https://doi.org/10.31891/csit-2023-2-7
UDC 004.89

Volodymyr SHYMANSKYI, Nikoletta KAZONNIKOVA
Lviv Polytechnic National University

VOICE ASSISTANT FOR FINDING ITEMS IN CLOTHING STORE

The paper presents the use of machine learning in audio file processing, the use of neural networks to recognize voice commands, and the development of a voice assistant for finding products in a clothing store.

The purpose of the work is to develop a voice assistant for finding items in a clothing store.

As a result, a dataset containing 30 categories and 3095 audio recordings was created, a neural network model was trained using the collected data, and an accuracy of 96.02% was achieved; WER: 0.0398; CER: 0.0087. The model was integrated with the search system into the voice assistant API, which allows you to record from a microphone, convert audio to text, break the text into keywords, and search the database using the obtained keywords. The speech recognition system has shown stable and high accuracy when recording from a microphone. This provides users with reliable and accurate recognition results even when using simple microphones. Search allows you to find results by keywords and names of items, and there is a function to recommend similar items if nothing was found at the user's request. The system's flexibility allows it to understand language contexts and does not depend on the order of words in a phrase, as the search is performed by keywords separately. The research analyzes the literature and compares existing approaches to voice command recognition. The methods of audio signal processing are described. The problem of searching for items in a clothing store was analyzed, and the current state of the market and demand for the technology were investigated. The practical value is that the developed voice assistant is specialized and optimized for searching for goods in a clothing store, allows solving the task of finding goods according to specified criteria, and simplifies the search task for different groups of people.

Keywords: machine learning, neural networks, voice recognition, Ukrainian language recognition, voice assistant, product search.

Володимир ШИМАНСЬКИЙ, Ніколетта КАЗЬОННІКОВА
Національний університет «Львівська Політехніка»

ГОЛОСОВИЙ ПОМІЧНИК ДЛЯ ПОШУКУ ТОВАРІВ У МАГАЗИНІ ОДЯГУ

У даній статті досліджено застосування машинного навчання в обробці аудіо файлів, використання нейронних мереж для розпізнавання голосових команд, створенню голосового помічника для пошуку товарів в магазині одягу, створенню набору даних для тренуваннння нейронної мережі..

Метою роботи є розробка голосового помічника для пошуку товарів у магазині одягу.

У результаті було створено набір даних, що містить 30 категорій та 3095 аудіозаписів, натреновано модель нейронної мережі з використанням зібраних даних та досягнуто точності 96.02%; WER: 0.0398; CER: 0.0087. Було інтегровано модель разом із системою пошуку в API голосового помічника, який дозволяє зробити запис з мікрофону, конвертувати аудіо в текст, розбити текст на ключові слова та провести пошук по базі даних з використанням отриманих ключових слів. Система розпізнавання мови показала стабільну та високу точність під час запису з мікрофону. Це забезпечує користувачам надійність та точність результатів розпізнавання навіть при використанні простих мікрофонів. Пошук дозволяє знаходити результати за ключовими словами та назвою речей, є функціонал рекомендації схожих речей, у випадку, якщо нічого не було знайдено за запитом користувача. Гнучкість системи дозволяє розуміти мовні контексти та не залежить від порядку слів у фразі, оскільки пошук відбувається по ключовим словам окремо. В роботі було проаналізовано літературу та проведено порівняння існуючих підходів розпізнавання голосових команд. Описано методи обробки аудіо сигналів. Було розглянуто проблему пошуку товарів у магазині одягу та досліджено сучасний стан ринку та попит на технологію.

Практична цінність полягає у тому, що розроблений голосовий помічник є спеціалізований та оптимізований для пошуку товарів у магазині одягу, дозволяє вирішувати задачі пошуку товару за заданими критеріями, спрощує завдання пошуку для різних груп людей.

Ключові слова: машинне навчання, нейронні мережі, розпізнавання голосу, розпізнавання української мови, голосовий помічник, пошук товарів.

Introduction

A voice assistant can be useful for many people with different levels of fashion expertise and different physical abilities. For example, people with disabilities may have difficulty finding the desired items, and for people with dyslexia, a voice assistant will make the search process more comfortable and efficient. Even for average people, searching with a voice assistant can save a lot of time, as there is no need to go around the entire store. In addition, such a voice assistant can partially replace a real consultant, selecting the necessary things for the buyer. Thus, a voice assistant can be useful for different categories of people with different needs. It will help make the process of searching for goods more efficient and comfortable.

The aim of the work is to develop a voice assistant for finding items in a clothing store.

Research objectives:

- research of the subject area;
- analysis of existing approaches and algorithms;
- architecture design and algorithm development;
- development of a system for using the algorithm;
- testing the developed voice assistant and demonstrating the results of its work;

The object of research is the process of searching for goods and voice recognition.

