

Development of the Speech-to-Text Chatbot Interface Based on Google API

Nataliya Shakhovska^[0000-0002-6875-8534], Oleh Basystiuk^[0000-0003-0064-6584],
Khrystyna Shakhovska^[0000-0002-9914-229X]

Lviv Polytechnic National University, Lviv 79013, Ukraine
nataliya.b.shakhovska@lpnu.ua, obasystiuk@gmail.com,
kristin.shakhovska@gmail.com

Abstract. The paper describes possibilities, which are provided by open APIs, and how to use them for creating unified interfaces using the example of our bot based on Google API. In last decade AI technologies became widespread and easy to implement and use. One of the most perspective technology in the AI field is speech recognition as part of natural language processing. New speech recognition technologies and methods will become a central part of future life because they save a lot of communication time, replacing common texting with voice/audio. In addition, this paper explores the advantages and disadvantages of well-known chatbots. The method of their improvement is built. The algorithms of Rabin-Karp and Knut-Pratt are used. The time complexity of proposed algorithm is compared with existed one.

Keywords: natural language processing, speech-to-text, Google API, Python, Flask, chatbot, hashing, time complexity, prefix-function

1 Introduction

Today, the creation of programs simulating human communication remains relevant. The simplest model of communication is the database of questions and answers to them [1]. In this case, there is the problem of describing the knowledge base and the implementation of the interpreter program. The markup language of the knowledge base can include question patterns and corresponding response patterns, as well as the background history of the dialogues to them and the name of the corresponding topic of communication.

Chat bot can perform additional functions, such as music search, pictures, facts, calculator, weather forecast, display of exchange rates. Most of these functions have an implementation on the Internet and are available as an external API. The aim of the paper is the recognition of user's gender for making chatbot's answer more likeness that is human. The algorithm for analyzing and parsing the user's text for automatically generating the response of the chatbot is developed. This algorithm takes into account the topics of correspondence and morphology of the text. The algorithm's work will be based on prefix function and hash function. To add, a comparison of the developed algorithm with the existing ones will be made. This research will describe the way in which was created an interface for Telegram chatbot (in future easily connect it to

Slack, Facebook, etc.), whose main aim is to translate audio (in the future video) messages into text.

2 State of art

The chatbots market after the stage of PR and experiments moved to the stage of technology race and measuring the effectiveness of the tool. The concept of "chat bot" is replaced by a larger and more complete. Now it is a conversational artificial intelligence, Conversational AI. Companies use it to communicate with customers on websites, instant messengers, mobile applications and smart devices.

In order to "understand" the speech, the chat bot compares what has been said with the phrases of a large number of other people in whom the bot has been trained. It finds similarities with the phrase patterns, determines the topic of the question, and performs the action programmed for this type of question.

A person gets the illusion that the bot understands him: he acts logically, reacts like a person and maintains a conversation. The famous Turing test is based on this illusion: if the judge could not determine if the bot communicates with him or the person, the technology passes the test.

Understanding is a subjective "value" that is difficult to measure. Microsoft conducted a study of the effectiveness of the speech recognition system and the system of answers to questions on a given set of tests - in 2017-2018 both were more effective than people, passing the test with a minimum of errors.

The main technology of modern bots is natural language understanding (NLU). It allows the machine to understand users and run the necessary parameters for processing requests. All technological solutions related to the processing of natural language are needed for this.

These include the approach to processing (rule-based, statistical, hybrid), speech recognition and speech synthesis technologies, chatbot deployment technologies in the company's business processes (cloud or local). This is a huge and very promising market, which experts predict growth to \$ 16 billion by 2021.

Machine learning technologies that are the most competitive in the market are based on the neural networks principle, when a chatbot is trained in response samples. He is shown examples of customer phrases, and he learns to put such phrases and words that are similar to them in the right class. The more efficient the algorithm, the fewer examples you need to train a bot.

Speech recognition and speech synthesis technologies, natural language understanding technologies, machine learning algorithms are behind the "conversational spirit". The company Just AI is developing the designer of conversational bots Aimylogic. The best online artificial intelligence (AI) online chats are Mitsuku, Rose, Poncho, Right Click, Insomno Bot, Dr. AI and Melody.

