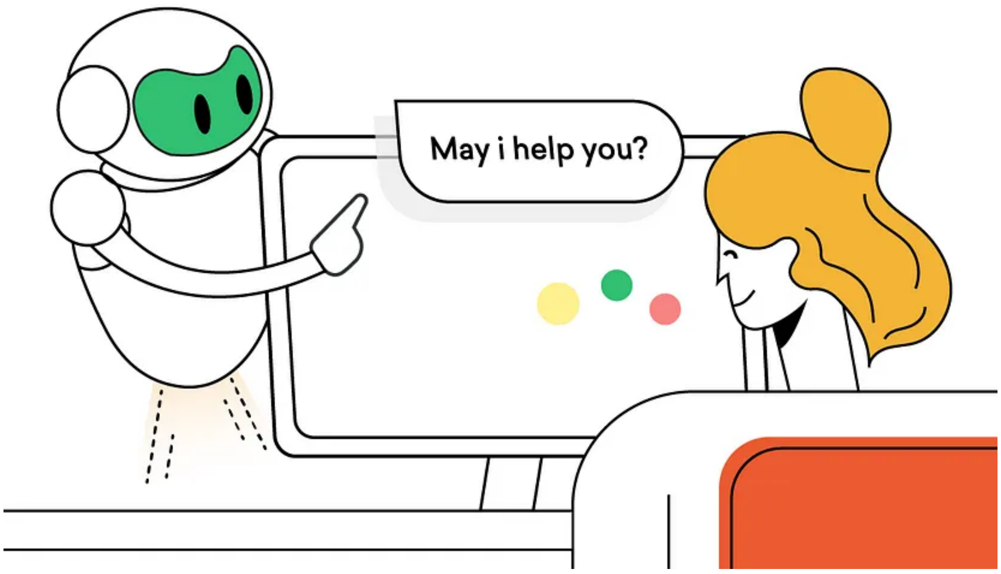
Build your first chatbot using NLTK and Keras

Hitesh Mishra · Follow
8 min read · Apr 1, 2021

44



Chatbot using NLTK and Keras

In this article, we will learn about chatbots using Python and how to make chatbots in python using NLTK and Keras.

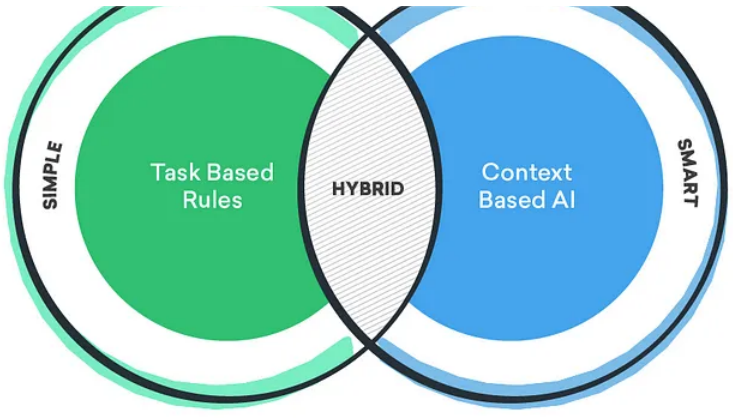
What is Chatbot?

A chatbot is a software application used to conduct an online chat conversation via text or text-to-speech instead of providing direct contact with a live human agent. A chatbot is a type of software that can automate conversations and interact with people through messaging platforms.

Chatbots are now responsible for almost 30% of all activities. Chatbots are used by companies to provide services such as customer care, knowledge generation, and more.

Types of chatbots?

Chatbots can be broadly categorized into three types.



