DOI 10.34185/1562-9945-2-133-2021-06
UDC 004.415.2.031.43

V.V. Spirintsev, D.V. Popov, O.V. Spirintseva

VIRTUAL DIGITAL ASSISTANT WITH VOICE INTERFACE SUPPORT

Abstract. A virtual digital assistant which can work with arbitrary systems and provide an effective solution of narrowly focused user tasks for interaction with Ukrainian services voice interface supported has been proposed. The developed web service was implemented by using the PHP programming language, Wit.ai service for audio signal processing, FANN library for neural network construction, Telegram service for creating an interface.
Keywords: digital assistant, API, databases, neural network, voice interface.

Problem. The increase in the total amount of information that a person must process has necessitated the creation of systems for daily tasks automation. In addition, modern machinery is equipped with components sufficient to create an interactive interface. So this allows to develop software products for fast processing of text and voice requests of the user.

Analysis of researchers. Today, the software market offers the user lots of different virtual digital assistants (VDA), designed for the porpose of effective interaction with external online services for everyday problems solving (scheduling, organization and execution of everyday affairs, contextual information retrieval, etc.) [1]. However, the existing universal solutions (Google Assistant, "Алиса", "Дуся", Bixby, Siri) do not allow the user to solve narrowly focused tasks (e.g., control of parcels at leading postal operators, ordering transport tickets, taxi call and etc.) effectively without the request specifying (choice of service among those are proposed by assistant and further manipulation with this one). Most of VDAs are focused on the foreign market and do not have close integration with Ukrainian services (the applications have only the basic functionality of device management and provide the work with a limited set of systems), so it reduces the efficiency of their work. The efficiency of the VDA is also affected by the way the commands are entered (text, voice, etc.) to execute a specific request. Voice interface support provides the most complete implementation of the program compared to the text version of the command. Therefore, the development of the virtual digital assistant that will ensure simulta-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

neously close integration with domestic services and support of voice interface is relevant and needs to be studied further.

The aim of the work is to summarize and systematize the results of research [2] according to the development of the virtual digital assistant with a voice interface.

The main part. The general structure of the project using the development modular approach is shown in the Figure 1.

The core of the software product allows you to receive data from the input module, process them (depending on the request) and transmit to the output module. This system allows not to depend on external services, giving the opportunity to switch to other service easily if necessary. This feature of the architecture will allow to implement both the web interface and messages with third-party programs (e.g., Telegram) easily.