1. Mitsuku is one of the best bots in AI [2]. Also, is the current Laurel Laureate of Loebner. This bot can talk about something, unlike other ones, done for a specific task.
2. Rose. The ChatBot, which won the Loebner Prize in 2014 and 2015 [3].

3. RightClick. Launches a program that creates websites. He asks general questions during the conversation: "What is the area of your interests?" and "Why do you want to create a website?" Based on the analysis of the replies received, the chatbot creates custom templates. Quite adequately responsive to non-site-related topics [4].
4. Poncho is a meteorological specialist. He sends notifications at least twice a day with the user's consent and is smart enough to answer such questions as "Should I take an umbrella today?" [5].
5. Insomno Bot is designed for "night owls". This applies to all people with sleep problems. This bot has the ability to keep conversation on any topic [6].
6. Dr. AI bot asks for symptoms, body parameters and disease history then lists the most and least likely causes of the symptoms and sorts them in order of severity [7].
7. Baidu melody. This application collects medical information about people and then passes it to doctors in a form that facilitates their use for diagnostic purposes [8].

Nowadays, Speech-to-text methods and systems have been wide-spread to solve a variety of tasks. Highly used in the artificial assistants' field as a method to understand users' desires. As a lot of these speech-to-text technologies is nowadays represented in different APIs and frameworks, that means creating any new software for testing or production based on these technologies become a really easy task, as each of famous API provides samples [<https://cloud.google.com/speech-to-text/docs/>].

Today, based on the classic Speech-to-text methods, there are online solutions such as APIs and libraries [15], where you can push information and in real time receive a response. However, the classical neural network will provide more flexibility, but it required prepared data for learning, in this case, it's possible to use open source audio to text data [16], but collect own data, helps to understand users better, what kind of requests they prefer, long or short, that themes, which words [17]. All information received by this system based on Google API will provide answers to all these questions.

The natural language processing systems are carried out mainly on the analysis of time series (data arrays). The time series include methods of nonlinear analysis and unsupervised learning. Highly used in this field are RNN (Recurrent Neural Networks), HMM (Hidden Markov Model), Bayesian network. RNN became a sophisticated solution for all kind of translation works because it's made it possible to work with different input and output length, you need to use a recurrent neural network [17].

The papers [16 - 20] emphasizes the need to protect personal data. Security reasons are well described in papers [20], as data need to be stored in the database, which will help clearly understand and customize recognition algorithm.

This research will describe the way in which was created an interface for Telegram chatbot (in future easily connect it to Slack, Facebook, etc.), whose main aim is to translate audio (in the future video) messages into text.

3 The system architecture

3.1 Data collection

Audio and video data collection process consists of two main steps. The first one is the remote storage of social network. Users will provide access to the data, so the main task for this step is using audio or video data fetching, for our local storage. There are two reasons, why this data needs to be fetched for the local storage. First, because we need to upload files directly into the Google Speech-to-Text API. Second, because it'll be useful for statistics of requests, tracking wrong and correct answers, and in future as the database for learning own specific neural network, based on collected data.

The decision to use Google Speech-to-Text API was made because Google provides clear documentation, with great examples of API features and they fully satisfied request of 100% uptime, reliability and has a lot of supporting languages, so it helps to make the product interesting for more people.



Fig. 1. The simplified representation of created system

Based on the figure above, it's become clear that software work as an interface to the speech-to-text recognition system, so in this case we have some specific design requirements. There are two main ideas of how bot for the social network can works.

First, in general, is called polling, that's mean like ones in a small period, bot requests to the social network server, asking about if there are any new users requests to him. This design solution has main pros that bot creation, based on polling system it's easier, you don't need have fully created a web app, which will handle HTTP requests and main cons that this method suitable for a small app, which won't receive a significant amount of messages.

Second - web hook. That's mean, a web application is hosting and social network servers, redirect all requests to this address, this type of bot become more independent, less overloaded and handle requests faster. Visual representation of how both design methods work on Fig 2.

