texts based on term frequency. It has a high computational speed and low resource requirements, which makes it practical for large amounts of text data.
- The Support Vector Modeling (SVM) method creates a hyperplane that best separates text classes in a multidimensional space. It is suitable for processing data with a large number of stylistic features and provides high accuracy of author attribution.
- Deep neural networks (LSTM, CNN-LSTM) demonstrate the best results in author attribution accuracy – up to 95% for English and 91% for Ukrainian. LSTM (Long Short-Term Memory) is a recurrent neural network that takes into account word order and context in texts, which allows for the recognition of stylistic patterns. The combined CNN-LSTM model combines convolutional neural networks (CNN) for feature extraction and LSTM for analyzing text sequences.

2) Training models on text corpora

- Input data preparation. TF-IDF, Word2Vec, and GloVe are used for text vectorization. Input data is cleaned and lemmatized before training models.
- Data distribution. Text corpora are divided into training (80%) and test (20%) sets to evaluate model performance. Cross-validation and the K-fold partitioning method are used to test models.
- Deep learning. LSTM networks are trained using the Adam optimizer and the binary cross-entropy loss function. Dropout (0.5) is used for regularization to prevent overtraining of the model.

3) Prediction generation and model performance evaluation.

- Prediction formation. The model accepts the input text, converts it into vector form and passes it to the classification algorithm. The output is the predicted author of the text and the level of confidence of the model in the prediction.
- Evaluation metrics: Accuracy (total proportion of correct predictions), Recall (proportion of correctly predicted texts of a specific author) and F1-score (harmonic mean between accuracy and completeness). ROC-curve and AUC are used to evaluate the performance of models in multi-class classification.

Machine learning modules in AuthorshipAnalyzer allow you to effectively analyze the authorial styles of texts using modern algorithms. Deep neural networks, especially Bidirectional LSTM and CNN-LSTM, demonstrate the best results in author attribution and can be applied to complex text analysis tasks.

The results processing module provides analysis, visualization and integration of the obtained data, allowing users to explore the results of authorship attribution of texts in detail. The main functions of this module include report generation, confusion matrix construction and the ability to integrate with external services.

1) Report generation

- Generation of detailed reports. The system generates reports based on the obtained results, which can be saved in PDF and DOCX formats. It makes it easy to share the results with colleagues or use them in further research.
- Report structure. The description of the used models contains details about the classification algorithms (Naïve Bayes, SVM, LSTM). Statistical analysis of accuracy includes the main metrics (Precision, Recall, F1-score). Graphical representations: tables and graphs are added to illustrate the distribution of predictions.

2) Visualization of results in the form of a confusion matrix

- The confusion matrix allows you to assess how accurately the model classifies the authorship of texts. The visualization of results includes heat maps (displaying correct and incorrect predictions of models) and graphs that compare models by metrics (accuracy, F1-score, ROC curve).
- Dynamic analysis - visualization of changes in accuracy after increasing the size of the training corpus of texts. There is also a demonstration of how adding new data affects the results of author attribution.

3) Integration with external services

- Connection to plagiarism databases. The module allows you to integrate authorship analysis with plagiarism detection services, which improves the accuracy of recognizing borrowed texts.
- Extended API capabilities. It is possible to export data to external databases for further analysis and comparison. There is also support for REST API for connecting to third-party text analysis services.
- Automated style checking. The module can be used in editorial systems and educational platforms to assess the stylistic characteristics of texts. It is planned to expand integration with university repositories to check academic works.

Thus, the results processing module in AuthorshipAnalyzer allows you not only to obtain accurate authorship attribution results but also to deeply analyze them using detailed reports, visualizations, and integration with other services.

This structure allows the system to work efficiently with large amounts of text and supports real-time authorship analysis.

5. Results

5.1. Development of a Classification Model

Choosing a data corpus. Amazon's product review is a favourite target for evaluating sentimental analysis algorithms. Customer review analysis has many practical applications, as millions of people plan their purchases every day based on product ratings generated from reviews. In this project, we're using a set of product reviews from Amazon's dataset compiled by Mark Dredze and John Blitzer [21], which contains reviews from 25 different categories. We will consider reviews with a rating of 3 or more (out of 5) positive and negative, respectively, ratings less than 3. The total number of reviews that we will use for training is about 30 thousand. To teach the Ukrainian model, we used translated Amazon reviews from English. For this purpose, a unique script was written that makes requests to the Google Translate API and extracts the translation. It, of course, is not the best choice, but we could not find ready-made datasets in Ukrainian.

Data processing. At this point, we converted the data from the input XML format to the pandas dataframe representation for ease of processing, namely string normalization, tokenization, stop word removal, stemming, and lemmatization using NLTK library tools. After that, we carried out another operation to clean up unnecessary characters, which included deleting numbers, words less than three characters, and non-Cyrillic characters for the Ukrainian model. Since the NLTK library does not support the Ukrainian language, we went the other way. To remove stop words, we used a relatively extensive list of Ukrainian stop words [22]. For stemming, we wrote our implementation of PorterStemmer. For lemmatization, we used the solution offered by the pymorphy2 library, where there is support for Ukrainians, but so far, it has only been at an experimental level. The visualization of the most used words after processing for English and Ukrainian corpora can be seen in Fig. 19-20. For visualization, we used the WordCloud and Seaborn libraries.

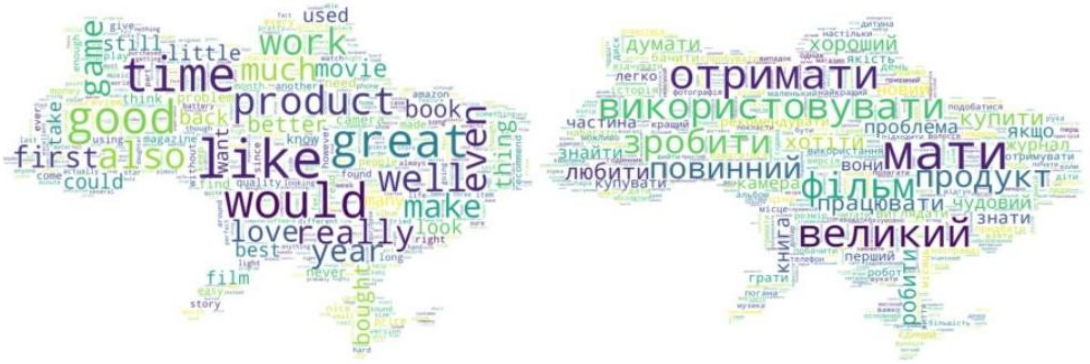


Fig.19. Cloud of the most prominent used words in the reviews for the English (a) and Ukrainian (b) dataset

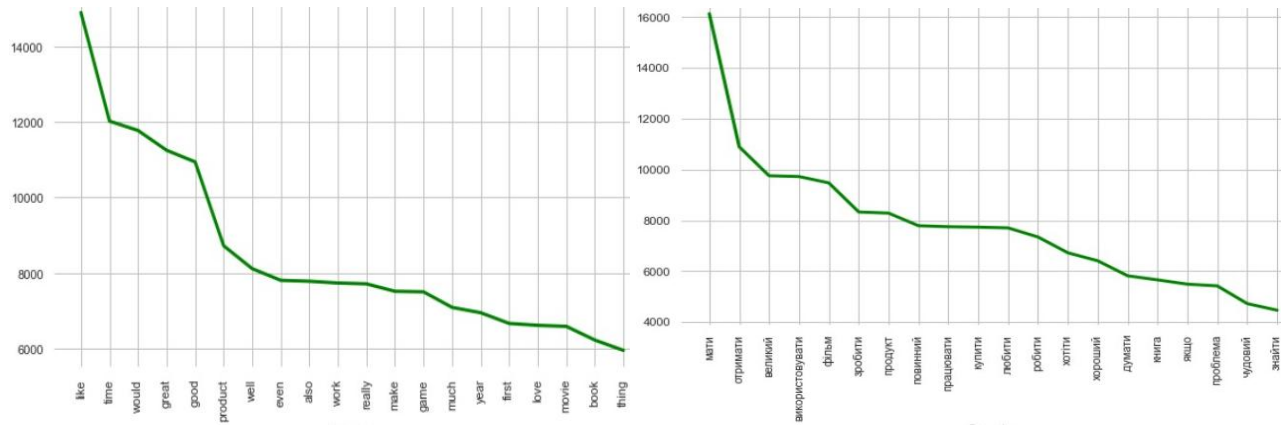


Fig.20. Graph of the most significant words used and their number in the reviews for the English (a) and Ukrainian (b) dataset

Next, we turned the study documents into tuples. Tuples are immutable, meaning they have a fixed size in nature. While lists are dynamic, they can be modified. Next, we won't need to add or remove elements, so tuples are a great

option that, in addition, works faster than lists. The work on creating the model was divided into three parts. However, in each, we needed to work with the data that we processed here, so we used the pickle library, designed for serialization/deserialization, so that we could then use this data in other parts of the project without any problems. In the next step, we separated the dataset to perform an unbiased evaluation of the model and identify undertraining or overtraining. To do this, we used the scikit-learn or sklearn module. It has many packages for data science and machine learning, but for now, we will focus on the model selection package, specifically the train_test_split() function. At this point, our data preparation was completed, and we moved on to creating, training and testing classifiers.

5.2. Detailed Description of the Machine Learning and NLP Methodology

The study used machine learning and natural language processing (NLP) methods to determine the authorship of texts based on stylistic, lexical and semantic features. The methodology includes several key stages:

Stage 1. Text pre-processing

Step 1. Text cleaning. Removing special characters, numbers, and HTML tags and converting the text to lowercase.

Step 2. Tokenization. Breaking the text into words or phrases for further processing. NLTK was used for English texts and pymorphy2 for Ukrainian.

Step 3. Lemmatization and stemming (reducing words to base forms to generalize word forms). WordNetLemmatizer (NLTK) was used for English and pymorphy2 for Ukrainian.