Types of chatbots

Simple chatbots

- Simple chatbots have limited capabilities, and are usually called rule-based bots.
- They are task-specific. This means the bot poses questions based on predetermined options and the customer can choose from the options until they get answers to their query.
- These chatbots are best suited for straightforward dialogues.
- They are very simple to build and train.
- Example: Ordering Pizza

Smart chatbots

- AI-enabled smart chatbots are designed to simulate near-human interactions with customers.
- They can have free-flowing conversations and understand intent, language, and sentiment.
- These chatbots require programming to help them understand the context of interactions.
- They are much harder to implement and execute and need a lot of data to learn.
- Example: Virtual Assistants

Hybrid chatbots

- They are a combination of simple and smart chatbots.
- Hybrid chatbots meet that middle ground.
- Hybrid chatbots have some rule-based tasks, and they can understand intent and context.

- This makes them a balanced tool for businesses to interact with

- Example: Medical Diagnosis

How to Make Chatbot in Python?

To create a chatbot in python you should have good knowledge of Python, Keras, and *Natural language processing (NLTK)*.

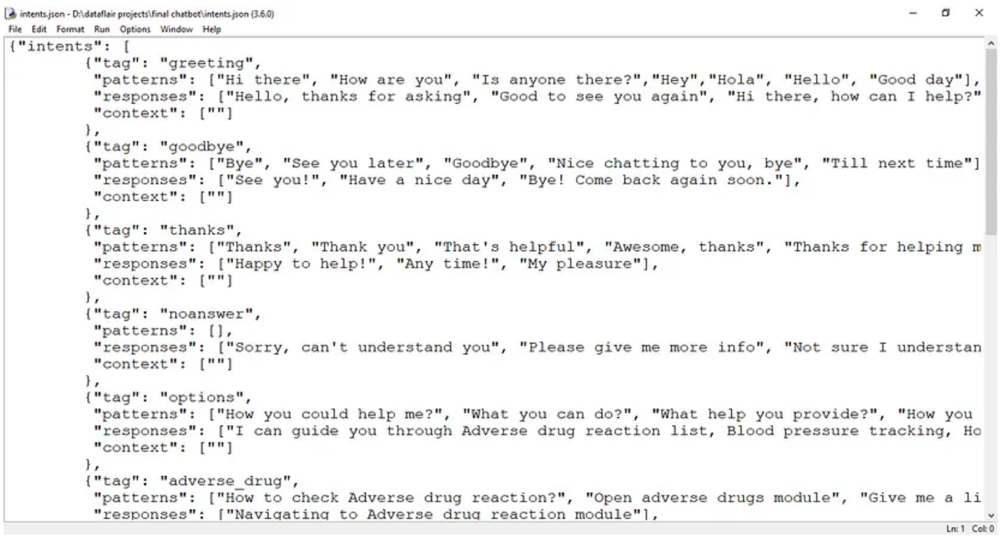
Below are the 6 steps to create a chatbot in Python:

1. **Install required modules** you can install the required modules with the help of the python-pip command

```
pip install tensorflow, keras, pickle, nltk
```

2. Import and load the data file

- Create intents.json. This is how our intents.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 m
         "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 understan
         "context": [""]
        },
        {"tag": "options",
         "patterns": ["How you could help me?", "What you can do?", "What help you provide?", "How you
         "responses": ["I can guide you through Adverse drug reaction list, Blood pressure tracking, Ho
         "context": [""]
        },
        {"tag": "adverse_drug",
         "patterns": ["How to check Adverse drug reaction?", "Open adverse drugs module", "Give me a li
         "responses": ["Navigating to Adverse drug reaction module"],
```

intents.json

- First, make a file name as train_chatbot.py. We import the necessary packages for our chatbot and initialize the variables we will use in our Python project.
- The data file is in JSON format so we used the JSON package to parse the JSON file into Python.

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
```

```
classes = []
documents = []
ignore_words = ['?', '!']

data_file = open('intents.json').read()
intents = json.loads(data_file)
```

3. Preprocess data

- When working with text data, we need to perform various preprocessing on the data before we make a machine learning or a deep learning model. Based on the requirements we need to apply various operations to preprocess the data.
- Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words.
- Here we iterate through the patterns and tokenize the sentence using nltk.word_tokenize() function and append each word in the words list. We also create a list of classes for our tags.

```
for intent in intents['intents']:
    for pattern in intent['patterns']:
        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))
        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

- Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is converting a word into its lemma form and then creating a pickle file to store the Python objects that we will use while predicting.