The subject of research is machine learning algorithms and neural networks for processing and recognizing voice commands.

Related works

Developing a voice control system requires an in-depth analysis of existing research and methods for processing voice commands. During the research process, a significant amount of relevant literature was found, which makes it possible to conduct a detailed analysis and use best practices in the development of a voice control system.

For example, an article by Edward L., who investigated methods of detecting voice fraud using machine learning [1]. The final neural network model had an error of 10.8. Vincenzo D. used deep learning to recognize covid-19 through breathing and coughing by feeding the network with audio recordings without pre-processing [2]. Muhammad E. used the k-means algorithm to recognize audio signals [3]. Han G. used deep learning to classify music genres [4]. The model is trained on 5 different genres and recognizes them with an accuracy of 90.32%. Richard M. used convolutional neural networks to classify cough sounds using features such as mel spectrograms and mel-frequency coefficients and achieved an accuracy of 82.96% [5]. Satish K. also conducted research in the field of medicine and used deep learning to monitor the pulse status of patients and achieved 85% accuracy [6]. Elham N. used an ensemble method to detect Parkinson's disease based on patient's voice metrics and achieved 90.6% accuracy [7]. George S. studied the recognition of whole sentences using existing trained models for speech recognition [8]. Eleni T. used processed spectrograms as input to a convolutional neural network and achieved an accuracy of 91.25% [9]. Elizabeth V. conducted a comparative study of audio signal classification using a convolutional neural network and a generative adversarial network [10].

It is also worth noting the work of A.G. Krivokhata, O.V. Kudin, and A.O. Lisnyak, which describes the processing of audio signals using ensemble learning and neural network classifiers [11]. Neural network ensembles are actually effective because it is possible to choose a different number of algorithms, undemanding to resources, and it is possible not to adjust to the quality of the audio recording by choosing a specific algorithm, but to choose the results with the highest accuracy each time.

Presenting main material

The task is to use neural networks to create a voice assistant. There are also subtasks that consist of using algorithms for searching and processing voice commands for the neural network. To create voice assistant we need to define general flow of functionality and to determine implementation steps (Fig. 1).

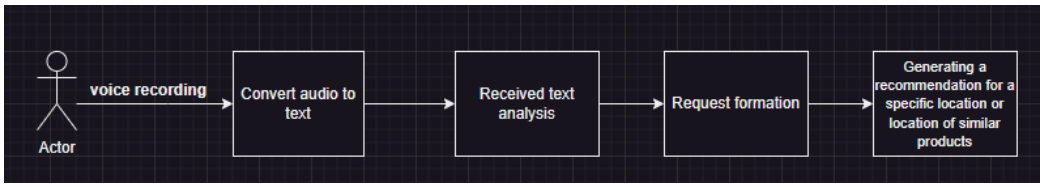


Fig.1 Voice assistant functionality flow

Firstly, we need voice recognition algorithm which can take audio as input and convert it to text. We need to identify a set of words that can be used in voice commands, collect audio data, and train a neural network model on it. Next step is to create a system that can process received text, highlight keywords and form a search query. After that we can use the query to look for certain items in database with search algorithm. Next step is the recommendation system to provide suggestions to the user if nothing was found at their request.

The development of a neural network model can be divided into two stages. The first is, of course, audio pre-processing, such as encoding, splitting audio into frames, or creating mathematical models such as spectrograms, chromagrams, etc. Many of these methods were described in their work by Palanisamy K., Shingania D., Yao A., and explained in which cases it is better to prepare the data in which way [12]. Next, you need to choose the classification method itself, such as the support vector method, k-means, neural networks, or deep learning.

When determining the assistant's functionality, you need to consider whether the store is electronic or physical, whether it is designed to search for specific products, whether it can provide up-to-date information about product availability, discounts, and promotions. In our case, the assistant will be able to include commands of the following type:

- "Where can I find product X?"
- "How much does product X cost?"
- "Find product X in size Y"

One approach to finding products can be to analyze and classify them according to their description and characteristics. With the help of machine learning, you can train algorithms to recognize certain features of products and use this information for search. Such characteristics can be name, color, brand, article, price, category (age, gender, etc.). In addition to classification algorithms, you can use search algorithms such as binary search, search trees. You need algorithms that are fast enough and able to work with large amounts of data.

