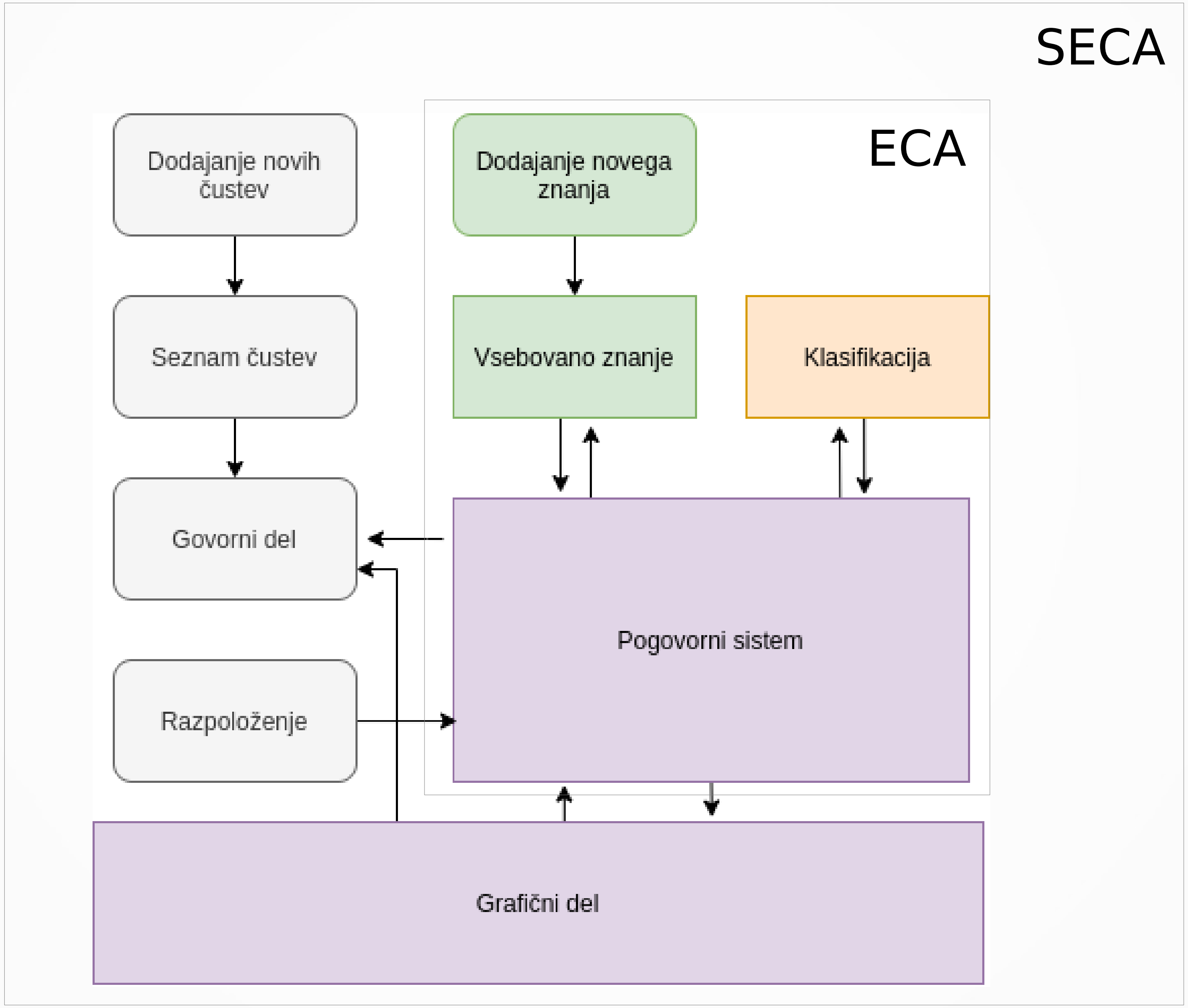
Operacijske raziskave v telekomunikacijah