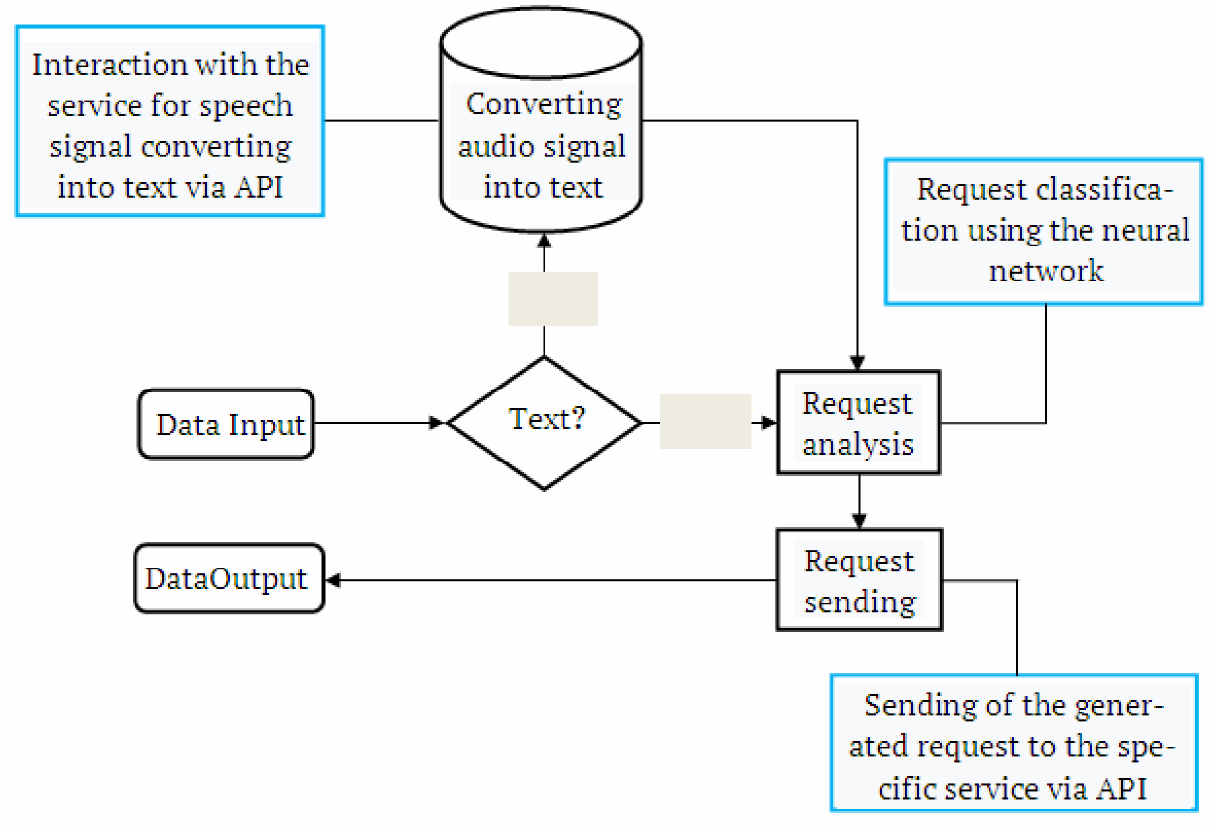

Interaction with the service for speech signal converting into text via API
Converting audio signal into text
Request classification using the neural network
Data Input
Text?
Request analysis
DataOutput
Request sending
Sending of the generated request to the specific service via API

Figure 1 - Software product architecture

The general architecture consists of (fig.1):

- Data input module. Allows the user to interact with the system by entering the text or voice command

- Data verification module. Performs the analyzing function applied to the entered information. If the data is text, the request is sent to the analysis module. If it`s audio, then the request is sent to the conversion module

Module of data sending for conversion. Allows to convert user`s request quickly and return a text response

- Request analysis module. Analyzes the user's request and compares it with the software command.

- Request sending module. After the request has been generated, the program sends it to the appropriate subroutine.

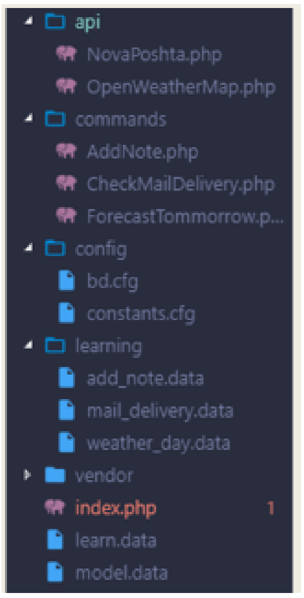
- Data output module. After the information has been received, its formation and output is performed

Figure 2 shows the directory structure of the developed application.

API. This folder contains files that are responsible for the system's interaction with the external services. They allow to send a request to the definite resource, depending on the user's command, and then return the response

Learning. It`s necessary a neural network to be created to process user requests intelligently. To teach it the request type determining, it`s needed to create the training data, which will later be located in this folder.

Vendor. To accelerate the development of the software product, it is necessary to use the third-party frameworks. According to the current standard, they are sent to this folder


```
api
    NovaPoshta.php
    OpenWeatherMap.php
commands
    AddNote.php
    CheckMailDelivery.php
    ForecastTommorrow.p...
config
    bd.cfg
    constants.cfg
learning
    add_note.data
    mail_delivery.data
    weather_day.data
vendor
index.php                 1
learn.data
model.data
```
Figure 2 - Directory structure

Config. In order not to add data for the connection to the database in each executable file, access constants and other project settings quickly, all configuration files are located in this folder

Command processing tools

After defining the entered command, the task of its implementation appears. You need to develop an appropriate routine to do this. Implemented tasks can be divided into several classes:

- Basic interactions with user data.

- Request of the data from external online services.

- User data processing that requires operating system tools.

Interaction with the database

In this case, the work is carried out with the connected database. Regardless of the selected database software implementation, there are such requests. The first

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

one is the request for data available in the database and the second one is the recording of new data into the database (Fig. 3).