To train the neural network, a lot of data had to be collected, including audio recordings. First, a set of recognition words was identified that could be used to search for clothes: various command words, such as "find", "show", etc.; words for product names, such as "shirt", "skirt", etc., and various colors. The dataset also contains a variety of complete sentences to train the model to distinguish between cases. A large number of entries of the same word will help to better distinguish certain characteristics of words, which will significantly improve the classification accuracy. In addition, the dataset contains data with different background sounds and mispronunciations. The architecture of the dataset is as follows (Fig. 2)

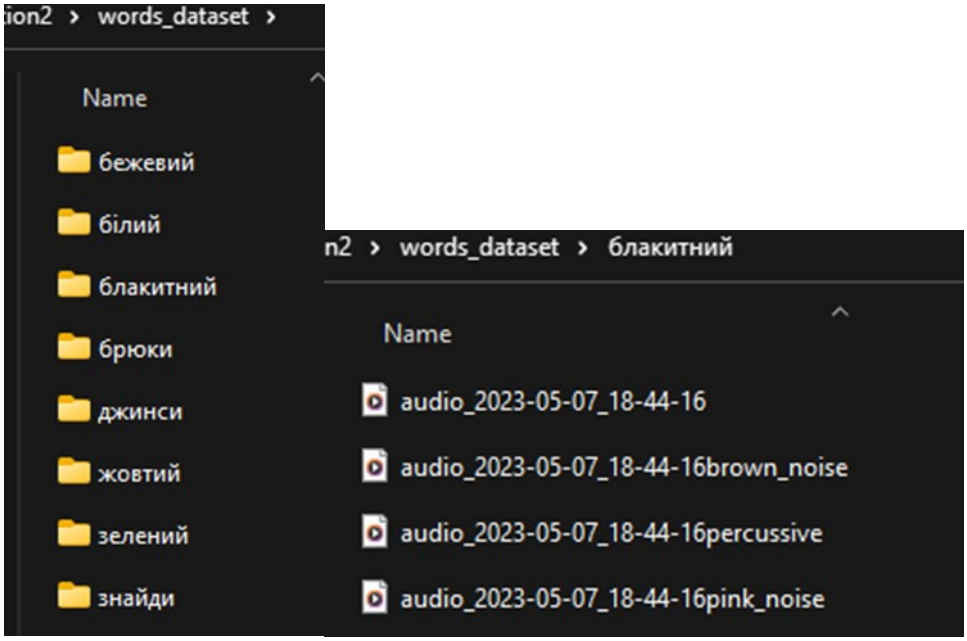


Fig.2. Dataset architecture

As you can see, the data is organized into categories, each category has the name of the recorded command. The files themselves are saved in the ".wav" format, with the date they were recorded and the sequence number in the name. Also, to increase the amount of data, some effect or noise was applied to each recording, which is also indicated in the recording name. The recorded sentences are stored in a separate folder and contain the text of the audio recording in the title.

In total, the dataset contains 30 different categories and a total of 3095 audio files. Each category contains an average of 90-100 records, and the sentence folder contains the most files - 287.

To train on this data, we first need to normalize all the data, i.e., reduce it to a single volume, frequency, and convert it to a single channel (mono) format.

In addition to evaluating the accuracy of speech recognition, it is also important to consider aspects of the development environment and technical requirements associated with the model.

The development environment includes aspects such as programming languages, libraries, and infrastructure required to deploy and use the model. To ensure optimal use of the model, additional technical components may also be required, such as building an API interface for easy interaction.

The functionality of training, improving and testing the model, audio data preprocessing will be developed in Python programming language version 3.8 using such libraries as NeMo, Numpy, Pandas, Torch, Librosa and others. For optimal development, it is recommended to use an integrated development environment (IDE) such as PyCharm or Visual Studio Code, which provide an extended set of tools.

Given the power of the hardware, training will take place on the CPU, although for faster training, such models should be trained on the GPU.

Python will also be used to develop the voice assistant system. The Django framework will be used to develop the API, and HTML, CSS, and JavaScript will be used for the user interface.

It was decided to use a pretrained ASR (Automatic Speech Recognition) model (theodotus/stt_uk_squeezeformer_ctc_ml) provided by Nvidia in its NeMo library. ASR models, also known as STT (Speech To Text) models, are systems that are designed to automatically convert speech to text. These models are used in a variety of applications, including voice command matching, transcription of audio and video files, speech translation, and much more.

ASR models are usually built on the basis of deep neural networks, in particular recurrent neural networks (RNNs), such as LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit), or transformers, such as in the Transformer model. The ASR model process can be divided into several stages:

- Audio pre-processing: The audio signal is first subjected to pre-processing, which includes functions such as resampling (changing the sample rate), noise filtering, volume normalization, and possibly other operations to improve the sound quality and reduce the impact of noise.
- Feature extraction: The audio signal is fed to the input of the ASR model, where a feature extraction process is performed. Typically, this process involves converting the audio signal into a spectrogram or other vector representation that can be fed to the neural network.
- Acoustic model: This stage uses a neural network called an acoustic model. It takes as input the feature vectors obtained in the previous step and tries to predict the probability of each input vector belonging to different phonemes or phonetic units. This model is trained on a large amount of labeled audio data with text transcriptions.
- Lexical model: After the acoustic model, the lexical model is used to select the correct word sequence based on the received phonetic sequences. The lexical model can include a dictionary or other resources to help determine the most likely word sequence for a given phonetic sequence.
- Language model: The next step is the language model, which is used to select the most likely text based on the word sequence derived from the lexical model. The language model takes into account the linguistic properties of the language and the context to produce a more natural and understandable textual result.
- Decoding: The last step is decoding, where the most likely text is selected based on the output probabilities of the language model. This may include post-processing, such as error correction or contextual adaptation.

