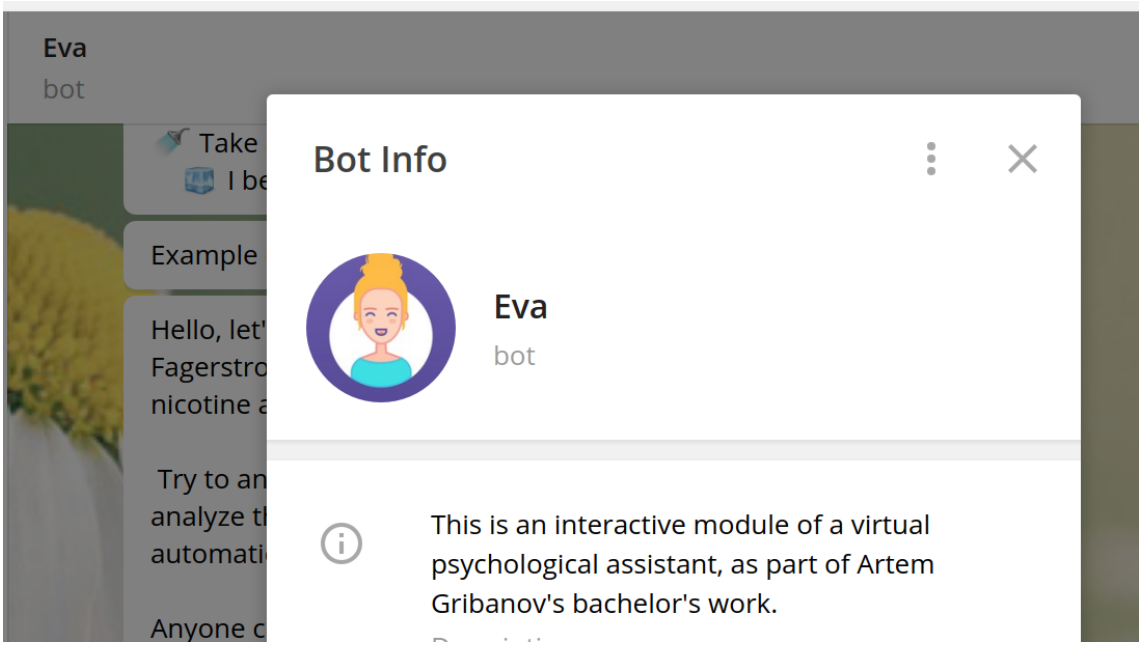
Source

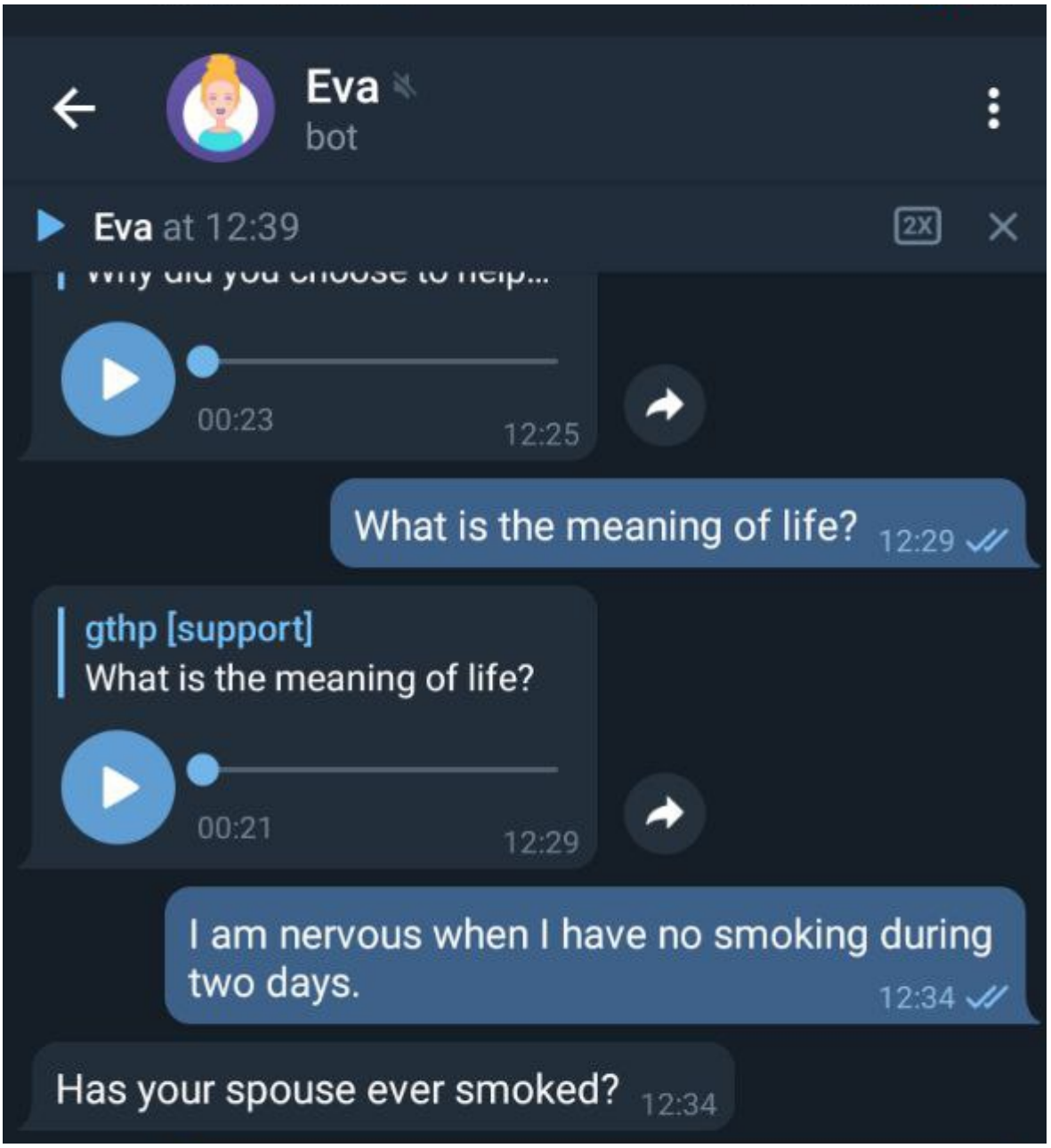
Introduction

Eva is an interactive customizable chat bot based on a predictive text generation algorithm. Thus, it was possible to solve the problem of script bots - the inability to respond to unexpected output. Eva endowed with the ability to respond by voice.



In order to direct the bot's unprotected responses towards the nicotine addiction topic, the concepts of the context of dialogue, personality and memory were invented and implemented.
In addition, the bot has predefined scenarios:

- help to avoid an attack of cravings for smoking
- send an educational video about the dangers of smoking
- conduct a survey on the strength of cravings for smoking



Key concepts

To generate responses, the bot uses prepared dialogue templates, at the end of which a human message is inserted. As a result of the operation of the algorithm, we get the most probable continuation of the text, which contains the "meaningful answer" of the bot to the question posed by the user.

Personality

In the course of communication with the bot, there is a feeling that the bot identifies itself with a virtual assistant for nicotine addicts. This was achieved by introducing itself on behalf of the bot at the beginning of the template with the dialog into which the user's message is inserted:

'"**Psychologist**:Hello! My name is Eva. I am a virtual psychologist and your personal psychological assistant. I study nicotine addiction. I know a lot about the dangers of nicotine and cigarettes. I will help you cope with nicotine addiction. ' \

'**Me**:I am your patient with nicotine addiction. How could you help me? ' \

'**Psychologist**:As a specialist, I know what nicotine addiction is and how it affects your body. I know what the harm of smoking is and what the devastating consequences can be. I am an expert in the field of psychological addiction to nicotine. I know a lot about the dangers of smoking. I will help you with my advice and knowledge. I want you to stop smoking. I want you to recover from your nicotine addiction.' \