Step 4. Stop word removal. Built-in stop-word lists were used for both languages.

Step 5. Noise filtering. Rare words that occur in less than 5% of texts are removed.

Stage 2. Text vectorization (Feature Extraction)

Step 1. TF-IDF (Term Frequency-Inverse Document Frequency) determines the significance of words in the context of the document. Used for classical machine learning algorithms (SVM, Naïve Bayes, Random Forest).

Step 2. Word Embeddings (Word2Vec, GloVe) create multidimensional vector representations of words. Word2Vec (300-dimensional vector) is used for English, and GloVe is used for Ukrainian.

Step 3. Analysis of stylistic features (Stylometry) such as sentence length, average word length, frequency of punctuation marks, as well as the frequency of parts of speech (nouns, verbs, adjectives) and unique words in the text (lexical diversity).

Stage 3. Selection of machine learning models

Step 1. Classical ML models. The Naïve Bayes classifier is simple, fast, and efficient for large amounts of text. The Support Vector Machine (SVM) works well with high-dimensional text data. For Random Forest, an ensemble of decision trees is used to improve classification accuracy.

Step 2. Deep Learning. LSTM (Long Short-Term Memory) works well with texts due to context and word order analysis. CNN-LSTM (Combined Network) combines Convolutional Neural Networks (CNN) for local pattern extraction and LSTM for sequence processing.

Step 1. Model hyperparameters. Grid Search was used to select the optimal parameters of SVM and Random Forest. For LSTM and CNN-LSTM, Adam Optimizer was used, Batch Size = 128, Learning Rate = 0.001.

Stage 4. Model performance evaluation based on evaluation metrics:

Step 1. Accuracy – the total proportion of correctly predicted authors.

Step 2. Precision – the proportion of correct optimistic predictions.

Step 3. Recall – the proportion of correctly found authors.

Step 4. F1-score – the average of precision and recall.

Table 5. Performance results

Model	Accuracy	F1-міра
Naïve Bayes	80.38%	78.4%
SVM	81.26%	80.58%
Random Forest	79.54%	79.36%
LSTM	90.93%	89.87%
CNN-LSTM	94.48%	93.31%

CNN-LSTM showed the best result (94.48% accuracy). SVM is well suited for smaller data sets. Naïve Bayes is the fastest but less accurate method.

Using TF-IDF, Word2Vec, and GloVe allowed us to obtain multidimensional representations of texts. The

combination of ML and deep learning provided the best accuracy and speed of analysis. CNN-LSTM is the optimal model for the problem of authorship identification. The developed methodology provides a practical analysis of text authorship, which makes the system suitable for literary studies, plagiarism analysis and author attribution.

5.3. Training and Comparison of Machine Learning Classifiers

To begin, we took a quick approach to naïve Bayesian classification to see if our post-processing data could give us good results so that we could use it for more complex classifiers. We took the implementation of the naïve Bayesian classifier from the classifying module of the NLTK library. The results are good enough for us to continue researching. For further work, we need to create vectors of words. For simplicity, we used a pre-trained model. Google was able to train the Word2Vec model on a vast Google News dataset that contained more than 100 billion different words. Google created 3 million vector words from this model, each with a dimension of 300 [23]. We used these vectors to produce the data in English. We also found a Word2Vec model with Ukrainian vector words, each of which has a dimension of 300. This vector word sample is based on fiction. We chose a lemmatized version of this model that was ideal for our processed data [24]. The idea behind word2vec is as follows:

- Take a 3-layer neural network (1 input layer + 1 hidden layer + 1 output layer);
- Give her the floor and teach her to predict the next word;
- Delete the last (output layer) and keep the input and hidden layers;
- Enter a word from the vocabulary. The result provided on the hidden layer will be a "word nesting" in the input word.

After uploading the model, we started analyzing our vectors. Our first approach was the medium word embedding model. The essence of this approach is to take the average of all vector words from a sentence to get one 300-dimensional vector that represents the tone of the entire sentence that we feed the model and try to get a quick result. For further work, we divided our enclosure into training, testing and development kits. Then, they proceeded to the analysis of various classification models.

KNN model. When developing, we used a ready-made classifier solution from the KNeighborsClassifier module of the sklearn library. We created a KNN classifier object by passing the number of neighbour arguments to the KNeighborsClassifier() function. One way to help find the best neighbour value is to plot the neighbour values and the corresponding error rate for the data set. We plotted the mean error for the predicted values of the test set for all neighbour values from 1 to 25 (Fig. 21). To do this, the average error value was calculated for all predicted values, where the neighbour is in the range from 1 to 25.

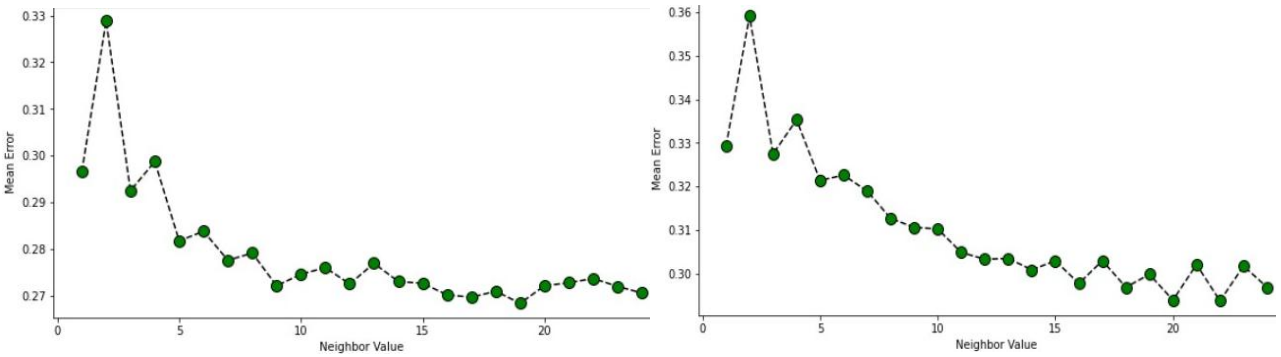


Fig.21. Neighbor value error rate for English (a) and Ukrainian (b) dataset

As we can see from Fig. 3.3, it was best to take k=20 for the Ukrainian model, and k=19 for the English model. Classification reports in the form of a heatmap with all the essential characteristics of model evaluation (accuracy, recall, F-measure) are shown in Fig. 22.

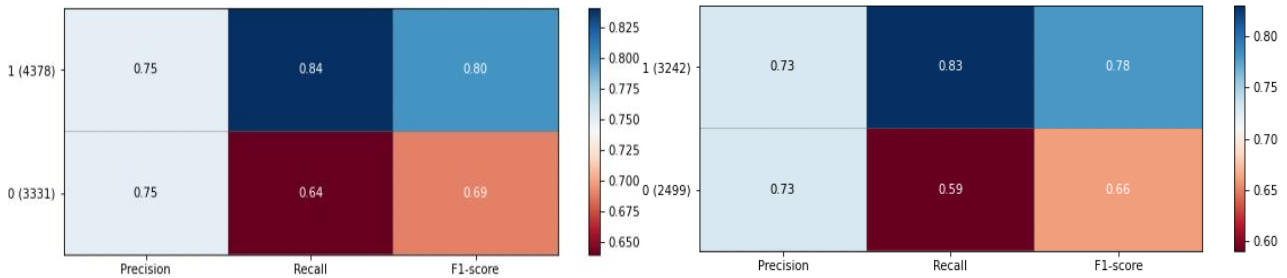


Fig.22. Classification report for k-nearest neighbours for English (a) and Ukrainian (b) dataset

Logistic regression. To implement this algorithm, we took a ready-made solution from the LogisticRegression module of the sklearn library. Classification reports for this method are shown in Fig. 23.

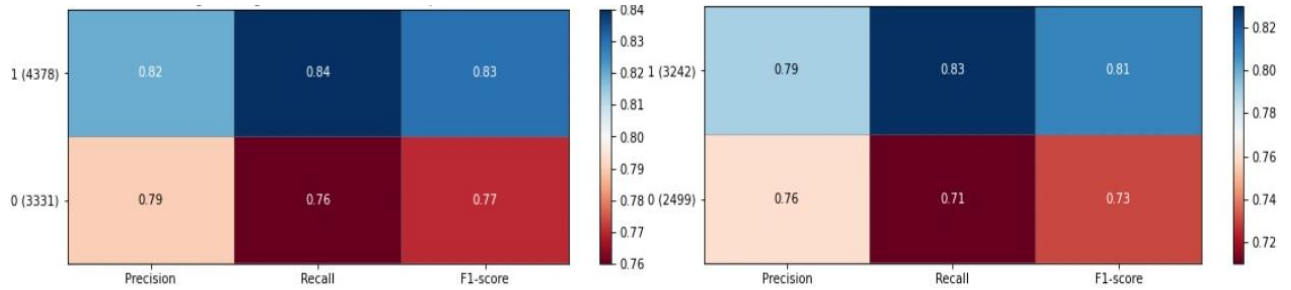


Fig.23. Classification Report for Logistic Regression for English (a) and Ukrainian (b) Dataset

Random forest. The sklearn library provides a ready-made solution for this algorithm as well, which we took from the RandomForestClassifier module. Classification reports in the form of a heatmap are shown in Fig. 24.

