Proje kodları açıklamalı

Merhaba bugün Makine öğrenmesinden faydalanarak basit bir chatbot yapmayı analiz edeceğiz.

Chatbotlar doğal dil anlama ve doğal dil oluşturma methotlarını kullanır.

Öncelikle gerekli kütüphaneleri yüklüyoruz.

Kütüphane yükleme

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle
nltk.download("punkt")
```

Yüklenen kütüphaneleri anlama

- **Numpy**

Numpy kütüphanesi bize diziler, matrisler rastgele sayılar oluşturma ve çeşitli sayısal konularda yardımcı olur.

- **Keras**

Keras pyton kaynaklı bir sinir ağı kütüphanesidir.

- **NLTK**

NLTK doğal dil işleme için bir araç kitidir. bu kit kelimeleri ayırma (Tokenization) , kök bulma (Sremming) gibi işlemleri içerir.

WordNetLemmatizer, doğal dil işleme uygulamalarında sıklıkla kullanılan bir lemmatizasyon aracıdır. Lemmatizasyon, kelimenin kökünü bulmak için kullanılan bir yöntemdir.

WordNetLemmatizer, Python dilindeki nltk (Natural Language Toolkit) kütüphanesinde bulunur. Bu araç, bir kelimenin kökünü bulmak için WordNet adlı bir İngilizce sözlük kullanır. Kelimenin eş anlamlıları ve ilişkili kelimeler de WordNet tarafından sağlanır. WordNetLemmatizer, kelimenin eş anlamlılarına göre kelime köklerini bulmak için kullanılan bir yöntem olan morfolojik analiz yapar.

Örneğin, "running" kelimesi için WordNetLemmatizer, "run" kelimesini kök olarak bulacaktır. Bu, metin tabanlı uygulamalarda kelime sınıflandırması, anlamsal analiz, kelime benzerliği gibi çeşitli görevler için önemlidir.

Python'da `import json`
komutu, JSON (JavaScript Object
Notation) veri formatını işlemek için kullanılan bir kütüphaneyi yükler.
JSON, veri değiş tokuşu ve depolama için yaygın olarak kullanılan bir
formattır.

- **Pickle**

Pickle, Python'da bulunan bir modül ve veri serileştirme işlemi için kullanılan bir protokoldür. Serileştirme, verilerin bir dosyaya yazılması veya ağ üzerinden başka bir makineye gönderilmesi gibi işlemlerde kullanılır.

Pickle modülü, Python nesnelerini, yani listeleri, sözlükleri, sınıfları ve hatta fonksiyonları da dahil olmak üzere her türlü nesneyi serialize etmek ve deserialize etmek için kullanılabilir. Serialize etmek, bir nesneyi bir bayt dizesi gibi bir biçimde temsil etmektir. Deserialize etmek ise, bu bayt dizisini tekrar bir Python nesnesine dönüştürmektir.

Pickle, kullanımı kolay ve esnek bir serileştirme aracıdır. Özellikle büyük veri setleri ile çalışırken veya verileri ağ üzerinden gönderirken kullanışlıdır. Ancak, dikkatli kullanılmadığında güvenlik açıkları doğurabilir. Bu nedenle, güvenilir kaynaklardan gelen verileri pickle etmek ve depickle etmek önemlidir.

İntent.json dosyasında belirli cevaplar ve yanıtları sıralanmış

```
intents_file = open("intents.json").read() #intents.json dosyasını okuma
intents = json.loads(intents_file) #json dosyasını okuma
```

bazı boş listeler ve görmezden gelinmesi gereken noktalam işaretleri için atamalar