'**Me**:Can you help me? ' \

'**Psychologist**:I will try to help you as a psychologist. I will help you with tips on how to cope with your cigarette cravings. Tell me about your problem or ask a question and I will answer you. ' \

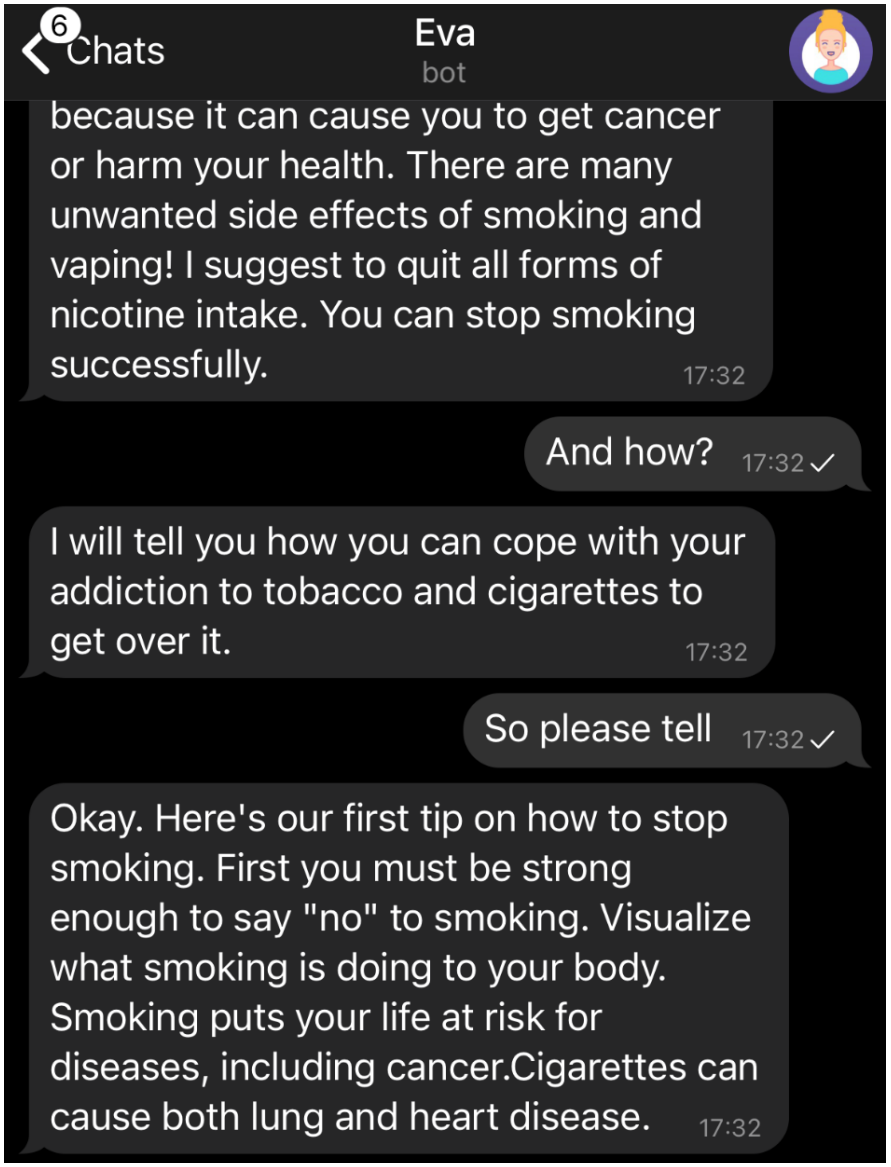
'**Me**: <user input message> \

'**Psychologist**: {Generated response}

Dialogue context

The bot can tell something, it can answer questions, it can ask questions. This is achieved by using not one template, but a group of templates.