Univerza *v Ljubljani*

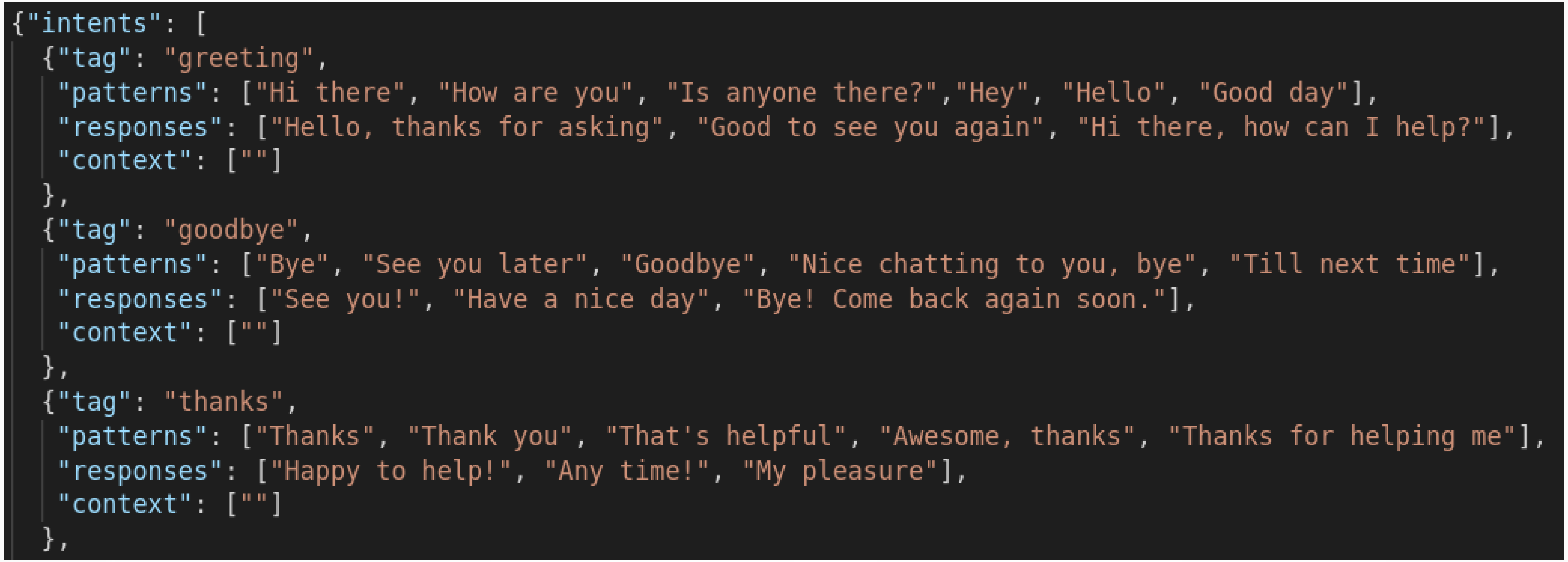
Pogovorni agent

Bojan Faletič, 27.maj.2020

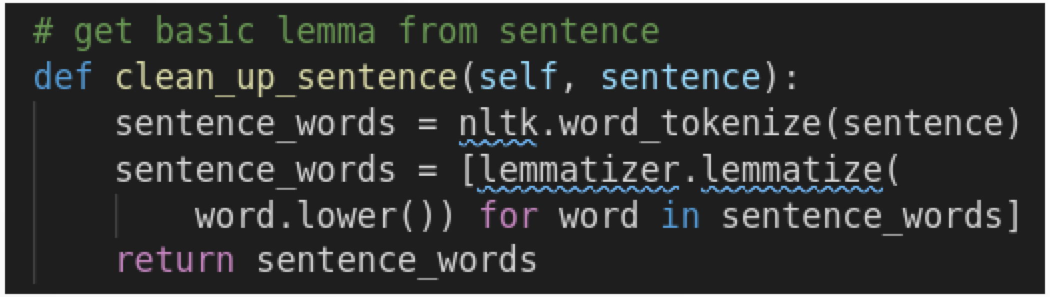
Shema agenta



Zbirka znanja

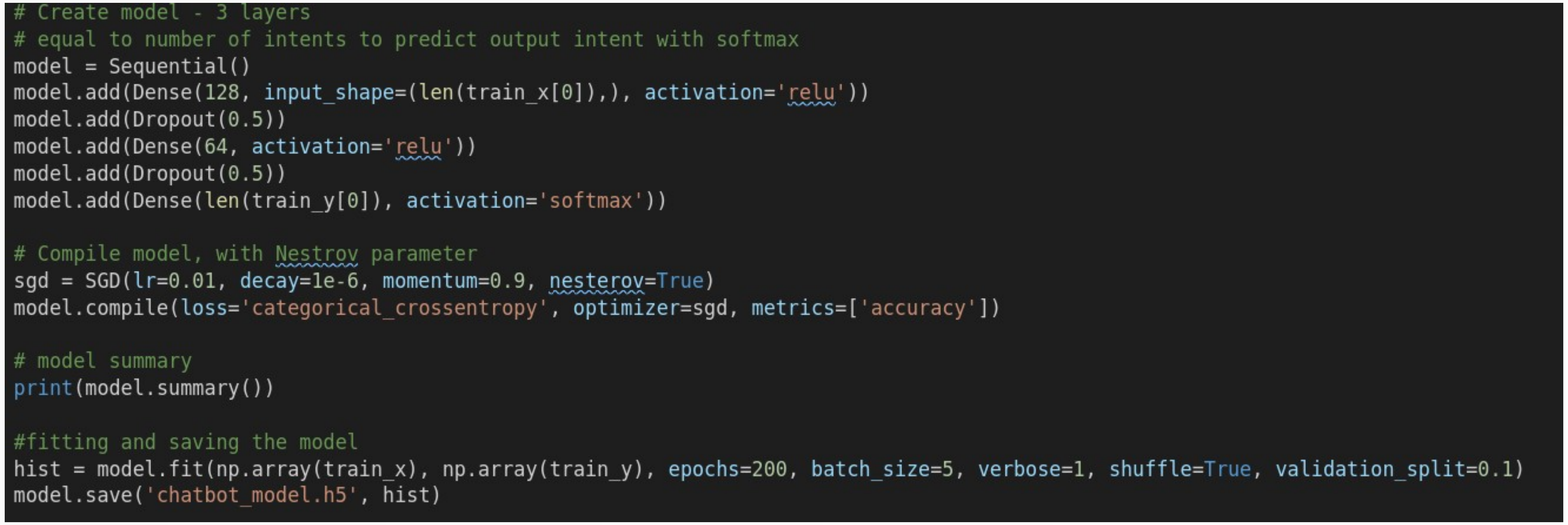


```
{"intents": [
  {"tag": "greeting",
   "patterns": ["Hi there", "How are you", "Is anyone there?","Hey", "Hello", "Good day"],
   "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"],
   "context": [""]
  },
  {"tag": "goodbye",
   "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