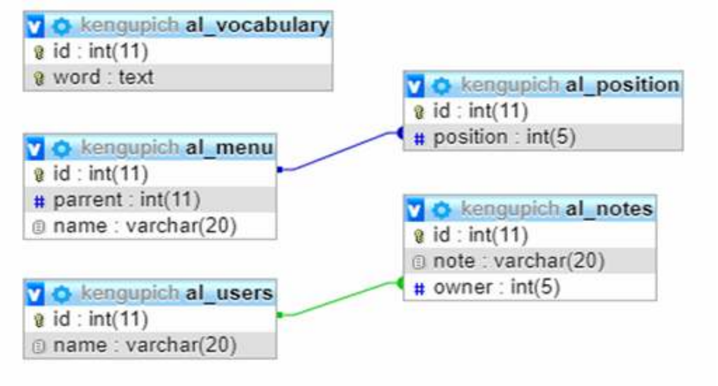
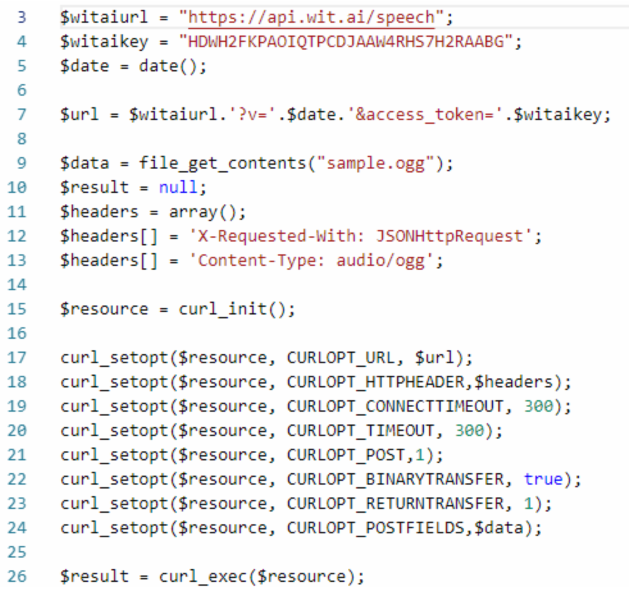


Figure 3 - Database design

This procedure is undemanding to the machine resources, so it is performed rapidly.

Interaction with third-party service and the choice of the services

To implement user interaction with the system through voice commands, it`s necessary to send the received audio file to the appropriate service for decoding. This method can significantly reduce development time, as it offers higher accuracy. To solve this problem, the Wit.ai service was used to convert the audio signal to a symbolic form by interacting with them via the API. The system receives an audio file in "ogg" format and returns the finished text in JSON format. To interact with this service, one needs to send a voice file using the cURL function (Fig. 4).



```php
$witaiurl = "https://api.wit.ai/speech";
$witaikey = "HDWH2FKPAOIQTPCDJAAW4RHS7H2RAABG";
$date = date();

$url = $witaiurl.'?v='.$date.'&access_token='.$witaikey;

$data = file_get_contents("sample.ogg");
$result = null;
$headers = array();
$headers[] = 'X-Requested-With: JSONHttpRequest';
$headers[] = 'Content-Type: audio/ogg';

$resource = curl_init();

curl_setopt($resource, CURLOPT_URL, $url);
curl_setopt($resource, CURLOPT_HTTPHEADER,$headers);
curl_setopt($resource, CURLOPT_CONNECTTIMEOUT, 300);
curl_setopt($resource, CURLOPT_TIMEOUT, 300);
curl_setopt($resource, CURLOPT_POST,1);
curl_setopt($resource, CURLOPT_BINARYTRANSFER, true);
curl_setopt($resource, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($resource, CURLOPT_POSTFIELDS,$data);

$result = curl_exec($resource);
```

Figure 4 – Sending a voice file for processing

Also, based on customer`s requirements, the service OpenWaetherMap was chosen. It allows receiving weather information for the current and some next days for free. Another external service is the Ukrainian company «Нова Пошта». The program allows you to re-check the status of the parcel quickly without using the third-party applications.

