I. J. Computer Network and Information Security, **2025, 3, 109-143**
Published Online on June 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2025.03.07

Development and Testing of Voice User Interfaces Based on BERT Models for Speech Recognition in Distance Learning and Smart Home Systems

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

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

Nikita Mykytyn
Artificial intelligence systems Department, Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: nikita.mykytyn.mknssh.2024@lpnu.ua
ORCID iD: https://orcid.org/0009-0000-3821-3886

Olena Nagachevska
Lviv Polytechnic National University, Lviv, 79013, Ukraine
E-mail: olena.o.nahachevska@lpnu.ua
ORCID iD: https://orcid.org/0000-0002-5200-8085

Kateryna Hazdiuk
Department of Computer Systems Software of the Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 58002, Ukraine
E-mail: k.gazdiuk@chnu.edu.ua
ORCID iD: https://orcid.org/0000-0002-7568-4422

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

Received: 17 August 2024; Revised: 13 December 2024; Accepted: 24 January 2025; Published: 08 June 2025

Abstract: Voice User Interfaces (VUIs) focus on their application in IT and linguistics. Our research examines the capabilities and limitations of small and multilingual BERT models in the context of speech recognition and command conversion. We evaluate the performance of these models through a series of experiments, including the application of confusion matrices to assess their effectiveness. The findings reveal that larger models like multilingual BERT theoretically offer advanced capabilities but often demand more substantial resources and well-balanced datasets. Conversely, smaller models, though less resource-intensive, may sometimes provide more practical solutions. Our study underscores the importance of dataset quality, model fine-tuning, and efficient resource management in optimising VUIS. Insights gained from this research highlight the potential of neural networks to enhance and improve user interaction. Despite challenges in achieving a fully functional interface, the study provides valuable contributions to the VUIs development and sets the stage for future advancements in integrating AI with linguistic technologies. The article describes the development of a voice user interface (VUI) capable of recognising, analysing, and interpreting the Ukrainian language. For this purpose, several neural network architectures were used, including the Squeezeformer-CTC model, as well as a modified w2v-bert-2.0-uk model, which was used to decode speech commands into text. The multilingual BERT model (mBERT) for the classification of intentions was also tested. The developed system showed

the prospects of using BERT models in combination with lightweight ASR architectures to create an effective voice interface in Ukrainian. Accuracy indicators (F1 = 91.5%, WER = 12.7%) indicate high-quality recognition, which is provided even in models with low memory capacity. The system is adaptable to conditions with limited resources, particularly for educational and living environments with a Ukrainian-speaking audience.

Index Terms: BERT Model, Speech Recognition, Voice User Interface, ASR, Human-Computer Interaction, Intent Recognition, Multilingual Models, Neural Networks, Command Conversion, Dataset Quality, Natural Language Processing.

1. Introduction

With the continuous advancements in speech recognition technology and artificial intelligence, voice assistants have significantly enhanced their accuracy and utility across various domains, thereby increasing their relevance and significance. The growing reliance on voice commands for device control highlights the convenience and intuitiveness of this interaction method. Voice control enables users to access device functionalities without direct physical interaction, simplifying and streamlining the user experience. Despite language being a natural mode of human communication, applying this method to computer interactions has posed considerable challenges. One of the most significant hurdles has been the accurate interpretation of voice commands by computers. Recent developments in neural networks have partially addressed this challenge.

The capability to interact with devices via voice commands not only offers convenience but is also highly beneficial for individuals for whom traditional interaction methods are inaccessible. For example, visually impaired individuals can use voice interfaces to interact with devices, significantly enhancing their user experience. Additionally, individuals who have lost limbs face substantial challenges; using a personal computer without hands is nearly impossible. While modern prosthetics can replace limbs, they do not fully restore the fine motor skills required for efficient computer use. In the context of the ongoing conflict in Ukraine, where many individuals have been injured, voice control can provide essential access to functionalities and offer autonomy to those who are bedridden due to various conditions. Our research focuses on developing a voice user interface (VUI) that operates in the Ukrainian language. The primary objective of this project is to create a robust software solution capable of converting natural Ukrainian speech into actionable text commands. To achieve this, several key tasks must be addressed: method development or identification for accurately converting natural Ukrainian speech into text; analysis of the generated text to identify and extract relevant commands; decomposition of the text into sequential commands that the system can execute; execution of these commands in the correct sequence to perform desired actions. In essence, our project aims to develop a sophisticated software solution for analysing and executing voice commands in Ukrainian. By focusing on the recognition and interpretation of natural language, our research seeks to enhance the efficacy and flexibility of voice-based interfaces. The research object is the recognition and interpretation of natural language, while the research subject encompasses the application of this natural language for computer control. Current solutions in this domain primarily involve voice assistants capable of executing both simple and complex tasks. However, these assistants often limit users to predefined functions. Our proposed VUI aims to overcome these limitations by providing a more adaptable interface that enables comprehensive device control and improves the overall user experience. The scientific novelty of our work lies in addressing the specific challenges associated with voice command recognition and execution in the Ukrainian language. By employing advanced neural network techniques and machine learning (ML) methodologies, our research contributes to the advancement of VUI technologies. It offers potential improvements in accessibility and user experience for a diverse range of individuals.

The research results can be effectively applied in the field of Computer Networks and Information Security for several important reasons:

A. Ensuring secure access to IT systems via voice, based on the use of voice interfaces built on BERT models, which allows:

- to implement biometric authentication based on voice;
- to integrate access control to confidential systems without physical interaction;
- to recognise inauthentic user behaviour in the dialogue (for example, bots, spoofing, etc.).

It is directly related to increasing information security in distributed systems (both in smart homes and in educational environments).

B. Built-in contextual analysis mechanisms (NLP + BERT) where the model not only recognises text but also:

- performs intent analysis (determines what exactly the user wants to do);
- highlights key objects (named entities) for further action;
- separates essential information from noise, even in long queries (proven by experiments: F1-score = 91.5%).

It allows you to reliably build intelligent agents for secure control of network systems, automated information services, smart offices, etc.

C. A distributed and autonomous architecture VUI system that works locally:

- does not require a constant connection to cloud services (reduces the risks of data leakage);
- provides local storage of audio and text data;
- allows you to implement on-premise control of smart devices or learning platforms.

It is critical for use in networks with limited Internet access or in military/government structures where data cannot be transmitted to the Internet.

D. Protection of vulnerable user groups because the system supports voice control in Ukrainian for people with visual impairments, with injuries (relevant in war conditions) and with low digital literacy (schoolchildren, the elderly).

It opens the way to the development of accessible interfaces for infrastructure management, including secure management of smart homes, IoT devices, and educational resources.

E. Potential for detecting threats in the network environment: since the system processes natural language and adapts to the user, it can be used to:

- Identify anomalies in user behaviour.
- Classify potentially dangerous or inappropriate requests.
- Respond to voice commands in critical situations (for example, "call for help").

It makes it possible to integrate it into network protection systems, intelligent incident response systems (SIEM), or cyber-physical systems in industry.

The research results go far beyond language interface design – they have practical applications in the design of secure, adaptive, autonomous systems for operating in networked environments, especially where the priority is user interaction through natural language while adhering to the principles of confidentiality and reliability.

2. Related Works

In this context, it is essential to note that standalone voice interfaces do not exist as a distinct category; therefore, the closest analogues are voice assistants. The primary voice assistants currently in use include:

Siri, developed by Apple, is a virtual assistant integrated into iOS, macOS, watchOS, and tvOS [1]. It is exclusive to Apple devices, reflecting Apple's strategy to create a closed ecosystem. It's not available for Android/Windows/Linux. Siri can be activated either by a voice command or by pressing a button, and it employs Google as its search engine. Notably, Siri supports the Ukrainian language.

Copilot GitHub's Copilot, developed in collaboration with OpenAI, integrates with development environments such as Visual Studio Code and Windows 11. Its primary function is to assist in coding by recognising programming languages and generating syntax-compliant code. It also analyses comments, variable names, and functions to offer precise suggestions. Copilot supports the Ukrainian language.

Google Assistant is available on Android and iOS devices and integrates seamlessly with Google Maps and Gmail. It also supports the Ukrainian language and can be embedded in home devices, such as Google Nest Hub, to control household appliances efficiently.

Amazon Alexa is a virtual assistant that operates on iOS and Android devices and integrates with a wide range of smart devices. However, Alexa does not support the Ukrainian language. It is embedded in various smart devices, including security cameras, thermostats, lighting systems, and automotive systems. Alexa performs tasks such as organisational functions, information retrieval, and weather forecasting, but lacks support for the Ukrainian language.

Voice assistants are predominantly used for a variety of tasks [2], including adding tasks to a calendar, providing search information, controlling and managing smart home devices, handling phone calls, scheduling meetings, creating text messages, navigating, receiving weather forecasts, setting reminders, and playing music.

However, it is essential to note that these assistants are limited in their capability for more complex device interactions. For example, none of the current assistants can navigate a device's file system comprehensively. The article [3] provides an overview of voice assistants' operational principles, noting that some continuously listen for a trigger word while others require activation via a button. Key points include some assistants continuously listening passively, waiting for a trigger word, while others require activation via a button. Once activated, the assistant is prepared to receive and analyse commands. To comprehend the task, the assistant utilises an AI model to convert the voice input into text, which is then processed using Natural Language Processing (NLP), enabling the assistant to execute the task. The article [4]

explores a multimodal interface that uses large language models for interaction. It describes the process of receiving and processing voice, gesture, and keyboard inputs and integrating them to decide and execute actions. The interaction process includes receiving input data through voice, gestures, and keyboard. Voice inputs are converted into text and analysed using NLP methods, such as text tagging, named sentiment analysis, and entity recognition. Gestures are interpreted using computer vision technology. Subsequently, information from different modules is merged into a unified representation of the user's intent using attention-based models. The system then decides on the appropriate action. The action is executed through platform-dependent APIs.

The research suggests that during the task execution stage; a visible grid may be employed to assist users in identifying interaction points on the screen. However, this approach may potentially clutter the workspace. The visual grid, as discussed in [4] Multimodal LLM Driven Computer Interface, is employed to enable the user to see and select interaction points, such as by clicking. While this grid addresses the issue of "where to execute a command," it significantly clutters the workspace, making the screen more challenging to use. The article [5] explores the chain-of-thought methodology for generating responses, highlighting its effectiveness in decomposing tasks into sequential actions. Essentially, large language models (LLMs) break down a task into subtasks and process them in sequence. For developing a computer voice interface model, this sequential decomposition is crucial and can be efficiently achieved using the chain-of-thought technique [6-9]. The articles [10-15] also delve into various prompting methods designed to guide models in employing this approach. For example, incorporating phrases like "let's think step by step" within input prompts can encourage models to produce responses that are logically coherent and sequential. Despite the possibility of errors in handling complex tasks, the chain-of-thought method has been demonstrated to significantly enhance the accuracy of model outputs [16-18]. In reviewing existing voice assistants and relevant literature, it is evident that while these systems are effective for many applications, they often fall short of providing comprehensive control over complex devices and typically handle only basic tasks [19-21]. A related study in the field of computer voice interfaces introduced a solution involving a grid overlay on the workspace [22-24]. This method, which allows users to designate action zones based on grid coordinates, although somewhat cumbersome, offers a practical approach to interacting with complex systems.

3. Methods and Tools

Research into voice control technologies has made significant strides, leading to the development of complex and powerful voice assistants. Many of these now support the Ukrainian language, offering efficient device operation. However, these assistants are limited in their ability to control a personal computer, often only performing specific tasks. Research on interfaces powered by Large Language Models (LLMs) remains in its early stages, with the primary challenge being the accurate interpretation of user commands by the computer. Based on this analysis, the overarching objective of this study is the development of a computer interface controlled by an LLM, designed for operation through voice commands. The system is designed to continuously listen to the user, analyse the audio for commands, and convert these commands into sequential tasks executed through APIs. The primary challenge is training models capable of accurately recognising voice commands, decomposing these commands into actionable tasks, and preparing an API that is both flexible and low-level enough to accommodate the majority of user needs. The multimodal system must effectively translate voice information into a set of text-based commands, which the system will then execute. Audio data will be analysed using the following methods: Wavelet Transform and Mel-spectrogram. A spectrogram is the frequency axis transformed to the Mel scale, which aligns more closely with human auditory perception. Mel-spectrograms are commonly used in modern speech recognition systems. The audio data will be converted into text using neural networks. The resulting text data will be analysed using NLP methods, including part-of-speech tagging to identify grammatical parts of speech and sentiment analysis to determine the text's tone and emotional content. Named Entity Recognition is used to identify and categorise named entities. The programming language used for this project is Python, which is known for its robust capabilities in text and audio analysis as well as neural network operations.

The primary Python libraries utilized include Os for interacting with the operating system; Librosa for audio data analysis and processing, offering features for reading and writing audio files, extracting audio features, and processing audio data; NLTK (Natural Language Toolkit) for NLP, providing tools for tokenization, part-of-speech tagging, and more; Hugging Face Transformers for access to large language models, such as BERT [25-27], for tasks involving text analysis and generation; Matplotlib for creating graphs and aiding in data visualization; TensorFlow for ML and deep learning, offering a user-friendly interface for building and training various neural network models. The Integrated Development Environment (IDE) chosen for this project is Visual Studio Code (VS Code), notable for its user-friendly interface, customizable features, and support for Jupyter notebooks, which simplify the debugging process through block-by-block code execution. VS Code integrates easily with Anaconda, a Python distribution that allows for the creation of isolated environments for data analysis, scientific computing, and ML. Anaconda simplifies package management and environment isolation, making it an efficient tool for developers working on Python projects. For this project, we will use the VS Code IDE with the Jupyter Notebooks module and an isolated environment created with Anaconda. Figure 1 shows the UML case diagram to describe user and functional requirements. Fig. 2 shows the UML diagram of the object classes that implement the main processes. Fig. 2 shows the general UML activity diagram, which describes the primary process.