```
words = [] #kelimeleri tutmak için boş liste
classes = [] #sınıfları tutmak için boş liste
documents = [] #dokümanları tutmak için boş liste
ignore_letters = ["!","?",",","."] #ignore_letters değişkenine !,?,.,, değerlerini atama
```

daha sonra doğal dil işleme projesinde kullanılan eğitim verilerini hazırlanır.

Veriler, bir JSON dosyasından yüklenir ve ardından her bir niyetin (intent) ve her bir niyetin farklı örneklerinin (patterns) altındaki cümleleri işlenir.

Her bir cümle, nltk.word_tokenize() fonksiyonu kullanılarak ayrıştırılır ve ardından elde edilen kelimeler, `words` adlı bir liste içinde toplanır. Bu kelimeler, ayrı bir liste içinde `documents` adlı bir liste içinde saklanır. Her bir örnek, kelimeleri ve niyeti (`intent["tag"]`) birleştirerek bir tuple olarak `documents` listesine eklenir. Ayrıca, her niyetin bir listesi de `classes` adlı bir liste içinde toplanır.

Son olarak, `documents` değişkeni yazdırılır, yani her bir örnek (örneğin, bir kullanıcının sormuş olabileceği bir soru) ve onun hangi niyetle ilişkili olduğu görüntülenir. Bu veriler daha sonra makine öğrenimi modellerinin eğitiminde kullanılabilir.

```
for intent in intents["intents"]: #intents["intents"] değişkenindeki değerleri intent değişkenine atama
    for pattern in intent["patterns"]: #    intent["patterns"] değişkenindeki değerleri pattern değişkenine atama

        word = nltk.word_tokenize(pattern)  #word değişkenine pattern değişkenini tokenize etme
        words.extend(word)#words değişkenine word değişkenini ekleme
        documents.append((word,intent["tag"])) #documents değişkenine word ve intent["tag"] değerlerini ekleme
        # sınıflarımızı listeye ekleme
        if intent["tag"] not in classes: #intent["tag"] değişkenindeki değerler classes değişkeninde yoksa
            classes.append(intent["tag"]) #classes değişkenine intent["tag"] değişkenini ekleme
print(documents) #documents değişkenini yazdırma
```

bazı hataları gidermek için

```
#nlt wordnet error
nltk.download("wordnet") #wordnet kütüphanesini indirme

#nltk nltk.download('omw-1.4') error
nltk.download('omw-1.4')
```

```
## lemmatize and lower each word and remove duplicates #lemmatize ve her kelimeyi küçültme ve tekrar edenleri kaldırma
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_letters] #words değişkenine lemmatizer kütüphanesini kullanarak
words = sorted(list(set(words))) #words değişkenine sıralama ve tekrar edenleri kaldırma
#sort classes
classes = sorted(list(set(classes))) #classes değişkenine sıralama ve tekrar edenleri kaldırma
#documents = combination between patterns and intents #documents = patterns ve intents arasındaki kombinasyon
print(len(documents),"documents") #documents değişkeninin uzunluğunu yazdırma
#classes = intents
print(len(classes),"classes",classes) #classes değişkeninin uzunluğunu yazdırma
#words = all words, vocabulary
print(len(words),"unique lemmatized words", words) #words değişkeninin uzunluğunu yazdırma