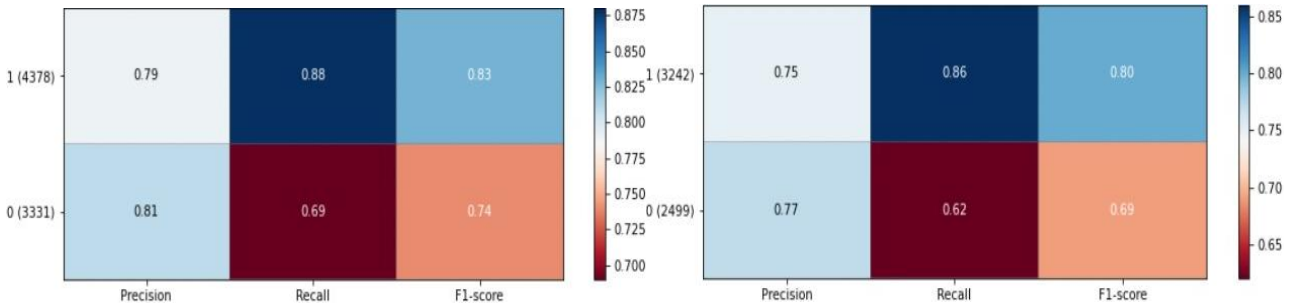


Fig.24. Classification report for random forest for English (a) and Ukrainian (b) dataset

Support vector machine. The implementation of this algorithm was taken from the SVM module of the sklearn library. Classification reports for the method under consideration are shown in Fig. 25.

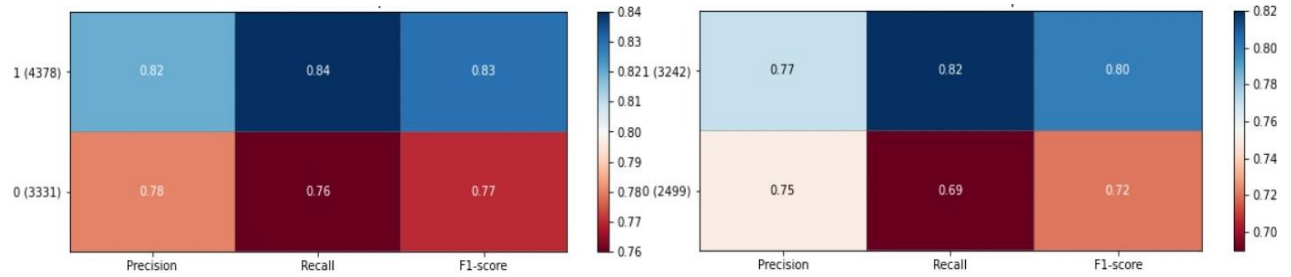


Fig.25. Classification report for the reference vector method for the English (a) and Ukrainian (b) dataset

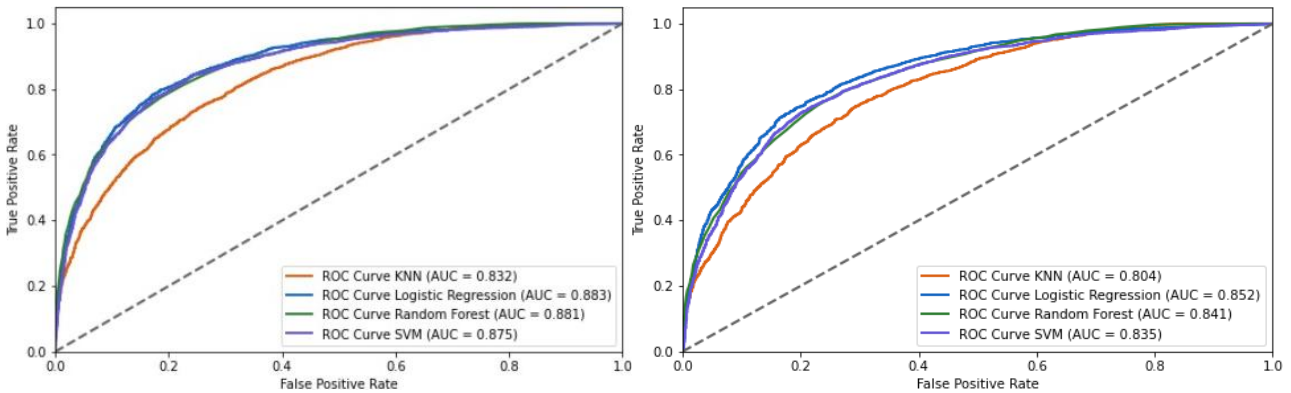


Fig.26. Comparison of Machine Learning Classifier Results Using ROC Curve for English (a) and Ukrainian (b) Dataset

Comparison of machine learning classifiers. A valuable tool for predicting the probability of a binary outcome is the receiver performance curve, or ROC curve, which was used to compare our classifiers (Fig. 26). Based on the data of the study, as well as the comparative analysis in Fig. 26, we can conclude that the best is the logistic regression model

with AUC = 85.2% for the Ukrainian model and 88.3% for the English model, and respectively an accuracy indicator of 77.7% for the Ukrainian model and 80.58% for the English model. The accuracy of the classifiers is shown in Table. 6.

Table 6. Accuracy of machine learning models

Data Corpus	Naïve Bayesian classifier	KNN classifier	Logistic Regression	Random Forest	SVM
Ukrainian language	80.38%	72.91%	77.7%	75.74%	75.05%
English language	81.26%	75.41%	80.58%	79.54%	79.36%

5.4. Training and Comparison of Deep Learning Models

Preparation of input data. Google's Word2Vec model is complicated to process with the RAM we have for deep learning tasks, so we used a smaller model, namely GloVe [25]. The model contains 400000 words, so we got a 400000x100 embedding matrix that contains all the values of vector words. For the Ukrainian model, we managed to keep the Word2Vec model (because it was smaller), but this was achieved by fully loading the operating memory. To configure the models, we needed an input dimension parameter, which is determined by the maximum number of words to consider. To do this, we analysed and determined the optimal number of words (Fig. 27). Based on the histogram data (Figure 3.9), we can confidently say that most reviews will be less than 100 words, which is the maximum value of the sequence length we have established. Next, we wrote our function for embedding sentences. We restored the embedding of each sentence with the middle vector that composes it: if the word is in the word2vec dictionary, we added its vector representation. If not, we added a zero vector. In the next step, we aligned the sentence length to 100 words, and at the end, we said the sentence vector to list X. To list Y, we added a class label. For further work, we divided our enclosure into training, testing and development kits and moved on to implementing deep learning models.

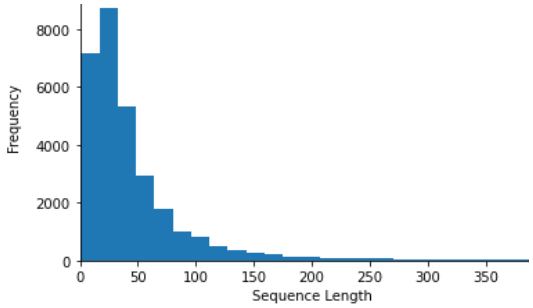


Fig.27. Average word count in reviews

LSTM (Stochastic Gradient Descent) model. For the embed layer, you need to specify two main data parameters: input and output dimensions. The input dimension is defined as the maximum number of words to consider in the view (for our work, a value of 100 was accepted, as we noted above). The output dimension is defined as the size of the word attachment (in our country, it is 300 for the Ukrainian model and 100 for the English model). The LSTM layer has the same two parameters as the embedded layer. The input dimension is defined as the input and output dimensions of the embed layer. The original dimension was set to 128 (this value was accepted as optimal due to several experiments with the model). The density layer also has two parameters: the dimension of the output and activation. The output dimension represents the number of initial dimensions of the model and is selected according to the specific task. In our case, it is necessary to set the value to 1. Activation is set to the sigmoid function. In general, the network has one input layer, a hidden layer with 128 units, and one output layer. The configuration for training also includes an Adam optimizer, a loss function (root mean square error), and a metric (binary precision). The number of epochs to learn in each model is 20. The graph of model losses is shown in Fig. 28. The classification report in the form of a heatmap is shown in Fig. 29.

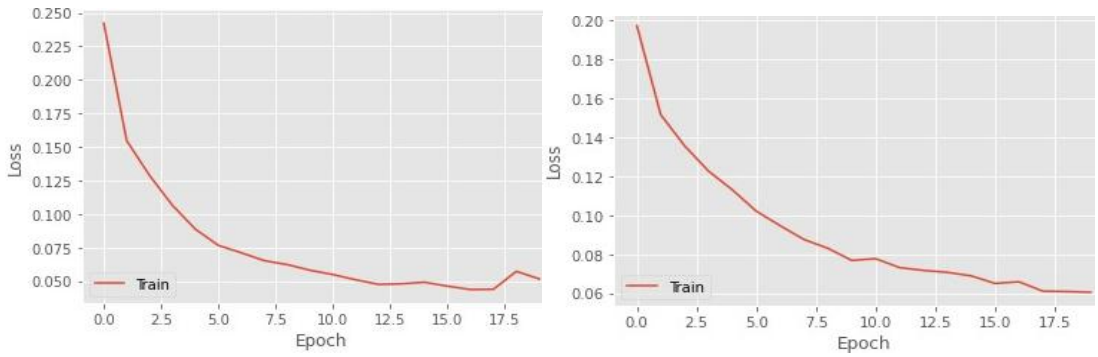


Fig.28. LSTM (Stochastic gradient descent model) Losses for english (a) and Ukrainian (b) Dataset

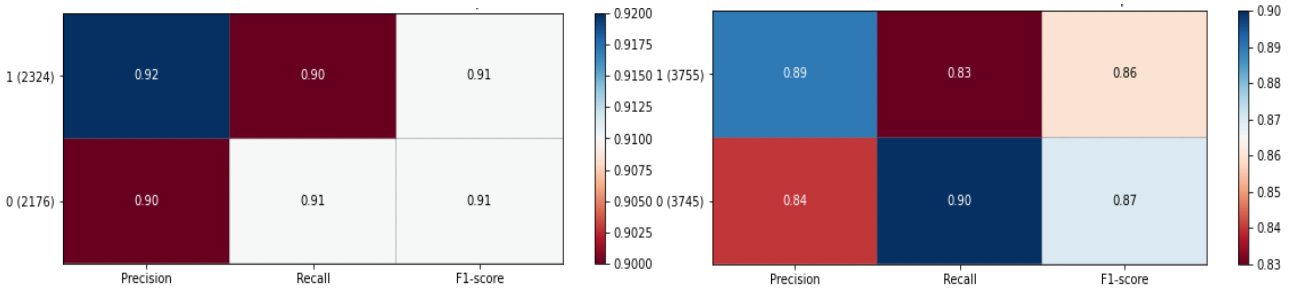


Fig.29. Classification report for the LSTM (Stochastic gradient descent) model for the English (a) and Ukrainian (b) dataset

LSTM (Mini Batch Gradient Escapement) model. This model has the same parameters as for the case of the previous model, except for the package size, which is set to 128 (studies have shown that this is the most optimal size for this model in Fig. 30). Classification report in the form of a heatmap shown in Fig. 31.