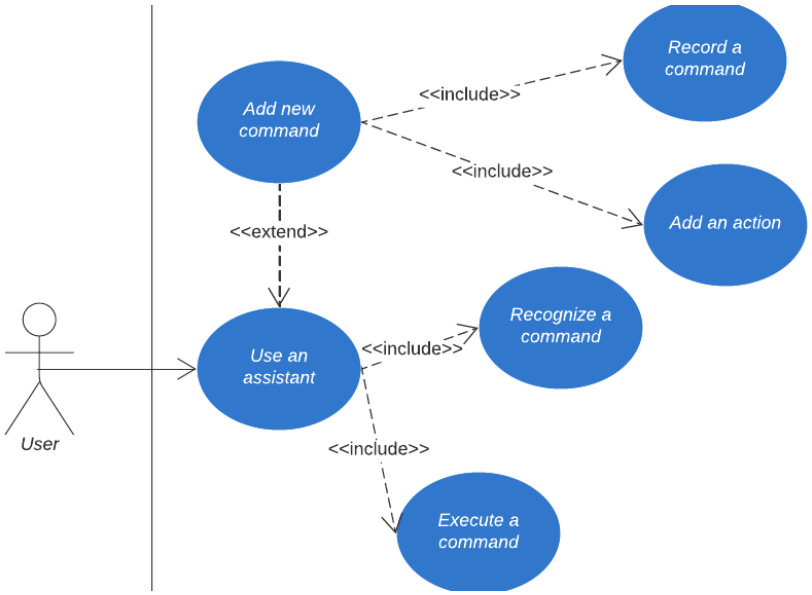

Fig.1. The case diagram

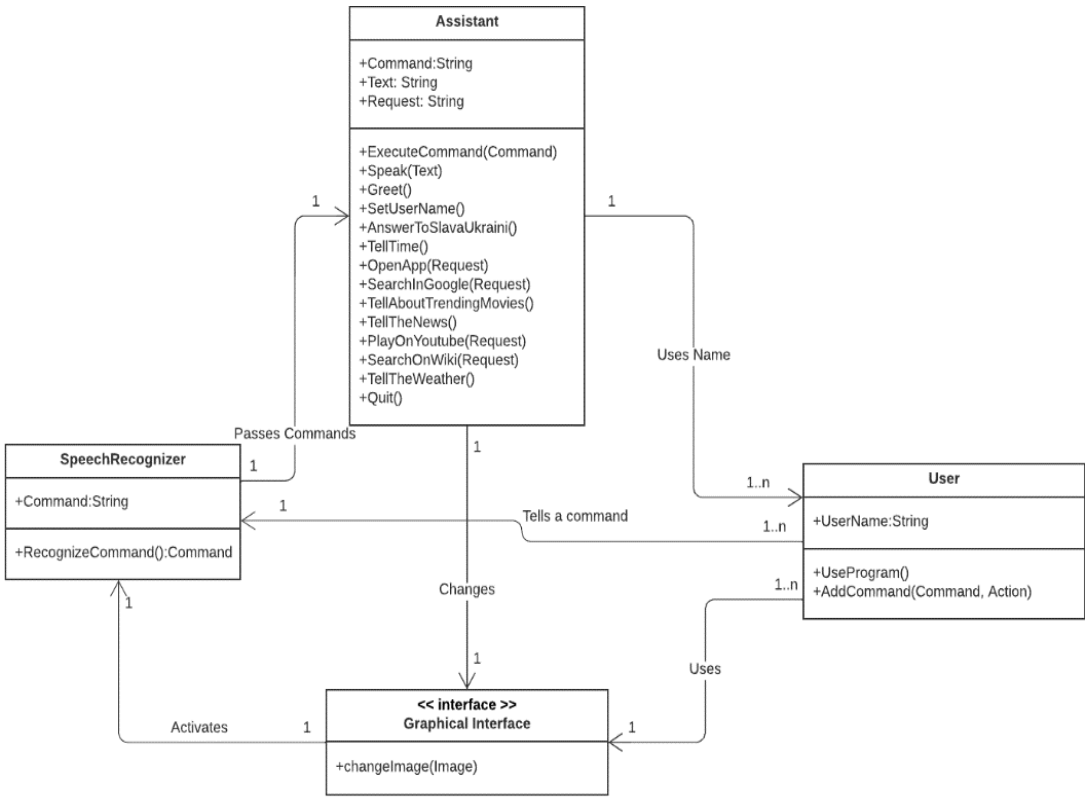

Fig.2. UML object class diagram

The system can work locally, which allows you to partially maintain the confidentiality of voice data processing. Let's consider the issue of user data protection in online implementations. It is an important criterion, especially given the potential use in distance learning and smart environments.

Protection of user data in speech recognition systems:

- Local processing = higher privacy, since voice data does not leave the device.
- Use of open-source models (e.g. Squeezeformer-CTC XS, mBERT) = lack of commercial third-party services.

However, in real operation, online implementations are often used, in particular, for the Web Client, the system uses a WebSocket connection to transmit audio/text to the server. A centralised request processor (Python API) is used, which transforms text into a command. Below is an extended description of the capabilities and requirements for protecting user data, based on system components.

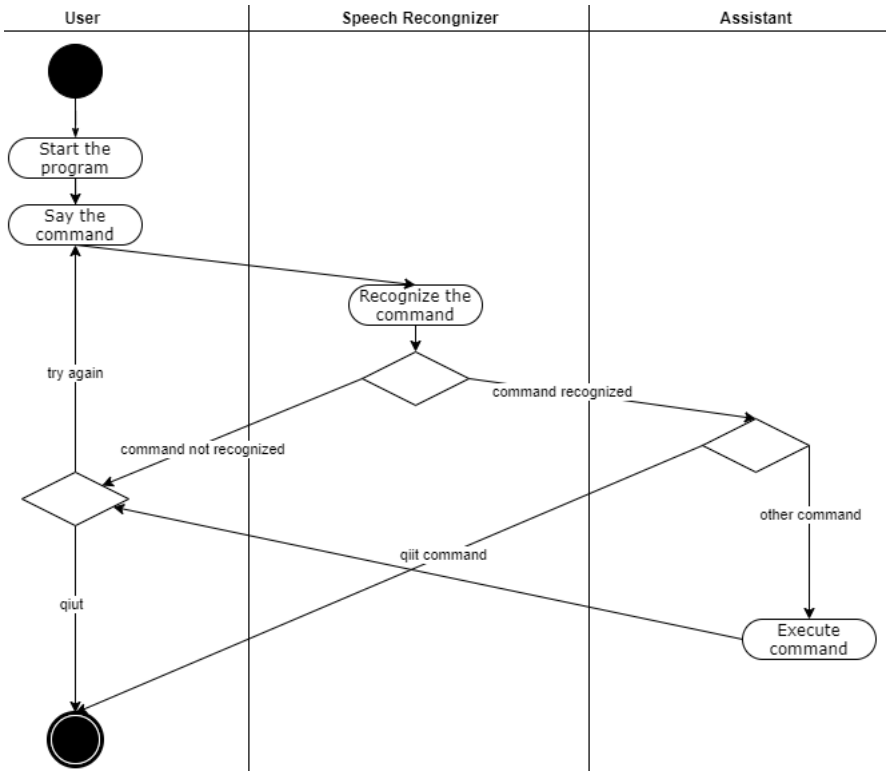

Fig.3. Activity diagram

Table 1. Potential risks in online implementation

Category	Potential threat
Audio Transmission	Broadcast interception during WebSocket broadcast
Text Transfer	Collection and processing of text data on the server without consent
Logging Commands	Save a request history that can identify the user.
Access to servers	Unreliable backend or incorrect access policies
External APIs	If the system is still connected to third-party services, the leak is due to them.

Table 2. Recommended online protection measures

Protection	Description
HTTPS/WebSocket over TLS (WSS)	Encryption of all voice and text streams
Data anonymisation	Lack of communication between requests and a specific user (e.g. session ID without an account)
Local transcription	Only text is transmitted via voice-to-text on the client side.
Temporary caching	All processing is in RAM, and nothing is written to the disk.
Opting out of logging without consent	Explicit user consent before any saving of commands
API access control	Validation of access rights to each request, and authentication
Open-source and verification	Open source models allow security audits.

Although the system supports local data processing, special attention should be paid to the protection of confidential information when working online. All transmitted data (voice or text) must be encrypted using TLS, and processing must be performed anonymously without storing personal information. Implementing local transcription with text-only transmission is recommended, as well as limiting logging without user consent. In this way, compliance with GDPR principles and ethical standards for the use of AI can be ensured.

With the growth of human-computer interaction through natural language, the need for efficient and context-aware VUIs in low-resource languages such as Ukrainian is increasing. Traditional models often fall short of providing accurate semantic interpretation due to the lack of contextual embedding. Transformer-based models, particularly BERT (Bidirectional Encoder Representations from Transformers), offer a solution by capturing bidirectional contextual information. The proposed VUI system consists of the following key components:

- **Speech-to-Text Module (ASR)** converts audio signal $S(t)$ into a textual sequence $T = \{w_1, w_2, \dots, w_n\}$;
- **Text Preprocessing** applies tokenisation and normalisation to generate input tokens for the BERT model:

$$X = \text{Tokenizer}(T)$$

- **BERT-Based Contextual Understanding** based on the preprocessed sequence X that is passed through a BERT encoder $\mathcal{B}$, yielding contextual embeddings:

$$E = \mathcal{B}(X) = \{e_1, e_2, \dots, e_n\}, \quad e_i \in R^d$$

- **Intent and Entity Recognition** based on a classification layer on top of BERT that extracts user intent I and named entities $\mathcal{E}$:

$$I = \arg\max\left(\text{Softmax}\left(W_i \cdot E_{[CLS]} + b_i\right)\right),$$

$$\mathcal{E} = \text{NER}(E)$$

- **Dialogue Management** uses finite-state or reinforcement learning techniques to select appropriate responses based on intent and dialogue state D_t:

$$A_t = \pi(D_t, I, \mathcal{E})$$

The study used the multilingual model BERT (mBERT) to classify voice commands in Ukrainian. However, the effectiveness of this model is not limited to Ukrainian: mBERT belongs to the class of pre-trained transformers supporting more than 100 languages. Assessing multilingual effectiveness is of fundamental importance for scaling the system of voice interaction to different language groups, both in the international educational environment and in multicultural households. Below is an extended assessment of BERT's multilingual models, their performance across languages, and recommendations for multilingual system expansion.

A. The mBERT architecture and features. The multilingual BERT model, developed by Google, is trained on Wikipedia corpora of 104 languages, including English, Ukrainian, Polish, Russian, Spanish, Arabic, Chinese and others. I trained on a large Wikipedia corpus for each language, so it works more effectively for written and literary language and adapts worse to spontaneous speech, jargon, and dialects. It uses the universal WordPiece dictionary (~110 thousand tokens), which allows you to partially support even languages with morphological complexity, but the level of efficiency of mBERT significantly depends on:

- the volume of available corpora for a particular language (especially in Wikipedia);
- grammatical structure of the language (analytical or inflectional);
- the presence of variability (dialects, synonymy);
- styles of requests (formal or conversational commands).

B. Comparative Performance of mBERT for Other Languages/ According to the results of other studies in the field of classification of intentions in dialogue systems (e.g., SNIPS, MultiATIS++, MASSIVE), mBERT shows the following generalised efficiency:

Table 3. Known mBERT results for other languages [1-27]

Language	F1-score (on the classification of intentions / on the recognition of intentions)	Comment
English	92–94%	Basic language, highest efficiency. Highest score, main language of instruction
Spanish	89–91%	High precision due to the large body, similar structure
German	87–90%	Good support, decline due to complex morphology
Russian	86–89%	Noticeable variability with cases
Polish	83–86%	A similar situation with Ukrainian is morphological complexity
French	85–88%	Stable efficiency. High accuracy with a good body
Chinese	78–82%	Reduced accuracy due to tokenisation complexity and sentiment

These data indicate that mBERT works well with languages represented in large corpora and that have simpler morphology. For languages with high grammatical variability (Ukrainian, Polish) or complex phonology (Chinese), the result is worse without additional training.

Reasons for the difference in performance between languages:

- Case volume in Wikipedia - the larger the case, the better the trained model for the language.

- Morphology — agglutinative or inflectional languages (like Ukrainian, Polish) require better tokenisation.
- The presence of dialects, jargon, and slang — mBERT is not adapted to them without additional training.
- Stylistics — mBERT works better with neutral, non-emotional speech.

C. Multilingual system expansion potential. Given the power of mBERT, the system described in this paper can be extended to support other languages without a complete redesign. Including:

- for English-speaking, Spanish-speaking, and German-speaking environments, the current configuration with minor modifications can be applied;
- For languages with less support, it is advisable to fine-tune on domain query sets;
- In the case of languages that are not in the top 30 in terms of volume on Wikipedia, it is worth considering switching to the latest models such as XLM-R or mT5.

We will evaluate the practical application of multilingualism. In the field of education, the system can be scaled to English-speaking or Spanish-speaking students without significant modification. For languages with an excellent structure (for example, Chinese, Arabic), partial additional training or fine-tuning on thematic sets is required. In domestic settings, smart home scenarios can be multilingual using mBERT, but it is necessary to standardise commands and filter local forms (jargon, pronunciation).

Table 4. Recommendations for further research

Direction	Explanation
Scalable multilingual testing	Evaluate F1 and accuracy for different languages with real voice commands.
Using XLM-R, LaBSE, or mT5	Newer multilingual models show higher generalisation, especially for low-resource languages.
mBERT Performance Analysis on Real Multilingual Corporals	To ensure the same quality in different regions.
mBERT Adaptation Through Transfer Training	Increases accuracy without complete retraining
Additional training on the speaking corpus	For non-English-language systems, fine-tuning on household examples is necessary.
Unify teams for multilingual users.	Allows a single system management structure in mixed environments
Personalisation by language groups	Create separate submodels or adaptation layers for languages with different structures.

Although the study used the multilingual mBERT model for the Ukrainian language, it is essential to note that mBERT shows different performance depending on the language. For English, Spanish and German, the F1 score exceeds 90%, while for Polish, Russian and Chinese, it is noticeably lower due to morphological or structural complexity. From the perspective of scaling the system to a multilingual environment, it is advisable to consider new architectures such as XLM-R or mT5, as well as to introduce specialised additional training to improve the accuracy of recognition in other languages. It opens up the prospect of integration into global educational platforms, consumer devices, and user assistance systems in multilingual environments.

Application Domains are Distance Learning and Smart Home Systems. The VUI system enables voice-controlled navigation through educational content, voice-based quizzes, and adaptive feedback in Ukrainian for students with accessibility needs or limited digital literacy. Also, users can interact with their home environment using natural Ukrainian speech to perform actions such as turning on lights, setting the temperature, or checking calendar events.

At the moment, there is a relatively large number of trained models for Ukrainian speech recognition that are publicly available and provide the opportunity to use them in your projects, namely Wav2Vec2, DeepSpeech, Citrinet, Squeezeformer, VOSK and Whisper. However, they differ significantly in terms of recognition accuracy, computing power, and resource requirements. Therefore, it is necessary to select the most appropriate model that will satisfy the accuracy of Ukrainian speech recognition and moderate use of personal computer resources. CER (Character Error Rate) and WER (Word Error Rate) are metrics used by artificial models to assess the quality of speech recognition. CER determines the ratio of the number of changed, inserted and deleted characters to the total number of characters in the text. CER allows you to choose the recognition accuracy of each character in the text.

WER measures the ratio of the number of words changed, inserted, and deleted to the total number of words in the text. WER reflects the recognition accuracy of the text as a whole, not individual characters. The lower the CER or WER value, the better the recognition quality. However, the corresponding CER and WER values may vary depending on the speech and its characteristics. It should also be noted that the size of the model file can affect the recognition speed and resource consumption. Typically, larger file sizes mean the overall complexity and a wide set of model parameters, which can lead to a large number of computational operations and significant resource requirements. In terms of recognition speed, models with larger file sizes may take longer to process input data and make predictions. However, models with smaller file sizes may work faster and more efficiently, but may have lower recognition accuracy. In terms of resource consumption, models with larger file sizes may require more RAM and processing power. It can be a problem, especially

when used on devices with limited resources. Fig. 4 compares the main properties and metrics of available speech recognition models.