   "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
   "context": [""]
  },
  {"tag": "thanks",
   "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
   "responses": ["Happy to help!", "Any time!", "My pleasure"],
   "context": [""]
  },
```

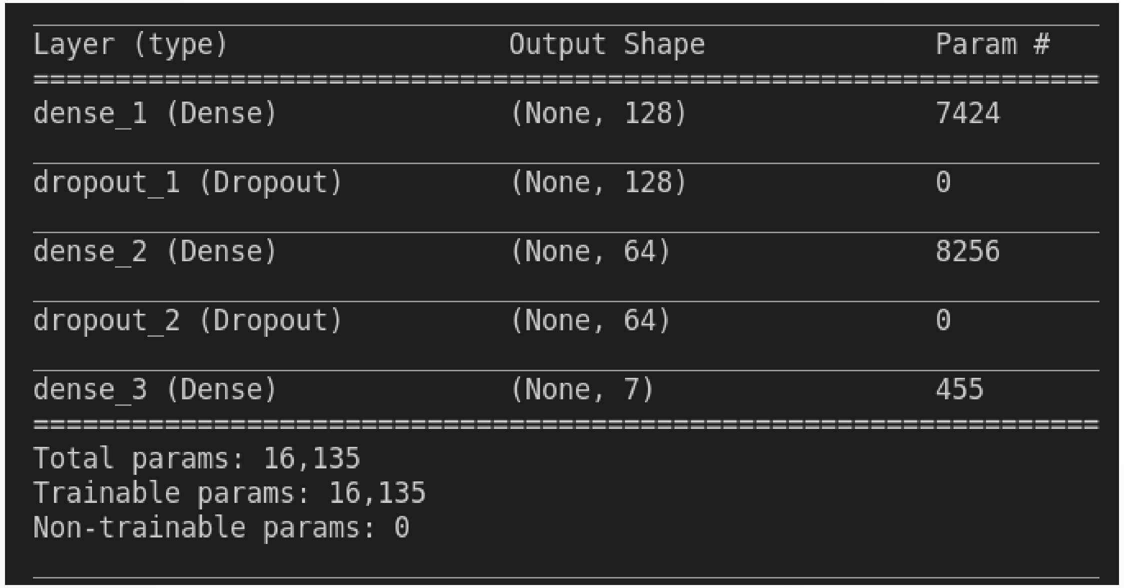


```
# get basic lemma from sentence
def clean_up_sentence(self, sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(
        word.lower()) for word in sentence_words]
    return sentence_words
```