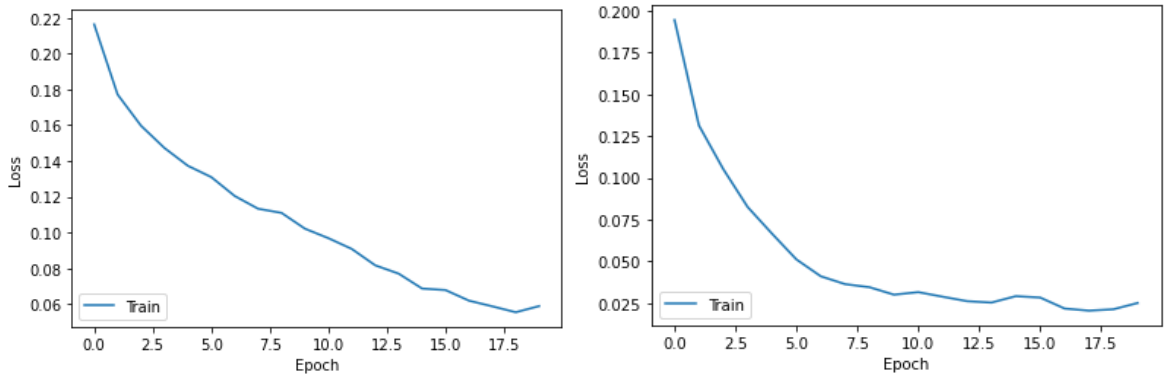


Fig.30. Losses of the LSTM (Mini-batch gradient descent) model for the English (a) and Ukrainian (b) dataset

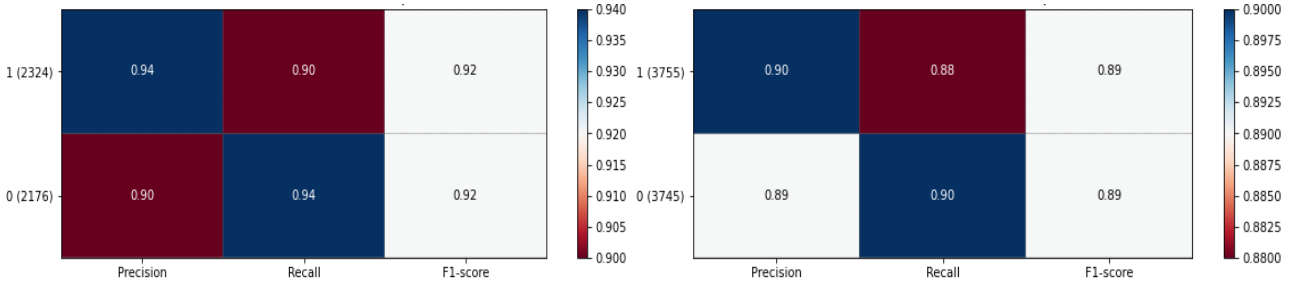


Fig.31. Classification report for the lstM (Mini-batch gradient descent) Model for english (a) and Ukrainian (b) Dataset

LSTM Stacked Layers have the same configurations for training, embedding layer and density layer as in previous models. The package size also remained at 128. The output dimension of the first layer of LSTM is 100 units, for 2 and 3 times 32. Also, the first two layers of the LSTM stack have a parameter return_sequence=True, which returns the last output in the original sequence rather than the complete sequence. The loss graph for this model is shown in Fig. 32. The classification report in the form of a heatmap is shown in Fig. 33.

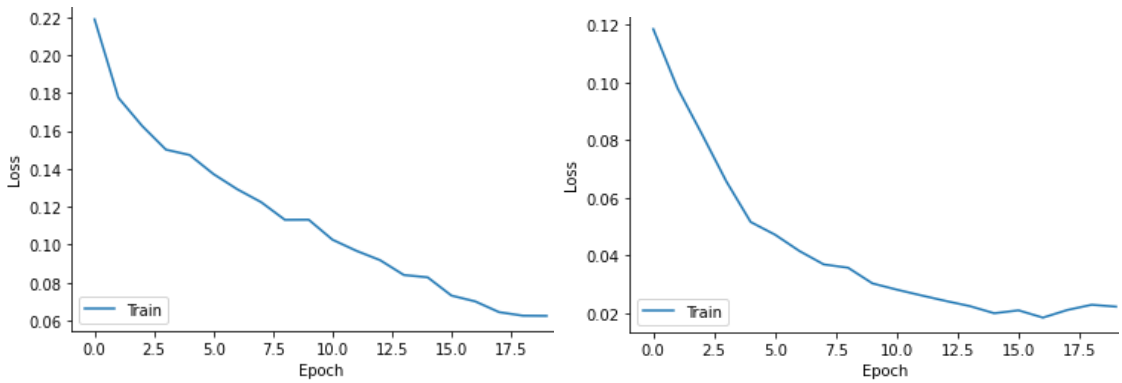


Fig.32. Model losses from LSTM stacked layers for english (a) and Ukrainian (b) Dataset

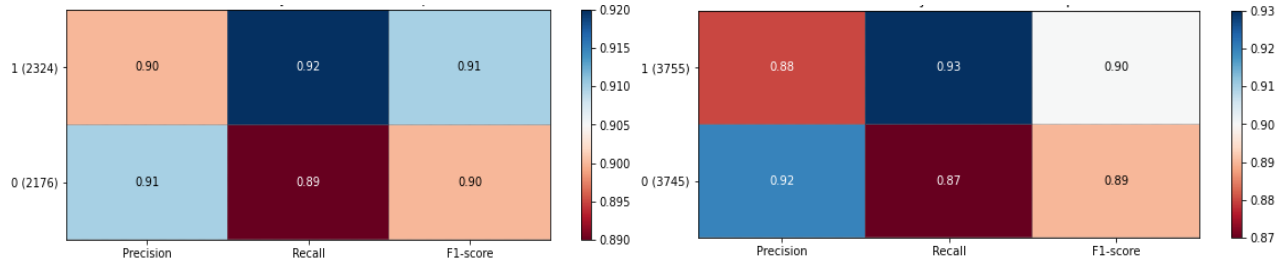


Fig.33. Classification report for the LSTM stacked layers model for english (a) and Ukrainian (b) Dataset

Bidirectional LSTMs. As with previous models, the settings for the embed layer and the density layer remain unchanged. The output dimension of both layers of bidirectional LSTM is 64 units. Also, an exclusion layer has been added for this model, the purpose of which is to accidentally set part of the input units to 0 during each update during training, which helps prevent the model from being over-trained. The optimal value of the parameter of the proportion of units to be removed is 0.5 (50%) [26]. For the loss function in the training configuration of this model, binary cross-entropy was chosen [26]. The loss graph for this model is shown in Fig. 34. The classification report in the form of a heatmap is shown in Fig. 35.

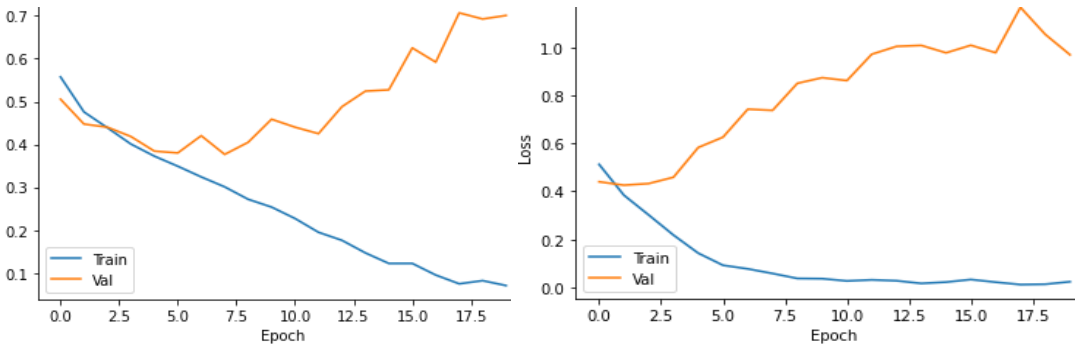


Fig.34. Bidirectional LSTM model losses for english (a) and Ukrainian (b) dataset

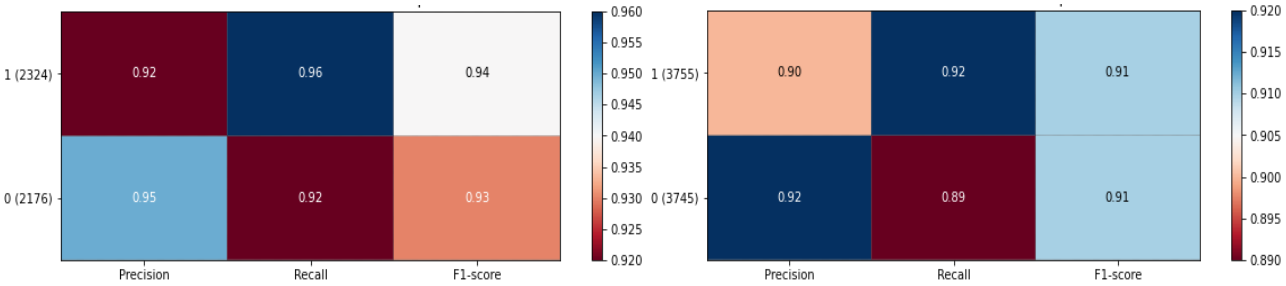


Fig.35. Classification report for the bidirectional LSTM model for the english (a) and Ukrainian (b) dataset

Combined Network (CNN-LSTM). The embedding and density layers should remain unchanged parameters for training, such as in the previous model. The differences are in the convolution, LSTM, exclusion, and maximization aggregation layers. There are five essential parameters in convolutional layers: number of convolutional cores, filter length, and activation. The number of convolutional cores is selected as 128, the filter length is 4, and activation is defined as a function of ReLU (layer of truncated linear nodes). The max-pooling layers have only one parameter, which is the length of the pool. Pool length is defined as the size of the region to which maximization aggregation is applied. In this architecture, this parameter was assigned a value of 3. The output dimension of the LSTM layer is 32 units. For the exclusion layer, we chose a value of 0.2 for the fraction of units to be removed since it showed better accuracy. The loss graph for this model is shown in Fig. 36. The classification report in the form of a heatmap is shown in Fig. 37.