Creating a neural network

Since a strictly limited set of commands will make it difficult to interact with the assistant (when scaling the application, the number of commands will increase, and this will cause difficulties in the program mastering), there is a necessity to convert the received characters into a specific query. It`s necessary to perform the intelligent processing of the original text to achieve this. An effective solution to this problem is the use of neural networks. To teach the system determining the query while using the associative memory method [3], it`s necessary to convert words from the original text into numbers and compose the vector, and then repeatedly correlate the specific vector with a specific query. In this manner the map of relationships is created, and according to it any text will be converted into a specific query. After the request specification it`s necessary to start a cycle of works with a certain service by means of the presented API. In practice, there is often a need to use the ready-made solutions that significantly can improve the development of the software product. We used a ready-made library - FANN (Fast Artificial Neural Network) in this work. This allowed us to implement an artificial neural network with the specified parameters quickly. It offers the high speed and flexibility in use, so one can solve the most of problems. To create a neural network, the function of the FANN library - fann_create_standart_array () is used. The arguments of the following function are

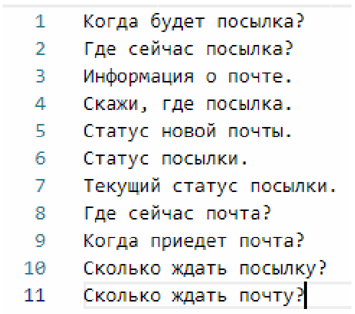
1. The number of layers.
2. The array with the following data:
   - Number of the network entries. This is the total volume of the dictionary in this task.
   - Number of the hidden layers. It is determined experimentally.
   - Number of the network outputs.

The dictionary is determined on the basis of all the unique words that are presented in the training data. To optimize the dictionary, it is necessary to process each word with the function of determining its root.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Preparation of the educational data

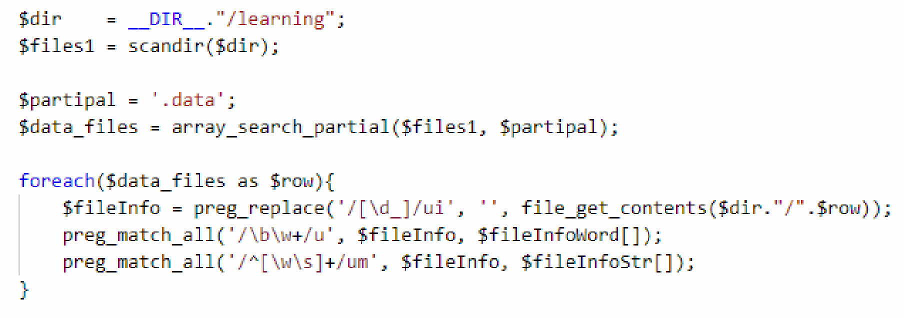
To train a network it`s necessary to create files with the names that correspond to the names of the commands they belong to. The file should consist of phrases that can be assigned to a specific command. Each line must end with the sign according to Fig. 5.



```
Когда будет посылка?
Где сейчас посылка?
Информация о почте.
Скажи, где посылка.
Статус новой почты.
Статус посылки.
Текущий статус посылки.
Где сейчас почта?
Когда приедет почта?
Сколько ждать посылку?
Сколько ждать почту?
```

Figure 5 – Sample of the educational data

To ensure the support for any number of commands, the search for the appropriate files in a given directory is carried out (Fig. 6).