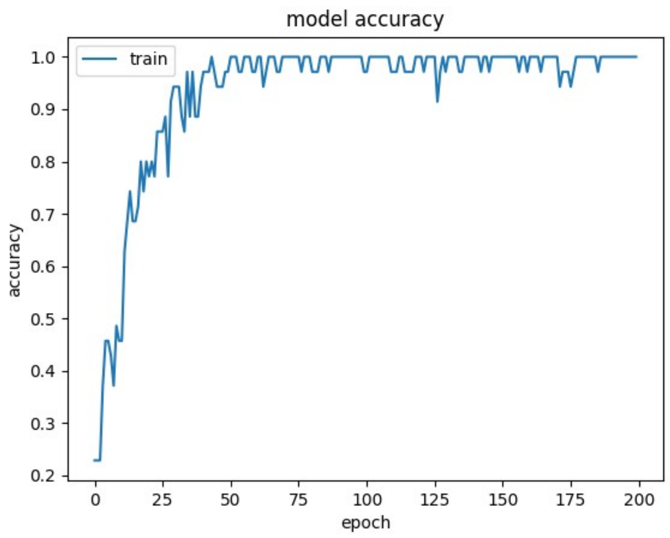
Ustvarjen model



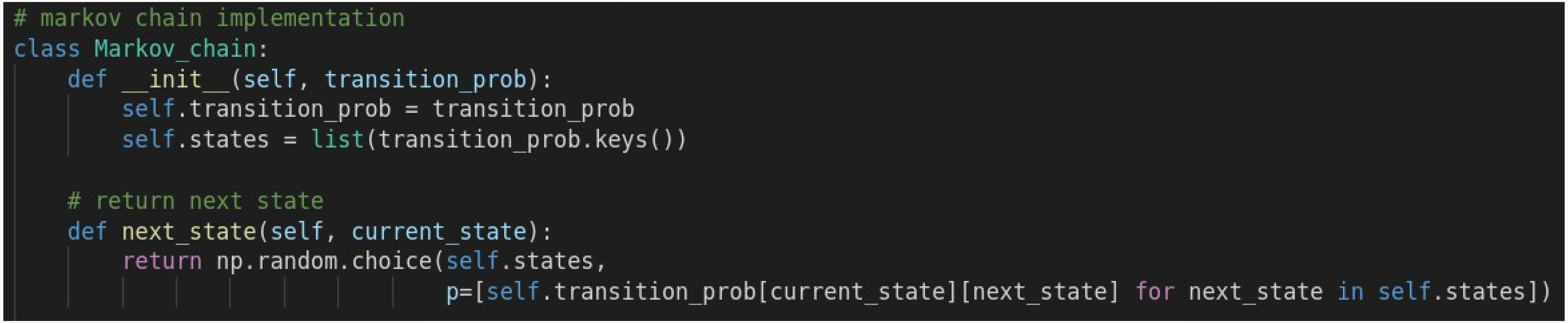
```
# Create model - 3 layers
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model, with Nestrov parameter
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# model summary
print(model.summary())

#fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1, shuffle=True, validation_split=0.1)
model.save('chatbot_model.h5', hist)
```