```
# lemmatize, lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower())
```

```
# documents = combination between patterns and intents
print(len(documents), "documents")

# classes = intents
print(len(classes), "classes", classes)

# words = all words, vocabulary
print(len(words), "unique lemmatized words", words)
pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
```

4. Create training and testing data

- and the output.
- Our input will be the pattern and output will be the class our input pattern belongs to. But the computer doesn't understand the text so we will convert text into numbers.

```
# create our training data
training = []

# create an empty array for our output
output_empty = [0] * len(classes)

# training set, bag of words for each sentence

for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent
related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in
pattern_words]

# create our bag of words array with 1, if word match found in
current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)
    # output is a '0' for each tag and '1' for current tag (for each
pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1
    training.append([bag, output_row])

# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)

# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")
```

5. Build the model

- We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this.
- After training the model for 200 epochs, we achieved 100% accuracy on our model. Let us save the model as 'chatbot_model.h5'.

```
# Create model - 3 layers. First layer 128 neurons, second layer 64
neurons and 3rd output layer contains number of neurons
# 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. Stochastic gradient descent with Nesterov
accelerated gradient gives good results for this model
```

```
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
```

```
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200,
batch_size=5, verbose=1)
model.save('chatbot_model.h5', hist)
print("model created")
```

6. Predict the response (Graphical User Interface)

- To predict the sentences and get a response from the user, create a new file 'chatapp.py'.
- We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve a random response from the list of responses.
- Again we import the necessary packages and load the 'words.pkl' and 'classes.pkl' pickle files which we have created when we trained our model:

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

import pickle
import numpy as np
from keras.models import load_model
model = load_model('chatbot_model.h5')

import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

- To predict the class, we will need to provide input in the same way as we did while training. So we will create some functions that will perform text preprocessing and then predict the class.

```
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in
sentence_words]
    return sentence_words
# return bag of words array: 0 or 1 for each word in the bag that
exists in the sentence

def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0]*len(words)

for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                    # assign 1 if current word is in the vocabulary
```

```
    return(np.array(bag))

def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []

for r in results:
        return_list.append({"intent": classes[r[0]], "probability": 
str(r[1])})
    return return_list
```

- After predicting the class, we will get a random response from the list of intents.

```
def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result

def chatbot_response(text):
    ints = predict_class(text, model)
    res = getResponse(ints, intents)
    return res
```

- Now we will develop a graphical user interface. Let's use the Tkinter library which is shipped with tons of useful libraries for GUI.
- We will take the input message from the user and then use the helper functions we have created to get the response from the bot and display it on the GUI. Here is the full source code for the GUI.

```
#Creating GUI with tkinter
import tkinter
from tkinter import *

def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)
    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "You: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Verdana", 12 ))
        res = chatbot_response(msg)
        ChatLog.insert(END, "Bot: " + res + '\n\n')
        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)

base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)
#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50",
font="Arial",)
```

```
ChatLog['yscrollcommand'] = scrollbar.set
#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send",
width="12", height=5, bd=0, bg="#32de97",
activebackground="#3c9d9b",fg='#ffffff', command= send )

#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5",
font="Arial")

