Development of an Intelligent Chatbot for a Hospital Website

Tetiana A. Vakaliuk [1,2,3], Dariia G. Skripchenko[1], Mariia O. Medvedieva[4] and Mykhailo G. Medvediev[5]

[1] *Zhytomyr Polytechnic State University, 103 Chudnivsyka Str., Zhytomyr, 10005, Ukraine*
[2] *Institute for Digitalisation of Education of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine*
[3] *Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*
[4] *Pavlo Tychyna Uman State Pedagogical University, Sadova str., 2, Uman, Cherkasy region, 20300, Ukraine*
[5] *ADA University, School of Information Technologies and Engineering, 61 Ahmadbey Aghaoghlu Str., Baku, AZ1008, Azerbaijan*

Abstract
The study examined the ideas, procedures, and algorithms for creating chatbots and clinic websites. The study also looked at the prospects of chatbots and whether or not they are necessary for digital resources in general. A chatbot was created without using the ChatterBot framework, and its structure and data source were fully explained. The methodologies, algorithms, and libraries used to design and train the chatbot were also described. In addition, we carried out a visual analysis, discussed the findings, and made judgments regarding the chatbot's functionality.

Keywords
chatbot, AI, DNN, deep learning, neural networks

1. Introduction

Although medicine has long been viewed as a profession based solely on face-to-face communication, it has become increasingly clear that the use of IT, in general, and electronic communications, in particular, provides not only cost-effective healthcare but also a way to ensure remote communication between medical staff and patients or their carers.

The use of the Internet in medical activities has become a reality in recent years, and telemedicine, or virtual hospitals, has moved from the stage of science fiction to reality [1]. The development of the information society, the increase in the number of people with access to computers, even the elderly (including the development of introductory computer courses for them), the diversification of information, and the need for rapid interaction have led to the adaptation of medical units to the means of information transmission to make them more attractive and more accessible to operate and use. Developing systems with guaranteed quality information should be a constant concern of public health professionals, who should work with IT specialists to create their web pages.

Many Internet sites are trying to establish themselves as sources of medical information and as a means of connecting patients with doctors. The number of visitors to these sites and the considerable number of sites indicate public interest in these services. Therefore, they need to use the available technologies to communicate effectively with patients, continue to improve them and create new ones.

The development and improvement of clinic and hospital websites should be the focus of attention for healthcare and informatics professionals. Clinic and hospital websites should ensure compliance with ethical rules, be functional and professional from an informatics point of view, and provide adequate information to potential clients of medical units. Websites are a means of communication and a convenient IT tool for better management of these structures.

tetianavakaliuk@gmail.com (T. A. Vakaliuk); dariiaskripchenko@gmail.com (D. G. Skripchenko); medvedeva-masha25@ukr.net (M. O. Medvedieva); miserablewisdom@ukr.net (M. G. Medvediev)
0000-0001-6825-4697 (T. A. Vakaliuk); 0009-0000-5540-8677 (D. G. Skripchenko); 0000-0001-9330-5185 (M. O. Medvedieva); 0000-0002-3884-1118 (M. G. Medvediev)

The study aims to selection the methods and tools for developing a chatbot for a clinic/hospital information resource.

2. Theoretical background

2.1. Analysing the need to use chatbots

Researchers have looked into how people engage with chatbots in both lab and real-world environments in order to analyze user behavior. In contrast to user interactions during chats with other users, [1] observed that user interactions with chatbots involved longer conversations with more and shorter messages, as well as lower saturation than user interactions during chats with other users. In [4] studied the real-world use of a voice conversational agent and found that the real-world context posed challenges to the conversation and led to frequent failures in the chatbot interaction. Such studies provide insights into how users behave when interacting with or responding to chatbots. As a result, they offer helpful advice for designing chatbot interactions, for instance, to reduce usability problems and enhance conversational recovery.

In [5] investigated users' motivations for using chatbots. As a result, they offer helpful advice for designing chatbot interactions, for instance, to reduce usability problems and enhance conversational recovery. Researchers [6] also looked into user preferences for various chatbot personas and discovered that people like personas that show engagement and productivity. [7] studied users' experiences with voice chatbots and found that users struggle to understand chatbot features and interact effectively with them; they suggested that a game could be a helpful entry point for learning chatbot features and interacting effectively.

Relevant insights into chatbot use and user experience can also be gleaned from the literature on dialogue system evaluation, for example, regarding frameworks and metrics for dialogue evaluation [8], which target constructs such as efficiency, effectiveness, and user satisfaction with tasks: task-oriented systems, relevance, and human-likeness for systems focused on small talk [9]. Furthermore, competitions for dialogue systems, like the Alexa Prize [10], in which groups compete to create voice chatbots capable of social interaction, greatly advance our knowledge of user preferences and views of chatbots. But further research is needed to determine how pragmatic and hedonic characteristics, taken separately or together, affect the chatbot user experience as a whole.

Therefore, lessons have been learned that apply, in particular, to task-oriented chatbots [11].