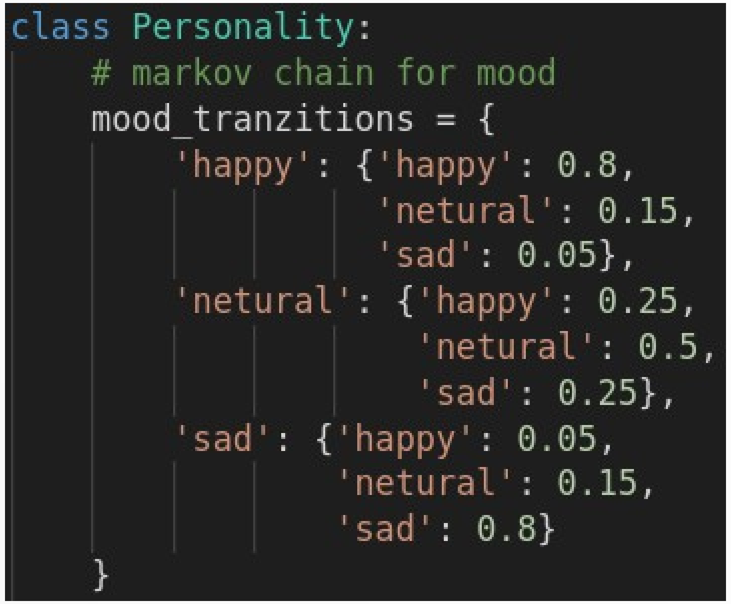
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               7424
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 455
=================================================================
Total params: 16,135
Trainable params: 16,135
Non-trainable params: 0
_________________________________________________________________
```



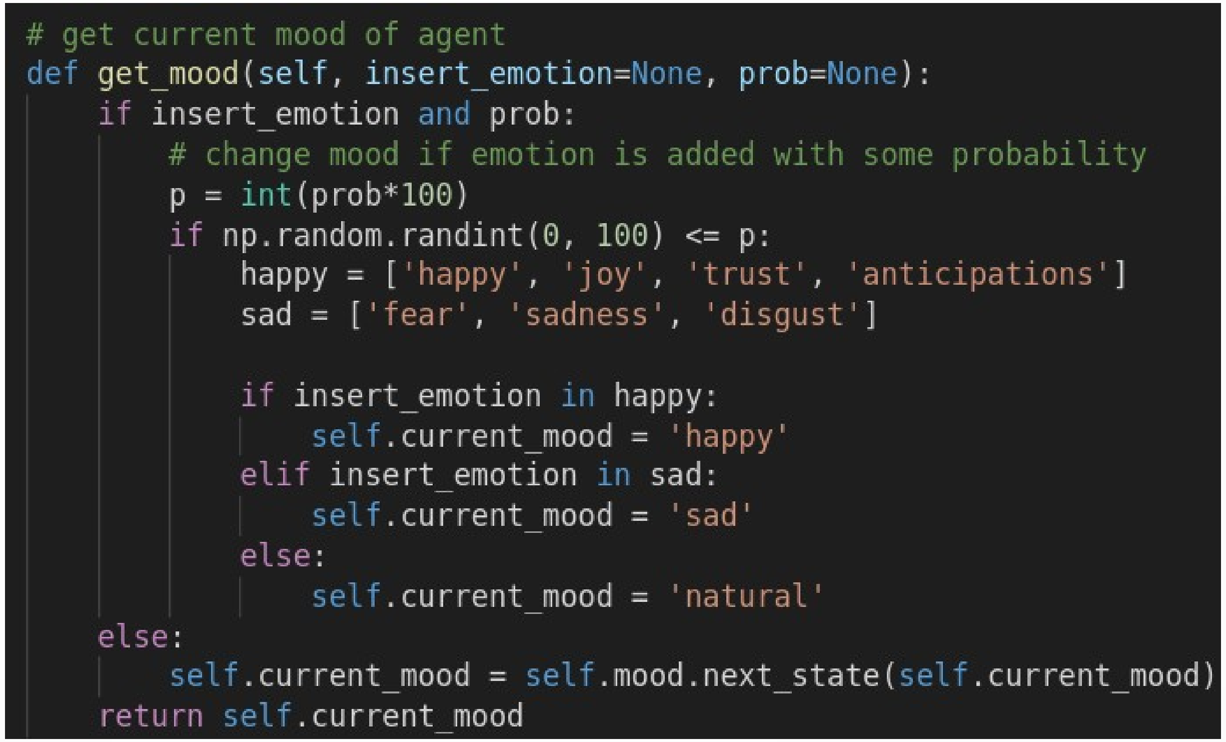
Razpoloženje



```
# markov chain implementation
class Markov_chain:
    def __init__(self, transition_prob):
        self.transition_prob = transition_prob
        self.states = list(transition_prob.keys())

    # return next state
    def next_state(self, current_state):
        return np.random.choice(self.states,
                                p=[self.transition_prob[current_state][next_state] for next_state in self.states])
```

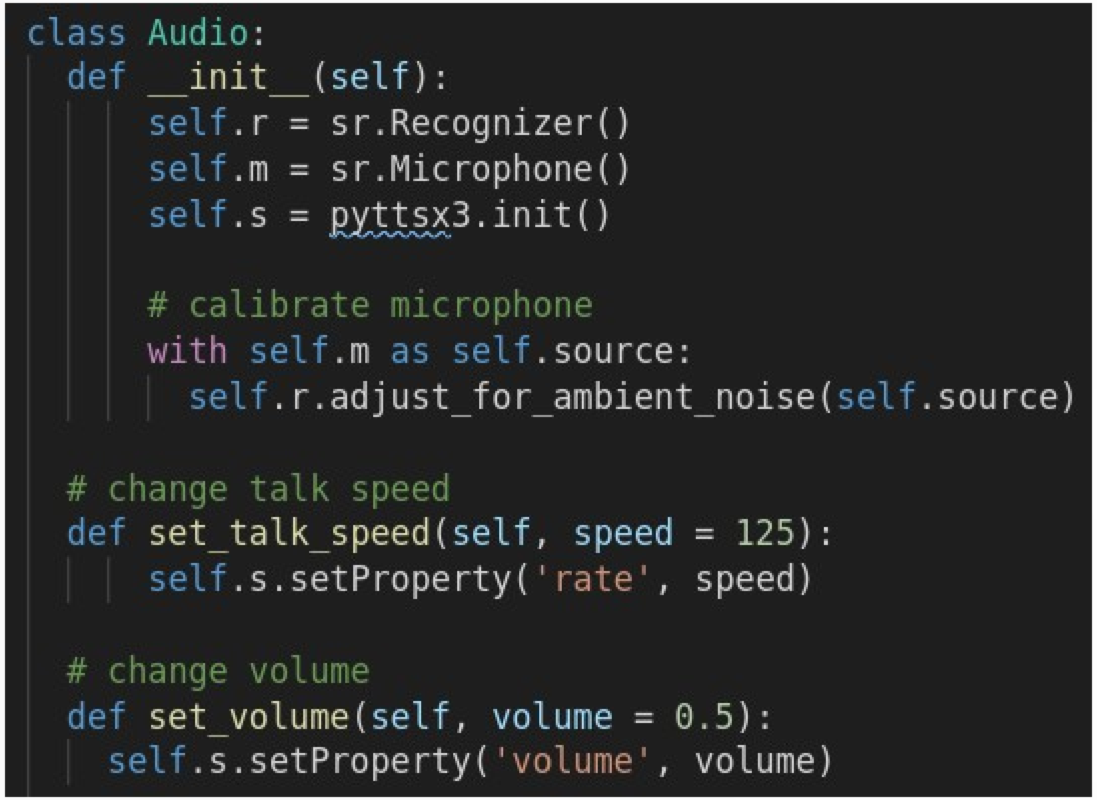