pickle.dump(words,open("words.pkl","wb")) #words değişkenini words.pkl dosyasına yazma
pickle.dump(classes,open("classes.pkl","wb"))
```

Bu kod bloğu, doğal dil işleme projelerinde kullanılan bir terim "lemmatization" işlemi ile ilgili işlemleri gerçekleştirir. Burada, projede kullanılacak olan veri kümesindeki tüm kelimelerin lemmatize edilmesi, küçük harfe dönüştürülmesi ve tekrar eden kelimelerin kaldırılması işlemleri yapılır.

Öncelikle, nltk kütüphanesi içinde yer alan `WordNetLemmatizer()` sınıfı kullanılarak, `words` listesi içindeki tüm kelimeler lemmatize edilir ve `ignore_letters` adlı bir listede yer alan harfler dikkate alınmaz. Lemmatize edilmiş kelimelerin küçük harfe çevrilmesi, `lower()` fonksiyonu kullanılarak gerçekleştirilir. Bu işlem, kelime köklerini elde etmek için yapılır.

Daha sonra, `words` listesi sıralanır ve `set()` fonksiyonu ile tekrar eden kelimeler kaldırılır. Elde edilen kelimeler, `words` listesi içinde saklanır. Aynı işlem `classes` listesi için de uygulanır.

`documents` adlı bir liste, `patterns` ve `intents` verilerinin birleştirilmesi ile oluşturulur. Bu veriler, daha sonra makine öğrenimi modellerinin eğitimi için kullanılacak olan eğitim verileri olarak kullanılabilir.

Son olarak, `documents`, `classes` ve `words` listelerinin uzunluğu ekrana yazdırılır ve ayrıca `words` ve `classes` listeleri, `pickle` kütüphanesi kullanılarak `words.pkl` ve `classes.pkl` dosyalarına yazdırılır. Bu dosyalar, daha sonra modellerin eğitimi sırasında kullanılabilir.

```
#create the trainin data
training = [] #boş liste oluşturma
#create empty array for the output #çıktı için boş dizi oluşturma
output_empty = [0]*len(classes) #output_empty değişkenine classes değişkeninin uzunluğu kadar 0 değeri atama
#trainin set, bag of words for every sentence #trainin set, her cümle için bag of words
for doc in documents: #documents değişkenindeki değerleri doc değişkenine atama
    #initializing bag of words
    bag = [] #bag değişkenine boş liste atama
    #list of tokenized words for the pattern
    word_patterns = doc[0] #word_patterns değişkenine doc[0] değişkenini atama
    #lemmatize each word - create base word, in attempt to represent related words #her kelimeyi lemmatize etme - temel kelimeyi oluşturma,
    word_patterns = [lemmatizer.lemmatize(word.lower()) for word in word_patterns] #word_patterns değişkenine lemmatizer kütüphanesini kull
    # create the bag of words array with 1, şf words şs found in current pattern #1 ile bag of words dizisini oluşturma, bulunan kelimeler
    for word in words: #words değişkenindeki değerleri word değişkenine atama
        bag.append(1) if word in word_patterns else bag.append(0) #bag değişkenine word değişkenindeki değerlerin word_patterns değişkenind

    #output is a "0 for each tag and "1" for current tag (for each pattern)
    output_row = list(output_empty) #output_row değişkenine output_empty değişkenini atama
    output_row[classes.index(doc[1])] = 1 #output_row değişkenine doc[1] değişkenindeki değerin indexini atama
    training.append([bag,output_row]) #training değişkenine bag ve output_row değerlerini ekleme
#shuffle the features and make numpy array
random.shuffle(training) #training değişkenindeki değerleri karıştırma
training = np.array(training) #training değişkenini numpy dizisine çevirme
#create training and testing lists. X- patterns, Y - intents
train_x = list(training[:,0]) #train_x değişkenine training değişkenindeki 0. değerleri atama
train_y = list(training[:,1]) #train_y değişkenine training değişkenindeki 1. değerleri atama
print("Trainin data is created") #trainin data is created yazdırma
```

Bu kod bloğu, bir chatbot için eğitim verileri oluşturmak için kullanılır.

Öncelikle, "documents" adlı bir liste oluşturulur ve bu listede her bir öğe, bir cümle ve bu cümlenin hangi amaçla kullanıldığını belirten bir etiket içerir.

Daha sonra, "classes" adlı bir liste oluşturulur ve bu liste, chatbot'un anlamaya çalışacağı farklı amaçları temsil eden etiketler içerir.

Eğitim verileri oluşturmak için bir döngü kullanılır. Her bir belge ("doc"), "bag" adlı bir boş liste oluşturulur ve cümlenin kelimeleri üzerinde dolaşarak her bir kelimenin lemmatize edilmiş hali "word_patterns" adlı bir liste oluşturulur. Daha sonra, bir iç içe döngü kullanılarak, her bir kelime "words" listesinde dolaşılır ve bu kelime, "word_patterns" listesinde varsa "1" olarak "bag" listesine eklenir, aksi takdirde "0" olarak eklenir. Bu işlem, belge için bir "bag of words" (kelimelerin torbası) modeli oluşturur.

"output_row" adlı bir boş liste oluşturulur ve bu liste, "classes" listesinin uzunluğu kadar sıfır içerir. Daha sonra, belgenin etiketi, "classes" listesindeki indeksine karşılık gelen "1" değeri olarak "output_row" listesine eklenir.

Son olarak, "training" adlı bir liste oluşturulur ve bu liste, her bir belgenin "bag" listesi ve "output_row" listesi olarak bir öğe içerir. "training" listesi karıştırılır ve "train_x" ve "train_y" listeleri oluşturulur. Bu iki liste, sırasıyla, "bag" listeleri ve "output_row" listelerini içerir ve chatbot'un eğitiminde kullanılır.