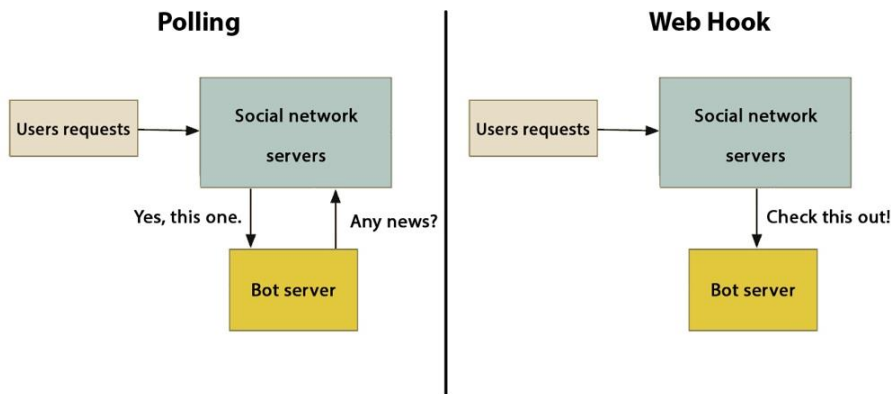


Fig. 2. The simplified representation of webhook and polling technologies

3.2 Proposed method

The prefix function [9, 10] from the string S and the position i in it is the length k of the largest prefix of the substring $S[1..i]$, which is also a suffix of this substring. That is, at the beginning of the substring $S[1..i]$ of length i you need to find such a prefix of maximum length $k < i$, which would be the suffix of the given substring ($S[1..k] == S[(i-k+1)..i]$). For line S it is convenient to represent a prefix function in the form of a vector of length $|S|-1$. You can consider the prefix-function of the length $|S|$, putting $\pi(S, 1) = 0$. Example of the prefix of the function for the line «abcdabcabcdabcdab»:

• $S[i]$	a	b	c	d	a	b	c	a	b	c	d	a	b	c	d	a	b
• $\pi(S,i)$	0	0	0	0	1	2	3	1	2	3	4	5	6	7	4	5	6

Fig. 3. Example of the prefix of the function for the line «abcdabcabcdabcdab»

Hashing [11] is converting an array of input data of arbitrary length (an array of rows) into (output) a bit string of fixed length executed by a certain algorithm. A hash function, or convolution function, is a function that converts the input data of any (usually large) size to a fixed-size data. Hashing transforms the input array of data of arbitrary length into a source bit string of fixed length. Output data is called an input array, key, or message. Hash functions are linked with checksum, control digits, fingerprints, randomization of functions, error-correcting codes, and ciphers. Although these concepts coincide to some extent, each one has its own application and requirements and is developed and optimized in many ways.

The algorithm of the chatbot “Hashbot” will consist of searching for the keyword (words) of the conversation and the formation of the person's termination of the verb.

We search for a keyword by means of hashing. To do this, we use the calculation formula:

$$h(S) = S[0] + S[1] * P + S[2] * P^2 + S[3] * P^3 + \dots + S[N] P^N, \quad (1)$$

where P is a simple number. We should choose P , which is approximately equal to the number of characters in the input alphabet. If the strings are composed only of small Ukrainian letters, then the good choice will be $P = 37$.

One of the most obvious and simple ways of hashing is the middle square method, when the key is erected in a square and taken several digits in the middle. The key is initially reduced to an integer, for carrying out arithmetic operations with it. However, this method works well until a large number of zeros on the left or right is missing. If the item of the table with the index from hash function is already occupied, a concatenated list joins it. If for several different key values returns the same value of the hash function, then at that address there is an indicator on the linked list that contains all the values.

We use the Rabin-Karp algorithm to search the substring in the string for $O(N)$ combining it with hash-function.

Let us we have text T consisting from lines $|S|$. Each line consists of small Cyrillic letters. It is necessary to find all occurrences of the line S in the text T for the time $O(|S| + |T|)$. So, the chatbot answering algorithm "Hashbot" consists of two steps:

1. Keyword searching;
2. Verb ending finding.

The keyword search algorithm consists of the following steps:

1. To count the hash for line S .
2. To count the hash value for all prefixes of subclasses T .
3. To choose all subclasses T of the length $|S|$. Each of them can be compared with other lines of length $|S|$ in time $O(1)$.