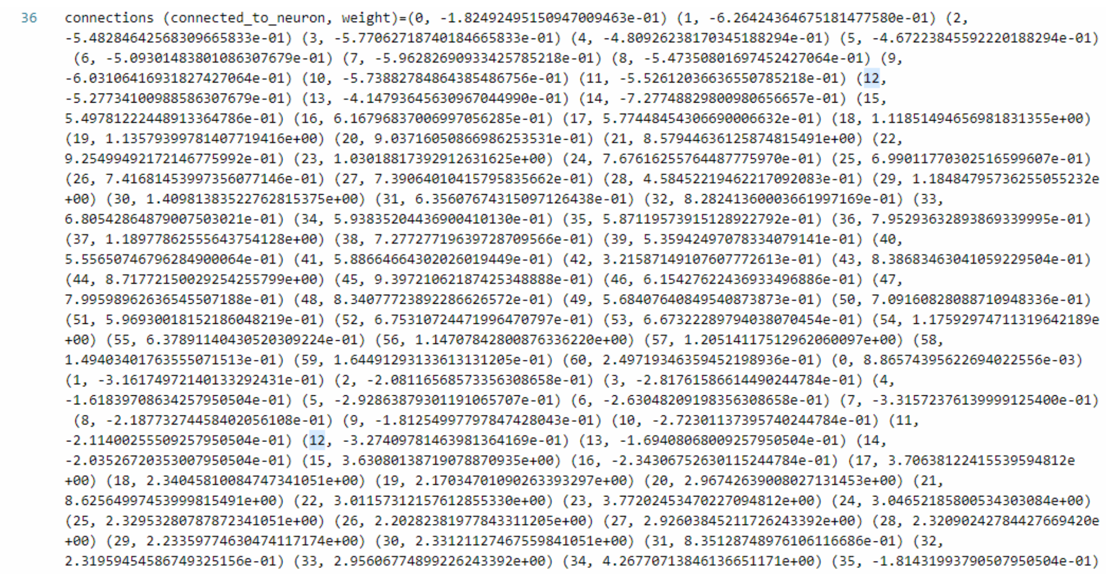


```
$dir    = __DIR__."/learning";
$files1 = scandir($dir);

$partipal = '.data';
$data_files = array_search_partial($files1, $partipal);

foreach($data_files as $row){
    $fileInfo = preg_replace('/[\d_]/ui', '', file_get_contents($dir."/".$row));
    preg_match_all('/\b\w+/u', $fileInfo, $fileInfoWord[]);
    preg_match_all('/^[\w\s]+/um', $fileInfo, $fileInfoStr[]);
}
```

Figure 6 – The search for training files

Next, it`s necessary to analyze all the files given to the system, separate the roots from them using the Porter stemmer, and create the dictionary of the words. The volume of this dictionary will determine the size of the vector that will be fed to the input of the neural network. Then the vectors of a given length and consist of zeros are created. After that, the words obtained from the training files are processed with the function, which returns the sequence number of each processed word from the created dictionary. The obtained numbers are used as position indices of ones in the created vector. After the module has been executed, a new file "learn.data" is created. Then it is fed to the neural network for learning. Then the model with the coefficients of all neurons is created (Fig. 7).