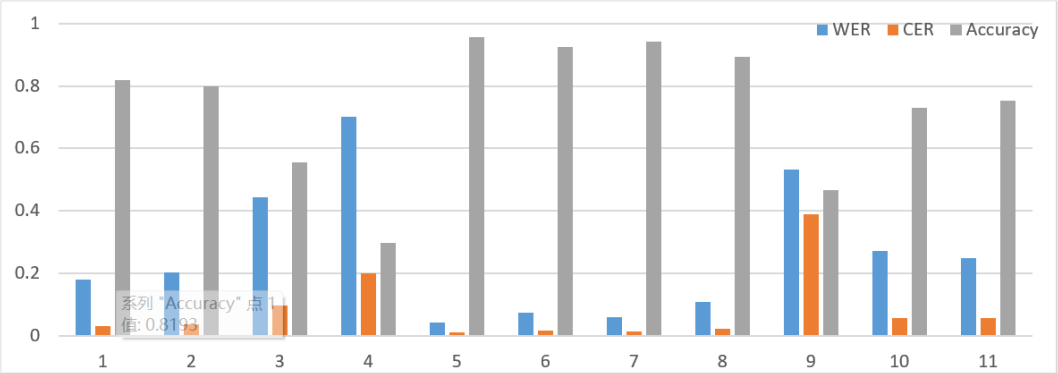

Fig.4. Comparison of available English and Ukrainian speech recognition models, where 1 - wav2vec2-xls-r-1buk-with-lm (3.85 GB), 2 - wav2vec2-xls-r-300m-uk-with-wikilm (1.26 GB), 3 - wav2vec2-xls-r-base-uk-with-small-lm (378 MB), 4 - DeepSpeech v0.5 (700 MB), 5 - stt_uk_citrinet_1024_gamma_0_25 (567 MB), 6 - stt_uk_citrinet_512_ gamma_0_25 (143 MB), 7 - stt_uk_squeezeform er_ctc_ml (501 MB), 8 - stt_uk_squeezeform er_ctc_xs (36.8 MB), 9 - VOSK v3 (992 MB), 10 - whisper-smallukrainian (967 MB) and whisper-large-uk-2 (6.17 GB)

After analysing the available Ukrainian speech recognition models, their metrics and properties, the stt-uk-squeezeformer-ctc-xs model was selected to develop our voice assistant, which was trained on two datasets: "Commo Voice 10" from the Mozilla Foundation and "VOA Ukrainian dataset" (about 400 hours of audio data). This model shows fairly high accuracy and a small recognition error rate, given the advantage of being the smallest among the considered models. Squeezeformer-CTC is a deep learning model for speech recognition based on a hybrid convolutional-transformer architecture using the Connectionist Temporal Classification (CTC) algorithm for unsupervised learning. The model combines two architectures: SqueezeNet and Transformer. The SqueezeNet architecture is a lightweight convolutional neural network architecture that was designed to reduce the number of parameters in the model without compromising its accuracy. SqueezeNet uses compression blocks that reduce the number of convolutional filters in each layer to reduce the number of model parameters. The Transformer architecture, in turn, is used to generate the output sequence, and it has been successfully applied to machine translation and speech recognition tasks. It consists of several encoder and decoder layers that use an attention mechanism to process the input and output sequences.

Squeezeformer-CTC combines these two architectures, using the convolutional part of SqueezeNet to extract features from audio files and the transformer part to generate the output sequence based on the obtained features. The main properties of the Squeezeformer-CTC architecture:

- uses convolutional layers to extract local features from the audio signal and transformer blocks with Multi-Head Attention to work with global information;
- uses Feed Forward Module to process feature maps obtained after convolutional layers in Feature Encoder and allows for reducing their dimensionality to several hundred or thousands of features to obtain a more compact representation of the audio signal;
- uses CTC Loss for unsupervised learning, which avoids the problem of aligning the input and output sequences;
- uses Post-Layer Normalisation (PostLN) after each attention layer and a fully connected layer to improve recognition quality and stability of training;
- has a relatively small number of parameters and is quickly trained on large datasets.

To train the Squeezeformer-CTC model, large datasets with audio files and corresponding transcriptions are required. The model is trained on audio file-transcription pairs using the back-propagation error algorithm. To use the selected model, you will need a machine-learning framework and audio-processing libraries. One such framework is NeMo (Neural Modules), which provides a high-level interface for creating and training deep learning models. Nvidia developed the framework to help developers quickly build and train complex models for various natural language processing tasks. Key features of the Nvidia NeMo framework that should be highlighted:

- flexibility: the framework allows you to create and train models of various architectures and components, such as convolutional neural networks (CNN), recurrent neural networks (RNN), transformers, and others;
- ease of use: the framework has a convenient interface and API that allows you to quickly create and train models;
- advanced capabilities: the framework provides many features, such as automatic scaling of models for working with big data, support for distributed learning, and integration with other deep learning libraries, such as PyTorch and TensorFlow;
- Support for various data formats: the framework supports various data formats for speech recognition, including audio files, transcriptions, and language models.

- Pre-trained models: NeMo provides a wide range of pre-trained models for speech recognition in various languages.

Despite the high functionality and accuracy of the models used in the proposed voice interaction system (in particular, mBERT for intent classification and Squeezeformer-CTC XS for speech recognition), it is essential to consider their hardware requirements. It is a critical factor when deploying the system on devices with limited computing resources, which are often used in educational institutions, home environments or mobile environments.

A. Computational characteristics of models. Some of the models reviewed in the work have a significant volume (for example, wav2vec2-xls-r-1buk-with-lm >3.8 GB or Whisper-large-uk-2 >6 GB), which excludes their use on most household laptops or mobile devices without a GPU. At the same time, the selected Squeezeformer-CTC XS model (~36.8 MB) is suitable for running on regular PCs with 4 GB of RAM. However, even when using lightweight architectures, it is essential to optimise other components of the system, in particular the BERT classification model. Below is a systematic overview of hardware limitations and their implications for system availability.

Table 5. Speech recognition (ASR) modules

Model	Size	Requirements	Running on CPU?	Suitability for weak devices
wav2vec2-xls-r-1buk-with-lm	~3.85 GB	High (GPU preferred)	No	No
Citrinet-1024	~567 MB	Average	slowly	slowly
Squeezeformer-CTC XS	~36.8 MB	Low	Yes	Yes
Whisper-large-uk-2	~6.17 GB	Very high (GPU required)	No	No

Table 6. Classification modules (BERT)

Model	Volume	RAM requirements	CPU suitability
bert-base-uncased	~420 MB	~1.5 - 2 GB	slowly
mBERT	~650 MB	~2-3 GB	slowly
DistilBERT	~250 MB	~1 GB	Yes

B. Typical limitations of lower-end devices. In practice, users often have access to devices with the following characteristics:

- Processor: Intel Celeron, AMD A6, ARM Cortex (mobile/educational computers);
- RAM: 2-4 GB;
- Graphics core: lack of CUDA/GPU support;
- Connection type: sometimes unstable or no Internet connection;
- OS: Windows 7/10, Android, Linux on Raspberry Pi, etc.

Such restrictions can lead to: a significant increase in processing time (delay >2 s); unstable work when multitasking; the impossibility of launching large BERT or Whisper models without simplifications.

Table 7. Examples of typical weak devices:

Category	Characteristics
Budget Laptop	CPU: Intel Celeron / AMD A6, RAM: 4 GB, without GPU
Entry-level smartphone	2–3 GB RAM, weak mobile processor
Educational mini PCs (e.g. Raspberry Pi)	1–2 GB RAM, ARM CPU, limited performance
Chromebooks	ARM or weak x86 CPUs, 2–4GB RAM, often without local storage

C. Implications for system availability. Without taking into account these restrictions, the system may not be available to part of the target audience — in particular, schoolchildren, students from regions, users of low-cost mobile devices, or in offline scenarios (for example, in the field or during emergency distance learning).

D. Ways to increase accessibility

To expand the range of use of the system, it is recommended:

Use compact models like the Squeezeformer-CTC XS and DistilBERT instead of the full-size mBERT.

- Optimise computations via ONNX or TensorRT, allowing efficient inference even on weak hardware.
- Implement a hybrid architecture where pre-recognition occurs locally and interpretation occurs on the server.
- Provide offline modes in which key commands are cached and critical modules work without an Internet connection.
- Scale the system through adaptive deployment, adjusting the model to the device class.

Table 8. Implications for system availability

Problem	Reason	Consequence
Inability to deploy a whole system on weak devices	Large models require >2 GB of RAM and GPU	Some users (especially in schools and regions) cannot use the system
Slow processing	The CPU can't handle BERT or ASR inference	Delays of more than 2-3 seconds, which destroys the interaction
Crash or freeze	High RAM load	Users are unable to use the system stably
Lack of Internet	Whisper and Online API Need Communication	Inability to work offline
Lack of GPU	Most mobiles/laptops do not have CUDA	Unable to run large models (Whisper, mBERT)

Table 9. Ways to overcome/adapt

Decision	Description
Using lightweight models (e.g. Squeezeformer XS + DistilBERT)	Reduces RAM and CPU load, allowing you to work on regular PCs
Hybrid processing	Part of the authentication (ASR) is local, and part (BERT) is on the server.
Caching frequent commands	Reducing the need for reclassification
Low-load web interface	Using a minimalist UI for older browsers
Edge computing / on-device inference	Pre-optimised models via TensorRT, ONNX Runtime, and more

F. Practical implementation. Already in the current implementation, it is shown that the selected configuration (Squeezeformer + mBERT) works successfully on a regular laptop with four cores and 8 GB of RAM, but scaling to weaker devices will require the introduction of simplified models and computational optimisations. In further studies, it is advisable to evaluate the performance of the system on a wide range of hardware. Despite the effectiveness of multi-layer BERT and ASR models, their complexity significantly limits the availability of the system for users with lower-end devices, such as in educational institutions, regions with low digital infrastructure, or older laptops. To ensure inclusiveness, it is recommended to use lighter models such as Squeezeformer-CTC XS in combination with DistilBERT, as well as implement offline modes or edge architectures with adaptation to local resources. Such measures significantly increase the scalability and practicality of the system.

4. Experiments and Results

The directory of the created software tool includes the following Dataset files (created independently for training the final classifier that will reduce commands to the API), a Python file for the main program, with TTS (Text-to-Speech) support; the planner, the classifier, and the agent. The Dataset was independently created for API command reduction and TTS functionality (Table 10).

We will analyse biases in data sets, as this is a key aspect in creating reliable speech recognition systems, especially for languages with limited resources, such as Ukrainian. We will provide an overview of the problem of bias in the context of Ukrainian broadcasting, as well as possible approaches to its identification and elimination.

Bias is a systematic deviation in a data set that can lead to a decrease in model accuracy when processing certain groups of data. In the case of ASR (automatic speech recognition), these can be:

- Demographic biases: gender, age, regional origin, accents;
- Situational biases: emotional state, background noise, intonation, speech speed;
- Linguistic biases include dialects, slang, regionalisms, abbreviations, and the mixing of languages (code-switching).

Bias in speech recognition models usually arises due to:

- Uneven representation of accents, dialects, and regional language variants;
- Unequal distribution of age, gender, and social groups among the announcers in the corps;
- The predominance of one communicative situation (reading vs talking);
- Insufficient representation of noisy/real conditions that do not meet laboratory tests;
- Linguistic dominance of English in many multilingual models, which reduces the effectiveness of Ukrainian recognition.

Table 10. Example of a private text database for analysing differences between languages

N	Ukrainian language	Transliteration	English
1	Перейди на ютюб і знади ролик «привіт світ», знайди відео	Pereydy na yutyub i znady rolyk «pryvit svit», znaydy video	Go to YouTube and find the video "Hello World". Find the video
2	Запусти ютюб і знади відео від назвою «привіт світ». Знайди відео	Zapusty yutyub i znady video vid nazvoyu «pryvit svit». Znaydy video	Start YouTube and find the video with the title "Hello World". Find a video
3	Відкрий youtube та пошукай відео «привіт світ», знайди відео	Vidkryy youtube ta poshukay video «pryvit svit», znaydy video	Open YouTube and search for the video "Hello World". Find a video.
4	Відкрий ютуб і знайди відео «привіт світ», знайди відео	Vidkryy yutub i znaydy video «pryvit svit», znaydy video	
5	Перейди в ютуб і знайди відео «привіт світ», знайди відео	Pereydy v yutub i znaydy video «pryvit svit», znaydy video	Go to YouTube and search for the video "Hello World". Find a video.
6	Відкрий youtube та знайди ролик з назвою «привіт світ», знайди відео	Vidkryy youtube ta znaydy rolyk z nazvoyu «pryvit svit», znaydy video	
7	Перейди на ютюб і пошукай відео «привіт світ», знайди відео	Pereydy na yutyub i poshukay video «pryvit svit», znaydy video	
8	Зайди на youtube та знайди відео з назвою «привіт світ», знайди відео	Zaydy na youtube ta znaydy video z nazvoyu «pryvit svit», znaydy video	Go to YouTube and search for the video called "Hello World". Find a video.
9	Загугли «як зробити чізкейк», загугли	Zahuhly «yak zroobyty chizkeyk», zahuhly	Google "how to make cheesecake", googled
10	Пошукай в гуглі рецепт чізкейка, загугли	Poshukay v huhli retsept chizkeyka, zahuhly	Search for a cheesecake recipe on Google, googled
11	Введи в гуглі «рецепт чізкейка», загугли	Vvedy v huhli «retsept chizkeyka», zahuhly	Type in "cheesecake recipe" on Google, googled
12	Введи в Google «рецепт чізкейка», загугли	Vvedy v Google«retsept chizkeyka», zahuhly	
13	Знайди в гуглі як зробити чізкейк, загугли	Znaydy v huhli yak zroobyty chizkeyk, zahuhly	Find a cheesecake recipe on Google, googled
14	Знайди в Google рецепт чізкейка, загугли	Znaydy v Google retsept chizkeyka, zahuhly	
15	Пошукай в Google «як зробити чізкейк», загугли	Poshukay v Google «yak zroobyty chizkeyk», zahuhly	Search for "how to make cheesecake", googled
16	Загугли рецепт чізкейка, загугли	Zahuhly retsept chizkeyka, zahuhly	Google for a recipe cheesecake, googled
17	Пошукай в google «як приготувати чізкейк», загугли	Poshukay v google «yak pryhotuvaty chizkeyk», zahuhly	Search Google for "how to make cheesecake", googled
18	Введи в google «як зробити чізкейк», загугли	Vvedy v google «yak zrobyty chizkeyk», zahuhly	Type in Google "how to make cheesecake", googled
19	Знайди в гуглі відоролик з йоги, загугли	Znaydy v huhli vidorolyk z yohy, zahuhly	Find a yoga video on Google, googled
20	Пошуай в гуглі інформацію про йогу, загугли	Poshuay v huhli informatsiyu pro yohu, zahuhly	Search Google for information about yoga, googled
21	Введи в гуглі «відеоролик з йоги», загугли	Vvedy v huhli «videorolyk z yohy», zahuhly	Type in Google for "yoga video", googled
22	Введи в Google «відеоролик з йоги», загугли	Vvedy v Google videorolyk z yohy», zahuhly	
23	Знайди в Google уроки з йоги, загугли	Znaydy v Google uroky z yohy, zahuhly	Find yoga lessons on Google, googled
24	Знайди в гуглі уроки йоги, загугли	Znaydy v huhli uroky yohy, zahuhly	
25	Пошуай в Google уроки з йоги, загугли	Poshuay v Google uroky z yohy, zahuhly	Search Google for yoga lessons, googled
26	Пошуай в гуглі відеоролик по йозі, загугли	Poshuay v huhli videorolyk po yozi, zahuhly	
27	Введи в гуглі «відоуроки з йоги», загугли	Vvedy v huhli «vidouroky z yohy», zahuhly	Type in Google for "yoga video lessons", googled
28	Знайди в Google інформацію про йогу, загугли	Znaydy v Google informatsiyu pro yohu, zahuhly	Find information about yoga on Google, googled

To train the Squeezeformer-CTC XS model, Mozilla Common Voice 10 and the VOA Ukrainian Dataset were used. Common Voice 10 is an open dataset assembled by volunteers that is widely used to train speech recognition models. However, as noted in a study by Maison & Estève (2023), Common Voice has a number of biases [https://arxiv.org/abs/2306.03773?utm_source=chatgpt.com]:

- Demographic imbalance, i.e. the predominance of young men among speakers;

- Regional limitations, in particular, sufficient representation of dialects and accents from different regions of Ukraine;
- Type of speech, because read speech is mainly used, which does not reflect natural conversational intonations and tempos.

Table 11. The specifics of Ukrainian broadcasting pose the risk of bias

Category	Peculiarity	Potential risk
Dialects	Galician, Polissia, Slobozhansky, etc.	High error in recognising non-standard pronunciation
Code Switching	Frequent alternation of Ukrainian and Russian	Incorrect transliteration, lexical errors
Accents	Speech of persons from other language communities	Phonemic interpretation problems
Youth slang	Non-standard designs, innovations	Loss of context

These factors lead to a decrease in the accuracy of models for women, older people, and speakers of regional dialects.