When the keyword has been found, you must define the individual verb ending. So, we try to find the verb as the word located before or after the keyword found. We test the ending using a prefix function: by Knutt-Pratt algorithm, which does not contain explicit line comparisons and is executed for $O(n)$ actions.

Here is an algorithm scheme:

1. To count the value of the prefix function $\pi[i]$ for $i \in [1..N-1]$ ($\pi[0] = 0$).
2. To calculate the current value $\pi[i]$, use the variable j , which indicates the length of the current sample considered. First time j is equal $j = \pi[i-1]$.
3. To test a sample of length j , for which we compare the characters $s[j]$ and $s[i]$. If they are the same, then we consider $\pi[i] = j + 1, i = i + 1$. If the characters are different, then we reduce the length j , assuming it is equal to $\pi[j-1]$, and repeat this step of the algorithm from the beginning.
4. If we have reached the length of $j = 0$ and so did not find the same characters, then stop the sampling process and consider $\pi[i] = 0, i = i + 1$.

The general schema of proposed method is given in fig. 4.

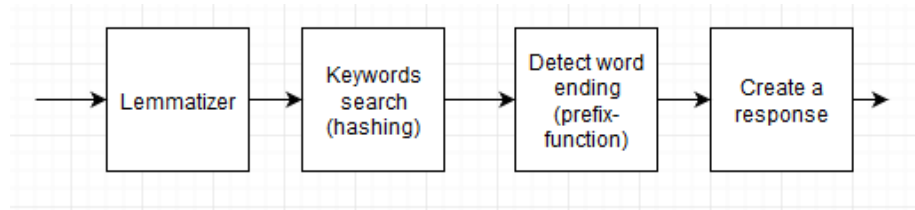


Fig. 4. The general schema of Hashbot algorithm

4 Results

4.1 Technical part

The program is implemented on Python language and web framework Flask. Solution based on direct communication with Telegram API, using direct communication is faster and more efficient for speech-to-text chatbot, but it's also possible to use libraries, pyTelegramBotAPI is the main library for interacting with the Telegram API via Python. Heroku Cloud Application Platform supports for an impressive range of languages, large documentation with tutorials for beginners, a user-friendly interface for the complete work with the application, including access via the console, the presence of databases. An important factor is the ability to "spin" on the server around the clock. The chatbot is built using a rule-based approach. To make possible interaction with Google API, we need to fetch voice message from chat and uploaded it to Google endpoint in one of the supported encodings. File storage is possible to organize in two ways: download and store in some directory with assigning a unique id for each audio file or download file and store all related data in the database. In the basic implementation of chatbot, which is represented in this article, storage was organized as a directory with files, so one of the main features to make a chatbot interface more flexible in the future is possible to implement file storage, for example in ORDBMS PostgreSQL. In the process, it is worth adding the use of a virtual environment with all related libs uploaded and included into it such as one of the important framework (Flask), google-speech-api, requests, etc.

Main script for chatbot given below, all script you can review on GitHub repository [18]:

```
def get_text_from_audio(file_name):  
    """Transcribes the audio file."""  
    from google.cloud import speech  
    from google.cloud.speech import enums  
    from google.cloud.speech import types  
    client = speech.SpeechClient()  
    with io.open(file_name, 'rb') as audio_file:  
        content = audio_file.read()
```

```

        audio = types.RecognitionAudio(content=content)
config=types.RecognitionConfig(encoding=enums.
RecognitionConfig.AudioEncoding.OGG_OPUS,sample_rate_hertz=16000
)
response = client.recognize(config, audio)
print("Before recognition")
result = 'Seems this file is empty'
for result in response.results:
    result = result.alternatives[0].transcript
    print("Ready to send result")
    return result
@app.route('/', methods=['POST', 'GET'])
def index():
    if request.method == 'POST':
        fetched_request = request.get_json()
        if "message" in fetched_request:
            send_response_message(fetched_request)
            return "ok!", 200
    return render_template('index.html')

```

And the last main point for organization interface is communication with Google API. Three possible communication solutions are: using client libraries, gcloud tool and command line requests, for chatbot more suitable is client library usage. To make it work we need set up google-client in the virtual environment, create a project in Google Cloud Platform dashboard, generate unique access credentials and download private key in JSON format and include it into the project and set up global environment variable `GOOGLE_APPLICATION_CREDENTIALS` to the file path of private key [19].