1. In task-oriented chatbots, usefulness is paramount. For task-oriented chatbot applications, solving user problems and helping users achieve their goals efficiently and effectively is critical to ensuring a quality chatbot experience. Providing valuable assistance and support is essential for sustained engagement, and for almost all chatbot applications, it is crucial to interpret user intent and provide appropriate responses correctly. Although chatbots are still a new form of interactive system, it is essential for service providers to carefully ensure that their chatbots serve their intended purposes and that these purposes are perceived as valuable by their users.
2. Hedonic attributes can improve the user experience in task-oriented chatbots. For many task-oriented chatbot applications, the user experience can be enhanced by combining pragmatic and hedonic chatbot attributes. While a very useful chatbot can provide a good user experience, this experience can be further enhanced by carefully incorporating content and chatbot characteristics perceived as pleasant, engaging, or playful.
3. User reports are valuable. Understanding how users work with chatbots is difficult. Nevertheless, understanding such experiences is crucial for chatbot service providers to increase the acceptance of chatbots among the public. With the research approach presented here, we have demonstrated the feasibility and benefits of collecting user reports using a questionnaire based on critical incident techniques. We hope this will serve as an example of how service providers can approach the collection of high-quality user reports that provide much-needed rich information about the chatbot experience.
4. Different users have different needs. The natural language interaction in chatbots makes them very suitable for personalisation. An example is our finding that pragmatic and hedonic chatbot attributes have different meanings for chatbot users of varying age

groups. However, the ability to personalise chatbots has not been sufficiently implemented. On the contrary, current chatbots typically display the same personality and provide the exact content regardless of user characteristics. Chatbot service providers can benefit from investing in understanding the needs of their different user groups and configuring chatbots that can adapt accordingly.

2.2. Concepts, methods and algorithms of chatbot development

A chatbot is a computer program developed based on neural networks and machine learning technologies that allow communication in audio or text format. A chatbot performs specific tasks (e.g., obtaining reference information, performing calculations) or for entertainment.

Typically, a chatbot prompts the user to choose an option from a list of options. That is, obtaining information or performing other tasks is done by selecting the suggested answers or categories, and the user does not enter text into the chat window. More sophisticated bots can enter text and receive information per the user's request.

Virtual chatbots allow you to place orders or communicate online.

Chatbots can be divided into the following areas of application:

- p2p - personal communications (for personal communication);
- b2c - consumer (support for company customers on the corporate website and in mobile applications).

Chatbots mainly use artificial intelligence to communicate with users, providing relevant content and up-to-date offers. They operate based on a set of instructions or use machine learning.

Some of the main concepts associated with chatbot technology are as follows [12]:

- Pattern matching, which is based on typical stimulus-response blocks. A sentence (stimulus) is entered, and a response (reaction) is generated according to the data entered by the user.
- AIML (Artificial Intelligence Markup Language) is based on Pattern recognition or Pattern matching concepts.
- Latent semantic analysis can be used with AIML to develop chatbots. It is used to identify similarities between words as a vector representation.
- Chatscript, the successor to AIML, is an expert system consisting of an open-source scripting language and an engine that runs it. Chatscript includes rules associated with topics, finding the best element that matches the user's query string, and executing the rule in that topic.
- RiveScript is a plain-text string scripting language for developing chatbots and other conversational objects. RiveScript is open source with interfaces for Go, Java, JavaScript, Perl, and Python.
- Natural language processing (NLP) is a field of artificial intelligence that studies the computer-assisted manipulation of texts or natural language speech.
- Natural Language Understanding (NLU) is the foundation of any NLP task. It is a technique for implementing natural user interfaces such as chatbots. NLU seeks to extract context and meaning from user input, which may be unstructured and respond appropriately to the user's intentions [1].

A bot's intelligence level depends solely on how it is programmed. A chatbot based on machine learning performs better because it understands commands and language. Therefore, users do not need to enter exact words to get relevant answers. In addition, the bot learns from interactions with customers and can quickly solve similar situations when they arise. Each such dialogue improves the bot's intelligence [12].

Chatbots can use neural networks as a language model [13]. For example, generative pre-trained transformers (GPTs), which use a transformer architecture, have become standard for building sophisticated chatbots. The "pre-training" in its name refers to the process of initial training on a large text corpus that provides a solid foundation for the model to perform well on subsequent tasks with a limited amount of task-related data. An example of a GPT chatbot is ChatGPT. Despite criticism about its accuracy, ChatGPT has attracted attention for its detailed answers and historical knowledge. Another example is BioGPT, developed by Microsoft, which focuses on answering biomedical questions.

Here are the most common limitations [14]:

- Since the I/O database is fixed and limited, chatbots can crash while processing an unsaved query.
- Chatbot performance heavily depends on speech processing and is limited by disturbances such as accents and spelling or grammatical errors.
- Chatbots cannot answer several questions simultaneously, limiting their communication capabilities.
- Chatbots require a large amount of conversational data to train them. Based on deep learning algorithms to create new answers word by word based on user input, generative models are usually trained on a large dataset of natural language phrases.
- It is difficult for chatbots to manage non-linear conversations that need to flow naturally and be relevant to the topic of the conversation with the user.
- As is usually the case with technological changes to existing services, some consumers, most often older ones, are uncomfortable with chatbots due to their limited understanding, which makes it evident that machines are processing their queries.

3. Proposed methodology.

Before you start working on a chatbot for a ready-made web resource, you should first understand the basic algorithms and concepts of chatbot technology development using Python.

It was decided that the chatbot would be based on a DNN (Deep Neural Network) to identify sentence patterns provided by the user as input and select a random answer related to this query. The chatbot will be implemented using Python libraries, such as the NLTK library in Python, which has functions that help identify the most relevant words in a sentence or paragraph, link them to their root meaning, and shorten them. This process is known as Stemming. The words are then converted to their corresponding numerical values, as neural networks only understand numbers. The process of converting text to numeric values is known as One-Hot Encoding. Once the data preprocessing is complete, we will create the neural networks using TFlearn and feed them with training data. After successful training, the model should be able to predict tags related to the user's query.

To train our chatbot, we need the "intents.json" file. Each intent contains a tag, templates, answers, and context. The templates are the data the user will likely enter, and the answers are the chatbot's results. This file contains general greeting phrases, questions about which doctors work in the clinic, and questions about the appointment, such as the appointment cost, which days you can make an appointment, and how you can make an appointment.

The challenge now is to get the model to learn the relationship between patterns and tags so that when a user enters a particular phrase, it can identify the corresponding tag and produce one of the answers resulting from the query.