```
class Personality:
    # markov chain for mood
    mood_tranzitions = {
        'happy': {'happy': 0.8,
                  'netural': 0.15,
                  'sad': 0.05},
        'netural': {'happy': 0.25,
                    'netural': 0.5,
                    'sad': 0.25},
        'sad': {'happy': 0.05,
                'netural': 0.15,
                'sad': 0.8}
    }
```

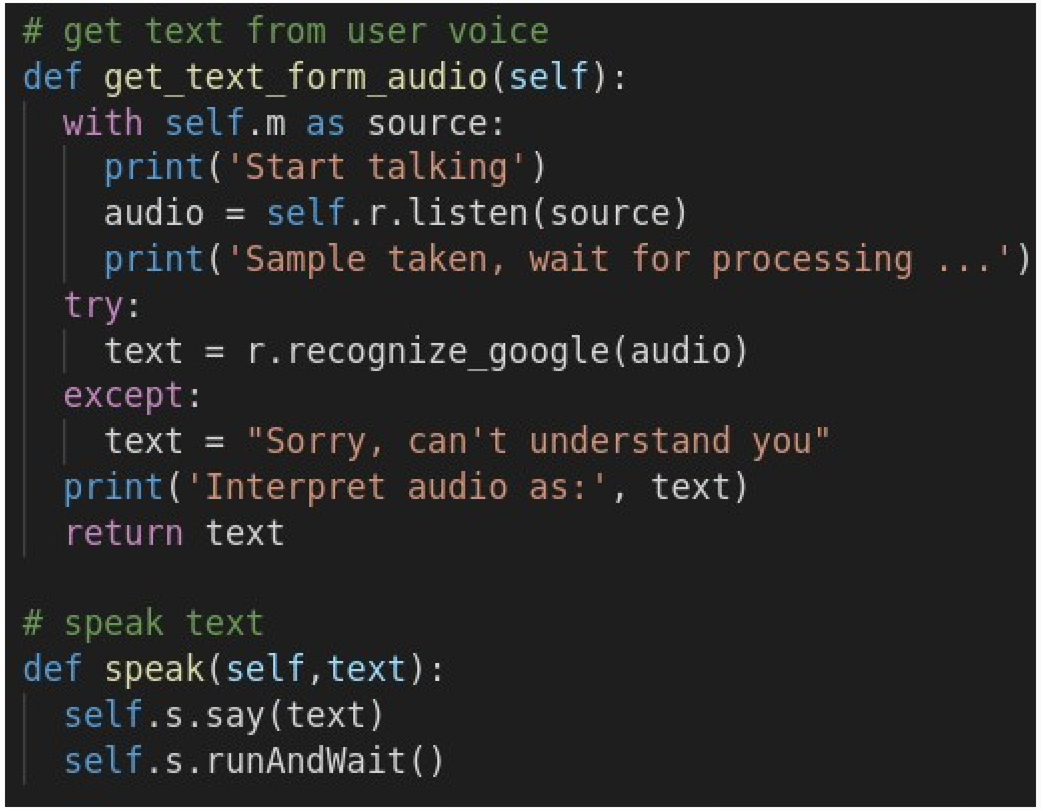


