



Published in Towards Data Science

---

You have **1** free member-only story left this month. [Sign up for Medium and get an extra one](#)



Kurtis Pykes

Follow

Mar 31, 2021 · 8 min read · ✨ · 🎧 Listen



Save



NATURAL LANGUAGE PROCESSING NOTES

# Build A Simple Chatbot In Python With Deep Learning

How To Build Your Own Friend



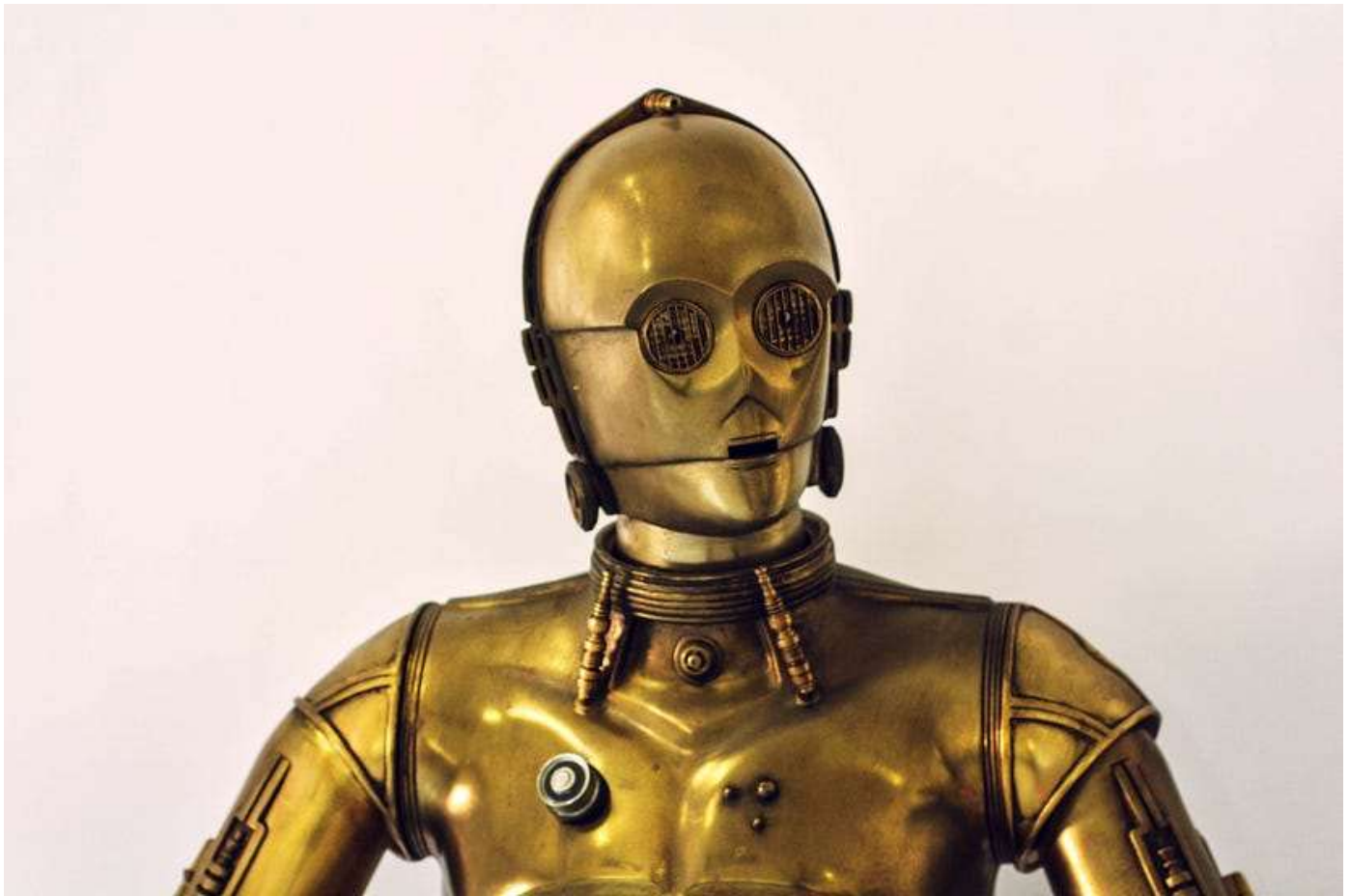


Photo by [Lyman Gerona](#) on [Unsplash](#)

Artificial Intelligence is rapidly creeping into the workflow of many businesses across various industries and functions. Due to advancements in Natural Language Processing (NLP), Natural Language Understanding (NLU), and Machine Learning (ML), humans are now able to develop technologies that are capable of imitating human-like interactions which include recognizing speech, as well as text.

### **A Primer On Conversational AI**

Learning About Conversational AI and How It Can Help Humans

[towardsdatascience.com](https://towardsdatascience.com)



In this article, we are going to build a Chatbot using NLP and Neural Networks in Python.

