TURING

Go to profile

FOR DEVELOPERS

How Artificial Intelligence Helps Create Automated Chatbots

Share

The arrival of artificial intelligence chatbots may seem recent, but the first chatbot was created during 1964-66 by Joseph Weizenbaum. Named ELIZA, the chatbot could even pass the Turing Test that was generally used to test the intelligent behavior of a machine. While ELIZA was remarkable, technology has come much further. In the intervening years, abilities have developed considerably.

“Artificial cognition will, over the next three to five years, become absolutely indispensable for any form of operations or support,” says Shannon Kalvar, research manager for IT service management and client virtualization at research firm, IDC.

This article will look at AI’s impact on automated chatbots, how chatbots work, their architecture, types, and more.

What are chatbots?

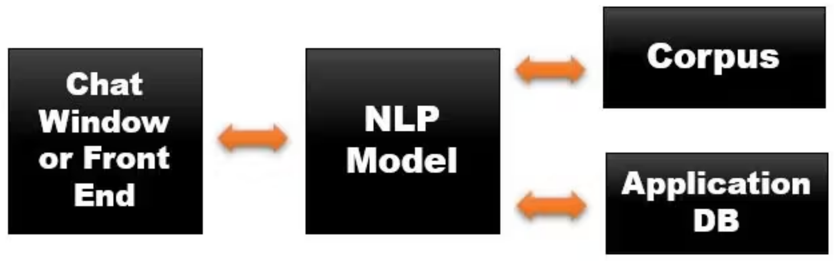
Chatbots are simulations that can grasp what humans are trying to convey and interact with them while performing specific tasks. Nowadays, most companies use chatbots to engage with their users on a basic level instead of actual humans.

Since chatbots can easily communicate with people and help them solve certain needs, they are a good fit for customer service. Some applications of chatbots include acting as virtual help desk assistants, virtual home assistants, and phone assistants. Such artificial intelligence help desks cut down customer waiting time and increase business productivity.

There are two main types of chatbots, simple and smart. With simple chatbots, the answers are already established in the system. The bot can handle simple queries but will fail to answer complex questions. Smart chatbots, on the other hand, use machine learning techniques when communicating with users, enabling them to build a database of answers.

By 2025, chatbots will be handling 95% of customer service interactions. They will be installed almost everywhere, but this doesn’t mean that all will function up to par.

- A front-end application interface through which the user will interact with the bot.
- A deep learning model to process the natural language. Models like BERT work best when making a chatbot.
- Corpus or data required to train the natural language processing (NLP) model. This is usually a huge amount of data that contains a lot of human interactions.
- A database for processing the actions to be performed by the chatbot.



NLP refers to any interaction between a machine and human language. However, this doesn't mean the machine will be able to understand it properly. There are various challenges when dealing with human language. For instance, it's hard for a computer to understand the tone of a person or a slang used, if any.

Even today, the best AI chatbots can't be confused with humans. Yet, we still speak with bots as they are useful - and perhaps even fascinating.

Types of chatbots

Flow-based chatbot

This chatbot is probably the simplest because it works by using a predefined conversational flow. It works on if-else statements. Whenever a client triggers a conversation, the chatbot guides them through the conversation flowchart, step by step.

There are several pros and cons of this type of chatbot:

Pros

- From the start of the chat, it can tell the user what they can obtain by communicating with it.
- Like long short-term memory (LSTM), it is easy for a flow-based chatbot to remember the context of a conversation.
- Since the conversation and replies are predesigned, if the user opts for option X, you know that the bot will respond with Y.
- The chatbot can be easily tweaked/edited for changes.

Cons

- The user can not have a conversation that is not designed by the chatbot admin as it follows a scripted decision tree.
- Flow-based chatbots can not learn or change their replies based on extra data or information. They always follow a defined path.
- If the user enters a keyword that the bot has never seen before, it will not respond.

Intent-based chatbot

Before we discuss what sets this type of chatbot apart, let's look at what 'intent' means here. For example, a user says 'Recommend me hip hop songs'. The intent behind this message is to browse the songs and suggest them to the user. Instead of relying on some predefined input, the chatbot understands the context of the message and then conveys its results to the user.

To offer this sort of scope, the chatbot needs NLP. This is an element in AI that allows machines to understand the human language. An intent-based chatbot is, of course, better than a flow-based chatbot since it uses artificial intelligence.

An intent-based chatbot looks for key terms and 'entities' in the messages it receives. If the user were to ask what the weather is like in a location on a certain day, the intent is to know the weather. The location and day are the entities as they specify the information the user is looking for.