The patterns alternate randomly, which creates the feeling of talking to a real person on the other side. Since sometimes the bot conducts a conversation and comments calmly, then it can ask you a question on the topic or remain silent, waiting for you to say.



Thus, I prepared a template for generating a neutral "comment" response (the first template) and generating questions in the context of a cigarette issue:

""**Question**: Do you know where you are?' \

'**Answer**: I am visiting a psychologist who is solving my problem of nicotine addiction to cigarettes. '\

'**Question**: How long have you been smoking? '\

'**Answer**: I have been smoking since childhood. '\

'**Question**: How many cigarettes do you smoke on average daily? '\

'**Answer**: My cravings for smoking depend on my psychological state. Ask the next question about my nicotine addiction, where are its roots. '\

'**Question**: {Generated question} ?

Memory

To increase the level of user interaction with the bot, the concept of memory was developed.

Increasing the conceptual coherence of the dialogue is achieved by the fact that several previous messages from the bot and the user are stored in the bot's memory.

Thus, the generated response contains the contextual vector of previous messages.

'**Psychologist**:I will try to help you as a psychologist. I will help you with tips on how to cope with your cigarette cravings. Tell me about your problem or ask a question and I will answer you. ' \

'**Me**: <user input message 1> \

'**Psychologist**: <bot output message 1> \

'**Me**: <user input message 2> \

'**Psychologist**: {Generated response}

Censorship

One of the weak points of the use of text generation algorithms in B2C products is the weak control of the response that the algorithm generates. Since the algorithm is trained on the corpuses of texts from the Internet and forums, from time to time it can give answers that are unacceptable: they contain obscene words, an intolerant attitude towards minorities and national prejudices.

To improve the user experience, the Censorship module was invented and implemented, which analyzes the content of the text generated by the algorithm for the presence of forbidden words and expressions. If such a word is found, the bot will generate a new response.

```
class Censorship:
    banned_words_list = ["***", "*****", "****", "*****", "****", "*****", "*****"]
    words_blacklist: [str]

    def is_output_safe(self, text):
        for word in self.banned_words_list:
            if word in text:
                return False
        return True

    def get_words_blacklist(self):
        return self.banned_words_list
```