### **The Data**

Before we can begin to think of any coding, we need to set up an intents JSON file that defines certain intentions that could occur during the interactions with our chatbot. To perform this we would have to first create a set of tags that users queries may fall into. For example:

- A user may wish to know the name of our chatbot, therefore we create an intention labeled with a tag called `name`
- A user may wish to know the age of our chatbot, therefore we create an intention labeled with the tag `age`
- etc etc

For each of the tags that we create, we would have to specify patterns. Essentially, this defines the different ways of how a user may pose a query to our chatbot. For instance, under the `name` tag, a user may ask someone's name in a variety of ways — “What’s your name?”, “Who are you?”, “What are you called?”.

The chatbot would then take these patterns and use them as training data to determine what someone asking for our chatbot's name would look like so that it could adapt to the different ways someone may ask to know our bot's name. Therefore, users wouldn’t have to use the exact queries that our chatbot has learned. It could pose the question as “What are you called?” and our chatbot would be able to infer that the user wants to know the name of our chatbot and then it would provide its name.


*Note: Our bot is not going to be super-duper intelligent so it would not always recognize what we are saying, but with enough examples, it would do a pretty decent job at deciphering.*

Within this intents JSON file, alongside each intents tag and pattern, there will be responses. However, for our simple chatbot, these responses are not going to be generated. What this means is that our patterns aren’t going to be as free-flowing as the patterns users can ask (it will not adapt), instead, the responses will be using static responses that the chatbot will return when posed with a query.



# used a dictionary to represent an intents JSON file

```
data = {"intents": [
    {"tag": "greeting",
      "patterns": ["Hello", "How are you?", "Hi there", "Hi",
        "Whats up"],
      "responses": ["Howdy Partner!", "Hello", "How are you
        doing?", "Greetings!", "How do you do?"]},
    {"tag": "age",
      "patterns": ["how old are you?", "when is your
        birthday?", "when was you born?"],
      "responses": ["I am 24 years old", "I was born in 1996",
        "My birthday is July 3rd and I was born in 1996", "03/07/1996"]},
    {"tag": "date",
      "patterns": ["what are you doing this weekend?",
        "do you want to hang out some time?", "what are your plans for this
        week"],
      "responses": ["I am available all week", "I don't have
        any plans", "I am not busy"]},
    ... ..
  ]}
```

Open in app 

Sign up

Sign In



Search Medium



```
    {"tag": "goodbye",
      "patterns": [ "bye", "g2g", "see ya", "adios", "cya"],
      "responses": ["It was nice speaking to you", "See you
        later", "Speak soon!"]},
    ]}
  ]}
```

## The Code

To code our bot, we are going to require some Python built-ins, as well as popular libraries for NLP, deep learning, as well as the defacto library NumPy which is great for dealing with arrays.

```
import json
import string
import random

import nltk
import numpy as np
from nltk.stem import WordNetLemmatizer

import tensorflow as tf
from tensorflow.keras import Sequential
```



```
from tensorflow.keras.layers import Dense, Dropout

nltk.download("punkt")
nltk.download("wordnet")
```

In order to create our training data, there is first some things we must do to our data .  
Here is the list:

- Create a vocabulary of all of the words used in the patterns (recall the patterns are the queries posed by the user)
- Create a list of the classes — This is simply the tags of each intent
- Create a list of all the patterns within the intents file
- Create a list of all the associated tags to go with each pattern in the intents file

Let's see this in Python...



295

| 5

```
# initializing lemmatizer to get stem of words
lemmatizer = WordNetLemmatizer()

# Each list to create
words = []
classes = []
doc_X = []
doc_y = []

# Loop through all the intents
# tokenize each pattern and append tokens to words, the patterns and
# the associated tag to their associated list

for intent in data["intents"]:
    for pattern in intent["patterns"]:
        tokens = nltk.word_tokenize(pattern)
        words.extend(tokens)
        doc_X.append(pattern)
        doc_y.append(intent["tag"])

    # add the tag to the classes if it's not there already
    if intent["tag"] not in classes:
        classes.append(intent["tag"])

# lemmatize all the words in the vocab and convert them to lowercase
# if the words don't appear in punctuation
```



```

words = [lemmatizer.lemmatize(word.lower()) for word in words if word
not in string.punctuation]

# sorting the vocab and classes in alphabetical order and taking the #
set to ensure no duplicates occur
words = sorted(set(words))
classes = sorted(set(classes))