Pros

AI-enabled intent-based chatbots are the superior choice as they 'understand' human language. They facilitate:

- **Natural conversation:** Talking to an intent-based bot feels similar to speaking with a person, thanks to the use of NLP. There is no need to use specific keywords for the bot to understand them. This makes for a much smoother and more engaging conversation.

TURING

Go to profile

Although live customer service is available in nearly every company, chatbots save a lot of time and reduce work for customer service agents. Utilizing AI technology frees up agents' time for issues that are more serious and cannot be handled through bots. They can focus on providing true customer service rather than handling low-level tasks that can be taken care of by the bots. Not only do AI chatbots depend on natural language processing to interact with customers, they get smarter with use as more data flows into the model working at the backend.

Building an AI chatbot

Let's look at how to build a working chatbot with Python and have a conversation with it. A prerequisite is knowledge of NLP.

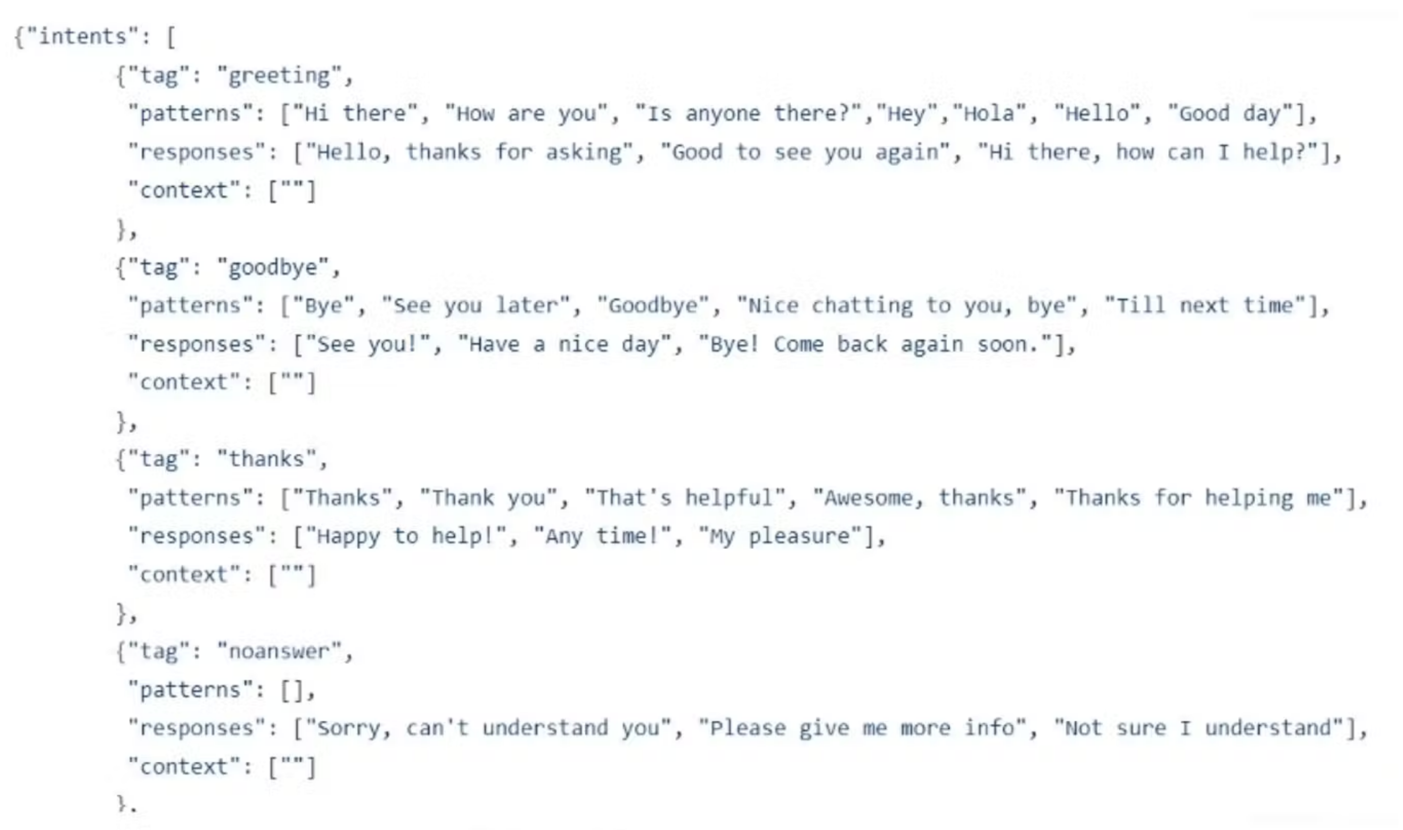
Importing libraries and data

Begin by importing the necessary libraries and making sure they are properly installed. You can use pip3 to install the packages.