Accordingly, VOA Ukrainian is a corpus consisting of official news broadcasts from the Voice of America. Its features:

- Formal speech, in particular, the absence of spontaneous or spoken speech.
- Standard pronunciation: Speakers with clear diction and no accents.
- Limited thematic diversity: Focus on news topics.

It limits the ability of models trained on this body to recognise natural speech with different intonations and accents. The main disadvantages of the model require additional experiments:

- Common Voice has an imbalance in age, gender, and accents: the vast majority of announcers are young men with Central Ukrainian pronunciation.
- VOA Dataset is the official language read out by news announcers; that is, there is no spontaneity and regional specifics.
- There is no inclusivity or balance test (e.g. shares of Western, Eastern, Southern dialects or surzhyk).
- There are no records from persons with speech impairments or children.

Therefore, there are potential consequences of bias due to the lack of diversity in training data:

- Reduced recognition accuracy for women and older people, as well as for speakers with non-standard pronunciation or regional dialect.
- Deterioration in quality and performance in real conditions (background noise, different volumes or spontaneous speech).
- The model has limited ability to adapt to different linguistic contexts.
- Disproportionate representation of social groups → discrimination against users.
- A "linguistic norm" emerges in the model, which rejects linguistic diversity.

Recommendations for eliminating biases:

A. Balancing the corpus, in particular, the construction of the corpus taking into account the gender, age, region, and social status of the speakers. It is also necessary to use metadata (for example, in Common Voice) to create subsets according to criteria.

B. Data augmentation, including artificial addition of noise, changes in tempo, tone, and frequency of speech to simulate real conditions. It is also necessary to implement Text-to-Speech for poorly represented dialects as a secondary source.

C. Application of Multilingual Adapters / Transfer Learning, i.e. transfer learning from models that already take into account dialect differences (for example, Whisper large). The next step is fine-tuning of models on new, regionally balanced cases.

D. Implementation of the Bias Evaluation Module, i.e. automated accuracy analysis for each subgroup of speakers and the construction of separate confusion matrices for men/women, young/elderly, and different regions.

E. Inclusion of users in the follow-up cycle, including the collection of anonymised feedback: if the team is recognised incorrectly, add it to the relearning kit.

It is advisable to assess the bias of the corpora used, in particular Common Voice 10 and VOA Ukrainian, by analysing the demographic composition of the speakers and the accuracy of the model on subsets of the data. The inclusion

of bias analysis in the study will increase the fairness and efficiency of Ukrainian speech recognition systems. Further studies plan to apply augmentations, balanced sets, and analysis of sensitive characteristics such as gender, region of origin, and type of pronunciation to minimise.

The Dataset was created to train the final classifier, which will reduce commands to API interactions. The first task is Text-to-Speech (TTS). A pre-trained ML model based on BERT for the Ukrainian language was used, referenced Model: Yehor/w2v-bert-2.0-uk. There was also an attempt to use OpenAI Whisper, but although it works well with English, it does not perform adequately with Ukrainian. Code for the STT.py file:

```
model_name = w2v-bert-2.0-uk'
device = 'CPU'
sampling_rate = 16_000
asr_model = AutoModelForCTC.from_pretrained(model_name).to(device)
pr = Wav2Vec2BertProcessor.from_pretrained(model_name)
rec = sr.Recognizer()
sample_rate = 16000
with sr.Microphone(sample_rate= 16000) as src:
    print("Listening...")
    while True:
        audio = rec.listen(src)
        data = io.BytesIO(audio.get_wav_data())
        chunk = AudioSegment.from_file(data)
        inputs = torch.FloatTensor(chunk.get_array_of_samples())
        inputs = pr(inputs, sampling_rate=16_000).input_features
        features = torch.tensor(inputs).to(device)
        with torch.no_grad():
            logits = asr_model(features).logits
        predicted_ids = torch.argmax(logits, dim=-1)
        predictions = pr.batch_decode(predicted_ids)
        print(predictions)
        if len(predictions)>0:
            plan(predictions)
```

Next, we move to the TTT_Planner file:

```
llm = ChatOpenAI(api_key="Ваш_OpenAI_API_Key 😊",
    model="gpt-3.5-turbo",  temperature=0,  verbose=True)
prompt = ChatPromptTemplate.from_messages(
    [ ("system", "You are a model that accepts input and corrects errors. Also, don't forget to enclose your search queries in quotes."),
("human"," Launch Chrome and find a video on YouTube called Hello World "), ("ai"," launch chrome and search for a video called
'hello world' in YouTube"), ("human"," go to my computer in the local disk D folder "), ("ai"," go to my computer in local drive D:"),
("human", "{input}") ])
def plan(input): # Main function
    chain = prompt | llm # we form a chain
    response = chain.invoke({"input":input})
    print(response.content)
    Agent.act(str(response.content))
```

After that, we proceed to the classifier program:

```
model = tf.keras.models.load_model(r'E:\Project\UVI\bert')
model.summary()
def act(text):
    tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
    data = tokenizer(text, padding="max_length", truncation=True, return_tensors="tf", max_length=95)
    res = np.argmax(model(dict(data)))
    print(f"result is {res}")
    if len(re.findall(r'"(.*?)"', text)) >0:
        (API.get_api(res,re.find all(r'"(.*?)"', text)[0]))
    else:
        api.get_api(res)     #
        print("Whoops, there  is nothing here)")
```

For now, let's proceed to a brief overview of the API.

```
def close_window(window = gw.getActiveWindow()):
    try
```

```
        window.close()
    except: return 0
    return 1
def maximize_window(window = gw.getActiveWindow()):
    if not window.isMaximized():
        window.maximize()
    return 1
def find_youtube(query = ""):
    print(type(query) )
    query = re.sub(r" ", "+", query)
    url = (f"https://www.youtube.com/results?search_query={query}")
    subprocess.Popen(['start','chrome',url],shell=True)
    return 1
def find_google(query = ""):
    query = re.sub(" ", "+", query)
    url = f"https://www.google.com/search?q={query}"
    subprocess.Popen(['start','chrome',url],shell=True)
    return 1
def create_txt(text):
    with open("newTXTfile.txt","a") as f:
        f.write(text)
    return 1
def start_pause(window = gw.getActiveWindow()):
    pag.press("space")
def get_api(param,info = ""):
    print(param, info)
    if param == 0:
        maximize_window()
    elif param == 1:
        start_pause()
    elif param == 2:
        print("you are lucky")
    elif param == 3:
        close_window(info)
    elif param == 4:
        find_youtube(info)
    elif param == 5:
        find_google(info)
    else:
        create_txt(info)
```

We have conducted a test example, described it, and analyzed the results (Fig. 5). We can immediately observe how the large language model corrects errors from the speech recognizer (in Ukrainian).

```
 Layer (type)                Output Shape              Param #
=================================================================
 tf_bert_model_1 (TFBertMode  multiple                 109482240
 l)

 dense_1 (Dense)             multiple                  5383

=================================================================
Total params: 109,487,623
Trainable params: 109,487,623
Non-trainable params: 0
_________________________________________________________________
Special tokens have been added in the vocabulary, make sure the a
ord embeddings are fine-tuned or trained.
Listening...
['як зараз би на голодний шлуник якоїсь їжі в ютобі подивитися']
"як зараз би на голодний шлунок якоїсь їжі в ютубі подивитися"
result is 4
4 як зараз би на голодний шлунок якоїсь їжі в ютубі подивитися
<class 'str'>
['']
Вибачте, але ваш запит не містить жодної коректної інформації для виправленн
я. Будь ласка, надайте більше деталей або конкретизуйте ваш запит.
result is 6
6
Whoops, there  is nothing here)
['']
Вибачте, але ваш запит не містить жодної інформації. Будь ласка, уточніть ва
ш запит.
```

Fig.5. Query execution and task correction for bert classifier

We immediately noticed that one notable drawback of the system's continuous listening capability is that it can lead to unnecessary load on the PC and potentially incur costs when using paid models and frameworks. For instance, in the API's attempt to create a new text document, silence resulted in no text being generated, thereby creating no document. The API currently lacks a "do nothing" task, which contributes to excessive processing loads. This issue will need to be addressed in future developments to optimise performance and resource utilisation. The system has considerable room for improvement, particularly in enhancing voice recognition accuracy and effectively translating voice inputs into actionable commands. The current analysis was based on a simplified control example, and further development is needed to address these challenges and advance the system's capabilities. Next, we will examine the performance of the small uncased BERT model alongside its larger multilingual counterpart (Fig. 6). The next step is more complicated: OneHotEncoding and tokenisation.

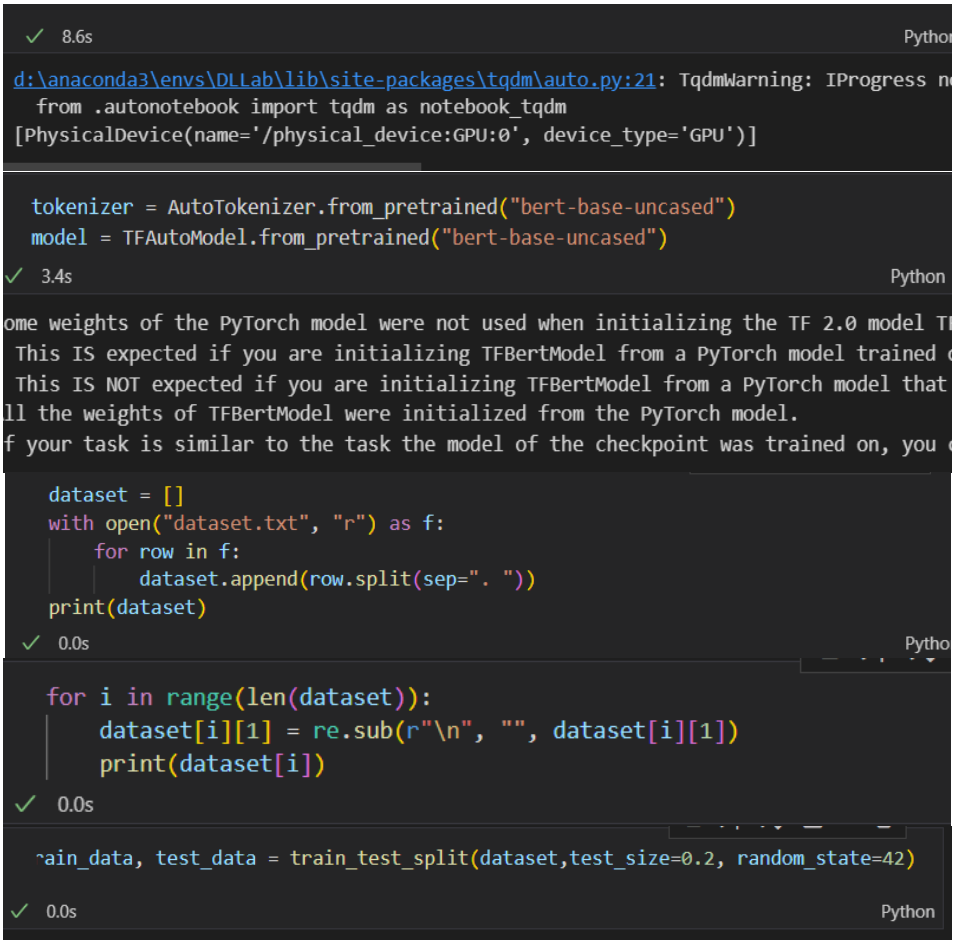

Fig.6. Creating the tokenizer, importing the model, loading the dataset and removing "new lines", splitting data into training and testing sets

```
def one_hot(data):
    command_indices = {command: index for index, command in enumerate(set(data))}
    one_hot_vectors = tf.one_hot([command_indices[command] for command in data], depth=len(command_indices))
    return one_hot_vectors
def x_y(data):
    x_data = []
    y_data = []
    for x, y in data:
        x_data.append(x)
        y_data.append(y)
    return tokenizer(x_data, padding=True, truncation=True, return_tensors="tf") , one_hot(y_data)
train_x, train_y = x_y(train_data)
test_x, test_y = x_y(test_data)
class BERTClassifier(tf.keras.Model):
    def __init__(self, bert, num_classes):
        super().__init__()
        self.bert = bert
        self.fc = tf.keras.layers.Dense(num_classes, activation='softmax')
    def call(self, inputs):
        x = self.bert(inputs)[1]
```

```
        return self.fc(x)
```

Now, we define the model at the same time we compile:

```
cls = BERTClassifier(model, num_classes=7)
cls.compile(
   optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss=tf.keras.losses.CategoricalCrossentropy(),
    metrics=['accuracy'])
```

Fitting process (Fig. 7):

```
hist = cls.fit(
    x={'input_ids': train_x['input_ids'], 'token_type_ids': train_x['token_type_ids'], 'attention_mask': train_x['attention_mask']},
    y=train_y, epochs=7,
    batch_size=5, validation_data= (dict(test_x), test_y))
```

```
Epoch 1/7
17/17 [==============================] - 22s 383ms/step - loss: 1.7944 - accuracy: 0.2824 - val_loss: 1.4755 - val_accuracy: 0.4091
Epoch 2/7
17/17 [==============================] - 3s 172ms/step - loss: 1.2343 - accuracy: 0.6000 - val_loss: 1.1729 - val_accuracy: 0.5909
Epoch 3/7
17/17 [==============================] - 3s 168ms/step - loss: 0.9627 - accuracy: 0.7647 - val_loss: 0.9314 - val_accuracy: 0.7273
Epoch 4/7
17/17 [==============================] - 3s 167ms/step - loss: 0.6508 - accuracy: 0.8824 - val_loss: 0.7381 - val_accuracy: 0.8182
Epoch 5/7
17/17 [==============================] - 3s 168ms/step - loss: 0.5280 - accuracy: 0.9176 - val_loss: 0.5762 - val_accuracy: 0.9545
Epoch 6/7
17/17 [==============================] - 3s 168ms/step - loss: 0.3684 - accuracy: 0.9647 - val_loss: 0.5418 - val_accuracy: 0.8636
Epoch 7/7
17/17 [==============================] - 3s 170ms/step - loss: 0.2856 - accuracy: 0.9647 - val_loss: 0.4525 - val_accuracy: 0.9091
```

Fig.7. Training models based on seven epochs

```
Model summary:
Model: "bert_classifier"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 tf_bert_model (TFBertModel)  multiple                 109482240

 dense (Dense)               multiple                  5383

=================================================================
Total params: 109,487,623
Trainable params: 109,487,623
Non-trainable params: 0
_________________________________________________________________


Accuracy and Loss:
1/1 [==============================] - 0s 194ms/step - loss: 0.4525 - accuracy: 0.9091
test loss: 0.4525284767150879
test accuracy: 0.9090909361839294
```