ASR models are trained using large datasets that contain audio recordings and corresponding text transcriptions. Training requires powerful computing resources and a long time. However, with the development of deep learning technologies, ASR models are becoming more accurate and able to work with a variety of speech types and accents.

The processors used to process this model can be dependent on the implementation and hardware used. Typically, GPUs (Graphics Processing Units) or specialized processors for accelerating computation, such as TPUs (Tensor Processing Units), are used to train and execute ASR models. The choice of processor depends on the availability of resources and the size of the task to be performed.

The theodotus/stt_uk_squeezeformer_ctc_ml model used for Ukrainian speech recognition is based on the Squeezeformer architecture and uses the CTC (Connectionist Temporal Classification) method. Figure 3 shows the architecture of the Squeezeformer model [13].

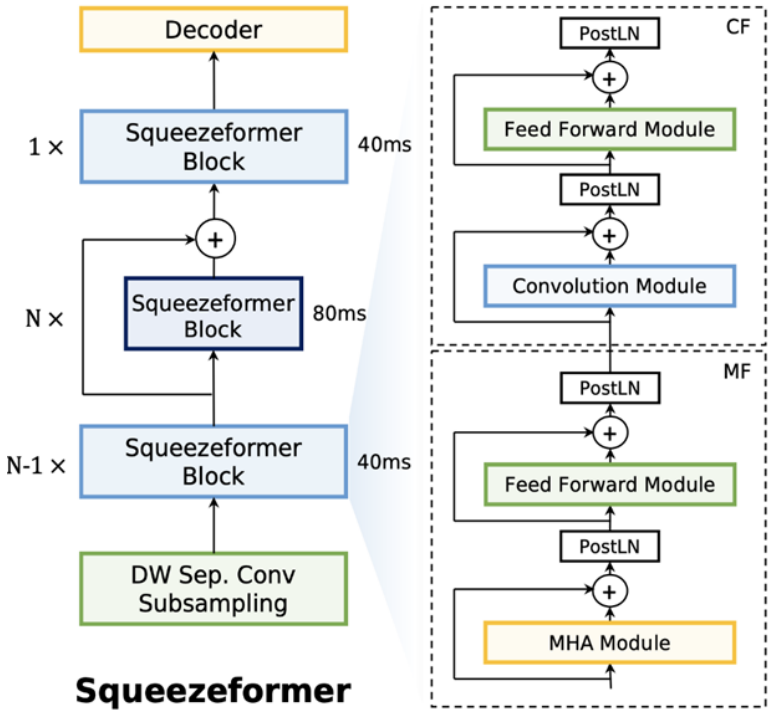


Fig. 3. Squeezeformer model architecture

Squeezeformer is an architecture that combines Squeeze-and-Excitation (SE) and Transformer to model channel importance in incoming audio and sequential dependencies in speech. It uses SE to compress channels and generate weights, and Transformer to model contextual dependencies in speech.

For training and decoding, the EncDecCTCModel class is used, which contains all the necessary functionality for setting up input data, training, and testing. Next described methods that are used in training process.

AudioToMelSpectrogramPreprocessor is a method that converts audio files of the ".wav" format into mel spectrograms. Also, at this stage, input data is processed before feature extraction, for example, resampling or noise filtering

SqueezeformerEncoder is used to implement the encoder. The encoder is used to convert the input features into a compact representation that is used for further speech recognition. These compressed features are then passed to the next layer.

The ConvASRDecoder processes the output features. Next, it determines the probabilities of characters and generates their sequences corresponding to the text of the input audio recording.

CTCLoss implements the CTC loss function to calculate the loss between the recognized character sequences and the actual labels. This loss is used to train the ASR model by reducing the discrepancy between the recognized and true character sequences.

SpectrogramAugmentation allows us to improve the model training by increasing the number of spectrograms.

WERBPE is used to calculate the accuracy of the model.

After the data has been processed and divided into training, validation, and test sets, you need to configure the neural network parameters. The main parameters are:

- Audio frequency(16000kHz)
- Path to the data set
- batch_size – a parameter that determines the number of data samples that are simultaneously submitted to the model before updating its weights(16)
- Number of processors on which the network will be trained(0)

The training will take place for 30 epochs, but functionality will be added to stop the training earlier if there is no loss reduction for three epochs. Since the model takes quite a long time to train, the ability to save the progress at each epoch and save the final model to a ".nemo" file has also been added.