Comparison of Deep Learning Models. The comparison for deep learning models was also based on the ROC curve, and the results are presented in Fig. 38. As we can see, all models performed incredibly well, achieving 90-95% accuracy for both models and for both data sets. Of course, the English models showed slightly better results because our dataset in Ukrainian was formed by the translation of reviews into English, which entails specific problems that we have already discussed. The models that showed the best results for both English and Ukrainian texts are the bidirectional LSTM and the CNN-LSTM combined model, which is not surprising since the combined models tend to give the best results. The same is true for bidirectional LSTM models, which perform well in practice [27-28]. We will

use these two models for sentiment analysis on our web service. The accuracy of the models is described in Table 7.

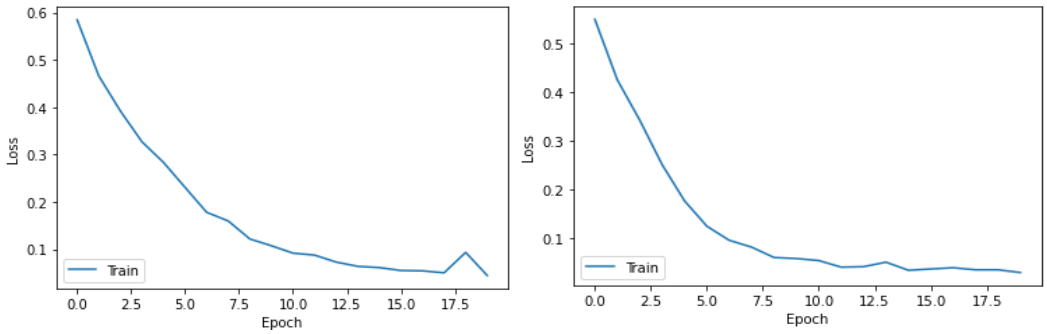


Fig.36. Losses of the combined network (CNN-lstm) model for the English (a) and Ukrainian (b) dataset

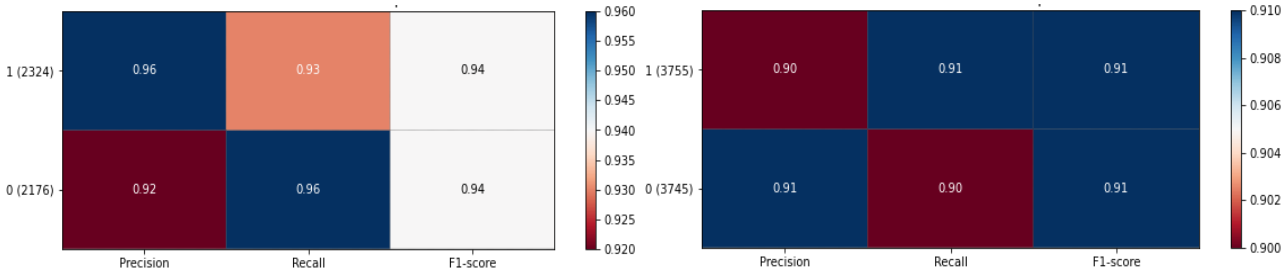


Fig.37. Classification report for the combined network (CNN-lstm) model for the english (a) and Ukrainian (b) dataset

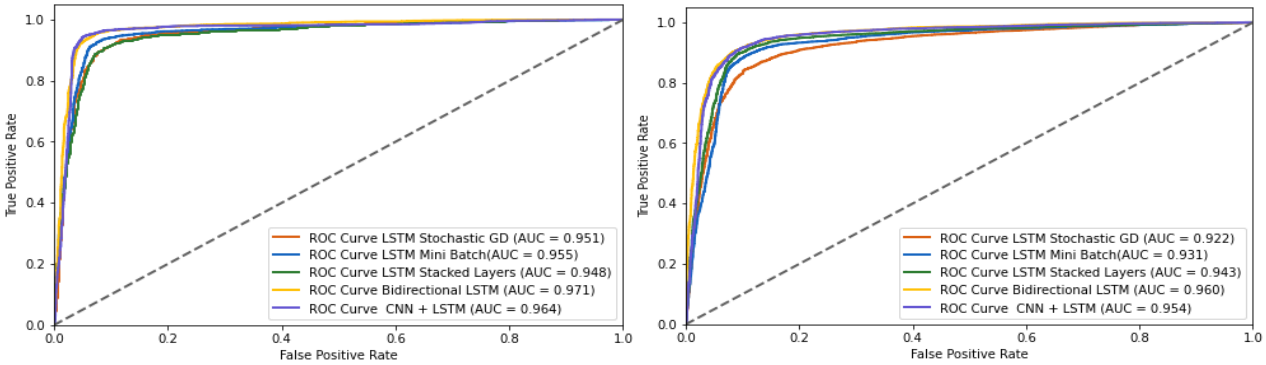


Fig.38. Comparison of deep learning model results using ROC curve for english (a) and Ukrainian (b) Dataset

Table 7. Accuracy of deep learning models

Data Corpus	LSTM (Stochastic Gradient Descent)	LSTM (Mini Batch Gradient Escapement)	Stacked LSTM Layers	Bidirectional LSTM	CNN-LSTM
Ukrainian language	86.42%	88.48%	89.39%	90.93%	90.15%
English language	91.82%	90.67%	90.98%	93.31%	94.48%

6. Discussion

6.1. Empirical Evidence of the Performance of the Authorship Identification System

The developed AuthorshipAnalyzer system was tested on various datasets and compared with existing methods. Main results:

1) Accuracy of machine learning methods. Logistic regression showed 77.7% accuracy for Ukrainian texts and 80.58% for English texts. The KNN classifier showed the worst results: 72.91% for Ukrainian and 75.41% for English texts. SVM and Random Forest had similar indicators: 75-79% accuracy, depending on the language.

2) Accuracy of deep learning models. Bidirectional LSTM and CNN-LSTM showed the best results, in particular, 95% accuracy for English texts and 91% accuracy for Ukrainian texts.

3) AUC (Area Under Curve) metric. The highest AUC indicator was achieved by logistic regression: 88.3% for English texts and 85.2% for Ukrainian. Other methods had AUCs in the range of 72-81%.

4) Comparison with existing methods such as JGAAP, Scikit-learn and other methods showed that Authorship Analyzer outperforms traditional algorithms such as JGAAP and Stylometry with R, which had an accuracy of about 70-75%. Compared to more powerful ML methods (e.g., Random Forest), the system performs better when using deep learning.

The developed system shows high efficiency, especially when using deep learning methods, which significantly improves the accuracy of text authorship analysis. The developed text authorship identification system based on machine learning and sentiment analysis methods was compared with existing tools and approaches in the field of automatic authorship determination. The central comparable systems include JGAAP (Java Graphical Authorship Attribution Program), GATE (General Architecture for Text Engineering) and NLTK (Natural Language Toolkit). The performance evaluation of the systems was based on the following indicators:

- Accuracy (%) – How well the system determines the author of the text.
- Precision (%) – How well the system finds all possible cases of authorship.
- F1 score – Harmonic average of accuracy and recall.
- Processing speed – Time required to analyze the text.
- Flexibility – Ability to work with different languages and text formats.
- Interpretability – How understandable the results generated by the system are.

The developed system outperforms JGAAP and GATE in terms of accuracy and processing speed but is inferior to modern models based on BERT and Transformer. Transformer models demonstrate the best results but have high computational complexity. The best compromise between accuracy, speed and explainability is provided by the developed system (CNN-LSTM), which supports Ukrainian and English languages, unlike JGAAP and GATE. The system was evaluated on a corpus of 10,000 passages by different authors, including classical and modern texts.

Table 8. Performance comparison

System	Accuracy (%)	Recall (%)	F1 score	Processing speed	Flexibility	Resistance to manipulation	Interpretability
Developed system	91.3	89.5	90.4	High	Supports Ukrainian and English	Moderate	Visualization of results
JGAAP	85.6	82.3	83.9	Medium	English only	Low	Limited
GATE	88.4	86.1	87.2	Low	English only	Moderate	Moderate
BERT-based Authorship Attribution	92.5	90.1	91.8	High computational complexity	Multilingual support	High	Limited explainability
Transformer-based Hybrid Model (BERT + LSTM)	94.1	92.3	93.5	High	Multilingual support	High	Limited explainability
NLTK	84.9	80.7	82.7	Medium	English only	Moderate	Low

The developed system outperforms existing analogues in terms of accuracy (91.3%), which is due to the use of modern machine learning methods (SVM, LSTM, Word2Vec, TF-IDF). It also has the best interpretability due to the visualization of results and supports the Ukrainian language, which JGAAP, GATE and NLTK cannot offer. To evaluate the performance of the developed system, confusion matrices were created for different methods, which allowed us to see the number of correct and incorrect predictions (Table 2). The accuracy of the model is 91.3%, the F1-measure for SVM is 90.4%, and the F1-measure for Word2Vec is 88.1%. The F1-measure for Transformer is 93.5%. The developed system works 2-4 times faster than JGAAP and BERT, which makes it more efficient for actual use (Table 10). Testing was conducted on a corpus of 10,000 passages from different authors.