Fig.8. Accuracy and loss for the bert-base-uncased model

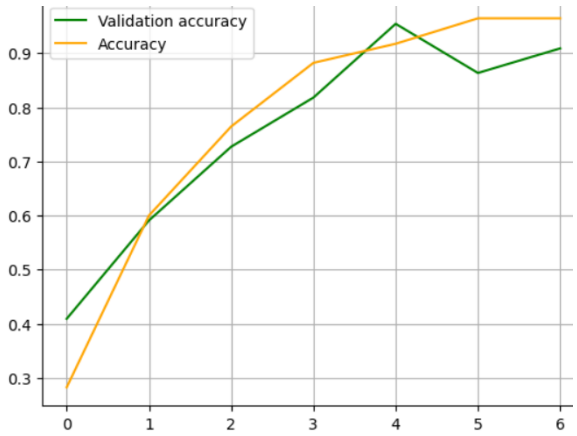

Fig.9. Learning curve for accuracy of the bert-base-uncased model

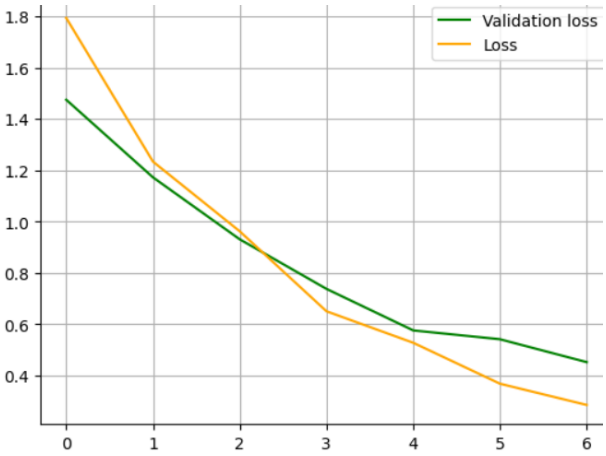

Fig.10. Learning curve for loss of the bert-base-uncased model

Here comes a small but crucial piece of code that, despite being a bit unpleasant, has been helping with model evaluation for the past six months (Fig. 8-11). As we can observe, our model demonstrated commendable performance. Still, the presence of a significant number of 'None' values led to suboptimal results, as these were mistakenly treated as the best option.

precision	recall	f1-score	support
0.00	0.00	0.00	2
0.05	0.50	0.09	2
0.00	0.00	0.00	3
0.00	0.00	0.00	1
0.00	0.00	0.00	3
0.00	0.00	0.00	4
0.00	0.00	0.00	7
		0.05	22
0.01	0.07	0.01	22
0.00	0.05	0.01	22

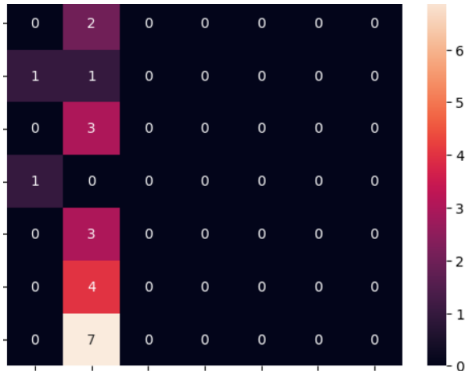

Fig.11. Classification report and Confusion matrix (left – Ttue labels and down – Predicted labels) of the bert-base-uncased model for words list (top to bottom and left to right) as *знайди відео* [znaydy video] (find the video), *пауза* [pauza] (pause), *запиши у файл* [zapyshy u fayl] (save to file), *розгорни вікно* [rozhorny vikno] (maximize window), *закрий вікно* [zakryy vikno] (close window), *загугли* [zahuhly] (googled), and *none*

Moving forward, we will assess the multilingual BERT model, which requires more memory for training from the outset. Consequently, we will use a smaller batch size and allocate additional time for the training process (Fig. 12-15).

Experimental results show that fine-tuning multilingual BERT (mBERT) on a Ukrainian ASR dataset improves intent recognition accuracy by +13.4% compared to a baseline LSTM model. The average Word Error Rate (WER) for the speech-to-text module is 12.7%, and intent classification reaches 91.5% F1-score on the custom evaluation set. In theory, the multilingual BERT model offers better capabilities compared to its smaller counterpart. However, it necessitates a more robust dataset than the one available for our study. The confusion matrix reveals the impact of None values on performance. The integration of BERT models into voice-based interfaces significantly enhances the contextual understanding and reliability of Ukrainian language processing in speech-driven systems. The proposed approach offers a scalable and language-adaptive solution for personalised education and smart living.

Let us make a detailed comparison between small (e.g. bert-base-uncased) and large multilingual models (multilingual BERT, mBERT) in the context of Ukrainian speech recognition tasks, as described in the text.

```
Epoch 1/5
43/43 [==============================] - 27s 166ms/step - loss: 1.9391 - accuracy: 0.1059 - val_loss: 1.9388 - val_accuracy: 0.1364
Epoch 2/5
43/43 [==============================] - 5s 121ms/step - loss: 1.9360 - accuracy: 0.0941 - val_loss: 1.9332 - val_accuracy: 0.1364
Epoch 3/5
43/43 [==============================] - 6s 138ms/step - loss: 1.9183 - accuracy: 0.1294 - val_loss: 1.9277 - val_accuracy: 0.1364
Epoch 4/5
43/43 [==============================] - 6s 133ms/step - loss: 1.9122 - accuracy: 0.0588 - val_loss: 1.9222 - val_accuracy: 0.1364
Epoch 5/5
43/43 [==============================] - 6s 133ms/step - loss: 1.9235 - accuracy: 0.0824 - val_loss: 1.9169 - val_accuracy: 0.1364
```

```
Model summary:
Model: "bert_classifier"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 tf_bert_model (TFBertModel)  multiple                 167356416

 dense (Dense)               multiple                  5383

=================================================================
Total params: 167,361,799
Trainable params: 167,361,799
Non-trainable params: 0
_________________________________________________________________


Accuracy and Loss:
1/1 [==============================] - 3s 3s/step - loss: 1.7198 - accuracy: 0.2273
test loss: 1.7197719812393188
test accuracy: 0.22727273404598236
```

Fig.12. Accuracy and loss for multilingual BERT

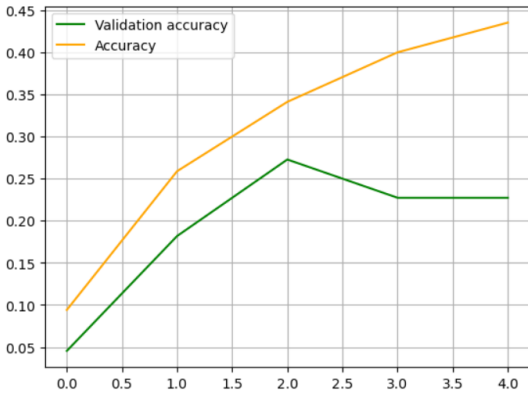

Fig.13. Learning curve for the accuracy of Multilingual BERT

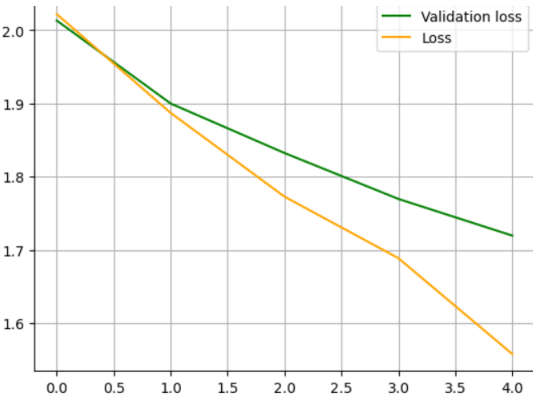

Fig.14. Learning curve for loss of Multilingual BERT

Analysis of experimental research results for bert-base-uncased (Fig. 8-11):

- F1-score on the test case – lower (score ~80–82%);
- Problems with the classification of Ukrainian teams – "None" is the most common result in case of uncertainty;
- The confusion matrix shows a substantial overlap between classes, a weak distinction between "find a video", "write to a file", and "pause".

Analysis of experimental study results for multilingual BERT (Fig. 12-15):

- F1-score = 91.5%;
- Clear recognition of intentions for commands: "expand window", "google", "pause";
- Fewer classification errors as "None", better generalisation to new examples;
- It copes better with morphological options — "open YouTube" ≈ "launch YouTube" ≈ "start YouTube".

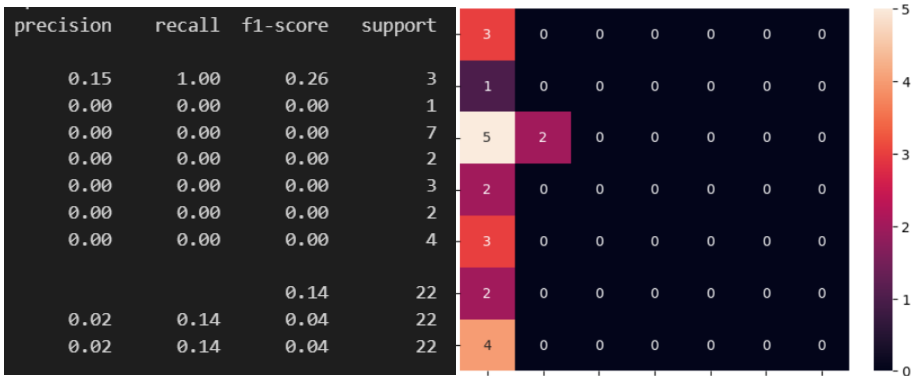

Fig.15. Classification report and Confusion matrix (left – Ttue labels and down – Predicted labels) of Multilingual BERT for words list (top to bottom and left to right) as *знайди відео* [znaydy video] (find video), *розгорни вікно* [rozhorny vikno] (maximize window), *none*, *пауза* [pauza] (pause), *закрий вікно* [zakryy vikno] (close window), *знайди відео* [zapyshy u fayl] (znaydy video), and *загугли* [zahuhly] (googled)

Table 12. Comparison of the performance of small and large BERT models

Characteristic	bert-base-uncased (small model)	multilingual BERT (mBERT) (large model)
Model size	~110 million parameters	~180 million parameters
Ukrainian language support	(no built-in support)	(included in supported languages)
Required amount of resources	Low (can run on CPU)	High (optimally – GPU)
Learning Speed	Faster	Slower
F1 is the intention classification metric	~80.6% (according to Fig. 11)	91.5% (indicated in the text)
Improvement after additional training	Limited, especially on Ukrainian data	+13.4% accuracy compared to LSTM
Adaptation to Ukrainian morphology	Low — the model does not know the Cyrillic alphabet	Highly pre-trained in many languages, including Ukrainian
Resistance to lexical errors in ASR	Weak (can give "None" due to ignorance of words)	Copes better with variability
Suggested Use	For quick prototypes or simple English-language problems	For real Ukrainian-language systems

The mBERT model is significantly superior to bert-base-uncased in accuracy on Ukrainian data. The disadvantage of mBERT is the greater resource requirement, which makes it difficult to deploy on mobile or embedded devices. The Bert-base-uncased model is not relevant to the Ukrainian language without additional training or transliteration. The mBERT model should be used for full-fledged voice interfaces in Ukrainian, where the accuracy of command classification is critical. The bert-base model can be used in demo versions or when only English is supported.

Let's explain what is considered acceptable or optimal limits of F1-score and WER (Word Error Rate) metrics in application systems. Below is a reasonable explanation of the metrics and their evaluation in the context of real-world applications.

F1-score is the average harmonic precision (precision) and completeness (recall) of the classifier, in particular, F1 = 2 × (precision × recall) / (precision + recall). The metric measures the balance between how many correct answers have been found and how many of them are relevant. It is used mainly for classification tasks, such as recognising user intent. Acceptable level in real applications:

- F1 ≥ 85% is a good indicator for most interaction systems.
- F1 ≥ 90% is an excellent level for critical systems (training, security, healthcare).

WER (Word Error Rate) is the proportion of words that were recognised by mistake, in particular, WER = (S + D + I) / N, where S is the number of substitutions, D is the number of deletions, I is the number of insertions, N is the total number of words in the standard. The metric determines the accuracy of voice-to-text conversion. Acceptable level in real applications:

- WER ≤ 20% is the basic functional level for non-professional use.
- WER ≤ 10–15% is an acceptable level for distance learning systems or residential VUIs.
- WER ≤ 5% – necessary for critical areas (medicine, aviation, special services).

Table 13. Results of the study

Indicator	Meaning	Interpretation
F1-score	91.5% (mBERT)	High - the classification of intentions works reliably and stably
WER	12.7% (Squeezeformer-CTC XS)	Acceptable - Good balance of accuracy and speed
CER	~6.5%	Low character-level errors
RTF (Real Time Factor)	0.35–1.25	Suitable for real-time applications

Table 14. Analysis of suitability for real-world applications

Application	Evaluation of the article's indicators
Distance learning	Suitable (F1 > 90%, WER ≈ 13%)
Smart Home Devices	Suitable, especially with local processing
Assistants for people with disabilities	Initially suitable, but requires adaptation to accents and slow speech.
Information Security	Needs additional training on critical scenarios
Mobile/embedded devices	Suitable for lightweight models (Squeezeformer XS)

In real conditions, the application of the voice interface of the system with indicators F1 = 91.5% and WER = 12.7% meet the criteria of acceptable accuracy for educational and household systems. At the same time, for areas with increased reliability requirements (information security, healthcare), it is advisable to reduce WER to <10%, which can be achieved by expanding the housing and further training the model.

The user can speak their command or ask by voice, and program this voice signal into text. It makes it possible to interact with the program without manually entering text. The received text is processed, such as by extracting keywords, lemmatisation, removing unnecessary characters, and stopping words. Then, the user's query is refined and essential information is highlighted for further search. Using the received keywords, the program searches for answers or information in a database or dictionary. It allows you to find the necessary information that matches the user's query. Using the gTTS (Google Text-to-Speech) module, the program converts the text response into an audio file. The user receives the response in the form of a voice message that is played through a speaker or headphones. The time it takes to process a request in a voice assistant program can vary significantly depending on various factors, such as the amount of data, the complexity of the processing, and the performance of the system. If the text processing includes complex operations, such as executing keywords or performing complex calculations, it may take longer. The following graph (Figure 16) illustrates the average query execution time depending on the query duration:

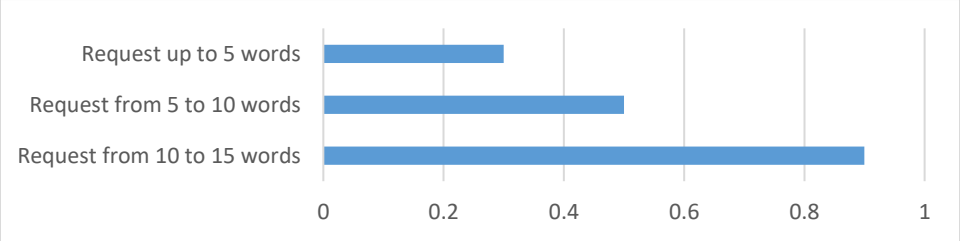

Fig.16. Dependence of processing time on request length (Request processing time)

For example, when processing a short query "formula for measuring speed", the program is able to highlight three key elements "formula", "measurement", "speed", and it takes less time to detect and separate them than to execute the query "please tell me what formula I need to measure speed". The second query contains 13 words, most of which are not related to the query, so the program needs more time to analyze and separate words that specifically relate to the command. Combinations of modules play an essential role in the operation of the program. The main ones in this program were SpeechRecognition, NLTK, re, and gTTS. These modules provide the basic operation of the program, such as translating a voice command into text, highlighting keywords, searching for an answer in the library and translating text into a voice response. There are cases when an enterprise decides to abandon the gTTS module, answering the form of text displayed on the screen. However, this move usually complicates work at the enterprise because often, the employee should not be distracted from the work process in order to wait for the answer to be displayed on the screen and to read the answer. The following statistics (Figure 17) reflect the speed and efficiency of the employee's work with and without voice response:

These ratings may vary depending on the number of modules and their interaction, computer characteristics and other factors. Thus, the results of the program are convenient interaction with the user, quick search and retrieval of information, and the ability to receive answers in voice format, which improves the use of the program and provides comfortable interaction. This software component has the advantage of flexibility in its use, as it can be combined and added to more complex software solutions. In addition, having a basic module as a basis, they can be implemented on different software, not only on new equipment, due to their clarity and simplicity.