The first step is to download the Punkt model provided by the NLTK library. In NLTK, PUNKT is an unsupervised learning model, meaning it can be trained on unlabeled data (data that has not been annotated with information about characteristics, properties, or categories is called unlabeled data).

Using unsupervised techniques, it generates a list of sentences from the text by developing a model for sentence starters, prepositional phrases, and contractions. It must be trained on a significant amount of open text in the target language without prior use.

Next, look at the libraries and modules that will be used to implement a chatbot. The NLTK supports research and teaching in NLP and related fields, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK is successfully used as a training tool and a platform for prototyping and building research systems. When working on any project in natural language processing, NLTK is the most critical module used.

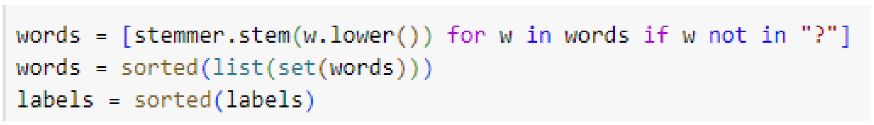
TensorFlow is an open-source machine learning software library for a range of tasks developed by Google to meet its need for systems that can build and train neural networks to detect and decipher patterns and correlations, similar to the learning and understanding used by humans. It is currently used for both research and product development at Google.

TensorFlow provides a library of ready-made algorithms for numerical computing implemented through data flow graphs. The nodes in such graphs execute mathematical operations or input/output points, while the graph's edges represent multidimensional data arrays (tensors) that flow between the nodes. Nodes can be assigned to computing devices and run asynchronously, processing all the parallel tensors that fit them together. This allows for the simultaneous operation of nodes in a neural network by analogy with the simultaneous activation of neurons in the brain.

However, TensorFlow is a low-level library. Using TensorFlow directly is challenging, as the API is highly verbose and prone to subtle actions. It is difficult to detect errors in TensorFlow. To overcome these difficulties, a shell called TFLearn was developed.

TFlearn is a modular and transparent deep-learning library built on top of TensorFlow. It was developed to provide a higher-level API for the TensorFlow library to make experiments easier and faster while remaining fully transparent and compatible.

After pre-loading the "intents.json" file and receiving the data, it is necessary to start pre-processing it, as shown in Fig. 1.



```
words = [stemmer.stem(w.lower()) for w in words if w not in "?"]
words = sorted(list(set(words)))
labels = sorted(labels)
```

Figure 1: Preliminary data processing

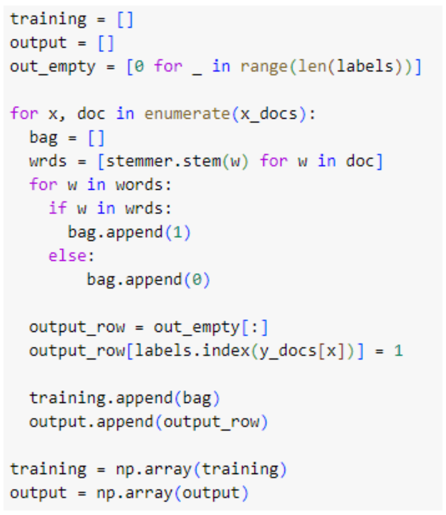
This recognises words and removes duplicate words from the word list. For this task use the Lancaster Stemmer algorithm to reduce words to their base (Figure 2). First, take a quick look at what stemming is. Stemming is the process of shortening a word to its base. A word stem is a word's base or root form and does not have to be an existing word.

Lancaster is one of the most aggressive stemmers because it tends to overstep many words. A Lancaster stemmer consists of a set of rules, where each rule specifies either the deletion or replacement of an ending. In addition, some rules are restricted to intact words, and some rules are applied iteratively as the word passes through them.

Because of the strict rules, there are two additional conditions to prevent different short-root words from occurring:

- If the word starts with a vowel, at least two letters must remain after the root (owing -> ow, but not ear -> e).
- If the word begins with a consonant, at least three letters must remain after the root letter, and at least one of them must be a vowel or "y" (speaking -> say, but not string -> str).

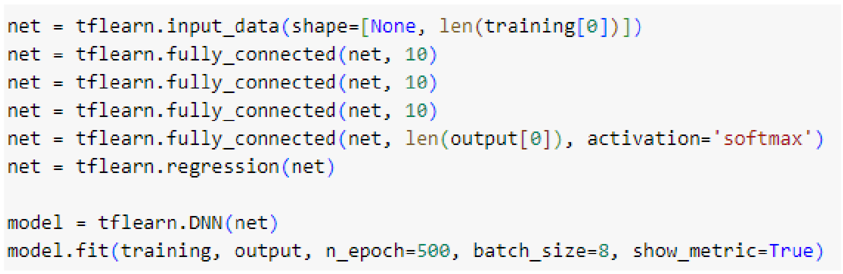
The Lancaster header stemmer has over 100 rules.



```
training = []
output = []
out_empty = [0 for _ in range(len(labels))]

for x, doc in enumerate(x_docs):
  bag = []
  wrds = [stemmer.stem(w) for w in doc]
  for w in words:
    if w in wrds:
      bag.append(1)
    else:
        bag.append(0)

  output_row = out_empty[:]
  output_row[labels.index(y_docs[x])] = 1

  training.append(bag)
  output.append(output_row)

training = np.array(training)
output = np.array(output)
```

Figure 2: One-Hot coding and training data preparation.

The resulting training and output data are encoded in One-Hot mode. The words are converted into ones and zeros, which are then added to the training list and the output list and then converted into NumPy arrays. We need to create our model using neural networks (Figure 3).