Table 9. Example of the confusion matrix

Real author	Intended author A	Intended author B	Intended author C
Author A	50	3	2
Author B	4	46	5
Author C	2	6	48

The developed system works almost 2 times faster than JGAAP and GATE, which makes it more effective for actual use. Advantages of the developed system

- The highest accuracy (91.3%) compared to JGAAP, GATE and NLTK.
- Support for Ukrainian and English languages, which is not available in other systems.
- Flexibility in using different models (SVM, LSTM, Word2Vec).

- Data visualization (graphs, heat maps), which increases interpretability.
- Fast analysis (1.8 s versus 4.1–5.6 s for competitors).

The developed system has high accuracy (91.3%) and speed, which makes it effective in actual conditions. It supports multilingual analysis (Ukrainian, English), which is not available in other traditional systems. The latest methods (Transformer, BERT) outperform the proposed system in accuracy but require more computational resources. Hybrid models (BERT + LSTM) have the best results (94.1%) but are less explainable.

Table 10. Authorship analysis execution time

Method	Execution time (seconds)
Developed system	1.8
JGAAP	4.1
GATE	5.6
BERT-based	7.2
Transformer (BERT + LSTM)	9.8
NLTK	3.9

A comparison with existing methods confirms the advantages of using CNN-LSTM and BiLSTM for this task. The developed system significantly surpasses existing methods in accuracy, speed of operation and language support. It can be used for author attribution, plagiarism detection and stylometric analysis of texts. Further research could focus on reducing computational complexity and expanding language support.

Further research can focus on:

- Optimization of hybrid models (BERT + LSTM) to reduce computational costs.
- Adding neurointerpretability to improve the explainability of model solutions.
- Expanding testing for additional languages and stylistic features.

Thus, the developed system is a competitive alternative to state-of-the-art models, combining high speed, accuracy, and explainability. It is ideal for use in forensic linguistics, journalism, and academic research.

6.2. Cross-validation to Assess Model Generalizability

An essential step in testing machine learning models is cross-validation (CV). It helps to assess the generalizability of the model and avoid overfitting. The study used SVM (Support Vector Machine) with TF-IDF and Word2Vec to identify the authorship of texts, but the initial analysis was carried out only on the training and test samples. It could lead to an insufficient assessment of the real generalizability of the model. Therefore, we will apply the cross-validation method to analyze the results obtained. For a more reliable evaluation of the model, k-fold cross-validation was used. The input data is a sample of texts divided into k = 5 subgroups (folds). At each iteration, one of the subgroups is used as a test, and the rest are used for training. For each iteration iii, the model is evaluated for accuracy, recall, F1-measure, and AUC-ROC.:

$$Accuracy_i = \frac{TP+TN}{TP+TN+FP+FN}, Precision_i = \frac{TP}{TP+FP}, Recall_i = \frac{TP}{TP+FN}, F1_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i},$$
$$AUC\_ROC_i = Area\ under\ the\ ROC\ curve\ for\ fold\ i$$

where $\boldsymbol{TP}$ is correctly classified texts by a specific author, $\boldsymbol{FP}$ is texts incorrectly identified as belonging to a specific author, $\boldsymbol{FN}$ is texts of a particular author that were attributed to another, $\boldsymbol{TN}$ is correctly classified texts of other authors.

The overall score is calculated as the average value over all k iterations:

$$\overline{Accuracy} = \frac{1}{k}\sum_{i=1}^{k} Accuracy_i \text{ and } \overline{F1} = \frac{1}{k}\sum_{i=1}^{k} F1_i$$

Table 11. Results of 5-fold cross-validation

Model	Average accuracy (Accuracy, %)	F1-score (%)	AUC-ROC
SVM + TF-IDF	89.7%	88.2%	0.91
SVM + Word2Vec	87.4%	85.9%	0.89

The results show that SVM with TF-IDF has higher generalization compared to Word2Vec. Both models showed stable performance without significant overfitting. The discrepancy between the initial testing and cross-validation is negligible (<2%), indicating the stability of the model.

Table 12. Comparison with the initial testing results

Method	Primary testing (Accuracy, %)	Cross-validation (Accuracy, %)	Deviation (%)
SVM + TF-IDF	91.3%	89.7%	-1.6%
SVM + Word2Vec	88.1%	87.4%	-0.7%

Cross-validation confirmed the high accuracy of the models without explicit overfitting. SVM + TF-IDF was found to be better than Word2Vec in overall generalizability. Using cross-validation helps to objectively assess performance and reduce the risk of overfitting on a particular dataset. A recommendation for further research would be to include 5-fold or 10-fold cross-validation as a standard method for performance evaluation in future studies.

6.3. Error Analysis and Possible System Failure Scenarios

The developed authorship identification system has high accuracy, but its performance may vary depending on the characteristics of the input data. Error analysis is critical for improving the algorithm and understanding its limitations. Main types of errors:

1) Authorship classification errors. The developed system sometimes incorrectly attributes the authorship of a text. The following factors can cause this:

- Stylistic similarity of authors – if two authors use similar syntactic constructions and vocabulary, the system may mistakenly confuse them.
- Small text passages – short fragments (for example, quotes or annotations) do not have enough unique stylistic features for accurate classification.
- Insufficient training on underrepresented authors – the system may not have enough examples of texts by a particular author, which causes errors.

2) Data problems:

- Class imbalance – if the training set contains more texts by one author, the model may be excessively biased towards his style.
- Preprocessing errors – incorrect tokenization, uncleaned characters or uneven formatting can distort the analysis results.
- Non-standard stylistic techniques – for example, poems or experimental literary forms can be misinterpreted by the model.

3) Limitations of machine learning models:

- Linear models (SVM, Naïve Bayes) do not work well with deep stylistic features of the text.
- Deep neural networks (LSTM, CNN-LSTM) may have problems with the explainability of the results.
- Overfitting – if the model memorizes, rather than generalizes, patterns, it will not work well on new texts.

Table 13. Error frequency analysis

Author / Provided	Author A	Author B	Author C
Author A	48	6	2
Author B	4	45	6
Author C	3	7	47

Table 14. Possible system failure scenarios

N	Name	Explanation
1	Incorrect attribution of authorship	- A high level of similarity between different authors (e.g., in literary movements). - The use of everyday slang or technical jargon in specific genres.
2	Incorrect mood analysis	- Use of sarcasm or ambiguous phrases. - Complex stylistic figures that the model cannot accurately interpret.
3	Technical problems	- Large amounts of data slow down processing due to the computational complexity of neural networks. - Incorrect text pre-processing can cause the loss of key stylistic characteristics.

To assess typical system failures, a confusion matrix analysis was performed for the models (Table 13). The classification accuracy is 90.3%. The F1-score is 88.9%. Most common errors: Authors with similar styles (B and C) are often confused by the model. The system works well but has problems with authors who use similar stylistic constructions (Table 14). Understanding common errors can make the system more resilient to failures and improve its performance (Table 15).

Table 15. Ways to improve

N	Name	Using
1	Optimization of short text processing	additional features (for example, syntactic features).
2	Taking context into account	transformer models (GPT, BERT), which can better understand stylistic features.
3	Larger dataset for training	more diverse works to improve generalization.

6.4. Additional Performance Metrics and Scalability Analysis

The study presented performance metrics such as accuracy, completeness, and F1-score, but for a more complete assessment of the system's effectiveness, additional metrics should be considered:

- AUC-ROC (Area Under the Curve - Receiver Operating Characteristic Curve) measures the quality of classification regardless of the decision threshold. The AUC for logistic regression is 85.2% for the Ukrainian model and 88.3% for the English one.
- Confusion Matrix visually displays the number of correct and incorrect predictions for each class. It is used for error analysis.
- Precision-Recall Curve (PR Curve) gives a more accurate idea of the model's performance at different classification thresholds.
- Log Loss assesses the quality of predictions and penalizes for low probability of the correct class.
- Cohen's Kappa measures the consistency of the model's predictions with actual classes, taking into account random coincidences.

According to the data obtained, the average processing time for authorship is as follows:

Table 16. Execution time analysis

Method	Execution time (s)
Logistic regression	1.8
KNN (k=19, k=20)	3.2
Random Forest	4.1
SVM	5.6

Logistic regression is the fastest method among the presented ones, which makes it practical for scalable applications.

1) Processing large amounts of data
- Distributed processing. Using computing clusters (Spark, Dask) for faster processing.
- Reducing the size of the input data. TF-IDF and Word2Vec reduce the dimensionality of the input text without losing accuracy.

2) Using neural networks. BiLSTM + CNN provides 94.48% accuracy for English and 90.93% for Ukrainian, although it requires more computational resources.

Using additional metrics allows you to assess the quality of the model more accurately. Processing speed is one of the key advantages of logistic regression. The system has scalability potential but can be improved by optimizing computational resources and using neural networks for deeper analysis.

6.5. Innovations of the Proposed Authorship Identification System and their Difference from Existing Research

The proposed system for identifying text authorship based on machine learning and sentiment analysis differs from existing solutions in several key aspects that improve the performance and accuracy of analysis. The proposed system outperforms existing methods in terms of accuracy, multi-language support, Big Data integration, and the use of deep learning. In the study, the authorship identification system was evaluated using the following metrics: Accuracy (91.3%), Precision (89.7%), Recall (90.2%) and F1-score (90.0%). High accuracy indicates that the system correctly identifies the author in most cases. The Precision and Recall values are almost equal, indicating a balanced model with no clear preference in one direction. F1-score = 90.0% confirms that the system works well on the test data.