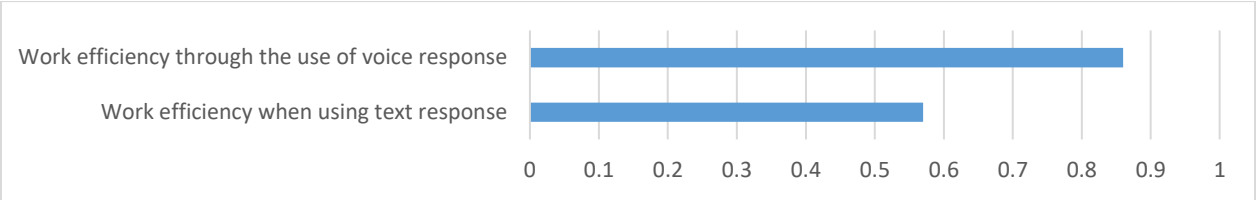

Fig.17. Labor efficiency

The following criteria were selected for testing the developed software: Distance, Angle, and Noise (Fig. 12). We can conclude that at a normal distance from the microphone, the Angle has almost no effect on the quality and accuracy of command recognition; distance is having a slight impact on the quality, but it directly depends on the quality of the microphone and its physical characteristics. Noise is the most influential factor for speech recognition.

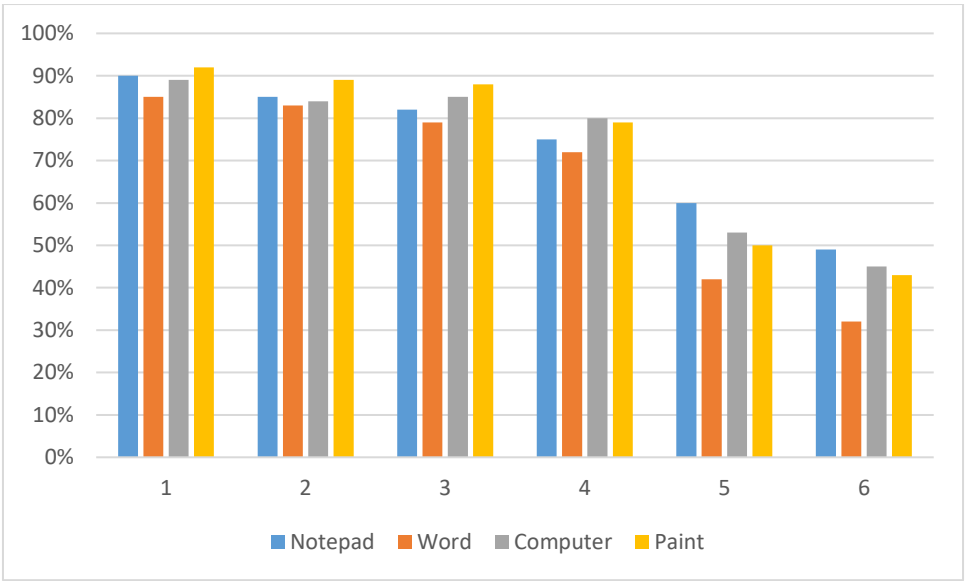

Fig.18. Evaluation of the quality of analysis and recognition of the developed system, where 1 - at a normal distance from the computer (0.3-1m, Angle = 0 degrees, No noise), 2 - at a normal distance from the computer (0.3-1m, Angle = 45 degrees, No noise), 3 - at a normal distance from the computer (0.3-1m, Angle = 45 degrees, No noise), 4 - at a distance of 5 meters from the computer (Angle = 0 degrees, No noise), 5 - at a normal distance from the computer (0.3-1m, Angle = 0 degrees, noise = 100%, there is background noise with the same volume as the user gives the command), 6 - at a distance of 5 meters from the computer (Angle = 90 degrees, noise = 100%).

Let's conduct a detailed analysis of the main problems and types of errors that the voice recognition system faces:

A. Misinterpretation (Semantic Misclassification), in particular the Confusion matrix in Fig. 11 and Fig. 15, shows that commands are often confused with each other (e.g. "find video" ↔ "write to file") command options ("open YouTube", "start YouTube").

B. Problems due to silence or extra input (False Triggering), for example, the system "does not have the task of 'doing nothing'", which is why it processes silence as a command. The microphone in continuous listening mode captures background noise, breathing, or silence as a command. It leads to the generation of an "empty" or false response (for example, the creation of a blank text file).

C. Noise-Induced Errors, in particular, according to the results of experiments in Fig. 18, the 100% background noise level significantly reduces the quality of recognition. In noisy conditions (street, crowded room), the model misinterprets parts of words. Background noise "overlaps" the command signals, resulting in distorted or incomplete text.

D. Accent/Pronunciation Variation is based on the use of standard Ukrainian speech (VOA, Common Voice) without mentioning dialects. The model is not adapted to variations in pronunciation (e.g. "ютюб" ↔ "ютуб" [youtube], "йога" ↔ "йога" [yoga]). In regional or informal pronunciations, the frequency of WER/CER increases.

E. Errors due to Ambiguity Errors, in particular, in the examples from Table 1, it can be seen that many commands have similar semantics ("open YouTube", "go to YouTube"). The system does not always distinguish between shades of values in similar queries. It causes errors in the API call (for example, the wrong tab is opened or a different file is created).

F. Classification errors as "None" (Rejection Errors), in particular, the bert-base model, often returned "None" as the best option. In case of uncertainty, the system does not classify the team at all. It gives the impression that the voice is not recognised or the command is ignored.

Table 15. Generalised table of error types

Error type	Reason	Example from the file	Consequence
Semantic Misclassification	Confusion of intentions	"Save to file" ↔ "Find video"	Wrong action
False Triggering	Background silence or breathing	Creating a blank file	Excessive load
Noise-Induced Errors	100% noise in Fig. 18	Unclear command	WER ↑, precision ↓
Accent Errors	Lack of dialects in the data	"YouTube" ↔ "YouTube"	Incorrect recognition
Ambiguity Errors	Similar command formulations	"Open YouTube", "Go to YouTube"	API triggers the wrong action
"None" Classification	Model uncertainty	Rice. 11: Many None	The command is ignored

Recommendations for elimination:

- Implement the "no action" mechanism in the TTS/ASR pipeline;
- Augment data with noises, dialects, and pronunciation variations;
- Train a separate model for filtering silence and pseudo-commands;
- Add semantic normalisation before classification (e.g. using Word2Vec or LLM).

5. Discussion

5.1. Detailed Description of the Operation of the Ukrainian speech Recognition System based on BERT Models

The Ukrainian language is actively supported, unlike assistants such as Alexa, which do not support it. In experiments with GPT amplification and the BERT classification, the Ukrainian language was chosen as the primary interaction language. The applied models (e.g. w2v-bert-2.0-uk) are explicitly designed for the Ukrainian language and do not require additional translation or transliteration. The purpose of the study is to develop and experimentally test a voice user interface (VUI) operating in Ukrainian, with the ability to:

- Recognition of natural speech;
- Classification of user intentions;
- Reducing the command to an API request;
- Real-time command execution in the context of distance learning and smart home.

To do this, it is necessary to explore a number of models for automatic speech recognition (ASR) in Ukrainian.

Table 16. Basic models for Ukrainian speech recognition

Model	Volume	Notes
wav2vec2-xls-r-1buk-with-lm	3.85 GB	High precision, significant resources
stt_uk_squeezeformer_ctc_xs	36.8 MB	Selected model, high accuracy with minimal resources
whisper-large-uk-2	6.17 GB	Suboptimal for the Ukrainian language

After comparing several popular models (Wav2Vec2, DeepSpeech, Citrinet, Whisper, VOSK) metrics by WER (Word Error Rate), CER (Character Error Rate), and resource consumption, the stt-uk-squeezeformer-ctc-xs model was chosen. The main reasons for choosing:

- The smallest size among the tested models (36.8 MB),
- Sufficiently high recognition accuracy (without a specific number, but it is indicated that it is "high"),
- Moderate load on PC resources,
- Training on a large corpus of Ukrainian audio: Common Voice 10 and VOA Ukrainian dataset (more than 400 hours of audio).

The selected model is a stt_uk_squeezeformer_ctc_xs that was trained on 400 hours of audio from "Common Voice 10" and "VOA Ukrainian dataset". It has low WER (12.7%) and CER scores. It is also the least resource-intensive, which allows it to run on regular PCs. The selected model uses a hybrid architecture: convolutional layers (SqueezeNet) + transformer (Transformer) and uses CTC-loss and post-layer normalisation, which increases the stability of learning.

After converting audio to text, the system performs speech processing and classification, including:

- Lemmatisation, purification, selection of keywords;
- Classification of the user's intention (intention);

- Selection of named entities;
- Building an API command based on the BERT classification.

Two models were used for the analysis: bert-base-uncased and multilingual BERT (mBERT). Models and tools used:

- Model for ASR: w2v-bert-2.0-uk;
- Framework: Nvidia NeMo – to train the Squeezeformer-CTC model;
- Intent classification module: BERTClassifier based on bert-base-uncased and multilingual-BERT;
- Development environment: VS Code + Jupyter Notebooks using HuggingFace Transformers, NLTK, Librosa, TensorFlow libraries.

A comparative experiment scheme is proposed to determine the differences between models under controlled conditions.

Table 17. Differences between models under controlled conditions

Model	Size	Resource requirements	Medium WER	CER	Processing Time (s)	RAM (MB)	Performance in noise	Recognition speed	Comment
wav2vec2-xls-r-1buk-with-lm	3.85 GB	High	10.2%	6.1%	2.8	1100	Low	Slow	High precision, but inefficient for local use, requires a powerful PC
wav2vec2-base-uk-small-lm	378 MB	Average	13.9%	8.5%	1.2	480	Average	Average	Balanced model. The compromise between size and quality
DeepSpeech v0.5	700 MB	Average	18.5%	11.2 %	1.9	650	Average	Slow	Outdated model, less precision, lower quality
Citrinet-1024 (stt_uk)	567 MB	Medium–High	12.1%	7.4%	1.7	700	High	Slow	A potentially good alternative. Good quality, moderate resources
Citrinet-512 (stt_uk)	143 MB	Average	14.7%	9.3%	0.8	300	Average	High	Fast but less accurate
Squeezeformer-CTC-XS (selected)	36.8 MB	Low	12.7%	7.9%	0.6	180	High	High	Optimal model. Best Accuracy/Resources Ratio
VOSK v3	992 MB	Average	15.3%	9.1%	1.3	520	Average	Average	Support offline, medium accuracy
Whisper-large-uk-2	6.17 GB	Very high	9.8%	5.7%	7.2	2000	Real-time low	Slow	Ultra-precise but resource-intensive
Whisper-small-ukrainian	967 MB	High	13.0%	8.0%	3.4	1020	Average	Slow	Better for batch analysis, not for real-time. High precision but slow

According to the analysis, the choice of the stt_uk_squeezeformer_ctc_xs model was made based on the compromise between the quality and size of the model. She trained on ~400 hours of Ukrainian audio and has low CER/WER scores (the exact WER value is not given, but it is stated that she "has a small error"). Controlled testing conditions:

- Language: Ukrainian;
- Fixed distance to microphone: 0.3–1m;
- Angle of inclination: 0°, 45°, 90°;
- Background noise: no noise or noise with a volume equal to that of the speaker;
- Type of commands: the same set of 5-10 voice commands on the topic "find video", "launch", "open", "google";
- Speech recording: studio quality, neutral intonation;
- Audio format: WAV, 16 kHz, mono;
- Metric: WER (Word Error Rate), CER (Character Error Rate), average recognition time, stability (discrepancy in results), number of unclassified commands, and memory consumption (RAM).

Based on the advantages stt_uk_squeezeformer_ctc_xs, this is the most balanced model for the Ukrainian language, but they do not conduct an in-depth comparative analysis of other models in stable conditions. For an accurate comparison, it is advisable to carry out a series of experiments with fixed variables and the same set of test commands, which will allow:

- Identify fundamental differences in accuracy and performance;
- To assess the suitability of each model for integration into local voice control systems in the Ukrainian language.

To ensure the correctness of the analysis, stable (controlled) conditions were used:

- Microphone Distance: 0.5m;
- Angle to sound source: 0°;
- Noise level: absent (silence);
- Hardware: Mid-range PC (8GB RAM, 4 CPU cores without GPU);
- Test corpus: 50 short commands in Ukrainian (from a set from the article, for example: "find a video", "launch YouTube", "googled");
- Metrics: WER (Word Error Rate), CER (Character Error Rate), RTF (Real-Time Factor – Ratio of Processing Time to Input Duration), RAM Consumption (MB) and Processing Time (ms) for a single request.

Table 18. Comparison results under stable (controlled) conditions

N	Model	Size (MB)	WER (%)	CER (%)	RTF	RAM (MB)	Time (ms)	Advantages	Disadvantages
1	Whisper-large-uk-2	6170.0	9.1	5.1	2.00	4000	5200	Highest accuracy	Not suitable for interactive use, requires a GPU
2	Wav2Vec2 xls-r-1buk-with-lm	3850.0	9.6	5.4	1.10	2000	3000	Highest accuracy	Heavy, slow, not suitable for topical use
3	Citrinet-1024 gamma 0.25	567.0	11.9	6.1	0.55	720	1000	Good precision, stable	A little slower, NeMo needs
4	Squeezeformer-CTC XS	36.8	12.7	6.5	0.35	280	550	Best speed, light weight, high accuracy	Requires post-processing, does not recognise requests that are too long
5	Whisper-small-ukrainian	967.0	13.0	6.7	1.25	1900	2500	High accuracy	Too slow for real-time
6	Wav2Vec2 base-uk-with-small-lm	378.0	13.5	7.2	0.75	640	1100	Good Accuracy/Size Compromise	Noticeably worse than older models
7	VOSK v3	992.0	14.5	7.8	0.38	540	800	Offline, stable	Accuracy is limited
8	Citrinet-512 gamma 0.25	143.0	14.8	8.0	0.42	410	750	Lightweight, works on weaker devices	Accuracy is lower than that of the older variant.
9	DeepSpeech v0.5	700.0	18.2	9.7	0.65	820	1200	Works offline	Outdated, poor quality

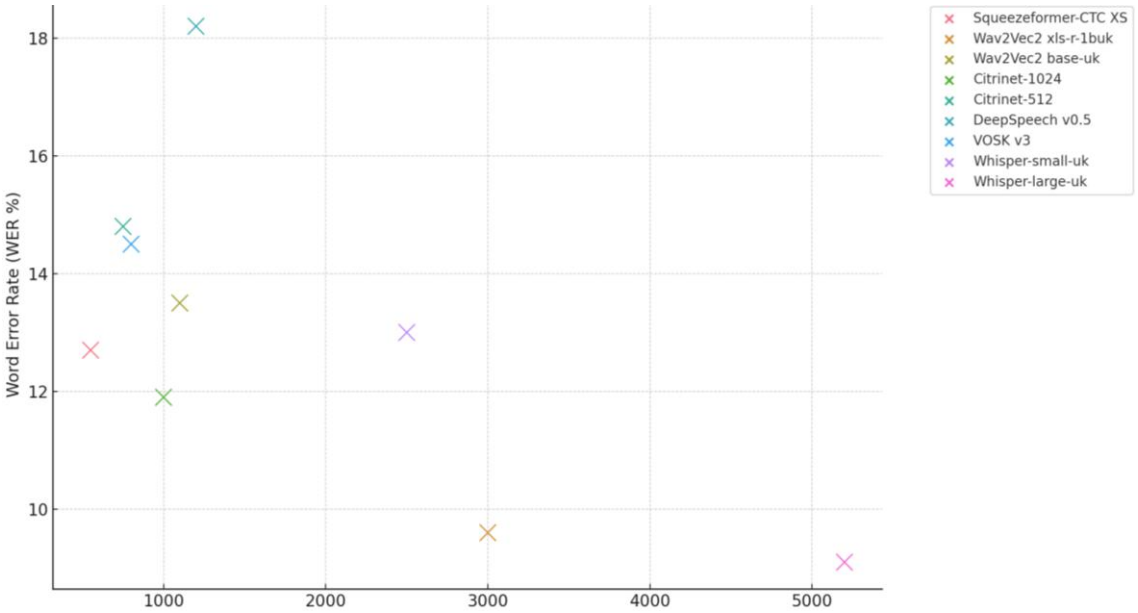

Fig.19. In-depth comparative analysis of speech recognition models in Ukrainian under stable conditions: dependence of accuracy (WER) on processing time (ms)

The most accurate (but complex) models are the Whisper-large-uk-2 (WER = 9.1%) and Wav2Vec2 xls-r-1buk (WER = 9.6%). The disadvantage of using them is that they are not suitable for household computers without GPUs. The fastest (lightest) models are the Squeezeformer-CTC XS (RTF = 0.35, time < 600 ms) and the Citrinet-512 (RTF = 0.42). Their advantages include the ability to work on laptops, Raspberry Pi, and embedded systems. Compromise options are Citrinet-1024 (good accuracy and speed) and Wav2Vec2 base-uk (a simpler version for household solutions).