There are various libraries like nltk that contain different tools for cleaning up text and preparing it for deep learning algorithms. json is used to load the json files directly into Python; pickle helps to load pickle files; numpy is used to perform linear algebra operations; and keras, a deep learning framework, will be used to build our model.

Initializing chatbot training

We have initialized all the lists where we will store the NLP data. We have our json file which contains the 'intents'.
Here's a snippet of what the json file looks like:



```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola", "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": [""]
        },
        {"tag": "noanswer",
         "patterns": [],
         "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],
         "context": [""]
        }.
```

Image source

```
for intent in intents['intents']:
    for pattern in intent['patterns']:
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        documents.append((w,intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

If you look at the json file closely, you will notice that it contains sub-objects within objects. For this, we will use a nested loop to extract all the words within 'patterns' and add them to our word list. We also add each pair of patterns within their corresponding tag to our document list.

We have cleaned the data and now it is ready to be fed into a deep learning model.

Building a deep learning model

```
training = []
output_empty = [0] * len(classes)
for doc in documents:
    bag = []
    pattern_words = doc[0]
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)
    output_row = list(output_empty)
    output_row =[classes.index(doc[1])] = 1
    training.append([bag,output_row])
random.shuffle(training)
training = np.array(training)
train_x = list(training[:,0])
train_y = list(training[:,1])
```

Now, let's separate the training data. In the above code, we are creating a big nested list that contains a bag of words of each of our documents. We converted the text into numbers to feed it to the deep learning model. We also have a feature called output_row that acts as a key for the list. After all these stems, we do a train_test_split with the patterns being the X variable and the intents being the Y variable.

```
model = Sequential()
model.add(Dense(128,input_shape = (len(train_x[0]),),activation_function='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)
sgd = SGD(lr=0.01,decay=1e-6,momentum = 0.9,nesterov = True)
model.compile(loss='categorical_crossentropy',optimizer='sgd',metric=['accuracy'])
hist = model.fit(np.array(train_x),np.array(train_y),epochs=200,batch_size=5,verbose=1)
```

Code source

Sequential model in Keras is one of the simplest neural networks. It has three hidden layers, with the first having 128 neurons, the second having 64 neurons, and the third having the number of intents as the number of neurons. The point of this network is to be able to predict which intent to choose, given some data.

Now that you've learned what goes into building a chatbot and a deep learning model, try implementing these codes on your own. Once your model is created, you can build your own chatbot interface and use this model at the backend.

Related articles

Training, Testing & Deployment of a Classification Model Using Convolutional Neural Networks and Machine Learning Classifiers

CNN, Convolutional Neural Networks, is a deep-learning-based algorithm that takes an image as an input...

Read more

Machine Learning or Artificial Intelligence - The Right Way Forward for Data Science

This article will aim to eliminate the blur between both these closely related terms to help you understand how AI and ML are tied to data science...

Read more

TURING

Go to profile

Read more

Complete Analysis of Artificial Intelligence vs Artificial Consciousness

Artificial intelligence (AI) is usually defined as the simulation of human intelligence processes by computer systems. It's become...

Read more

Java Virtual Machine (JVM): Introduction and Its Architecture

Looking to understand the concepts of JVM and its architecture? Here's a detailed guide...

Read more

Apply for remote Machine Learning developer jobs at top U.S. companies

FULL-TIME REMOTE JOB

Risk Systems Front-End...

A fast-growing company that is developing a world-class regulated multi-asset trading platform is looking for a Risk Systems Front-End Engineer. The engineer will play a crucial role in designing and implementing front-end...

Finance 251-10K employees

JavaScript Frameworks JavaScript

Apply now

FULL-TIME REMOTE JOB

Front-End Developer

An innovative U.S.-based business dedicated to reshaping the approach to product research, discovery, and buying experiences by pioneering a Product Intelligence Platform, is searching for a Front-End Developer. The...

Ecommerce 1-10 employees

React HTML/CSS JavaScript

Apply now

TURING

Go to profile

Blog

Know more about remote work.
Checkout our blog here.

Contact

Have any questions?
We'd love to hear from you.

Hire remote developers

Tell us the skills you need and we'll find the best developer for you in days, not weeks.

Hire Developers

Companies	Developers	Company	Contact
Hire Developers	Apply for Jobs	Blog	Contact Us
Book a Call	Developer Login	Press	Help Center
Explore Services	Remote Developer Jobs	About Us	Developer Support
Our Service Offerings	Developer Reviews	Careers	Customer Support
Hire for Specific Skills	Knowledge Base		
Customer Reviews	Resume Guide		
Interview Q/A	Jobs for LatAm		
Hiring Resources			

Sitemap Terms of Service Privacy Policy Privacy Settings