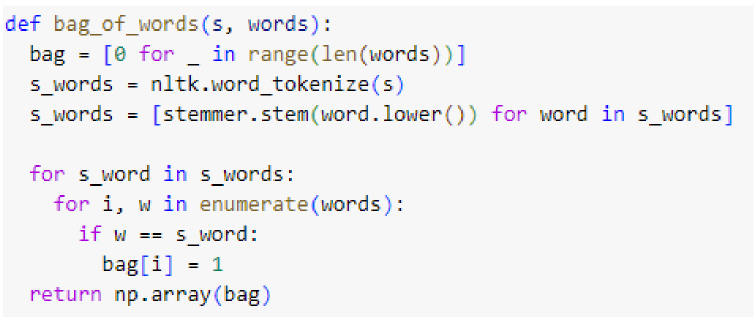
```
net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, 10)
net = tflearn.fully_connected(net, len(output[0]), activation='softmax')
net = tflearn.regression(net)

model = tflearn.DNN(net)
model.fit(training, output, n_epoch=500, batch_size=8, show_metric=True)
```

Figure 3: Creating a model using neural networks.

The first layer is the input layer, with a parameter of equally sized input data. The following three middle layers are the hidden layers responsible for all input processing. The output layer provides the probabilities of different words in the training data.

The training data is fitted into the model, and epochs are set to 500, during which the training will continue until it reaches 500 iterations. An epoch in a neural network trains the neural network with all the training data in one cycle. Using more than one epoch is necessary to get good performance from the non-training data (in practice, this can be roughly estimated using a set of delays); usually (but not always), this takes more than one pass of training data.

We have trained the model with a list of words, or a so-called bag of words, so we also need to do the same to make predictions. We can now create a function that will give us a bag of words to predict for our model (Figure 4).

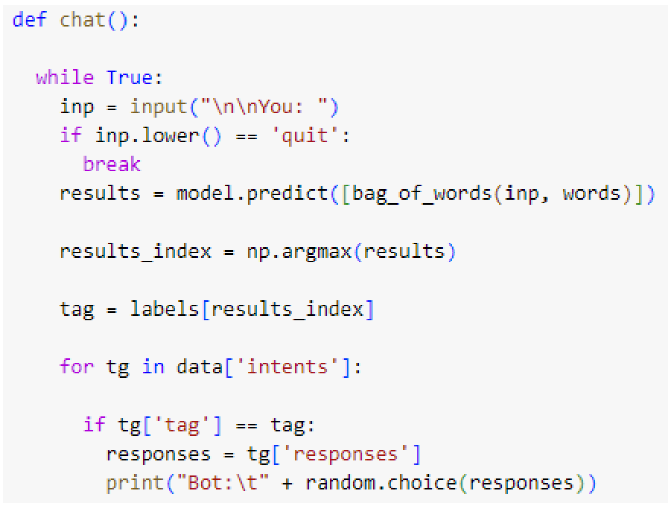


```
def bag_of_words(s, words):
  bag = [0 for _ in range(len(words))]
  s_words = nltk.word_tokenize(s)
  s_words = [stemmer.stem(word.lower()) for word in s_words]

  for s_word in s_words:
    for i, w in enumerate(words):
      if w == s_word:
        bag[i] = 1
  return np.array(bag)
```

Figure 4: Function to provide a set of words for model prediction.

This function helps to create a word pack for our model.

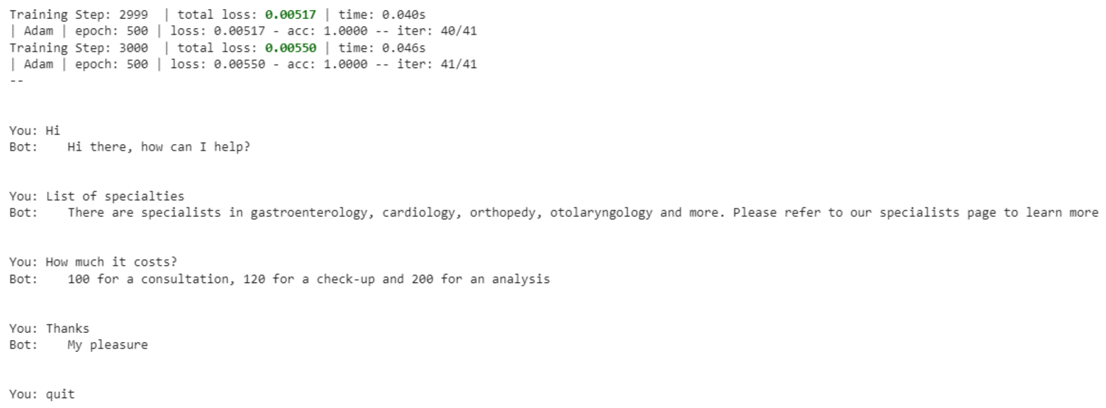
Now we need to create a chat function that brings all of this together (Figure 5).



```
def chat():

  while True:
    inp = input("\n\nYou: ")
    if inp.lower() == 'quit':
      break
    results = model.predict([bag_of_words(inp, words)])

    results_index = np.argmax(results)

    tag = labels[results_index]

    for tg in data['intents']:

      if tg['tag'] == tag:
        responses = tg['responses']
        print("Bot:\t" + random.choice(responses))
```

Figure 5: Chat function.

First, the model predicts outcomes using the word packet and user data and then returns a list of probabilities. Among the probabilities, the highest number is likely to be the result the user expects. So, we choose the index of the highest probability and find the tag and answers of that particular index. Then, we can select some random answers from the list of answers. Fig. 6 shows how a chatbot can identify a user input pattern and respond accordingly.