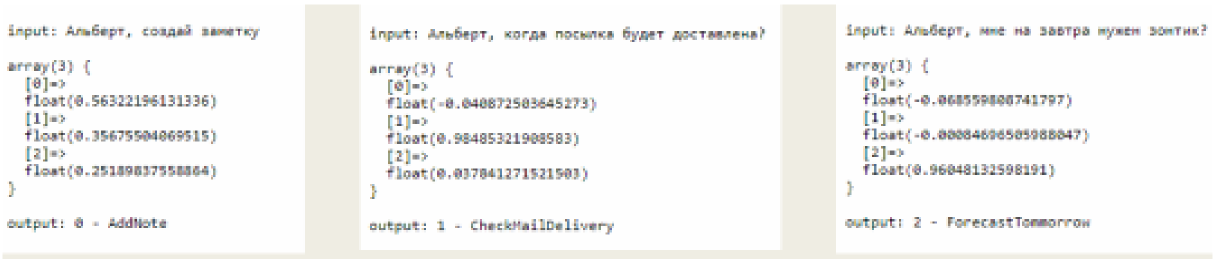
The neural network training is carried out through the analysis of the educational materials. To do this, we need to create the appropriate files with the "data" extension.



```
connections (connected_to_neuron, weight)=(0, -1.82492495150947009463e-01) (1, -6.26424364675181477580e-01) (2,
-5.48284642568309665833e-01) (3, -5.77062718740184665833e-01) (4, -4.80926238170345188294e-01) (5, -4.67223845592220188294e-01)
 (6, -5.09301483801086307679e-01) (7, -5.96282690933425785218e-01) (8, -5.47350801697452427064e-01) (9,
-6.03106416931827427064e-01) (10, -5.73882784864385486756e-01) (11, -5.52612036636550785218e-01) (12,
-5.27734100988586307679e-01) (13, -4.14793645630967044990e-01) (14, -7.27748829800980656657e-01) (15,
5.49781222448913364786e-01) (16, 6.16796837006997056285e-01) (17, 5.77448454306690006632e-01) (18, 1.11851494656981831355e+00)
(19, 1.13579399781407719416e+00) (20, 9.03716050866986253531e-01) (21, 8.57944636125874815491e+00) (22,
9.25499492172146775992e-01) (23, 1.03018817392912631625e+00) (24, 7.67616255764487775970e-01) (25, 6.99011770302516599607e-01)
(26, 7.41681453997356077146e-01) (27, 7.39064010415795835662e-01) (28, 4.58452219462217092083e-01) (29, 1.18484795736255055232e
+00) (30, 1.40981383522762815375e+00) (31, 6.35607674315097126438e-01) (32, 8.28241360003661997169e-01) (33,
6.80542864879007503021e-01) (34, 5.93835204436900410130e-01) (35, 5.87119573915128922792e-01) (36, 7.95293632893869339995e-01)
(37, 1.18977862555643754128e+00) (38, 7.27727719639728709566e-01) (39, 5.35942497078334079141e-01) (40,
5.55650746796284900064e-01) (41, 5.88664664302026019449e-01) (42, 3.21587149107607772613e-01) (43, 8.38683463041059229504e-01)
(44, 8.71772150029254255799e+00) (45, 9.39721062187425348888e-01) (46, 6.15427622436933496886e-01) (47,
7.99598962636545507188e-01) (48, 8.34077723892286626572e-01) (49, 5.68407640849540873873e-01) (50, 7.09160828088710948336e-01)
(51, 5.96930018152186048219e-01) (52, 6.75310724471996470797e-01) (53, 6.67322289794038070454e-01) (54, 1.17592974711319642189e
+00) (55, 6.37891140430520309224e-01) (56, 1.14707842800876336220e+00) (57, 1.20514117512962060097e+00) (58,
1.49403401763555071513e-01) (59, 1.64491293133613131205e-01) (60, 2.49719346359452198936e-01) (0, 8.86574395622694022556e-03)
(1, -3.16174972140133292431e-01) (2, -2.08116568573356308658e-01) (3, -2.81761586614490244784e-01) (4,
-1.61839708634257950504e-01) (5, -2.92863879301191065707e-01) (6, -2.63048209198356308658e-01) (7, -3.31572376139999125400e-01)
 (8, -2.18773274458402056108e-01) (9, -1.81254997797847428043e-01) (10, -2.72301137395740244784e-01) (11,
-2.11400255509257950504e-01) (12, -3.27409781463981364169e-01) (13, -1.69408068009257950504e-01) (14,
-2.03526720353007950504e-01) (15, 3.63080138719078870935e+00) (16, -2.34306752630115244784e-01) (17, 3.70638122415539594812e
+00) (18, 2.34045810084747341051e+00) (19, 2.17034701090263393297e+00) (20, 2.96742639008027131453e+00) (21,
8.62564997453999815491e+00) (22, 3.01157312157612855330e+00) (23, 3.77202453470227094812e+00) (24, 3.04652185800534303084e+00)
(25, 2.32953280787872341051e+00) (26, 2.20282381977843311205e+00) (27, 2.92603845211726243392e+00) (28, 2.32090242784427669420e
+00) (29, 2.23359774630474117174e+00) (30, 2.33121127467559841051e+00) (31, 8.35128748976106116686e-01) (32,
2.31959454586749325156e-01) (33, 2.95606774899226243392e+00) (34, 4.26770713846136651171e+00) (35, -1.81431993790507950504e-01)
```

Figure 7 – Coefficients for the words

To verify the operability of the software module, a number of tests were performed. The correctness of the ratio of the entered request to the specific program command was evaluated during these tests. For clarity of the result the function of output to the browser window was created. It`s arguments are the input phrase, the coefficients accepted during processing for each of possible outputs of the system and the system verdict based on the choice of the maximum coefficient among received ones (fig. 8).



```
input: Альберт, создай заметку

array(3) {
  [0]=>
  float(0.56322196131336)
  [1]=>
  float(0.35675504069515)
  [2]=>
  float(0.25189837558864)
}

output: 0 - AddNote
```