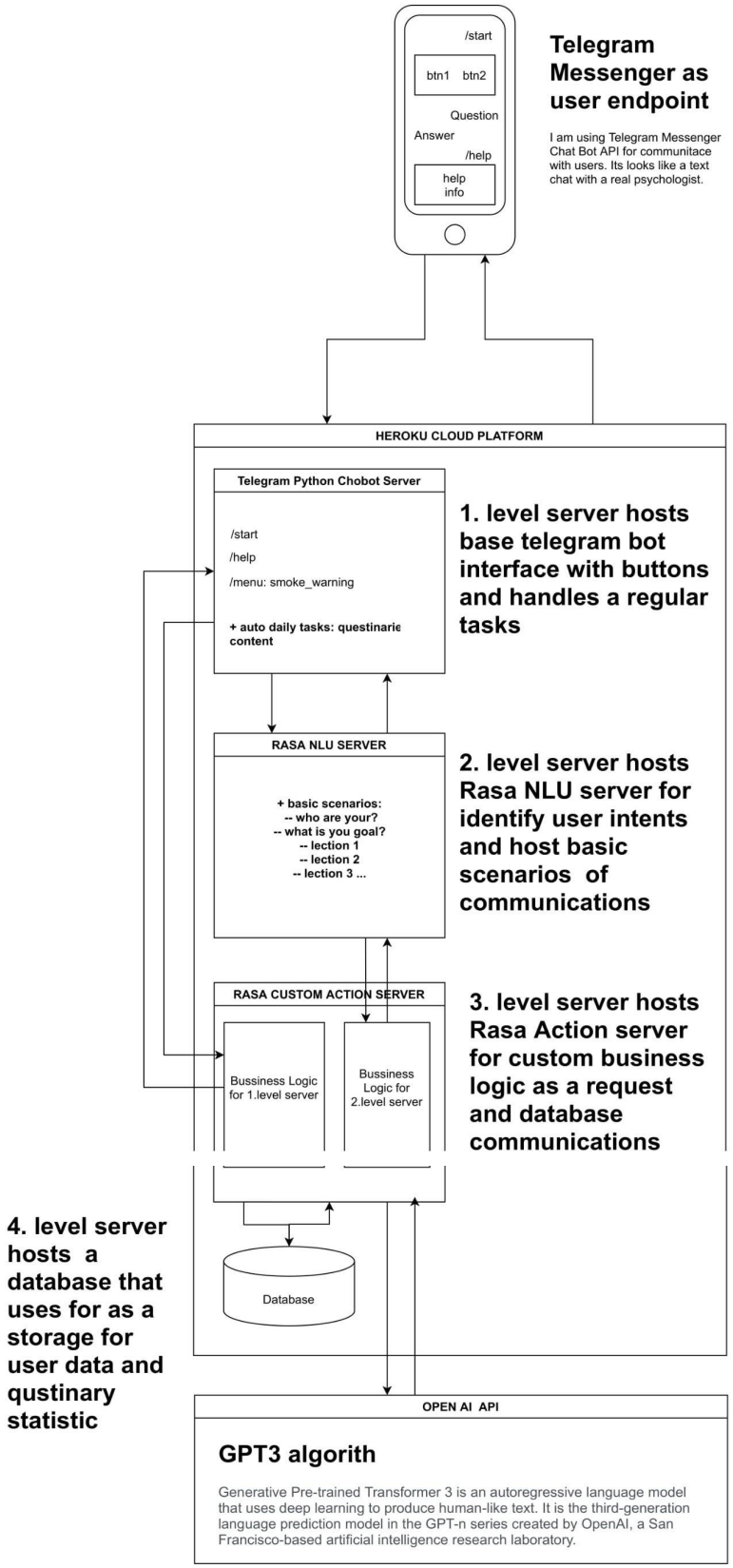
Emergency help

We consider the likelihood that the person who interacts with the bot may be in a suicidal state. If the user asks the bot how to harm or kill himself, the bot replies with the following text:

suicidal_warning =

'Perhaps this is not a good idea! Try to contact the psychological support service. 112 is similar to the American 911 general emergency service, active in all EU countries, and can be traced with caller ID.If you have a medical emergency (including acute psychiatric condition or suicidal state) an ambulance will arrive to take you to the nearest appropriate medical facility.'