#EntryBox.bind("<Return>", send)
#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)
base.mainloop()
```

7. Run the chatbot

- To run the chatbot, we have two main files; train_chatbot.py and chatapp.py.
- First, we train the model using the command in the terminal:

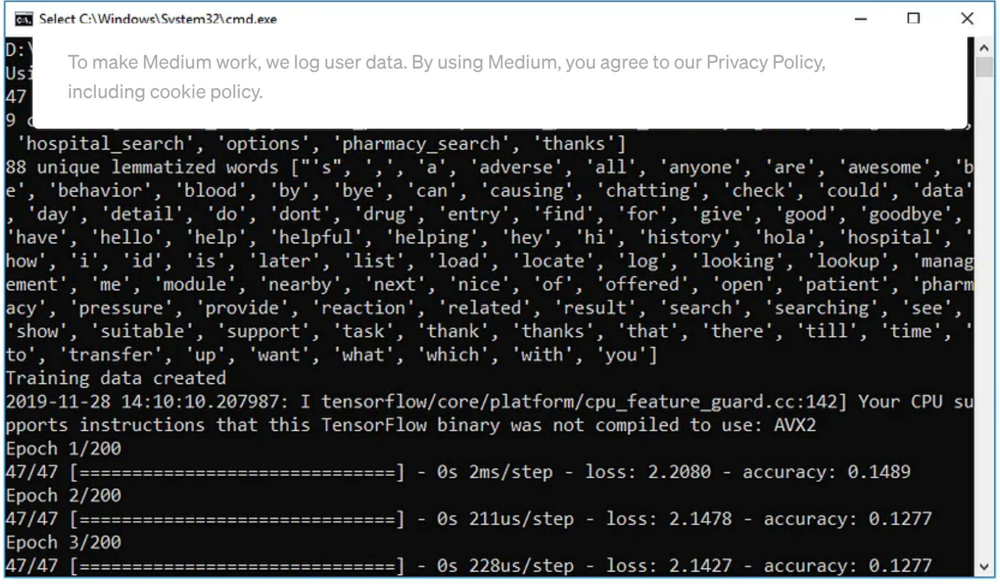
```
python train_chatbot.py
```

- If we don’t see any error during training, we have successfully created the model. Then to run the app, we run the second file.

```
python chatgui.py
```

- The program will open up a GUI window within a few seconds. With the GUI you can easily chat with the bot.

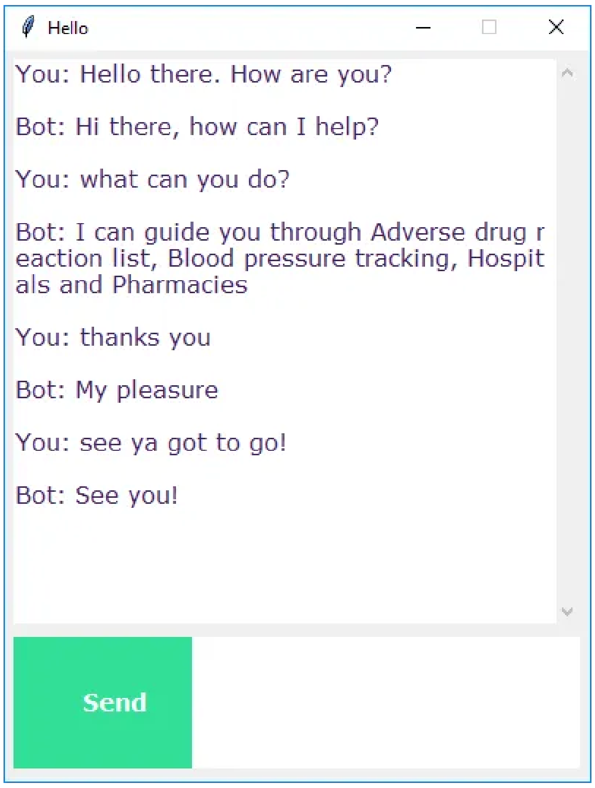
Screenshots:



```
Select C:\Windows\System32\cmd.exe
D:\
Usi
47
9 c
 'hospital_search', 'options', 'pharmacy_search', 'thanks']
88 unique lemmatized words ["'s", ',', 'a', 'adverse', 'all', 'anyone', 'are', 'awesome', 'b
e', 'behavior', 'blood', 'by', 'bye', 'can', 'causing', 'chatting', 'check', 'could', 'data'
, 'day', 'detail', 'do', 'dont', 'drug', 'entry', 'find', 'for', 'give', 'good', 'goodbye',
'have', 'hello', 'help', 'helpful', 'helping', 'hey', 'hi', 'history', 'hola', 'hospital', '
how', 'i', 'id', 'is', 'later', 'list', 'load', 'locate', 'log', 'looking', 'lookup', 'manag
ement', 'me', 'module', 'nearby', 'next', 'nice', 'of', 'offered', 'open', 'patient', 'pharm
acy', 'pressure', 'provide', 'reaction', 'related', 'result', 'search', 'searching', 'see',
'show', 'suitable', 'support', 'task', 'thank', 'thanks', 'that', 'there', 'till', 'time', '
to', 'transfer', 'up', 'want', 'what', 'which', 'with', 'you']
Training data created
2019-11-28 14:10:10.207987: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU su
pports instructions that this TensorFlow binary was not compiled to use: AVX2
Epoch 1/200
47/47 [==============================] - 0s 2ms/step - loss: 2.2080 - accuracy: 0.1489
Epoch 2/200
47/47 [==============================] - 0s 211us/step - loss: 2.1478 - accuracy: 0.1277
Epoch 3/200
47/47 [==============================] - 0s 228us/step - loss: 2.1427 - accuracy: 0.1277
```

To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

Chatbot Result



```
Hello
You: Hello there. How are you?

Bot: Hi there, how can I help?

You: what can you do?

Bot: I can guide you through Adverse drug r
eaction list, Blood pressure tracking, Hospit
als and Pharmacies

You: thanks you

Bot: My pleasure

You: see ya got to go!