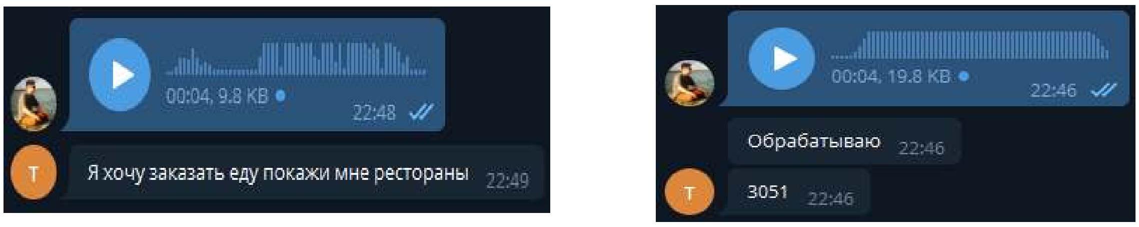
```
input: Альберт, когда посылка будет доставлена?

array(3) {
  [0]=>
  float(-0.040872503645273)
  [1]=>
  float(0.98485321908583)
  [2]=>
  float(0.037841271521503)
}

output: 1 - CheckMailDelivery
```

```
input: Альберт, мне на завтра нужен зонтик?

array(3) {
  [0]=>
  float(-0.068559808741797)
  [1]=>
  float(-0.00084696505988047)
  [2]=>
  float(0.96048132598191)
}

output: 2 - ForecastTommorrow
```

Figure 8 - The tests for answers correctness

Voice interface testing

After the voice request has been received, the system automatically sends the file to the processing service, which returns the text response. Figure 9 shows the operation of user requests that have been processed and converted to text.



Я хочу заказать еду покажи мне рестораны 22:49

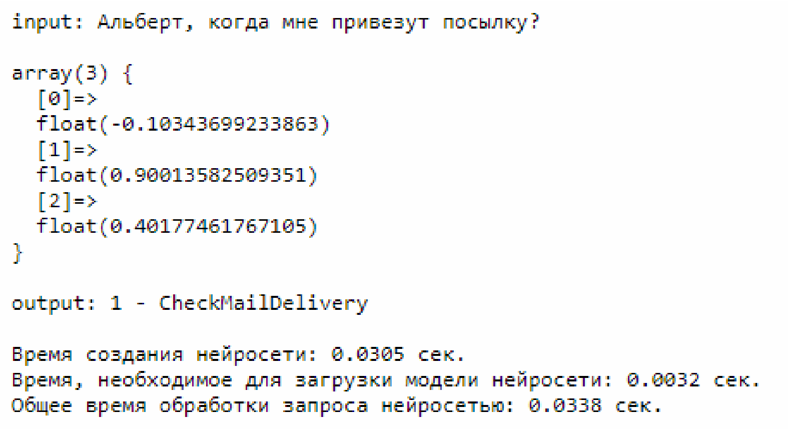
Обрабатываю 22:46

3051 22:46

Figure 9 - Voice interface testing

Speed analysis

To determine the speed of the system, it`s necessary to use a timer that will measure the execution time of the executable code. To implement this functionality in PHP, there is a function microtime (), which returns a Unix timestamp measured in microseconds. To assess the performance of the program before the execution of each of the code snippets, the temporary timestamps were added. After its execution the request to regain time was made. The difference between the two timestamps is the execution time. The results of the system speed testing are shown in table 1.



```
input: Альберт, когда мне привезут посылку?

array(3) {
  [0]=>
  float(-0.10343699233863)
  [1]=>
  float(0.90013582509351)
  [2]=>
  float(0.40177461767105)
}

output: 1 - CheckMailDelivery

Время создания нейросети: 0.0305 сек.
Время, необходимое для загрузки модели нейросети: 0.0032 сек.
Общее время обработки запроса нейросетью: 0.0338 сек.
```

Figure 10 – The test result

Table 1

Testing of the system speed

Inspection	Min	Avg	Max
Neural network creating	0.0186 c.	0.02655 c.	0.0509 c.
Model download	0.0027 c.	0.0058 c.	0.0322 c.
Total execution time	0.0216 c.	0.0325 c.	0.0833 c.

The obtained testing results allow asserting the high performance of the system.