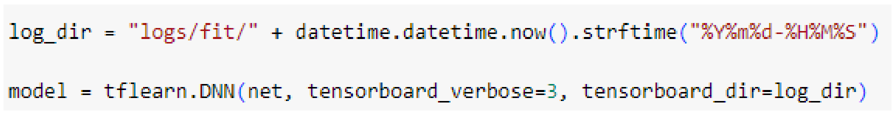
```
Training Step: 2999  | total loss: 0.00517 | time: 0.040s
| Adam | epoch: 500 | loss: 0.00517 - acc: 1.0000 -- iter: 40/41
Training Step: 3000  | total loss: 0.00550 | time: 0.046s
| Adam | epoch: 500 | loss: 0.00550 - acc: 1.0000 -- iter: 41/41
--


You: Hi
Bot:    Hi there, how can I help?


You: List of specialties
Bot:    There are specialists in gastroenterology, cardiology, orthopedy, otolaryngology and more. Please refer to our specialists page to learn more


You: How much it costs?
Bot:    100 for a consultation, 120 for a check-up and 200 for an analysis


You: Thanks
Bot:    My pleasure


You: quit
```

Figure 6: An example of how a chatbot works.

We can see that the chatbot provides correct answers to user queries after the training. Typing in common phrases results in the chatbot answering the question, and typing in quit results in the chatbot terminating the conversation.

3.1. Analysis of chatbot performance and its results in charts and graphs

TensorBoard was used to visually analyse the chatbot. TensorBoard is a tool for providing measurements and visualisations required during the machine learning workflow. It allows you to track experiment metrics such as loss and accuracy, visualise the model graph, plot embeddings in a lower dimensional space, and much more. First, you must update the code, as shown in Figure 7.



```
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

model = tflearn.DNN(net, tensorboard_verbose=3, tensorboard_dir=log_dir)
```

Figure 7: Listing of changes in the code.

TFLearn supports a detailed level for automatic management of the results:
0: Losses and performance (best speed).
1: Loss, metrics and gradients.
2: Loss, metrics, gradients and weights.
3: Loss, metrics, gradients, weights, activations, and sparsity (best visualisation).
For the tensoboard_verbose property, the value three was chosen for the best visualisation.

Accordingly, the logs will be stored in subdirectories with a time stamp so that you can easily select different training runs. To do this, create a log_dir variable that stores the subdirectory's name.

You can then launch Tensorboard to visualise the performance.

The TensorBoard Graphs dashboard is a tool for exploring your TensorFlow model. You can quickly view a conceptual graph of the model's structure to ensure it matches your intended design. You can also view the operational level graph to understand how TensorFlow understands the application. Examining the operational level graph can give you insight into how to modify your model. This is the tool that will be used for visual analysis.

As a result, we get the following graphs of accuracy and training losses, shown in Figs. 8-9.

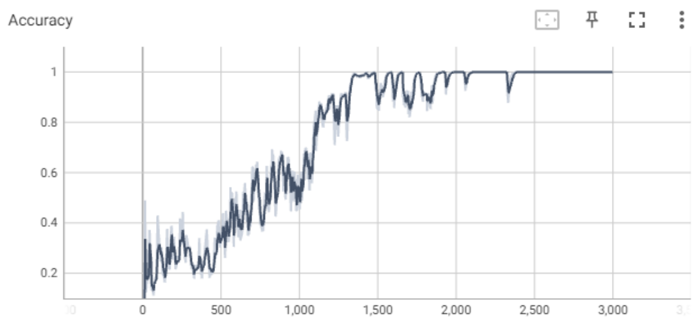


Figure 8: Accuracy chart.

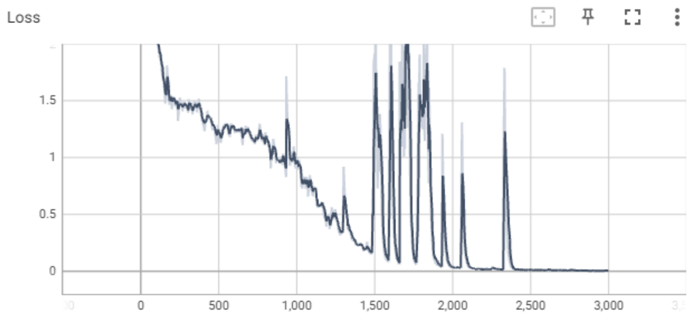


Figure 9: Schedule of losses during training.

In most cases, accuracy increases as loss decreases, but this is not always true. Accuracy and loss have different definitions and measures. They often appear to be inversely proportional, but the two have no mathematical relationship.

Low precision but high loss means the model makes significant errors in most data. However, if both loss and accuracy are lacking, the model makes minor errors in most data. However, if both are high, it results in significant errors in some data. Finally, if precision is high and loss is low, the model makes minor errors in only some data, which would be ideal.

Based on the data obtained, we can conclude that the chatbot learns quickly from the data provided and makes a small number of errors, indicating the low complexity of the data provided.

4. Results

An untrained ChatterBot instance starts without knowing how to communicate. Each time a user enters a statement, the library stores the text they entered and the text to which the statement was a response. As ChatterBot receives more data, the number of responses it can respond to and the accuracy of each response about the input statement increase.

The program selects the most appropriate answer by looking for the closest matching known statement that reaches the input and then choosing an answer from the list of available responses to that statement (Figure 10).

Taking into account the process of the frame, we first initialise the ChatBot instance as shown in Fig. 11.

ChatterBot preprocessors are simple functions that modify the input statement that the chatbot receives before processing the statement by the logic adapter.

Logic adapters define the logic of how ChatterBot chooses a response to a given input statement.

A typical Boolean adapter will use two main steps to return a response to an input statement. The first step involves searching the database for a known statement that matches or closely matches the input statement. Once a match is selected, the second step consists of establishing a general answer to the chosen match. Often, several existing statements will answer an available game, and the BestMatch logic adapter will determine the answer based on the best-known match with the given statement.