Architecture



autor: Gribanov Artem, 2021/2022

Technology stake

Language

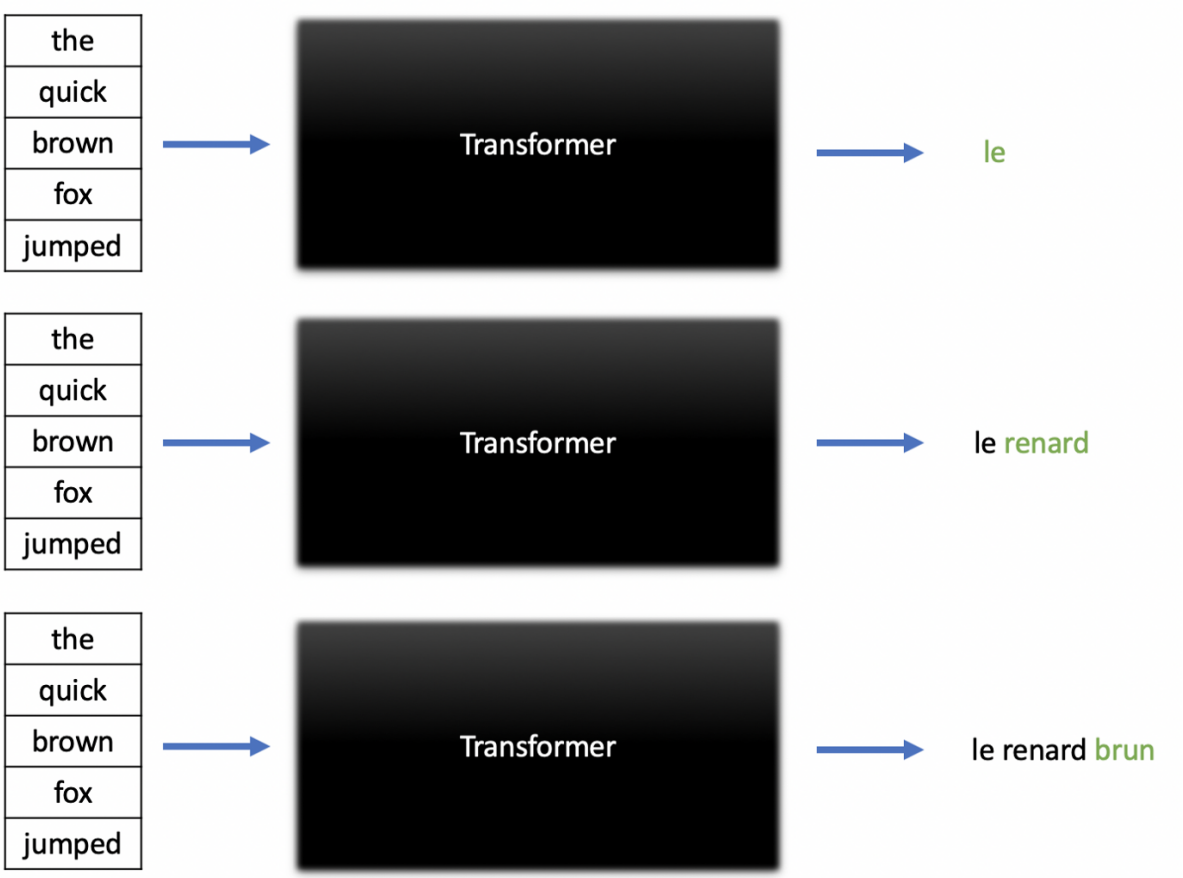
Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.

Libraries

Generative Pre-trained Transformer 3 (GPT-3)

GPT-3 (Generative Pre-trained Transformer 3) is a language model that was created by OpenAI, an artificial intelligence research laboratory in San Francisco. The 175-billion parameter deep learning model is capable of producing human-like text and was trained on large text datasets with hundreds of billions of words.



Source

At its core, GPT-3 is basically a transformer model. Transformer models are sequence-to-sequence deep learning models that can produce a sequence of text given an input sequence. These models are designed for text generation tasks such as question-answering, text summarization, and machine translation. The image below demonstrates how a transformer model iteratively generates a translation in French given an input sequence in English.