Bot: See you!
```

Looking for the Spanish version of this article, click here

Check out the code on my GitHub.

Thanks for reading. If you found the article useful don't forget to **clap** and do share it with your friends and colleagues. :) If you have any questions, feel free to reach out to me.
Connect with me on 👉 LinkedIn, Github :)

Chatbots Python Artificial Intelligence Keras Naturallanguageprocessing

Written by Hitesh Mishra

Follow

112 Followers

FullStack | Python | ReactJS | NodeJS | NextJS | Tech Writer

More from Hitesh Mishra

Hitesh Mishra in Geek Culture

Getting Started with Celery & RabbitMQ

Celery is an open-source asynchronous task queue or job queue which is based on...

3 min read · Apr 26, 2021

96

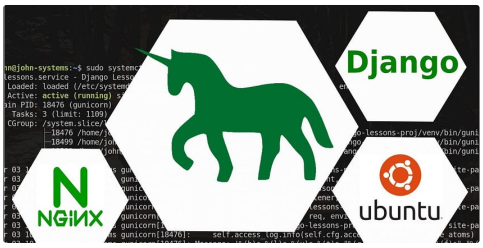
Hitesh Mishra in Geek Culture

How to Use ElasticSearch With Django

What is Elasticsearch?

4 min read · Feb 13, 2021

215 1



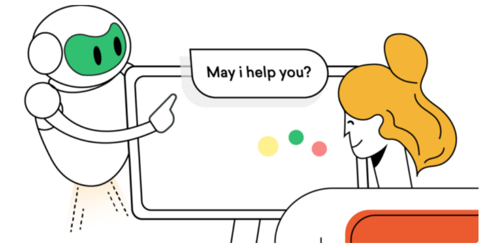
Hitesh Mishra

Deploy Django app with Nginx, and Gunicorn

NGINX makes the website faster & It can handle a high volume of connections.

3 min read · Dec 10, 2021

21 1



Hitesh Mishra

Construye tu primer chatbot usando NLTK y Keras

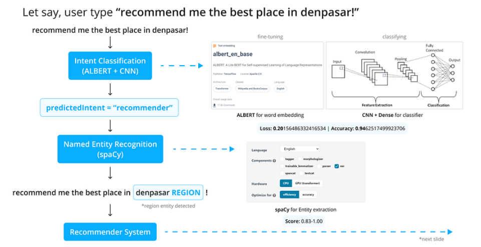
En este artículo, aprenderemos sobre chatbots usando Python y cómo crear...

8 min read · Apr 20, 2021

6

See all from Hitesh Mishra

Recommended from Medium


Let say, user type "recommend me the best place in denpasar!"
Intent Classification (ALBERT + CNN)
predictedIntent = "recommender"
Named Entity Recognition (spaCy)
recommend me the best place in denpasar REGION !
Recommender System

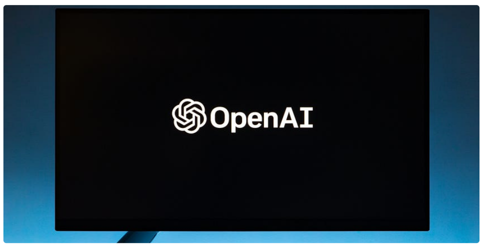
Muhamad Luthfi Rey

Building a Smart Chatbot with Intent Classification and Named...

Indonesia attracts travelers worldwide with its vibrant culture, breathtaking landscapes, an...

7 min read · Jul 14, 2023

48



InnovatewithDataScience

Create an Generative-AI chatbot using Python and Flask: A step by...

Introduction

4 min read · Sep 4, 2023

220 2

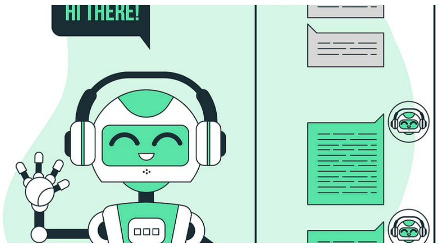
Lists

ChatGPT
23 stories · 372 saves

Predictive Modeling w/ Python
20 stories · 756 saves

AI Regulation
6 stories · 259 saves

ChatGPT prompts
34 stories · 916 saves



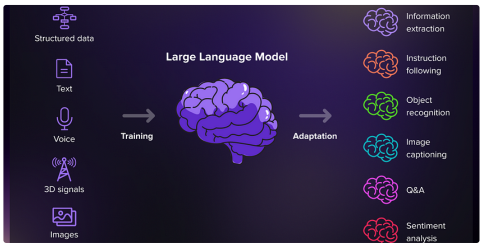
Gaurav Garg

How to Create Your Own Chatbot in 2023

Chatbots have become an integral part of many businesses' customer service and...

7 min read · Aug 28, 2023

11



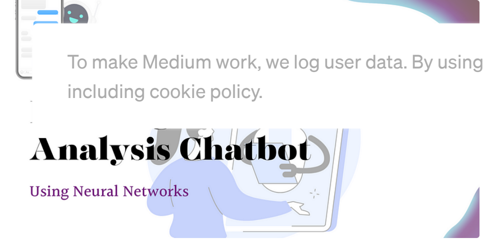
Alisha Saboowala

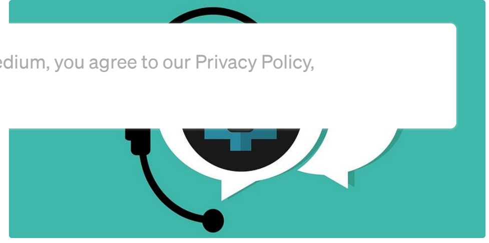
Build Your First LLM Chatbot

A beginner's guide to Large Language Models, LangChain, Vector DB's, and more

10 min read · Aug 10, 2023

75 3





Jude Nwabu... in Artificial Intelligence in Plain En...

Building a Sentiment Analysis Chatbot Using Neural Networks

Understanding and responding to user sentiments is crucial to building engaging a...

8 min read · Aug 31, 2023

48 1

Ramakrushna Mohapatra

Chatbot: Building From Scratch

Please read from start and end. If you will follow what I have mentioned here, you will...

15 min read · Aug 28, 2023

1

See more recommendations