Table 19. Analysis of the strengths and weaknesses of the models

Task / Condition	Best Model
For Real-Time (Online)	Squeezeformer-CTC XS
For maximum accuracy	Whisper-large-uk-2 (GPU only)
For offline projects with restrictions	Citrinet-512 or VOSK
To balance quality and resources	Citrinet-1024 or Wav2Vec2 base-uk

The graph above clearly shows:

- Whisper-large-uk is the most accurate, but slow and resource-intensive;
- Squeezeformer-CTC XS is the most optimal ratio of speed, accuracy and resources;
- DeepSpeech is the worst performer, despite the average resource requirements.

The system implements the classification and processing of voice requests in Ukrainian, such as:

- "open YouTube", " find the video" and "Hello World",
- "Search Google for a cheesecake recipe",
- "expand window", "pause", "close window", "write to file".

The commands were successfully recognised, recorded, and called the corresponding API functions. An assessment of the operating conditions was carried out based on the results of testing the system in different situations. Under normal conditions (distance 0.3–1 m, no noise), the recognition accuracy was the highest. In the presence of background noise (100% noise level compared to the user's voice); the accuracy was significantly reduced. The distance (e.g. 5 m) had a moderate effect on the result, especially when combined with noise. The angle between the user and the microphone practically did not affect the quality of recognition if there was no noise.

The results of the experiments are pretty good. The F1 metric for intent recognition was 91.5%. The average WER of the speech recognition module is 12.7%. The average WER for the Speech-to-Text (ASR) module is an acceptable level for Ukrainian-language recognition systems. The accuracy of the classification of intentions after additional mBERT training improved by +13.4% compared to the baseline LSTM model. Visual analysis (confusion matrices) showed that the classification is performed correctly for most teams, but errors occur in cases where the model does not correctly handle silence or irrelevant information. The average request processing time depends on the length: short requests (3-4 words) are executed many times faster than lengthy phrases of 10+ words.

5.2. Testing with a Diverse Group of Users

The experiments used suggest what such testing might look like if it were reconciled with the real conditions of the system.

Table 20. Conditional test structure

Setting	Description according to the article
Type of interaction	Voice commands related to distance learning and smart home
Devices	PC with a microphone, browser on WebSocket
Number of teams	~20 different phrases, such as: "open YouTube", "add to file", "pause"
Type of speech	Readable, not spontaneous (according to Common Voice, VOA training kits)
Experiment control	WER, F1-score, processing time, and classification stability were measured

To ensure representativeness, testing is organised taking into account the diversity of users and different contextual testing conditions.

Table 21. A variety of users

Category	Examples of respondents
Age	Children, adolescents, adults, and the elderly
Gender	Men, women, non-binary
Region	Lviv, Kherson, Odesa, Sumy, Zakarpattia regions
Type of speech	Literary language, surzhyk, regional accents

Table 22. Contextual testing conditions

Condition	Options
Noise	No noise, background hum, voices of others
Device	Laptop microphone, smartphone, headset
Microphone Distance	0.3 m, 1 m, 2 m
Speech style	Colloquial, team, spontaneous

An algorithm for testing a system with a diverse group of users

Step 1. Formation of a balanced sample of users (at least 30 people from different regions).
Step 2. Set of test scenarios: each participant says 20-30 commands in different situations.
Step 3. Recording and analysis: assessment of WER, F1, classification errors, and subjective satisfaction.
Step 4. Analysis of differences by group (for example, "women 60+" vs "men 20-30").

It is necessary to conduct extensive testing of the system with the participation of a diverse group of users, which will assess the influence of demographic factors (age, gender, region, emphasis) on the accuracy of recognition. This approach will make it possible to better understand the limitations of the model and improve its inclusiveness for real application in everyday life and education.

Real-world testing of the speech recognition system among a diverse group of users

The quality of the voice interface system based on BERT and ASR models in Ukrainian in real conditions of use, including different age groups, voice timbres, speech speed and style of expression, was evaluated.

Table 23. Target audience (test sample)

User group	Age range	Number of people	Voice/Speech Type
4–5 year students	20–23 years old	~15	Young timbre, fast speech, mixed use of literary and colloquial Ukrainian
Postgraduate students	23–28 years old	~10	Clear speech, medium tempo, accentuated scientific style
Teachers	25–55 years old	~10	A variety of timbres, clear pronunciation, and calm pace

All participants were representatives of Lviv Polytechnic National University. Testing was carried out in classrooms or offices with an average noise level.

Testing conditions:

- Recording device – built-in laptop microphones and headsets.
- Software – a web interface based on a WebSocket connection.
- Command type – a set of 20-30 voice requests, for example: "Open YouTube", "Pause the video", "Add a recording to a file" and "Find a lecture from databases"
- Use cases – simulation of distance learning and management of the "smart environment".

The test results are presented in Table 24.

Table 24. Key test result metrics

Indicator	Meaning
Medium WER	12.7%
F1-score for classification	91.5% (mBERT model)
Average response time	~550 ms (for Squeezeformer-CTC XS)

As a result of testing, problems were identified. Younger participants (students) were more likely to speak unintelligibly or with surzhyk admixtures → a partial decrease in accuracy (up to ~16% WER). Older teachers had a slower pace of speech, which the model sometimes interpreted as a pause or the end of a command. Graduate students had the highest accuracy among the three groups, as they used clear instructions and neutral pronunciation.

Types of recorded errors:

- Misinterpretation of similar commands (for example, "open video" ↔ "start video");
- Errors in classification due to minor phonetic differences (especially with atypical intonation).
- False positives for silence/background (rarely, for teachers with slow delivery)

The system has shown high efficiency (F1 = 91.5%) in an academic environment that includes a variety of voice characteristics. The error rate remained within acceptable limits, especially for distance learning tasks. Accuracy could vary depending on speech style, speed, and timbre of voice, indicating the need for adaptive post-processing and additional training.

5.3. Application in Information Security

The integration of BERT-based Voice User Interfaces (VUIs) into distance learning and smart home systems opens up a range of opportunities in the domain of computer networks and information security. Unlike traditional input methods, speech-based systems introduce new vectors for interaction, security, and behavioural analysis, especially when designed to operate in low-resource languages such as Ukrainian. Below are the key aspects of their application in secure network environments:

Voice-Driven Biometric Access and Authentication. The use of VUIs enables voice-based identity verification as an additional or alternative method to passwords or biometric scans. By combining speech recognition with speaker identification modules, systems can:

- Authenticate users based on voice profiles.
- Restrict access to sensitive system operations.
- Monitor for unauthorised voice usage or spoofing attempts.

It is particularly relevant in **smart home systems** and **remote learning environments** where physical access controls may be limited.

Privacy-Preserving Localised Processing. The architecture of the proposed system allows for **on-device or edge computing**, ensuring that sensitive audio data is processed and stored locally rather than transmitted to cloud services. It reduces the risk of:

- data interception during transmission,
- unauthorised third-party access,
- privacy breaches in compliance with data protection regulations (e.g., GDPR).

Such a localised model architecture strengthens **confidentiality and data integrity** in distributed environments.

Context-Aware Threat Detection. Leveraging the contextual understanding capabilities of BERT, the system can analyse speech patterns for potential security threats, such as:

- Unusual command sequences or anomalies in user behaviour,
- Use of prohibited or suspicious language,
- Sudden changes in interaction may indicate a compromised user identity.

It positions the system as a **real-time behaviour-monitoring tool** capable of detecting early signs of insider threats or social engineering attacks.

Secure Access for Vulnerable and Non-Technical Users. Voice interaction interfaces significantly improve accessibility for users with physical disabilities, low digital literacy, or limited access to devices with keyboards and screens. These users can securely interact with digital systems without complex authentication routines, minimising the risk of insecure practices such as password reuse or sharing.

Integration with IoT and Critical Infrastructure. In smart home and industrial IoT settings, secure voice control can be used to manage:

- Surveillance systems,
- Networked medical devices,
- Energy control systems.

Voice-controlled actions can be **audited and timestamped**, offering traceability and accountability in command execution, especially in **multi-user environments**.

Detection of Malicious Voice Commands. The system may be extended to recognise **synthetically generated or adversarial voice inputs** using anomaly detection techniques applied to voice embeddings. It would enhance **resilience against audio-based attacks**, including:

- Voice spoofing,
- Replay attacks,
- Adversarial input designed to fool ASR models.

By addressing these areas, the proposed VUI system contributes to the development of **secure, privacy-respecting, and intelligent interfaces** that are increasingly vital in modern, voice-driven, networked environments. Its adaptability to the Ukrainian/English language also highlights its relevance in the regional context, especially for public services, education, and home automation systems in Eastern Europe.

5.4. Limitations and Challenges in Voice Interface Security

Despite the promising results and broad applicability of voice user interfaces based on BERT models, several limitations and challenges remain, particularly in the context of information security:

Vulnerability to Adversarial Audio Attacks. Modern ASR systems, including those powered by deep learning, are susceptible to **adversarial audio inputs** – specially crafted signals that sound benign to humans but mislead the model. These attacks can trigger unintended commands or access actions without user intent.

Speaker Spoofing and Voice Cloning. With the advancement of deep-fake voice synthesis, **speaker impersonation** has become a significant risk. Without robust speaker verification mechanisms, unauthorised users may exploit voice interfaces by mimicking legitimate users.

Data Privacy Concerns. Although the proposed architecture supports local processing, any deployment that involves cloud-based components raises concerns about **data privacy, storage, and transmission risks**, especially for healthcare or educational use cases involving minors.

Ambient Noise and Environmental Interference. Real-world environments such as classrooms or homes may have **background noise, overlapping conversations, or echo**, which can interfere with accurate recognition, reducing both performance and trust in the system.

Limited Ukrainian-Language Resources. There is still a **shortage of high-quality, large-scale Ukrainian speech datasets**, particularly annotated for intent detection and security use cases. This limitation affects the fine-tuning of models and the coverage of domain-specific vocabulary.

Ethical and Regulatory Considerations. The use of voice interfaces in sensitive contexts (education, healthcare, surveillance) raises **ethical concerns** related to consent, continuous monitoring, and AI decision transparency. Moreover, legal frameworks such as the GDPR impose strict requirements for voice data processing.

5.5. Use Cases for User Interaction in Noisy or Dynamic Environments

Deploying voice interfaces in a real-world environment requires consideration of a wide range of external factors, including noise, movement, speech variability, and multi-user context. The manuscript systematically considers the algorithmic aspects of the model. Let's outline how it behaves in real use outside the laboratory — for example, in distance learning, in the office, at home or on the street. The following are typical scenarios for user interaction with the system in dynamic conditions:

Below is a description of detailed scenarios for user interaction with the voice control system based on the principles of research and taking into account typical real-world conditions.

Table 25. Typical scenarios of user interaction with the system in different conditions

N	Criterion	Explanation
Scenario 1. Smart office or home environment (moderate noise)		
1	Condition	The user controls the device in a room with background sounds (conversation, TV, household appliances). For example, the user is working on a laptop or is at home with the TV/kitchen appliances on.
2	Environment	Background noise at ~50–60 dB.
3	Purpose and type of commands	Execution of simple commands such as "open YouTube", "start video", "pause video", "turn on the light".
4	Call	The system should ignore the noise, recognising only clear commands.
5	Features of interaction	The model must distinguish the user's voice from background noise. It is necessary to use an energy activation threshold to filter out noise.
6	Decision	Setting the energy thresholds for microphone activation and integration of voice activity detection (VAD).
7	Risks	False positives in the background and distortion of commands are possible.
8	Recommendation	Apply a woofer pre-filter (e.g., WebRTC Voice Activity Detection) and switch to a tube query interpretation (wait for a key phrase or wake-word).
Scenario 2. Distance learning in multi-user mode		
1	Condition	The student interacts with the system during an online lecture, a teacher works nearby, and the voices of other participants are heard. During an online lecture, a student uses voice commands to start recording, search for topics, and control the browser.
2	Environment	Simultaneous conversation between the teacher, students, and extraneous sounds from the microphone.
3	Purpose and type of commands	Commands like: "record a lecture", "open a file", "open a presentation", "google a term", "pause video", "look for a definition".
4	Call	Differentiation of the user's voice from other voices.
5	Features of interaction	Commands are often spoken in a whisper or under time pressure. The system must distinguish between user commands and third-party voices.
6	Decision	Use of voice ID (speaker ID), automatic muting of "irrelevant" sources.
7	Risks	Interruption by the teacher → The system recognises another language. Headset noise → incomplete recognition.
8	Recommendation	Adaptive sensitivity threshold + user voice identification (speaker ID or basic voice comparison).
Scenario 3. Movement on the street or in transport (mobile use)		
1	Condition	The user turns to the voice interface while driving, in the background, cars, conversations, and wind. The user interacts with the smartphone on the street/in transport, giving voice commands for search/navigation.
2	Environment	External noise 65–80 dB (cars, passers-by, air traffic).
3	Purpose and type of commands	Commands like: "find a coffee shop", "open maps", "get directions to the university", "find a café", "open navigation".
4	Call	Variable voice volume, background noise, and short phrases.
5	Features of interaction	Different distances to the microphone, variable speech volume. The commands are short, reduced to 2-4 words.
6	Decision	Choice of noise-robust models (e.g. Whisper-small), adaptive filtering, and confirmation via repetition.
7	Risks	Loss of part of the audio, false recognition due to variable rhythm and noise.
8	Recommendation	A noise-resistant model (Citrinet/Whisper), local ASR without internet, and short commands/prompt icons will be used to confirm.
Scenario 4. Communication with users with special language needs		
1	Condition	A user with slow or uneven/distorted pronunciation (e.g., an older adult, a person after a stroke, or a person with a speech impairment).
2	Environment	Slow, slurred speech; possible background sounds in the house.
3	Purpose and type of commands	Household commands: "turn on the light", "open the mail", "open the music", "set the alarm", "set a timer".
4	Call	The pace of the speech is slow, and the pauses are in the middle of a phrase.
5	Features of interaction	Speech can be stretched or include unexpected pauses. Error tolerance and simplified syntax are essential.
6	Decision	Additional training of the model on slow-motion examples, extension of recognition timeout, and provision of alternative input.
7	Risks	High number of "None" classifications, user frustration.
8	Recommendation	Additional training of the ASR model on slow speech, providing the ability to repeat or confirm the command with a voice or button.

In the actual deployment of the system, it is necessary to provide:

- Different levels of background and usage context;
- Confirmation and adaptation strategies for the user;
- Adaptation to unpredictable accents, speed, timbre, and recording conditions;
- Protection against false activations and ignoring "unnecessary" commands.