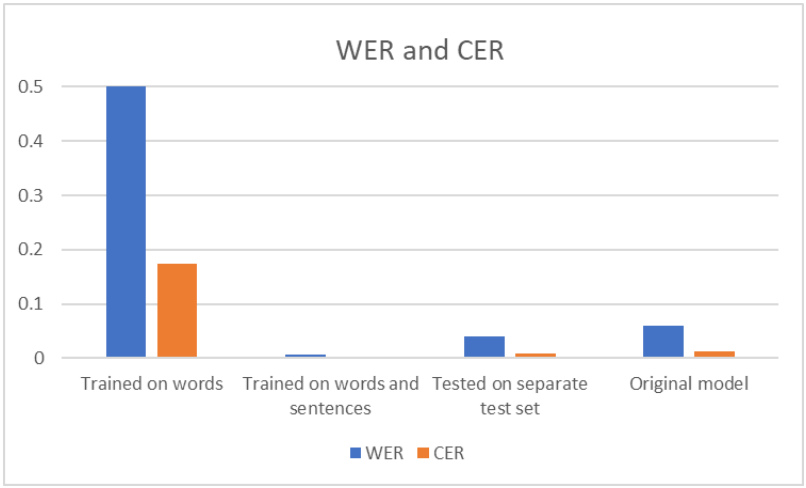
The TensorBoard library was used to determine the training accuracy and validate the model. Also, the model was tested on my and test data to determine the accuracy of the model.

WER (Word Error Rate) and CER (Character Error Rate) are metrics used to evaluate the accuracy of automatic speech recognition (ASR) and optical character recognition (OCR) systems, respectively. Both metrics measure the level of errors in the reproduction of text. WER is defined as the ratio of the total number of insertions, deletions, and substitutions made by the system to the total number of words in the input text. It measures the difference between the recognized text and the correct text at the word level. It is usually expressed as a percentage or as a decimal number. The CER, on the other hand, is defined as the ratio of the total number of insertions, deletions, and replacements made by the system to the total number of characters in the input text. It evaluates the recognition accuracy at the level of individual characters. Like WER, CER is usually expressed as a percentage or as a decimal number. If the WER or CER is close to zero, it means that the recognition system has achieved high accuracy. Values close to 100% indicate low recognition accuracy. WER and CER are useful metrics for comparing different speech recognition systems or evaluating the quality of ASR and OCR models. They allow you to evaluate the efficiency of the system in terms of accuracy and draw conclusions about the quality of its performance.

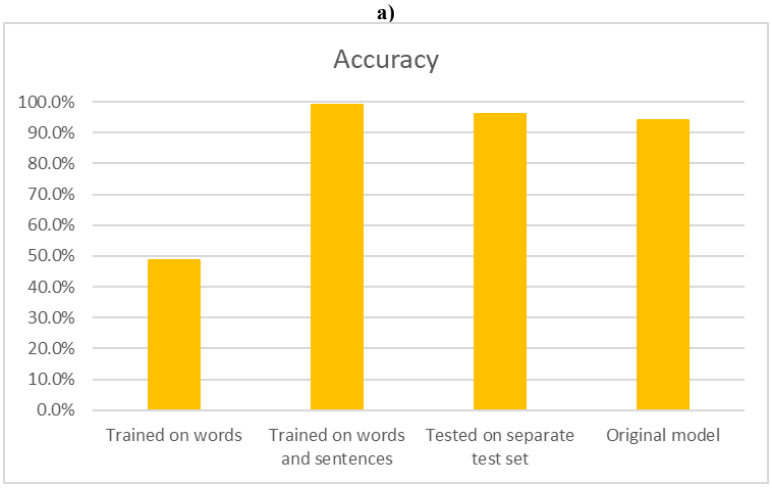
Initially, the model was trained on a set of words only, without full sentences. The training stopped at 7 epochs and had the following accuracy when tested on sentences: WER: 0.5121; CER: 0.1732, which meant that errors were found in almost 50% of the words. The problem was that the model could not recognize word cases. It was then decided to train the model on sentences as well.

This combination of data performed well during training, and in the next attempt the accuracy was like this: WER: 0.0069; CER: 0.0012, which meant that the number of errors was less than one percent. However, the same data was used in testing as in training. A separate test set with sentence recordings was collected for additional testing and the following accuracy was obtained: WER: 0.0398; CER: 0.0087. This test set covers the entire word set.