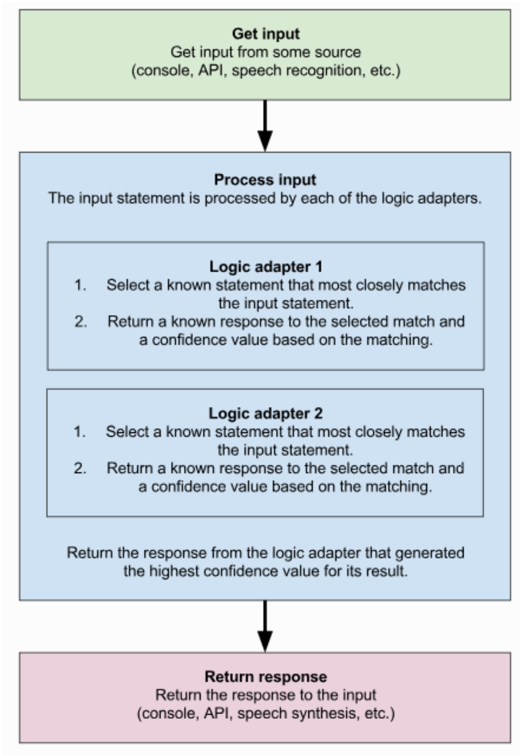
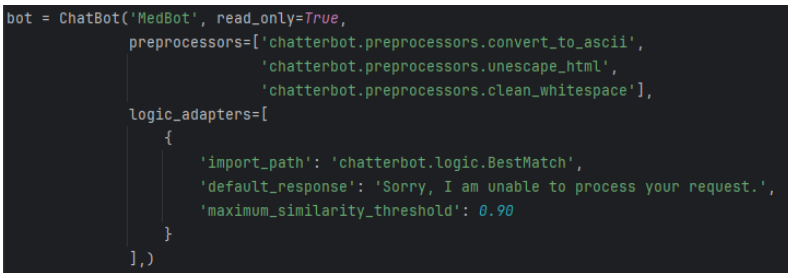

Get input
Get input from some source
(console, API, speech recognition, etc.)
Process input
The input statement is processed by each of the logic adapters.
Logic adapter 1
1. Select a known statement that most closely matches the input statement.
2. Return a known response to the selected match and a confidence value based on the matching.
Logic adapter 2
1. Select a known statement that most closely matches the input statement.
2. Return a known response to the selected match and a confidence value based on the matching.
Return the response from the logic adapter that generated the highest confidence value for its result.
Return response
Return the response to the input
(console, API, speech synthesis, etc.)

Figure 10: Diagram of the ChatterBot workflow.

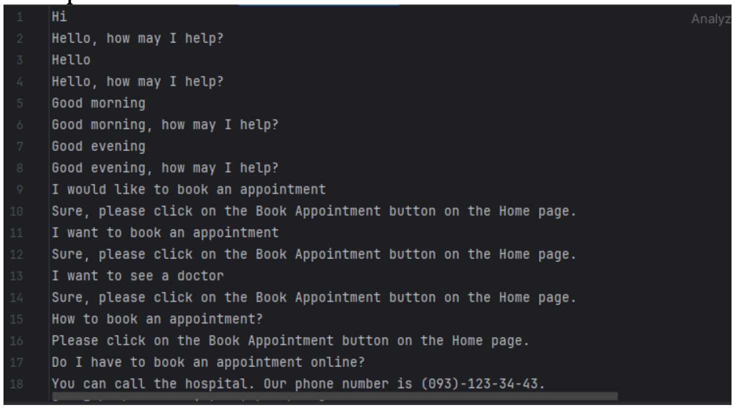


```
bot = ChatBot('MedBot', read_only=True,
              preprocessors=['chatterbot.preprocessors.convert_to_ascii',
                             'chatterbot.preprocessors.unescape_html',
                             'chatterbot.preprocessors.clean_whitespace'],
              logic_adapters=[
                  {
                      'import_path': 'chatterbot.logic.BestMatch',
                      'default_response': 'Sorry, I am unable to process your request.',
                      'maximum_similarity_threshold': 0.90
                  }
              ],)
```

Figure 11: Initialising a ChatBot instance.

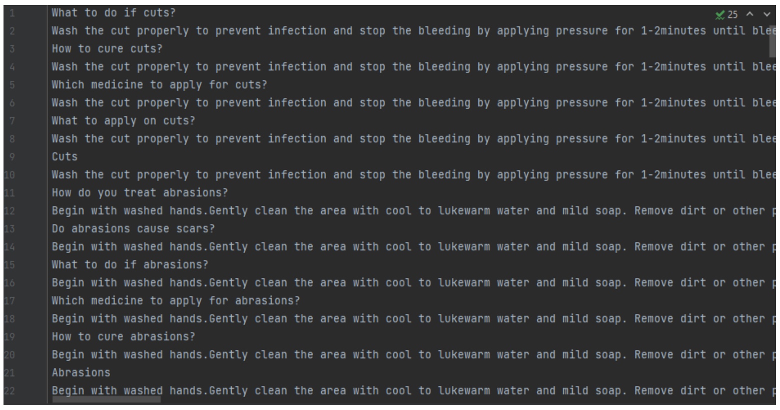
The best match adapter uses a function to compare the input statement with known statements. When it finds the closest match to the input operator, it uses another function to select one of the available answers to that operator.