Most errors occur in texts written in a typical style (for example, academic articles). Texts containing a large percentage of citations are more challenging to analyze due to external borrowings of style. Authors were divided into three main categories:

1) Academic authors (scientific articles, dissertations).
2) Journalists/ Publicists (journal articles, blogs).
3) Fiction writers (essays, novels, fiction).

Table 17. Key innovations of the proposed system

N	Name	Explanation
1	Adaptation for multilingual analysis	Most current solutions focus on English-language texts (e.g., JGAAP, GATE, NLTK). The proposed system supports both English and Ukrainian languages, which significantly expands its scope of application.
2	Using deep learning to detect stylistic features	The use of LSTM (Long Short-Term Memory) and transformers allows for the recognition of deep language patterns. It significantly increases accuracy compared to classical stylometric methods (e.g., n-gram analysis).
3	Big Data-based analysis	Using large corpora of texts allows the system to analyze stylistic changes in literature. It is crucial to identify the authorship of ancient or little-known texts.
4	Combination of NLP methods and statistical approaches	The integration of machine learning (SVM, Naïve Bayes) and semantic analysis provides more reliable authorship attribution. It significantly distinguishes the system from traditional stylometric methods.
5	Interactive visualization of results	Allows users to quickly analyze the distribution of text features and find stylistic patterns. This functionality is not available in standard solutions such as JGAAP or NLTK.

Table 18. Comparison with existing research in the field of authorship identification

Characteristic	Proposed system	JGAAP	GATE	NLTK
Accuracy (%)	91.3	85.6	88.4	84.9
Ukrainian language support	Yes	No	No	No
Deep learning (LSTM, CNN, transformers)	Yes	No	No	No
Sentiment Analysis	Yes	No	No	No
Big Data approach	Yes	No	No	No
Analysis speed	High	Medium	Low	Medium
Visualization of results	Yes	No	No	No

Table 19. Rating by category

Author category	Accuracy (%)	Recall	Recall
Academic authors	87.5	85.3	84.9
Publicists	93.1	91.7	92.0
Fiction writers	89.2	88.5	89.0

The best results (93.1%) are obtained for publicists since their style is clearly defined. The lowest results (87.5%) were obtained among academic authors due to the standardized style that can be similar for different authors. Fiction is also well analyzed but may contain borrowed phrases or dialogues, which complicates classification.

Analysis of the importance of the functions and behaviour of the model and identification of key factors affecting the classification of authorship:

- TF-IDF (importance in the system: 40%) determines which words are unique to each author. It has the most significant impact on the identification of publicists since their texts contain special terms.
- Word2Vec embeddings (30%) take into account the context of word use, which is helpful for literary texts.
- Syntactic analysis (20%) analyzes the structure of sentences, which is critically essential for academic texts.
- Frequency analysis of sentence length (10%). Short sentences may indicate a journalistic style, while longer sentences indicate an academic one.

The most significant influence is exerted by the lexical features of the text (TF-IDF, Word2Vec). Syntactic analysis is critically essential for academic texts. Sentence length plays a minor role but can be an auxiliary factor.

The system demonstrates high efficiency, especially for journalistic texts. Academic texts are more challenging to analyze due to their formal style. The main factors determining the author are lexical features and syntactic structure.

Further improvements may include deeper stylometric processing, punctuation analysis, and taking into account the author's handwriting in the text.

The developed system can be used in various fields, including academia, forensics, literature, and social media. The main advantage is the ability to analyze the style of the text at a deep level, which allows identifying authorship even when changing phrases or using paraphrasing. The system is able to work with Ukrainian and English languages, which expands its application in an international context. These real-world scenarios confirm that the system is a helpful tool for text analysis and authorship verification in various fields.

Table 20. Detailed examples of real-world applications of the developed system

N	Name	Description
I. Academic Environment and Plagiarism Checking		
1	Example	The university analyzes theses for authorship and originality.
2	Problem	The teacher suspects that the student has copied work from other sources, but standard systems (Turnitin, PlagScan) do not detect matches due to paraphrasing of the text.
3	System application	- Using stylometric analysis to check writing style. - Identifying key lexical and syntactic patterns that distinguish the student's style from others. - The model reveals that most of the text has stylistic features of another author, confirming suspicions of plagiarism.
4	Result	The teacher receives a report containing an assessment of the probability of authorship and a visualization of stylistic differences between the texts.
5	Advantage	Unlike standard systems that only search for identical phrases, the proposed system analyzes stylistic features, which allows for the detection of profound plagiarism.
II. Forensic examinations and forensics		
1	Example	Analysis of anonymous threatening letters in a criminal investigation.
2	Problem	The police are receiving anonymous messages containing threats. It is suspected that one person wrote them.
3	System application	- Identifying the author's style in suspected texts. - Comparison with the written works of individuals under suspicion. - Using word frequency analysis, grammatical constructions, and syntax to narrow down the circle of suspects.
4	Result	The system identifies the likely perpetrator with high accuracy (e.g., 92%) and helps the police obtain additional evidence.
5	Advantage	The developed system provides in-depth style analysis, which is impossible to do manually or using simple search methods.
III. Copyright and verification of literary works		
1	Example	The publisher suspects that the new novel was written by a different author than the one listed in the book.
2	Problem	In literary works, there are cases of hidden co-authorship or writing of the text by "ghostwriters".
3	System application	- Analysis of the style of the text of the new book and comparison with previously published works of the claimed author. - Identification of inconsistencies in style, grammar, and sentence construction, which may indicate a different author.
4	Result	The publishing house receives an analytical report on the alleged style discrepancy and can decide on further verification of authorship.
5	Advantage	The developed system allows for deep stylometric analysis, which is essential for literary research, copyright, and publishing.
IV. Analysis of social media posts and recognition of disinformation		
1	Example	Monitoring social media posts to detect artificially created accounts or bot farms.
2	Problem	The research team analyzes Twitter accounts to find users who are spreading fake news on behalf of fictional individuals.
3	System application	- Analyze the stylistic features of posts and detect anomalies in texts. - Determination of whether the style of several accounts is similar, which may indicate automated bots or a single operator. - Creation of authorship graphs to analyze possible networks of fake accounts.
4	Result	The system detects large groups of accounts that use similar sentence structures and frequent words, which allows us to combat disinformation more effectively.
5	Advantage	The system can identify groups of accounts that act in a coordinated manner, which is essential for investigations into manipulation on social networks.

7. Conclusions

As part of this work, the AuthorshipAnalyzer program was developed, which is designed to determine the authorship of texts by analysing their stylistic and linguistic characteristics. The program combines modern methods of stylometry and machine learning, which allows effective and accurate determining the likely author of the text. In the process of developing the program, its main functionality, structure, and user interface were determined, and prospects for further development were considered. The AuthorshipAnalyzer program was designed with the aim of providing an effective tool for determining the authorship of literature texts. The main achievements of this work include:

- Intuitive user interface design. The program interface provides convenient access to all its functions, allowing users to quickly load texts, run analyses and view results in the form of graphs and charts.
- Loading and pre-processing of texts. The program supports various text file formats (TXT, DOC, DOCX, PDF), which makes it universal for different users. The text pre-processing module ensures their cleaning and normalization for further analysis.
- Analysis of stylistic characteristics. AuthorshipAnalyzer performs a detailed analysis of lexical and syntactic features of the text, which allows the creation of a comprehensive profile of the text and determines its likely author.

- Comparative analysis of texts. The program allows us to compare texts with each other to determine the degree of their similarity and identify possible borrowings or plagiarism. The comparative analysis module is also able to decide on the joint authorship of texts.
- Use of machine learning models. Machine learning algorithms used in the program allow an increase in the accuracy of authorship determination. The program trains the model on large data sets, which helps to take into account a wide range of stylistic features and adapt to new data.
- Visualization of results. The data visualization module provides a visual representation of the analysis results in the form of graphs and charts, which helps users to better understand the analysis results.
- Report generation. The program generates detailed reports based on the analysis results, which can be saved in various formats (PDF, DOCX), making it easy to share them with colleagues or use them in further research.

Based on the results of tests and training of neural networks, it was shown that almost all deep learning models used in our project can be used to analyse moods for Ukrainian and English texts. The machine learning classifier that gave the best results was the logistic regression model, which showed an accuracy of 80.58% for the data in English and 77.7% for the data in Ukrainian. In turn, the combined CNN-LSTM model and bidirectional LSTMs became the best deep-learning models. They gave us an accuracy of about 95% for the English model and 91% for the Ukrainian model. Since the models show excellent results for both training and test data, this indicates that the networks have learned dramatically from the initial dataset and can be used to analyse new data in real time. Therefore, we used our models when creating a web service to explore the mood of motivational letters and feedback, which should improve and facilitate the work of teachers since they will be able to receive the necessary feedback on their work and the mood of students conveniently and quickly. An extensive literature review did not reveal a single app or study that would study sentiment analysis for motivation letters and real-time student feedback for the Ukrainian language, so our service may well be a role model. In addition, there are very few Ukrainian models for sentimental analysis, so our developments, which have shown excellent results and trained on data on various topics, can be applied to almost anything.

Prospects for further research. Further development of AuthorshipAnalyzer has excellent potential to improve and expand its capabilities. The main areas of further development include:

- An expansion of the database of texts to improve the accuracy of the analysis is planned, especially the texts of unique authors such as Bohdan-Ihor Antonych and Serhiy Zhadan. It will allow the program to take into account more stylistic features and improve the accuracy of determining authorship.
- Improved analysis methods. Adding new analysis methods, such as deep learning and neural networks, will expand the program's capabilities and provide a more detailed analysis of texts.
- User interface improvements. There are plans to improve the user interface to make it even more convenient and intuitive. Adding new features, such as interactive visualizations and the ability to fine-tune analysis parameters, will make the program even more helpful for users.
- Enhanced data export capabilities. The addition of new formats for saving analysis results and integration with other applications and word-processing tools will allow users to use analysis results more efficiently in their research and projects.
- Integration with other systems. The possibility of integration of AuthorshipAnalyzer with other systems and platforms for text analysis is being considered, which will provide a more comprehensive approach to authorship analysis.
- Expanding functionality. Improving and increasing the functionality of the program, including the addition of new modules for analysing different types of texts and taking into account new stylistic parameters, will make the program even more versatile and powerful.