In general, when testing the model, it was found that the command words are recognized best, and there are not many errors in the names of clothes. Most often, the network makes mistakes on colors when they are not in the nominative case. The final accuracy of the neural network is 96.02%. In the graphs in Figure 4, you can see the comparison of model accuracies in different experiments and the comparison with the original model (the accuracy of the original model was taken from the documentation [13]).



a)



b)

Fig.4 Comparison a) values of WER and CER; b) values of accuracy

As we can see, the trained model on words and sentences shows better accuracy than the original model. Figure 5 shows the loss and wer plots for training and validation.

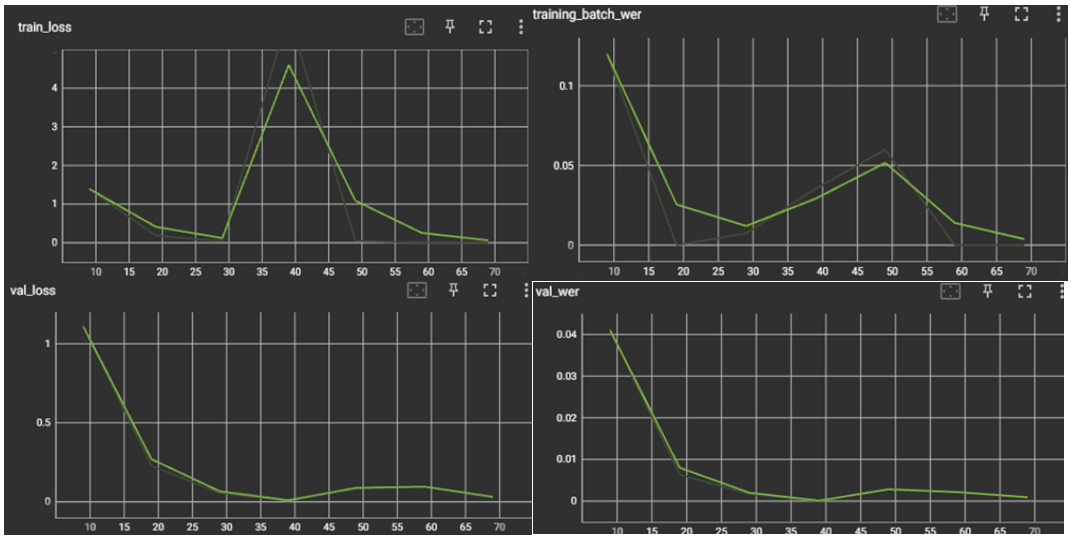


Fig.5 Accuracy and loss graphs for training and validation

We can see that basically the functions are stable, which means that the network is trained well. There is a slight increase in losses in the middle of the training, which may be due to the small size of the dataset, or a small amount of low-quality data that increased the losses. However, we can see that the losses and validation accuracy were steadily approaching zero.

As mentioned above, the voice assistant will be based on the REST API developed on the Django framework. In order to create a voice assistant, the following functionality was created:

- A user interface was created to record audio from the microphone and display the shelf number on the screen;
- The recognize_speech function that receives the recording from the microphone, normalizes it, feeds it to the neural network for transcription, and returns the text;
- The search_keywords function that accepts the text, splits it into words, discards the command words, searches the database, and returns the shelf number.

To pass a record from the interface to the recognize_speech method, the startRecognition JavaScript function was created, which, when you click the "Start" button, sends a request to the API, which in turn launches this method. To use our trained model, the NeMo library will be used to deploy and save the model from a file in the ".nemo" format. Next, the recording from the microphone should be normalized in the same way as for training, i.e., reduced to one channel and set to 16000. Now the audio can be fed to the neural network to be converted to text and passed to the search_keywords method.Figure 6 shows how the voice assistant works.