We need to train the bot with training data to provide the bot with some knowledge. The training data is filled in a list that will represent the conversation. ChatterBot uses natural language processing to learn the data provided to the bot. Fig. 12 shows a listing of the training data file with site navigation, and Fig. 13 lists the training data file for delivering essential medical advice for minor medical problems.



```
Hi
Hello, how may I help?
Hello
Hello, how may I help?
Good morning
Good morning, how may I help?
Good evening
Good evening, how may I help?
I would like to book an appointment
Sure, please click on the Book Appointment button on the Home page.
I want to book an appointment
Sure, please click on the Book Appointment button on the Home page.
I want to see a doctor
Sure, please click on the Book Appointment button on the Home page.
How to book an appointment?
Please click on the Book Appointment button on the Home page.
Do I have to book an appointment online?
You can call the hospital. Our phone number is (093)-123-34-43.
```

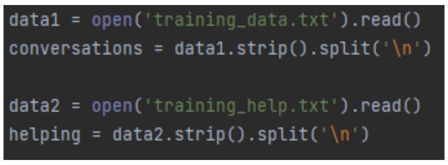
Figure 12: Listing a file with training data with site navigation.



```
What to do if cuts?
Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blee
How to cure cuts?
Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blee
Which medicine to apply for cuts?
Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blee
What to apply on cuts?
Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blee
Cuts
Wash the cut properly to prevent infection and stop the bleeding by applying pressure for 1-2minutes until blee
How do you treat abrasions?
Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
Do abrasions cause scars?
Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
What to do if abrasions?
Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
Which medicine to apply for abrasions?
Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
How to cure abrasions?
Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
Abrasions
Begin with washed hands.Gently clean the area with cool to lukewarm water and mild soap. Remove dirt or other p
```

Figure 13: Listing of a file with training data on giving basic health advice.

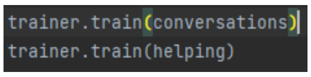
Further, these data should be processed as shown in Fig. 14.



```python
data1 = open('training_data.txt').read()
conversations = data1.strip().split('\n')

data2 = open('training_help.txt').read()
helping = data2.strip().split('\n')
```

Figure 14: Data processing.

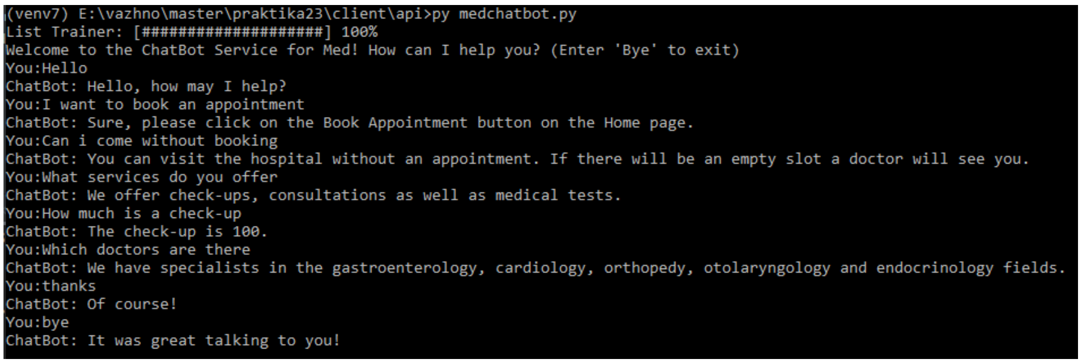
After that, we use the processed data to train the bot, as shown in Fig. 15.



```python
trainer.train(conversations)
trainer.train(helping)
```

Figure 15: Training a chatbot with processed data.

Now the chatbot is ready to use. Fig. 16 shows an example of how a chatbot works.



```
(venv7) E:\vazhno\master\praktika23\client\api>py medchatbot.py
List Trainer: [####################] 100%
Welcome to the ChatBot Service for Med! How can I help you? (Enter 'Bye' to exit)
You:Hello
ChatBot: Hello, how may I help?
You:I want to book an appointment
ChatBot: Sure, please click on the Book Appointment button on the Home page.
You:Can i come without booking
ChatBot: You can visit the hospital without an appointment. If there will be an empty slot a doctor will see you.
You:What services do you offer
ChatBot: We offer check-ups, consultations as well as medical tests.
You:How much is a check-up
ChatBot: The check-up is 100.
You:Which doctors are there
ChatBot: We have specialists in the gastroenterology, cardiology, orthopedy, otolaryngology and endocrinology fields.
You:thanks
ChatBot: Of course!
You:bye
ChatBot: It was great talking to you!
```

Figure 16: An example of how a chatbot works.

The client side of the site was developed using React. The main page of the site looks like this, as shown in Fig. 17.

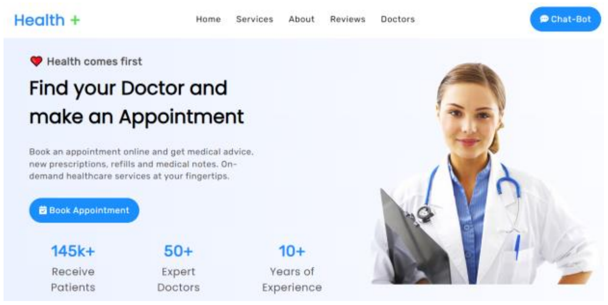

Health +
Home
Services
About
Reviews
Doctors
Chat-Bot
Health comes first
Find your Doctor and make an Appointment
Book an appointment online and get medical advice, new prescriptions, refills and medical notes. On-demand healthcare services at your fingertips.
Book Appointment
145k+
Receive Patients
50+
Expert Doctors
10+
Years of Experience

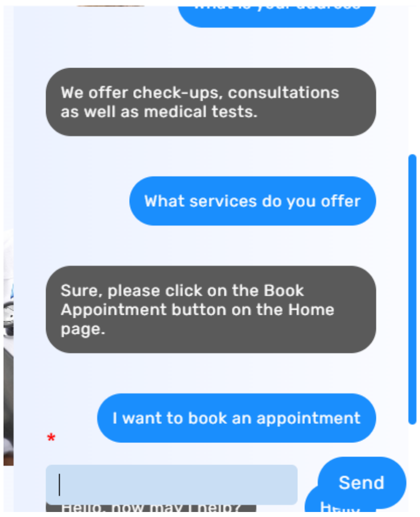
Figure 17: The main page of the site.

The main menu contains links to other sections of the website, which the user can access by clicking on. For example, clicking on the About link opens the section with information about the clinic.

When you click the Book Appointment button on the main page, a form for leaving a request for an appointment with a doctor opens.