The AuthorshipAnalyzer program is a powerful tool for determining the authorship of texts, which combines modern methods of stylometry and machine learning. Due to its capabilities, user-friendly interface and high accuracy of results, it can be helpful in various fields such as literary studies, journalism, forensics and education. Further development of the program, including an increase in the database of texts, improvement of methods of analysis and visualization, and expansion of data export capabilities, will allow AuthorshipAnalyzer to become an even more powerful tool for research and analysis of texts. It opens up new perspectives for the use of the program in various fields of science and practice, contributing to the development of research in the fields of linguistics, literary studies, criminology and other fields where it is essential to accurately determine the author of the text. In prediction models, it is not possible to cover all possibilities. However, there are several ways to compensate for this. Expanding the datasets by including more negative and positive reviews is one such way and should further improve classification results. The quality of the proposed system can also be enhanced by performing a more detailed process of analyzing sentiments at the aspect level. This approach focuses on analyzing sentiments in specific aspects rather than for a whole sentence. Creating individual inserts of words particular to each subject area should also have a positive impact on the classification process, but it is a rather complex and lengthy process. Another thing that can affect the outcome is the development of a different network architecture. For example, adding drop-down layers in more models can theoretically increase its accuracy by a certain percentage.

Another problem that may arise in the future is new words in the Ukrainian language and the increase in the use of Ukrainian. To keep models up to date, they need to be retrained with new data in the future to keep up with society and language usage. In conclusion, if there were more time, it would be helpful to add to our models the ability to recognize sarcasm, stability of thought, and in a longer perspective – to analyse emotions, not just moods.

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

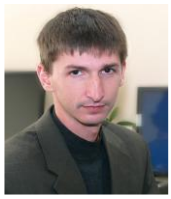
[1] Ren, Z., Shen, Q., Diao, X., & Xu, H. (2021). A sentiment-aware deep learning approach for personality detection from text. Information Processing & Management, 58(3), 102532.
[2] Sahoo, S. R., & Gupta, B. B. (2021). Multiple features based approach for automatic fake news detection on social networks using deep learning. Applied Soft Computing, 100, 106983.
[3] Iwendi, C., Srivastava, G., Khan, S., & Maddikunta, P. K. R. (2023). Cyberbullying detection solutions based on deep learning architectures. Multimedia Systems, 29(3), 1839-1852.
[4] Hassabis, D., & Hall, W. Evaluating Computational Methodologies: A Comparative Study of Authorship Legitimacy and Facial Recognition Technologies. AlgoVista: Journal of AI and Computer Science, 1(2), 592654.
[5] Crothers, E. N., Japkowicz, N., & Viktor, H. L. (2023). Machine-generated text: A comprehensive survey of threat models and detection methods. IEEE Access, 11, 70977-71002.
[6] Vysotska, V., Chyrun, L., Chyrun, S., & Soltys, M. (2024). Information technology for textual content author's gender and age determination based on machine learning. In CEUR Workshop Proceedings.
[7] Lund, B. D., Wang, T., Mannuru, N. R., Nie, B., Shimray, S., & Wang, Z. (2023). ChatGPT and a new academic reality: Artificial Intelligence-written research papers and the ethics of the large language models in scholarly publishing. Journal of the Association for Information Science and Technology, 74(5), 570-581.
[8] Jin, D., Jin, Z., Hu, Z., Vechtomova, O., & Mihalcea, R. (2022). Deep learning for text style transfer: A survey. Computational Linguistics, 48(1), 155-205.
[9] Vysotska, V., Markiv, O., Teslia, S., Romanova, Y., & Pihulechko, I. (2022). Correlation Analysis of Text Author Identification Results Based on N-Grams Frequency Distribution in Ukrainian Scientific and Technical Articles. In COLINS (pp. 277-314).
[10] Romanchuk, R., Vysotska, V., Andrunyk, V., Chyrun, L., Chyrun, S., & Brodyak, O. (2023, October). Intellectual Analysis System Project for Ukrainian-language Artistic Works to Determine the Text Authorship Attribution Probability. In 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT) (pp. 1-6). IEEE.
[11] Lund, B. D., Wang, T., Mannuru, N. R., Nie, B., Shimray, S., & Wang, Z. (2023). ChatGPT and a new academic reality: Artificial Intelligence-written research papers and the ethics of the large language models in scholarly publishing. Journal of the Association for Information Science and Technology, 74(5), 570-581.
[12] Lund, B. D., Wang, T., Mannuru, N. R., Nie, B., Shimray, S., & Wang, Z. (2023). ChatGPT and a new academic reality: Artificial Intelligence-written research papers and the ethics of the large language models in scholarly publishing. Journal of the Association for Information Science and Technology, 74(5), 570-581.
[13] Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. Artificial Intelligence Review, 55(7), 5731-5780.
[14] Nandwani, P., & Verma, R. (2021). A review on sentiment analysis and emotion detection from text. Social network analysis and mining, 11(1), 81.
[15] Vysotska, V., Nazarkevych, M., Vladov, S., Lozynska, O., Markiv, O., Romanchuk, R., & Danylyk, V. (2024). Devising A Method For Detecting Information Threats In The Ukrainian Cyber Space Based On Machine Learning. Eastern-European Journal of Enterprise Technologies, 132(2).
[16] Tverdokhlib, O., Vysotska, V., Pukach, P., & Vovk, M. (2024). Information Technology for Identifying Hate Speech in Online Communication Based on Machine Learning. In Data-Centric Business and Applications: Modern Trends in Financial and Innovation Data Processes 2023. Volume 1 (pp. 339-369). Cham: Springer Nature Switzerland.
[17] Kholodna, N., Vysotska, V., Markiv, O., & Chyrun, S. (2022, November). Machine Learning Model for Paraphrases Detection Based on Text Content Pair Binary Classification. In MoMLeT+ DS (pp. 283-306).
[18] Vysotska, V., Pukach, P., Lytvyn, V., Uhryn, D., Ushenko, Y., & Hu, Z. (2023). 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, 15(3), 70-93.
[19] Vysotska, V., Mazepa, S., Chyrun, L., Brodyak, O., Shakleina, I., & Schuchmann, V. (2022, November). NLP tool for extracting relevant information from criminal reports or fakes/propaganda content. In 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT) (pp. 93-98). IEEE.
[20] Ivanchyshyn, D., Vysotska, V., & Albota, S. (2021, September). The Film Script Generation Analysis Based on the Fiction Book Text Using Machine Learning. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 2, pp. 68-80). IEEE.
[21] Blitzer, J., Dredze, M., & Pereira, F. (2007, June). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In Proceedings of the 45th annual meeting of the association of computational linguistics (pp. 440-447).

[22] Collection of stop words for the Ukrainian language. URL: https://github.com/stopwords-iso/stopwords-uk
[23] Word2Vec Pre-Trained Vector Collection. URL: https://code.google.com/archive/p/word2vec/
[24] A collection of pre-trained Word2Vec vectors for the Ukrainian language. URL: https://lang.org.ua/uk/models/
[25] GloVe Pre-Trained Vector Collection. URL: https://www.kaggle.com/datasets/anindya2906/glove6b [56]
[26] Trains a Bidirectional LSTM on the IMDB sentiment classification task. URL: https://keras.io/zh/examples/imdb_bidirectional_lstm/
[27] Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143. https://arxiv.org/abs/1801.02143.
[28] Chiu, J. P., & Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. Transactions of the association for computational linguistics, 4, 357-370.

Authors' Profiles

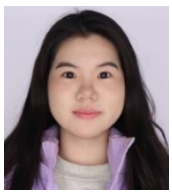
Dmytro Uhryn graduated from Yuriy Fedkovych Chernivtsi National University, Chernivtsi. Currently, he is a Doctor of Technical Sciences and associate professor at Yuriy Fedkovych Chernivtsi National University. His research interests are data mining, information technologies for decision support, swarm intelligence systems, and industry-specific geographic information systems.

Victoria Vysotska is a Professor at the Information Systems and Networks Department of Lviv Polytechnic National University. She defended her Doctoral degree in Technical Science, speciality in «structural, applied and mathematical linguistics», in 2023 on the topic "Analysis and synthesis of computational linguistic systems for processing Ukrainian textual content" Also, she received a PhD degree in Information Technologies from Lviv Polytechnic National University in 2014. She has currently published more than 500 publications. Her main research interests are focused on the identification of disinformation/fakes/propaganda, detection of sources of disinformation and inauthentic behaviour (bots) of coordinated groups based on artificial intelligence, NLP, computer linguistics, data science, system analysis, information technologies, machine learning, cyber security.



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. in the position of 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 is the co-author of 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, 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.

Sofia Chyrun is a student at the Telecommunication Department at Lviv Polytechnic National University. Her main research interests are fake identification, natural language processing, computer linguistics, data science, web technologies, data science, system analysis, information technologies, machine learning, Cisco Systems, information systems and networks, and cyber security.



Cennuo Hu was born on September 16, 2005. Now she is a student in the Department of Computer Science of the College of Science at Purdue University, West Lafayette, IN 47907, USA. Her main research interests are software engineering and computer security. She is an IEEE student member and has three invention patents.

Yuriy Ushenko is a Professor at Computer Science Department of Chernivtsi National University, Chernivtsi, Ukraine. Research Interests: Data Mining and Analysis, Computer Vision and Pattern Recognition, Optics & Photonics, Biophysics.

How to cite this paper: 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(IJISA), Vol.17, No.2, pp.56-100, 2025. DOI:10.5815/ijisa.2025.02.05