4.2 Algorithmic part

Due to big complexity of well-known algorithm, they are not as effective as we wish. According to this, we propose the usage of Rabin-Karp and Knut-Pratt algorithms, due to their effectivity. In consequence of hashing, we reduce a quantity of comparison, which let the algorithm works faster. To add, we can find special endings with linear complexity. Results of research can be used not only for chatbots and for finding key-words in the text.

An aggregated comparison of algorithms is given in Table 1.

As you can see, the Hashbot algorithm does not dominate only KMP-search in the case when we immediately find the required sample.

Table 1. Comparison of algorithms

<i>Algorithm</i>	<i>Complexity</i>
Hashbot algorithm	$O(S + T)$
Linear search	$O((T-S + 1) * T)$
KMP-search	from $O(S + T)$ to $O((T-S + 1) * T)$
LSA	$O(n^2 k^3)$, $n= S T $
SVM with tf-idf scheme	$O(Q S T)$

5 Conclusions

The paper presents scalable software solution for collecting and processing audio information to text. The program is implemented on Python language and web framework Flask. According to the results of research and after analysis of collected data set of audios and texts, will be developed a custom model based on Keras library using a recurrent neural network for training. Creation of such custom system will be the next phase of research.

After analyzing existing methods, we see that the complexity of the developed algorithm Hashbot for finding keywords is less than of well-known algorithm. Using the prefix function to form a bot response allows you to work with Cyrillic texts. The proposed algorithm improves the work of chatbots, which determines the gender of the interlocutor. This brings the bot closer to the level of human conversation

6 References

1. Shevat, Amir. *Designing bots: Creating conversational experiences* (First ed.). Sebastopol, CA: O'Reilly Media. ISBN 9781491974827 (2017).
2. Mitsuku: <http://www.mitsuku.com/>, last accessed 2019/01/21
3. Rose: <https://www.robeco.nl/service-contact/index.jsp>, last accessed 2019/01/21
4. Right click: <https://rightclick.io/#/>, last accessed 2019/01/21
5. Poncho: <https://poncho.is/>, last accessed 2019/01/21
6. Insomnobot: <http://insomnobot3000.com/>, last accessed 2019/01/21
7. Dr.A.I: https://www.healthtap.com/login?redirect_to=/symptoms, last accessed 2019/01/21
8. Baidu Melody's: <http://research.baidu.com/baidus-melody-ai-powered-conversational-bot-doctors-patients/>, last accessed 2019/01/21
9. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C.. *Introduction to algorithms*. MIT press (2009).
10. Knuth, D. E., Morris, Jr, J. H., & Pratt, V. R.. Fast pattern matching in strings. *SIAM journal on computing*, 6(2), 323-350 (1977)
11. Knuth, D. E. *Sorting and Searching*, 2nd edn. The Art of Computer Programming, vol. 3 (1998)
12. Shakhovska, N., & Shvorob, I. The method for detecting plagiarism in a collection of documents. In Scientific and Technical Conference" Computer Sciences and Information Technologies"(CSIT), 2015 Xth International (pp. 142-145). (2015).
13. Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (Eds.). *Handbook of latent semantic analysis*. Psychology Press (2013).

14. Aizawa, A. An information-theoretic perspective of tf-idf measures. *Information Processing & Management*, 39(1), 45-65 (2003).
15. Shakhovska, N., Medykovskyj, M., Bychkovska, L. Building a smart news annotation system for further evaluation of news validity and reliability of their sources. *Przegląd Elektrotechniczny*, 91 (7), 43-44 (2015)
16. du Preez, S. J., Lall, M., & Sinha, S. An intelligent web-based voice chatbot. In *IEEE EUROCON 2009* 386-391. (2009).
17. Boyko, N., Basytiuk, O., & Shakhovska, N. Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* 478-482 (2018).
18. <https://github.com/obasys/harry-bot>
19. Cloud Speech-to-Text Documentation, <https://cloud.google.com/speech-to-text/docs/>
20. Huang, J., Zhou, M., & Yang, D. Extracting Chatbot Knowledge from Online Discussion Forums. In *IJCAI* 7, 423-428 (2007).