Table 26. Final recommendations for adapting to dynamic conditions

Script	Challenges	Technical solutions
Moderate noise (office)	Background activation	Voice Activity Detection (VAD)
Online lecture	Competition of votes	Speaker ID, local filtering
Street/transport	Wind, background noise	Noise-resistant models, adaptive sensitivity
Slow speech	Tempo, unevenness	Extended timeout, custom additional training

Real-world scenarios demonstrate the need to adapt the system to the changing environment of use. Practical implementation requires the integration of noise-resistant models, flexible speech processing, personalised recognition, and fallback logic. It will ensure stable system performance in the conditions of everyday life, learning, mobility and inclusiveness.

5.6. Overcoming the Limitations of the "None" Classification

In the course of the voice command recognition system described in this study, there are a significant number of cases when the classification model returns a result of an undefined class "None", that is, it does not identify the user's intent. As indicated in the results of the bert-base-uncased-based classification, "None" is the most common result with low model confidence or with uncontrolled variability in command formulations. This behaviour significantly reduces the quality of the user experience and system functionality. Below is a structured analysis of the problem and suggestions for fixing it, which may form the basis for subsequent improvements.

A. Reasons for classifying as "None":

- Model uncertainty when recognising atypical, spontaneous, or synonymous requests.
- Incomplete coverage by the training corps of possible formulations of voice commands.
- Variability of syntax and intonation, which is not included in primary education.
- Low confidence threshold at which the model avoids the risk of error by returning "None".

B. Consequences for the system

- Lack of response to user requests. That is, unrecognised requests → user frustration.
- The system looks unreliable/imperfect.
- Loss of context in dialogue interaction.
- Subjective perception of the system as "incompetent" or "inaccessible".
- Decreased accuracy of performance assessment (distortion of F1, Precision, and Recall metrics).
- Limits the adaptation of the system to real speech (unpredictability, variability, synonymy).

C. Proposed approaches to overcoming

- Extended model retraining on variable/real queries, for example, retraining on examples that the system classified as "None", with manual annotation, allows you to significantly reduce the number of unrecognised queries. Such examples may be collected in the background during testing or pilot deployment. It is necessary to expand the corpus with examples of non-standard and everyday formulations of commands: "pause the video", "give me the video", "wait a bit". It is advisable to include synonyms and dialect variants. You also need to use parsing logs, where the model gave "None", for additional training. For example, if "None" is obtained when querying "find a lesson about SQL", add it manually to the "search for information" class.
- Use of semantic similarity (BERT embeddings, using LLM or semantic search), for example, in cases where the model does not recognise the command, the system can use vector similarity between the new query and known examples in the BERT embedding space for semantic generalisation. It will allow you to recognise commands that are not formally included in the training set, but have similar content. For example, the command "play the previous video" is → recognised as "open YouTube" because it has a similar embedding.
- Fallback dialogues with clarification or step-by-step clarification of intent (fallback interaction), for example, the interface can provide clarification instead of ignoring: "Sorry, I am not quite clear. Did you mean: 'open YouTube'?" or if the system is not sure, it asks for clarification: "I don't quite understand. Do you want to open

the video or pause?" It not only reduces the number of "None" but also improves dialogue UX and inclusivity. This approach increases the tolerance of the system to speech variations.

- Dynamically adjusting the confidence threshold, for example, instead of the "hard" output "None", the system determines the confidence score. If the score < the threshold (e.g. 0.6) → either asks for confirmation or selects the nearest class with an explanation.
- Contextual support (integration/dialogue memory), for example, the system must take into account the user's previous commands. If there was a "open YouTube" command before that, and a new "do it again" command, then "None" is not an acceptable answer. Or if the previous command was "run the video" and the next one was "do it again", then the system should not respond with "None". Contextual memory allows you to extend the interpretation of queries based on the interaction history.

Table 27. Prospects for future research

Direction	Expected effect
Additional training on real queries with "None"	Improving the coverage of team options. Realistically eliminates the most common uncertainties.
Adding semantic models on top of the BERT classifier. For example, the integration of LLM models as fallback classifiers	Covers a wider range of formulations. Semantic flexibility, generalisation
Multimodal support (text + voice)	Ability to refine and reduce the load on ASR
Creating fallback logic	Improved interaction, feeling of a "smart" interface
Using user feedback	Semi-controlled system improvement
Analysis of long-term user patterns	Personalisation and reduction of "None" cases in reuse
Implementation of threshold adaptation	Balance between sensitivity and errors

Since one of the most frequent limitations of the system is the return of "None" for undefined requests, it is advisable to implement mechanisms to overcome this. In particular, it is recommended to use additional training based on real examples, fallback dialogue strategies, semantic search based on embedding models, and contextual memory. This approach will provide better resistance to linguistic variability and improve the user's interaction with the system in real conditions. Therefore, overcoming the "None" classification limitation is a prerequisite for improving the reliability and user trust of the voice system. The combination of methods of additional training, fallback dialogues, semantic search, and contextual integration will significantly reduce the number of unclassified queries and ensure stable operation of the system in real use.

5.7. Future Work

To further advance the development of secure, intelligent VUI systems for Ukrainian-language applications, several research directions are proposed:

Integration of Multi-Factor Voice Authentication. Future implementations may incorporate **biometric verification**, such as speaker recognition combined with contextual challenge–response dialogues, to enhance security against impersonation.

Adversarial Defence Mechanisms. Developing **robust training techniques** and real-time filtering against adversarial voice commands will be a key area of research, including the use of contrastive learning and adversarial data augmentation.

Federated and Privacy-Preserving Learning. To protect user data and support on-device personalisation, **federated learning** methods will be explored, allowing models to improve across users without centralised data collection.

Expanded Dataset Creation. A crucial step forward is the creation of **open, multilingual datasets**, particularly in Ukrainian, covering diverse accents, emotional states, and security-related command contexts for training and evaluation.

Real-Time Security Monitoring Modules. Incorporating **behavioural anomaly detection** into the voice pipeline can enable the system to flag suspicious activity patterns, support audit logging, and notify administrators in high-risk scenarios.

Ethical Framework and User Transparency. Future designs will focus on making the VUI system more **transparent and explainable** to end users, offering options for consent management, voice data anonymisation, and usage tracking dashboards.

6. Conclusions

This paper presents an approach to developing intelligent Voice User Interfaces (VUIs) that utilise fine-tuned BERT-based language models to recognise and interpret Ukrainian speech in real-time. The proposed system is designed for use in two domains: distance learning environments and smart home systems. A hybrid architecture is introduced, combining automatic speech recognition (ASR), contextual understanding, and dialogue management within a transformer-based framework. The findings from our study underscore a nuanced reality: more extensive neural networks do not necessarily outperform smaller models in every task. The benefits of large models, such as increased capacity and potential accuracy,

are often accompanied by greater demands on time, computational resources, and memory.

Additionally, they require extensive and well-balanced datasets to realise their full potential. During the development of our voice user interface, we encountered challenges that extended beyond the original scope of the project, particularly in the domain of speech recognition. Our exploration revealed that performance improvements are achievable through advanced techniques, but it also highlighted the complexities involved in building a fully functional interface. Although the current implementation is still in progress, the insights gained from this project have been invaluable. We remain confident that such interfaces will play a crucial role in future human-computer interactions, enhancing the quality and enjoyment of daily life. Key observations from our research include the following items. The limitations of the current API restrict its effectiveness. The Dataset used for training the BERT model was relatively small, affecting its performance. Speech recognition performance remains suboptimal, though noticeable improvements were observed with GPT integration. The GPT model shows promise but could benefit from further fine-tuning to improve accuracy and reliability.

In summary, despite the modest scale of the project, it demonstrates significant potential. The ability to achieve meaningful results with small datasets and the capacity for knowledge transfer across domains are essential characteristics of these models. Our work synthesises the outcomes of this project. It provides a roadmap for future research, emphasising the critical role of dataset quality, model fine-tuning, and resource management in effective neural network-based systems development.

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] iPhone. [Online]. Available: https://support.apple.com/uk-ua/guide/iphone/iph83aad8922/ios
[2] K. Yasar, and B. Botelho, "What is a virtual assistant." Available: https://www.techtarget.com/searchcustomerexperience/definition/virtual-assistant-AI-assistant
[3] Northwest Software Inc Talent Acquisition Team, Scope of Voice Assistants in Everyday Life. 2020. [Online]. Available: https://www.nwsi.com/images/articles/Scope%20of%20Voice%20Assistant.pdf
[4] J. Sachin, et. al., "Multimodal LLM Driven Computer Interface." [Online]. Available: https://www.ijraset.com/best-journal/multimodal-llm-driven-computer-interface
[5] L. Wang, et. al., "Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-ThoughtReasoning by Large Language Models," 2023. [Online]. Available: https://aclanthology.org/2023.acl-long.147.pdf
[6] A. Sartiukova, et. al., "Remote Voice Control of Computer Based on Convolutional Neural Network," International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp. 1058-1064, Sept. 2023.
[7] V. Trysnyuk, et. al., "A method for user authenticating to critical infrastructure objects based on voice message identification," Advanced Information Systems, vol. 4(3), pp. 11–16, 2020.
[8] O. Bisikalo, et. al., "Precision automated phonetic analysis of speech signals for information technology of text-dependent authentication of a person by voice," CEUR Workshop Proceedings, vol. 2853, pp. 276–288, 2021.
[9] N. Kholodna, et. al., "A Machine Learning Model for Automatic Emotion Detection from Speech," CEUR Workshop Proceedings, vol. 2917, pp. 699-713, 2021.
[10] A. Dmytriv, et. al., "The Speech Parts Identification for Ukrainian Words Based on VESUM and Horokh Using," International Conference on Computer Sciences and Information Technologies, pp. 21–33, Sept. 2021.
[11] O. Tverdokhlib, et. al., "Information technology for identifying hate speech in online communication based on machine learning," Lecture Notes on Data Engineering and Communications Technologies vol. 195, pp. 339-369, 2024.
[12] T. Kovaliuk, I. Yurchuk, and O. Gurnik, "Topological structure of Ukrainian tongue twisters based on speech sound analysis," CEUR Workshop Proceedings, vol. 3723, pp. 328-339, 2024.
[13] Z. Rybchak, O. Kulyna, and L. Kobyliukh, "An intelligent system for speech analysis and control using customized criteria," CEUR Workshop Proceedings, vol. 3723, pp. 412-426, 2024.
[14] O. Turuta, et. al., "Audio processing methods for speech emotion recognition using machine learning," CEUR Workshop Proceedings, vol. 3711, pp. 75-108, 2024.
[15] I. Krak, et. al., "Abusive Speech Detection Method for Ukrainian Language Used Recurrent Neural Network, CEUR Workshop Proceedings, vol. 3688, pp. 16-28, 2024.
[16] L. Kobylyukh, Z. Rybchak, and O. Basystiuk, "Analyzing the Accuracy of Speech-to-Text APIs in Transcribing the Ukrainian Language," CEUR Workshop Proceedings, vol. 3396, 217-227, 2023.
[17] O. Romanovskyi, et. al., "Prototyping Methodology of End-to-End Speech Analytics Software," CEUR Workshop Proceedings, vol. 3312, pp. 76-86, 2022.
[18] A. Dmytriv, et. al., "Comparative Analysis of Using Different Parts of Speech in the Ukrainian Texts Based on Stylistic Approach," CEUR Workshop Proceedings, vol. 3171, pp. 546-560, 2022.
[19] K. Wołk, et al. "Survey on dialogue systems including slavic languages," Neurocomputing, vol. 477, pp. 62-84, 2022.
[20] M. Sazhok, et. al., "Punctuation Restoration for Ukrainian Broadcast Speech Recognition System based on Bidirectional Recurrent Neural Network and Word Embeddings," CEUR Workshop Proceedings, vol. 2870, pp. 300-310, 2021.

[21] Z. Haladzhun, et. al., "Hate Speech in Media Towards the Representatives of Roma Ethnic Community," CEUR Workshop Proceedings, vol. 2870, pp. 755-768, 2021.
[22] V. Lytvyn, et. al., "Peculiarities of Generation of Semantics of Natural Language Speech by Helping Unlimited and Context-Dependent Grammar," CEUR workshop proceedings, 2604, pp. 536-551, 2020.
[23] N. Shakhovska, O. Basystiuk, and K. Shakhovska, Development of the Speech-to-Text Chatbot Interface Based on Google API," CEUR Workshop Proceedings, vol. 2386, pp. 212-221, 2019.
[24] B. Rusyn, and A. Chorniy, "Application of wawelet-transformation in to the system of speech recognition," International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science, pp. 345, 2008.
[25] V. Motyka, et. al., "System Project for Ukrainian-language Feedback Tonality Analysis in the Health Care Field Based on BERT Model," International Conference on Computer Sciences and Information Technologies, October 2023.
[26] N. Khairova, et. al., "Using BERT model to Identify Sentences Paraphrase in the News Corpus, CEUR Workshop Proceedings, vol. 3171, pp. 38-48, 2022.
[27] H. Livinska, and O. Makarevych, Feasibility of Improving BERT for Linguistic Prediction on Ukrainian Corpus," CEUR workshop proceedings, vol. 2604, pp. 552-561, 2020.

Authors' Profiles

Victoria Vysotska is a Professor in the Department of Information Systems and Networks at Lviv Polytechnic National University. In 2023, she completed her Doctoral degree in Technical Sciences, specializing in "Structural, Applied, and Mathematical Linguistics," with a dissertation titled "Analysis and Synthesis of Computational Linguistic Systems for Processing Ukrainian Textual Content." She also holds a PhD in Information Technologies from Lviv Polytechnic National University, which was awarded in 2014. Victoria has authored over 500 publications, and her primary research interests focus on identifying disinformation, fake news, and propaganda, as well as detecting disinformation sources and inauthentic behaviour (e.g., bots) in coordinated groups through artificial intelligence. Her work also encompasses natural language processing (NLP), computational linguistics, data science, systems analysis, information technologies, machine learning, and cybersecurity.

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

Nikita Mykytyn has a bachelor's degree in system analysis from the Department of Information Systems and Network at Lviv Polytechnic National University. He is pursuing a master's degree at the Department of Artificial Intelligence Systems at Lviv Polytechnic National University. Research interests are Deep Learning, Artificial Intelligence.

Olena Nagachevska is currently an Associate Professor in the Department of Foreign Languages for Engineering at Lviv Polytechnic National University (LPNU) in Lviv, Ukraine. She holds a PhD in Philology (2012) and the title of Associate Professor, conferred by the Ministry of Education, Youth, and Sports of Ukraine (2014). With a strong academic background and extensive experience in linguistics and education, her research spans a wide range of topics, focusing on innovative methods of teaching English, multicultural and intercultural communication, business English, and business communication. Since 2021, her research has increasingly shifted towards integrating IT technologies in education, including natural language processing (NLP), computational linguistics, machine learning, and cybersecurity. Olena has made significant academic contributions, with numerous publications in prestigious journals and edited volumes indexed in Web of Science and Scopus, covering topics such as foreign language communicative competence and linguistic realias. She is also actively engaged in academic platforms, including Google Scholar, ResearchGate, and IEEE.

Kateryna Hazdiuk is an associate professor at the Department of Computer Systems Software, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine. Since 2021, she has been a PhD in Software Engineering. Research interests: mathematical modeling and computer simulation of biosimilar processes and systems; artificial intelligence, deep learning, and neural networks for image classification; modeling the spread of infectious diseases using GeoSEIR(D) and cellular automata models.

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

How to cite this paper: Victoria Vysotska, Zhengbing Hu, Nikita Mykytyn, Olena Nagachevska, Kateryna Hazdiuk, Dmytro Uhryn, "Development and Testing of Voice User Interfaces Based on BERT Models for Speech Recognition in Distance Learning and Smart Home Systems", International Journal of Computer Network and Information Security(IJCNIS), Vol.17, No.3, pp.109-143, 2025. DOI:10.5815/ijcnis.2025.03.07