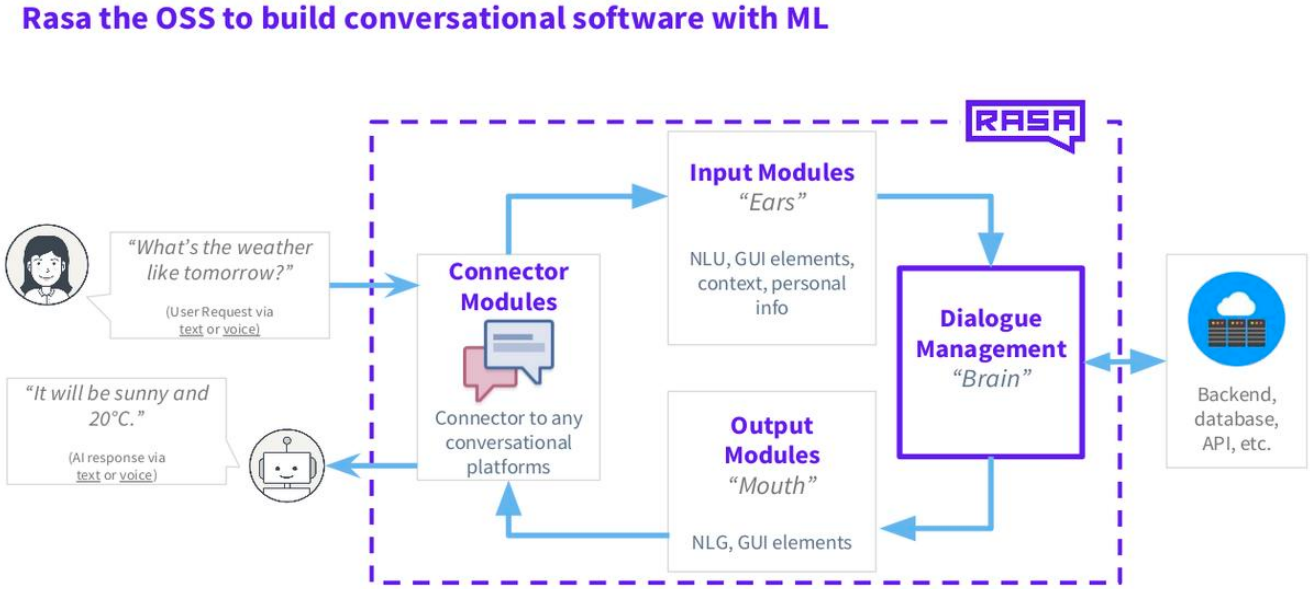
Frameworks

Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Rasa

Rasa is a **framework for developing AI powered, industrial grade chatbots**. It's incredibly powerful, and is used by developers worldwide to create chatbots and contextual assistants. In this project, we are going to understand some of the most important basic aspects of the Rasa framework and chatbot development.



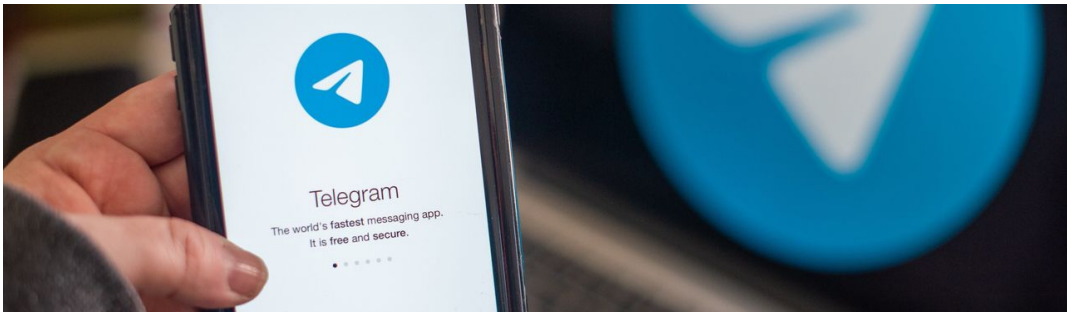
Source

autor: Gribanov Artem, 2021/2022

User manual

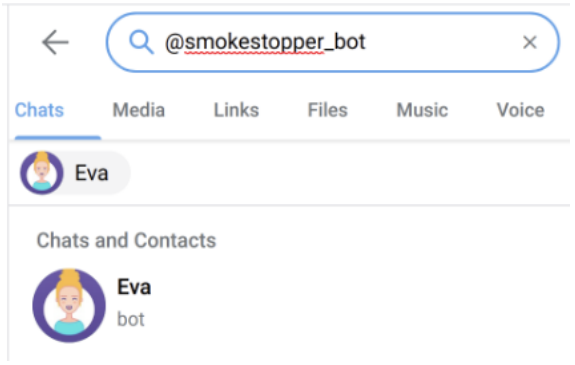
At the moment, the Eva is available in the Telegram messenger:

- Open the messenger on your computer or phone
    - https://telegram.org/



Source

- Find a bot in the search: **@smokestopper_bot**



- Start communication with command **/start**