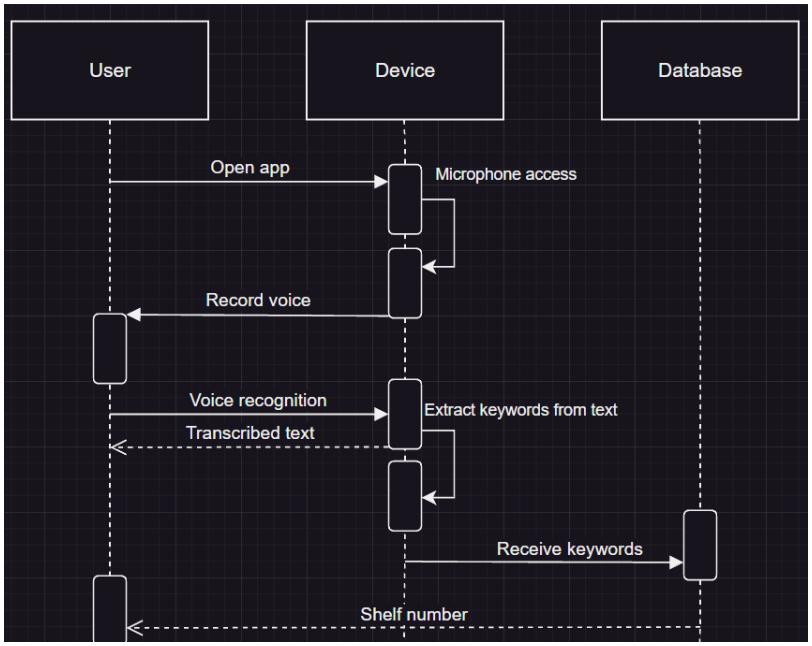


Fig.6. Sequence diagram of voice assistant

Next, the text is split into words, and all the command words that are not included in the search are defined in a separate file and will be removed from the text. Next, you need to take into account the difference in cases in the words in the voice command and in the database. For example, if you ask your assistant for a blue skirt in the accusative case, and the database contains a blue skirt in the nominative case, nothing will be found without taking this into consideration. Therefore, the UkStemmer library was used, which is a tool for processing the Ukrainian language and has the ability to highlight the stem of a word. The stem of a word is its basic form, which represents the essence of the word without endings and other morphological changes. The search algorithm itself is designed in such a way that it will find items for all keywords at once, but if it doesn't find one, it will continue to search by the name of the item. For example, if you wanted a red T-shirt and there is no such thing available, the assistant will offer you other T-shirts. The search results, i.e. all possible shelf numbers, are then returned to the main page. To test the algorithm, we need to create a test dataset with clothes to imitate database. Since we don't care about the authenticity of the products we will use we will use, this dataset can be generated. This dataset will contain such data as product names, color, size, location (shelf number). The items should have a certain dependency in location, for example, the same item of different sizes should be on the same shelf, and items of the same type can be on neighboring shelves. An example of how the assistant works can be seen in Figure 7.

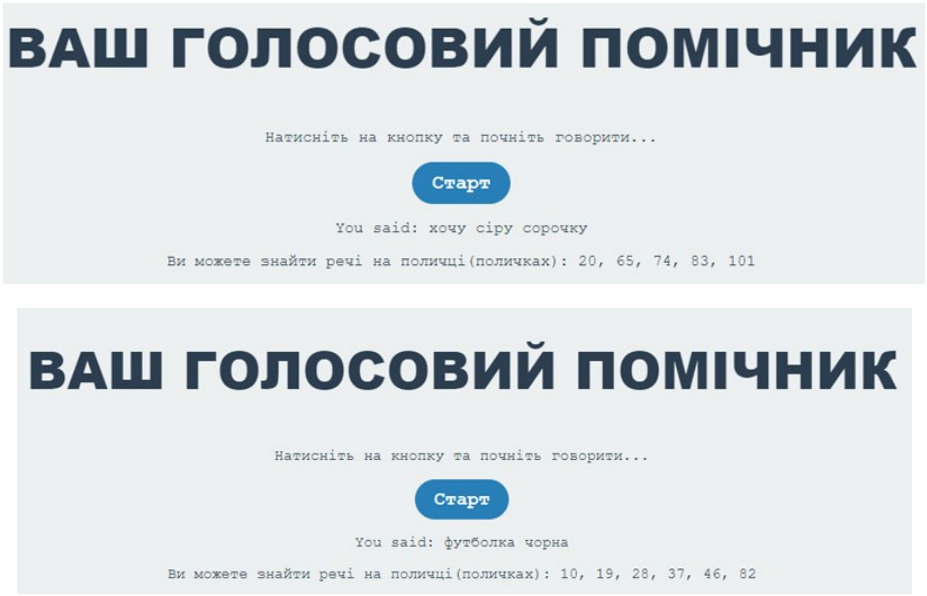


Fig.7. Example of how voice assistant works

As we can see, the speech recognition system demonstrates stable and high accuracy during microphone recording. This is important because it provides users with reliable and accurate recognition results even when using simple microphones. The search also works well, as it is performed on keywords individually, so the order of the words does not matter, which means that the system has flexibility and understanding of speech contexts. Users will find it convenient to use keywords to quickly find specific information.

The interface is designed with ease of use in mind, including for users with no previous experience with technology. The interface has intuitive controls that make it easy to use the program. At the top of the interface is the Start button, which is used to start recording audio. The color of the button changes to red when pressed, indicating to the user that the recording process has started. When the recording is complete, the button returns to blue, indicating that the recording is over. The recorded text is also displayed on the screen, allowing the user to check the correctness of the recognition. This can be useful for identifying possible errors or inaccuracies in the recognized text.