```
# get current mood of agent
def get_mood(self, insert_emotion=None, prob=None):
    if insert_emotion and prob:
        # change mood if emotion is added with some probability
        p = int(prob*100)
        if np.random.randint(0, 100) <= p:
            happy = ['happy', 'joy', 'trust', 'anticipations']
            sad = ['fear', 'sadness', 'disgust']

            if insert_emotion in happy:
                self.current_mood = 'happy'
            elif insert_emotion in sad:
                self.current_mood = 'sad'
            else:
                self.current_mood = 'natural'
    else:
        self.current_mood = self.mood.next_state(self.current_mood)
    return self.current_mood
```

Govorni del

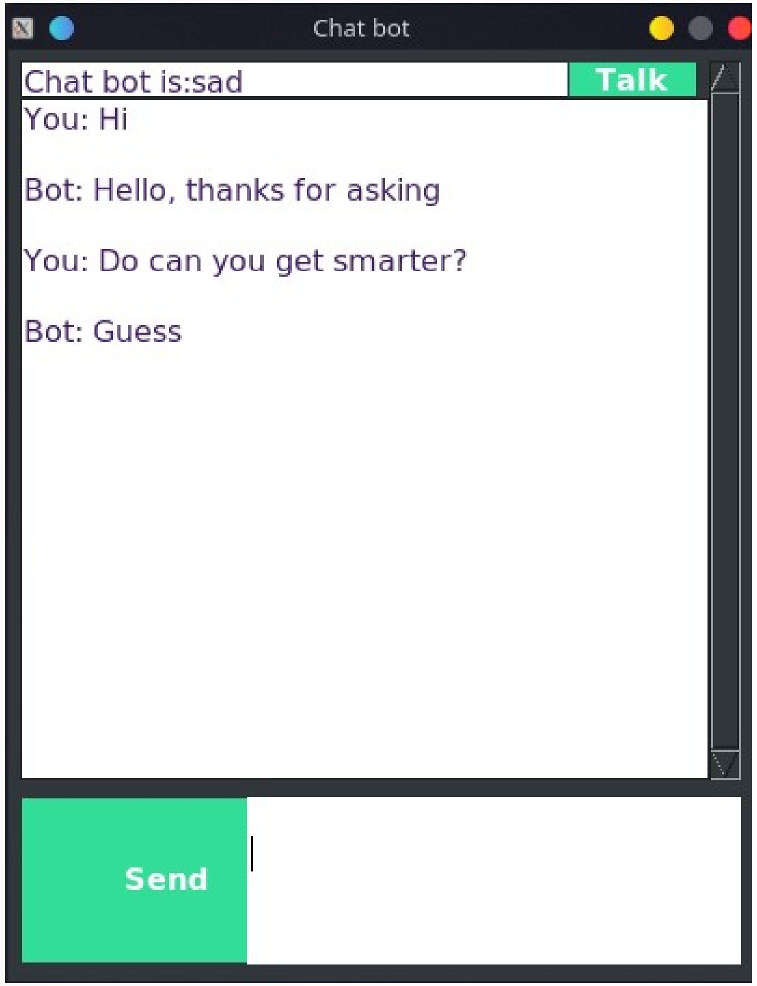


```
class Audio:
  def __init__(self):
      self.r = sr.Recognizer()
      self.m = sr.Microphone()
      self.s = pyttsx3.init()

      # calibrate microphone
      with self.m as self.source:
        self.r.adjust_for_ambient_noise(self.source)

  # change talk speed
  def set_talk_speed(self, speed = 125):
      self.s.setProperty('rate', speed)

  # change volume
  def set_volume(self, volume = 0.5):
    self.s.setProperty('volume', volume)
```



```
# get text from user voice
def get_text_form_audio(self):
  with self.m as source:
    print('Start talking')
    audio = self.r.listen(source)
    print('Sample taken, wait for processing ...')
  try:
    text = r.recognize_google(audio)
  except:
    text = "Sorry, can't understand you"
  print('Interpret audio as:', text)
  return text