The chatbot connects to a web resource using the Flask library on the backend, which was developed in Python, and the fetch method on the client side was developed with React.

The chatbot is available to users by clicking the Chatbot button in the main menu—an example of how the chatbot works is shown in Fig. 18.



We offer check-ups, consultations as well as medical tests.

What services do you offer

Sure, please click on the Book Appointment button on the Home page.

I want to book an appointment

Figure 18: An example of how a chatbot works.

5. Discussion

The developed chatbot communicates exclusively in English. Therefore, the disadvantages include localisation in one language.

This is not critical, as English is an international language, firstly, and secondly, there is room for improvement and the addition of other functions. To further add other languages of communication, it will be necessary to train the proposed models in other languages.

The question of how much a chatbot can provide a correct and adequate answer to a user is also debatable. For this purpose, it is worth conducting an experiment on the quality of communication between a healthcare professional as a user and a chatbot. As a result, it would be possible to identify the existing shortcomings that need to be eliminated.

6. Conclusions

The study examined the concepts, methods, and algorithms for developing websites for clinics and chatbots. The study also examined the necessity of using chatbots for general web resources and their future prospects.

A chatbot was developed without using the ChatterBot framework. A full description of the data source used to train the chatbot was provided, and its structure and the implemented methods, algorithms, and libraries used to develop and train the chatbot were presented and described. We also conducted a visual analysis, reported its results, and drew conclusions about the chatbot's performance. After that, a chatbot was developed using the ChatterBot framework, which will be used for the finished web resource. The implemented methods and the process of developing and training a chatbot were described, and an example of its operation was presented.

In addition, the article demonstrates how the developed chatbot was integrated with the developed web resource and demonstrates the operation of the chatbot. As a result, the article studies the concepts and technologies of chatbot development, develops its chatbot connected to the developed web resource, and explores its necessity and relevance.

Since this chatbot was developed specifically for a hospital website, it is distinguished by training models for medical terminology. According to the analysis of existing chatbots, this is rare in this sector.

Further research includes experimental testing of the proposed chatbot in medical institutions and its expert evaluation. The possibility of adding several languages for communication is also among the prospects for further research.

References

[1] Nikitchuk, T.M., Andreiev, O.V., Korenivska, O.L. and Medvediev, M.G., 2023. A model of an automated biotechnical system for analysing pulseograms as a kind of edge device. *Journal of Edge Computing* [Online], 2(1), pp. 64–83. Available from: https://doi.org/10.55056/jec.627

[2] Korenivska, O.L., Benedytskyi, V.B., Andreiev, O.V. and Medvediev, M.G., 2023. A system for monitoring the microclimate parameters of premises based on the Internet of Things and edge devices. *Journal of Edge Computing* [Online], 2(2), pp. 125–147. Available from: https://doi.org/10.55056/jec.614

[3] Escobar-Grisales, D., Vásquez-Correa, J.C. & Orozco-Arroyave, J.R. Evaluation of effectiveness in conversations between humans and chatbots using parallel convolutional neural networks with multiple temporal resolutions. Multimed Tools Appl 83 (2024) 5473–5492. doi: 10.1007/s11042-023-14896-y

[4] Kendall, L., Chaudhuri, B. & Bhalla, A. Understanding Technology as Situated Practice: Everyday use of Voice User Interfaces Among Diverse Groups of Users in Urban India. Inf Syst Front 22 (2020) 585–605. doi: 10.1007/s10796-020-10015-6

[5] Jindi Fu, Samar Mouakket, Yuan Sun. The role of chatbots' human-like characteristics in online shopping, Electronic Commerce Research and Applications, Volume 61 (2023) 101304. doi: 10.1016/j.elerap.2023.101304.

[6] Brinda Mehra. Chatbot personality preferences in Global South urban English speakers, Social Sciences & Humanities Open, 3:1 (2021) 100131. doi: 10.1016/j.ssaho.2021.100131.

[7] Rzepka, C., Berger, B. & Hess, T. Voice Assistant vs. Chatbot – Examining the Fit Between Conversational Agents' Interaction Modalities and Information Search Tasks. Inf Syst Front 24 (2022) 839–856. doi: 10.1007/s10796-021-10226-5

[8] Haoming Jiang, Bo Dai, Mengjiao Yang, Tuo Zhao, and Wei Wei. Towards Automatic Evaluation of Dialog Systems: A Model-Free Off-Policy Evaluation Approach. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 7419–7451.

[9] J. Deriu, A. Rodrigo, A. Otegi, et al., Survey on evaluation methods for dialogue systems, Artificial Intelligence Review 54 (2021) 755–810. doi: 10.1007/s10462-020-09866-x.

[10] Sarah E. Finch, Jinho D. Choi. Towards Unified Dialogue System Evaluation: A Comprehensive Analysis of Current Evaluation Protocols. In: Proceedings of the SIGdial 2020 Conference. 2020, pp. 236-245. https://www.sigdial.org/files/workshops/conference21/pdf/2020.sigdial-1.29.pdf

[11] A. Følstad, P. B. Brandtzaeg, Users' experiences with chatbots: findings from a questionnaire study, Qual User Exp 5, 3 (2020). Available online: https://link.springer.com/article/10.1007/s41233-020-00033-2#Sec26

[12] Adamopoulou E, Moussiades L. An Overview of Chatbot Technology. Artificial Intelligence Applications and Innovations. 6;58 (2020) 373–383. doi: 10.1007/978-3-030-49186-4_31.

[13] Kumar, Ramakrishna & Mahmoud, Maha. A Review on Chatbot Design and Implementation Techniques. International Research Journal of Engineering Science Technology and Innovation 07 (2020) 2791.

[14] Sarker, Iqbal. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. SN Computer Science 2 (2021). doi: 10.1007/s42979-021-00815-1.