```

This is how each list looks...

```

print(words)
>>> ['s', 'adios', 'are', 'birthday', 'born', 'bye', 'called',
'cya', 'do', 'doing', 'for', 'g2g', 'hang', 'hello', 'help',
'hi', 'how', 'is', 'later', 'name', 'old', 'out', 'plan',
'see', 'some', 'talk', 'thanks', 'there', 'this', 'time', 'to',
'up', 'wa', 'want', 'week', 'weekend', 'what', 'whats', 'when',
'who', 'ya', 'you', 'your']

print(classes)
>>> ['age', 'date', 'goodbye', 'greeting', 'name']

print(doc_X)
>>> ['Hello', 'How are you?', 'Hi there', 'Hi', 'Whats up', 'how
old are you?', 'when is your birthday?', 'when was you born?',
'what are you doing this weekend?', 'do you want to hang out some
time?', 'what are your plans for this week', "what's your name?",
'what are you called?', 'who are you?', 'Thanks for your help',
'bye', 'g2g', 'see ya', 'talk later', 'adios', 'cya']

print(doc_y)
>>> ['greeting', 'greeting', 'greeting', 'greeting', 'greeting',
'age', 'age', 'age', 'date', 'date', 'date', 'name', 'name',
'name', 'goodbye', 'goodbye', 'goodbye', 'goodbye', 'goodbye',
'goodbye', 'goodbye']

```

Now that we've separated our data, we are now ready to train our algorithm. However, Neural Networks expect numerical values, and not words, to be fed into them, therefore, we first have to process our data so that a neural network could read what we are doing.



In order to convert our data to numerical values, we are going to leverage a technique called bag of words. For more information on cleaning text and representing text as numerical values, check out my 2 posts that detail these techniques and how to perform them in Python.

## A Guide To Cleaning Text in Python

Preparing Natural Language for Machines To Read

[towardsdatascience.com](https://towardsdatascience.com)

## A Guide To Encoding Text In Python

Teaching Computers to Comprehend Human Language

[towardsdatascience.com](https://towardsdatascience.com)

```
# list for training data
training = []
out_empty = [0] * len(classes)

# creating the bag of words model
for idx, doc in enumerate(doc_X):
    bow = []
    text = lemmatizer.lemmatize(doc.lower())
    for word in words:
        bow.append(1) if word in text else bow.append(0)

    # mark the index of class that the current pattern is associated
    # to
    output_row = list(out_empty)
    output_row[classes.index(doc_y[idx])] = 1

    # add the one hot encoded BoW and associated classes to training
    training.append([bow, output_row])

# shuffle the data and convert it to an array
random.shuffle(training)
training = np.array(training, dtype=object)

# split the features and target labels
train_X = np.array(list(training[:, 0]))
train_y = np.array(list(training[:, 1]))
```



With our data converted into a numerical format, we can now build a Neural Network model that we are going to feed our training data into. The idea is that the model will look at the features and predict the tag associated with the features then will select an appropriate response from that tag.

```
# defining some parameters
input_shape = (len(train_X[0]),)
output_shape = len(train_y[0])
epochs = 200

# the deep learning model
model = Sequential()
model.add(Dense(128, input_shape=input_shape, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(output_shape, activation = "softmax"))

adam = tf.keras.optimizers.Adam(learning_rate=0.01, decay=1e-6)

model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=["accuracy"])
print(model.summary())
model.fit(x=train_X, y=train_y, epochs=200, verbose=1)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	5120
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 5)	325
Total params: 13,701		
Trainable params: 13,701		
Non-trainable params: 0		

Image By Author



In our sequential model, we used some dropout layers which are very effective at preventing deep learning models from overfitting to data. In another article, I went into slightly more depth about it.

### **Combating Overfitting in Deep Learning**

Stopping Your Deep Learning Model From Overfitting

[towardsdatascience.com](https://towardsdatascience.com)

Great. We've trained our Deep Learning model, but now we have to create the actual features that would allow us to use our model in a chatbot application. For this next task, I've created a set of utility functions that would allow us to easily perform this task.



The next part is simple. We must create a while loop that allows a user to input some query which is then cleaned, meaning we take the tokens and lemmatize each word. After that, we convert our text to numeric values using our bag of words model and make a prediction of what tag in our intents the features best represent. From there, we would take a random response from our responses within that intents tag and use that to respond to the query.



```
# running the chatbot
while True:
```

```
message = input("")
intents = pred_class(message, words, classes)
result = get_response(intents, data)
print(result)
```

Let's see our chatbot in action...



## Wrap Up

You've now seen how to create a very simple chatbot in Python using Deep Learning and NLP techniques. This in no way is an elegant Chatbot that you'd want to take to a hiring manager or to put on your CV — in my opinion — but it's a great start for anyone intrigued by conversational AI. If you happen to create a chatbot using this article as a guide, be sure to show me some previews of it on either my LinkedIn or Twitter of which I'll link below.

Thank you for reading! Connect with me on [LinkedIn](#) and [Twitter](#) to stay up to date with my posts about Data Science, Artificial Intelligence, and Freelancing.



## Related Articles

### Never Forget These 8 NLP Terms

Terminology All NLP Enthusiast Should Know

towardsdatascience.com

## 5 Ideas For Your Next NLP Project

Generating Some Ideas

towardsdatascience.com

## 7 Free Online Resources For NLP Lovers

Be Sure To Bookmark Them

pub.towardsai.net

Artificial Intelligence

Technology

Innovation

Ideas

Nlpnotes

---

## Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.



Get this newsletter

[About](#) [Help](#) [Terms](#) [Privacy](#)

Get the Medium app