# speak text
def speak(self,text):
  self.s.say(text)
  self.s.runAndWait()
```

Grafični modul



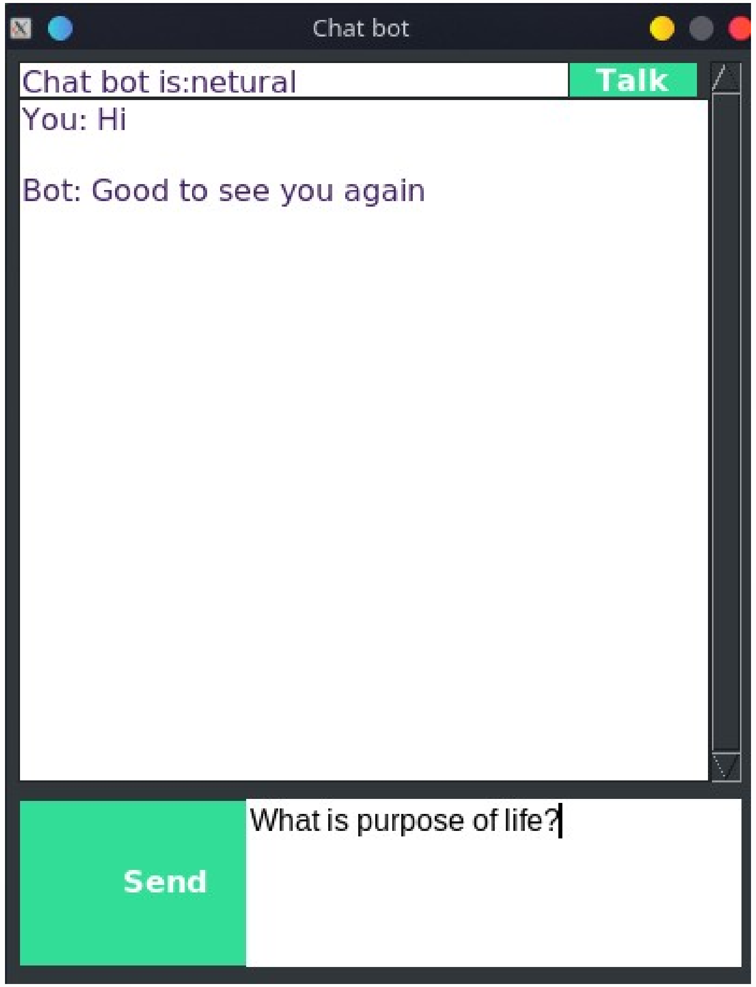
Chat bot is:sad

You: Hi

Bot: Hello, thanks for asking

You: Do can you get smarter?

Bot: Guess

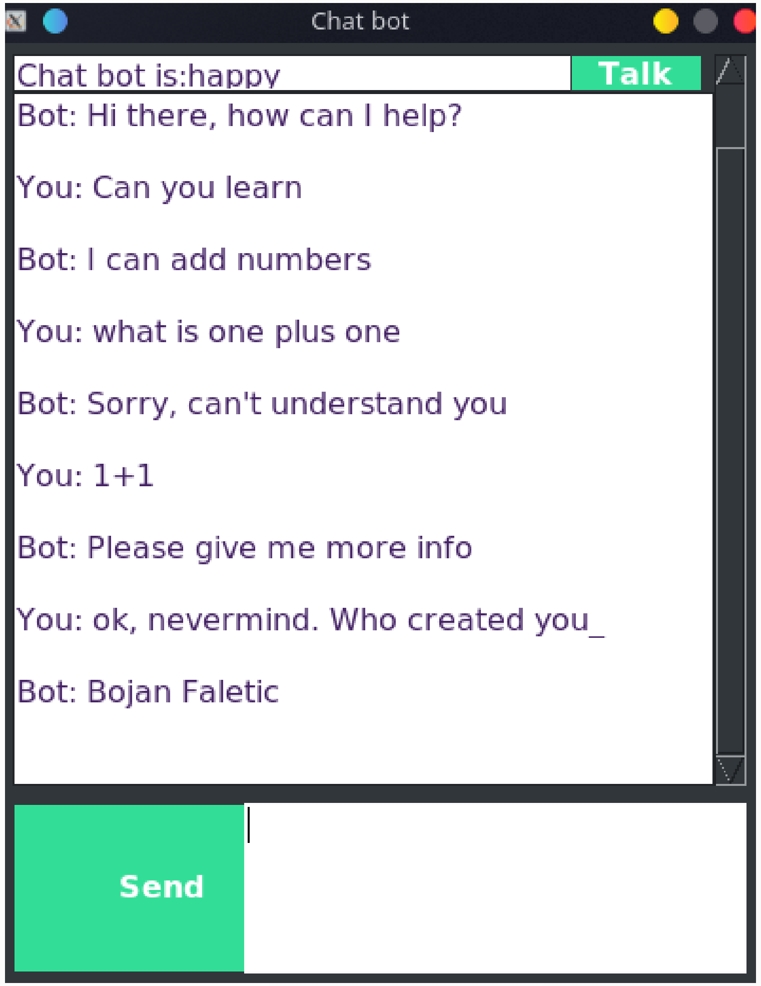


Chat bot is:netural

You: Hi

Bot: Good to see you again

What is purpose of life?



Chat bot is:happy

Bot: Hi there, how can I help?

You: Can you learn

Bot: I can add numbers

You: what is one plus one

Bot: Sorry, can't understand you

You: 1+1

Bot: Please give me more info

You: ok, nevermind. Who created you_

Bot: Bojan Faletic

https://github.com/BojanFaletic/chat_bot