Conclusions. The paper offers the virtual digital assistant with the voice interface support. The developed web service was implemented using the PHP programming language, the Linux operating system, IDE Visual Studio Code, Apache web server. The startup Wit.ai was used to process the audio signals. A ready-made solution - FANN (Fast Artificial Neural Network) was used to build the neural network. The Telegram was chosen as the service for the interface creating. This messenger allows third-party developers to program their own bots. These bots are the third-party utilities that run inside the program and significantly expand its capabilities. The finished product allows you to receive user requests from any application, process them and return the answer to the user both in the web-interface and in any application.

Further development of the proposed research can be carried out in the direction of expanding of the developed application functionality by adding of other required functions (calculator, reminder, integration with the taxi service, search queries, current exchange rate); by the Ukrainian language support adding; by adding of the new training data to increase the accuracy of the neural network request analysis; by expanding of the platforms list to interact with the application.

REFERENCES

1. Intelligent personal assistans [Electronic resource]. Access mode: https://www.predictiveanalyticstoday.com/top-intelligent-personal-assistants-automated-personal-assistants/

2. Спірінцев В.В. Розробка віртуального цифрового помічника з голосовим інтерфейсом/ В.В.Спірінцев, Д.В. Попов // Перспективні напрямки сучасної електроніки, інформаційних і комп'ютерних систем (MEICS-2018). Тези доповідей на III Всеукраїнській науково-практичній конференції: 21-23 листопада 2018 р., м. Дніпро. – Дніпро: Дніпровський національний університет імені Олеся Гончара, 2018. – С. 46-48.

3.Хайкин, Саймон. Нейронные сети: полный курс, 2-е изд.: Пер. с англ.- М.:ООО "И.Д.Вильямс".-1104 с.:ил.-Парал.тит.англ.

Received 09.02.2021.
Accepted 11.02.2021.

Віртуальний цифровий ассистент з підтримкою голосового інтерфейсу

Запропоновано віртуальний цифровий помічник, що може працювати з довільними системами та забезпечувати ефективне рішення вузьконаправлених задач користувача по взаємодії з українськими сервісами з підтримкою голосового інтерфейсу. Розроблений веб-сервіс було реалізовано за допомогою мови програмування PHP, сервісу Wit.ai для обробки аудіосигналів, бібліотеки FANN для побудови нейронної мережі, сервісу Telegram для створення інтерфейсу.

Виртуальный цифровой ассистент с поддержкой голосового интерфейса

Предложено виртуальный цифровой помощник, который может работать с произвольными системами и обеспечивать эффективное решение узконаправленных задач пользователя по взаимодействию с украинскими сервисами с поддержкой голосового интерфейса. Разработан веб-сервис был реализован с помощью языка программирования PHP, сервиса Wit.ai для обработки аудиосигналов, библиотеки FANN для построения нейронной сети, сервиса Telegram для создания интерфейса.

Спірінцев В'ячеслав Васильович - доцент, к.т.н., доцент кафедри програмного забезпечення комп'ютерних систем Національного технічного університету "Дніпровська політехніка".
Попов Дмитро Віталійович - фахівець із супроводу проектів ТОВ "РУШ".
Спірінцева Ольга Володимирівна - к.т.н., доцент кафедри електронних обчислювальних машин Дніпровського національного університету імені Олеся Гончара.

Спиринцев Вячеслав Васильевич - доцент, к.т.н., доцент кафедры про-программно обеспечение компьютерных систем Национального технически-го университета "Днепровская политехника".
Попов Дмитрий Витальевич - специалист по сопровождению проектов ООО "РУШ".
Спиринцева Ольга Владимировна - к.т.н., доцент кафедры электро-нных вычислительных машин Днепровского национального универси-тета имени Олеся Гончара.

Spirintsev Vyacheslav Vasilievich - Associate Professor, Candidate of Technical Sciences, Associate Professor of the Department of Pro-Software Support of Computer Systems of the National Technical University "Dnipro Polytechnic".
Popov Dmitry Vitalievich - project support specialist of RUSH LLC.
Olga Vladimirovna Spirintseva - Candidate of Technical Sciences, Associate Professor of the Department of Electronic Computing Machines, Oles Honchar Dnipro National University.