Conclusions

The task of the thesis was set, and the steps to accomplish it. The task was to offer our own solution to the problem and develop a system for its implementation. The methods of audio signal processing were analyzed, the literature was analyzed, and the existing approaches to voice command recognition were compared. The current state of the market and the demand for the technology were researched. The next step was to create a dataset and prepare it for training. The dataset contains 30 different categories and 3095 audio files. The number of records was increased by augmentation. The neural network model was trained and tested. The model's accuracy reached 96.02%. When tested on a separate data set, the WER was 3.98% and the CER was 0.87%. The neural network was integrated with the search system in the API. The speech recognition system showed stable and high accuracy during microphone recording. The search works well and allows you to find results by keywords and names of things. The system's flexibility allows it to understand speech contexts and does not depend on the order of words in a phrase. In general, the results indicate a successful implementation of a voice assistant with high speech recognition accuracy and efficient search.

References

1. Balamurali B. T., Lin K. E., Lui S. et al. Toward robust audio spoofing detection: A detailed comparison of traditional and learned features. IEEE Access. Vol. 7, 2019. P. 84229–84241. DOI:10.1109/ACCESS.2019.2923806.
2. Dentamaro V., Giglio P., Impedovo D. et al. AUCO ResNet: an end-to-end network for Covid-19 pre-screening from cough and breath. Pattern Recognition. Vol. 127, 2022. DOI:10.1016/j.patcog.2022.108656.
3. Esmaeilpour M., Cardinal P., Koerich A. L. A Robust Approach for Securing Audio Classification against Adversarial Attacks. IEEE Transactions on Information Forensics and Security. Vol. 15, 2020. P. 2147–2159. DOI:10.1109/TIFS.2019.2956591.
4. Hasib K. M., Tanzim A., Shin J. et al. BMNet-5: A Novel Approach of Neural Network to Classify the Genre of Bengali Music based on Audio Features. IEEE Access. 2022. P. 1–1. DOI:10.1109/ACCESS.2022.3213818.
5. Huang Y.-P., Mushi R. Classification of Cough Sounds Using Spectrogram Methods and a Parallel-Stream One-Dimensional Deep Convolutional Neural Network. IEEE Access. Vol. 10, 2022. P. 97089–97100. DOI:10.1109/ACCESS.2022.3205591.
6. Kantipudi M. V. V. P., Kumar S. A computationally efficient learning model to classify audio signal attributes. International Journal of Electrical and Computer Engineering. Vol. 12, Issue 5. P. 4926–4934. DOI:10.11591/ijece.v12i5.pp4926-4934.
7. Sheibani R., Nikookar E., Alavi S. An ensemble method for diagnosis of Parkinson's disease based on voice measurements. Journal of Medical Signals and Sensors. Vol. 9, Issue 4. P. 221–226. DOI:10.4103/jmss.JMSS_57_18.
8. Sterpu G., Harte N. Taris: An online speech recognition framework with sequence to sequence neural networks for both audio-only and audio-visual speech. Computer Speech and Language. Vol. 74, 2022. DOI:10.1016/j.csl.2022.101349.

9. Tsalera E., Papadakis A., Samarakou M. Comparison of pre-trained cnns for audio classification using transfer learning. Journal of Sensor and Actuator Networks. Vol. 10, Issue 4. DOI:10.3390/jsan10040072.
10. Vargas E., Hopgood J. R., Brown K. et al. On Improved Training of CNN for Acoustic Source Localisation. IEEE/ACM Transactions on Audio Speech and Language Processing. Vol. 29, 2021. P. 720–732. DOI:10.1109/TASLP.2021.3049337.
11. Кривохата А. Г. Огляд методів машинного навчання для класифікації акустичних даних / А. Г. Кривохата, О. В. Кудін, А. О. Лісняк // Вісник Херсонського національного технічного університету, Херсон, 2018. 331 с.
12. Palanisamy, Kamalesh; Singhania, Dipika; Yao, Angela. Rethinking CNN models for audio classification. National Institute of Technology, Tiruchirappalli, India. 2020.
13. Nvidia NeMo documentation [Electronic resource]. Retrieved from: https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/starthere/intro.html

Volodymyr Shymanskyi **Володимир Шиманський**	Ph.D., Associated Professor at the Department of Artificial Intelligence, Lviv Polytechnic National University, Lviv, Ukraine e-mail: volodymyr.m.shymanskyi@lpnu.ua https://orcid.org/0000-0002-7100-3263	Доцент кафедри системи штучного інтелекту Національного університету “Львівська політехніка”
Nikoletta Kazonnikova **Ніколетта Казьоннікова**	Student at the Department of Artificial Intelligence, Lviv Polytechnic National University, Lviv, Ukraine e-mail: nikoletta.kazonnikova.knm.2019@lpnu.ua https://orcid.org/0000-0003-3899-5894	Студентка кафедри системи штучного інтелекту Національного університету “Львівська політехніка”