```
#deep neural networds model #derin sinir ağı modeli
model = Sequential() #model değişkenine Sequential kütüphanesini atama
model.add(Dense(128,input_shape=(len(train_x[0]),), activation="relu")) #model değişkenine Dense kütüphanesini kullanarak 128 değerini atam
model.add(Dropout(0.5)) #model değişkenine Dropout kütüphanesini kullanarak 0.5 değerini atama
model.add(Dense(64,activation="relu")) #model değişkenine Dense kütüphanesini kullanarak 64 değerini atama
model.add(Dropout(0.5)) #model değişkenine Dropout kütüphanesini kullanarak 0.5 değerini atama
model.add(Dense(len(train_y[0]), activation="softmax")) #model değişkenine Dense kütüphanesini kullanarak len(train_y[0]) değerini atama
```

Bu kod bloğunda, bir yapay sinir ağı modeli oluşturuluyor. İlk olarak, Sequential() fonksiyonu kullanılarak bir model nesnesi oluşturuluyor. Daha sonra, add() fonksiyonu ile bu model nesnesine sırayla katmanlar ekleniyor.

İlk katmanda, input_shape parametresi ile giriş boyutu belirtiliyor. Burada, train_x dizisindeki her bir öğenin boyutu len(train_x[0]) olarak belirtiliyor. Bu değer, eğitim verilerinin boyutunu temsil ediyor. Dense() fonksiyonu kullanılarak, 128 nöronlu bir katman oluşturuluyor. Activation parametresi "relu" olarak belirtiliyor. Bu, ReLU (Rectified Linear Unit) aktivasyon fonksiyonunun kullanılacağı anlamına geliyor.

İkinci katman, Dropout() fonksiyonu ile bir dropout katmanıdır. Dropout, overfittingi önlemek için kullanılan bir tekniktir. 0.5 değeri, her bir nöronun %50'sinin rastgele olarak bırakılacağı anlamına gelir.

Üçüncü katman, yine Dense() fonksiyonu kullanılarak, 64 nöronlu bir katmandır. ReLU aktivasyon fonksiyonu kullanılır.

Dördüncü katman, tekrar Dropout() fonksiyonu ile bir dropout katmanıdır.

Son katmanda, Dense() fonksiyonu ile çıkış boyutu belirtilir. Burada, len(train_y[0]) olarak belirtilir. Bu, çıkış katmanındaki nöron sayısını temsil eder. Activation parametresi "softmax" olarak belirtilir. Bu, çıkışın sınıflandırma için kullanılacağını gösterir.

Bu şekilde, bir yapay sinir ağı modeli oluşturulmuş olur.

```
#Compliling mode. SGD with Nesterov accelerated gradient gives good result for this model #Compliling mode. SGD with Nesterov hızlandırılmı
sgd = SGD(lr=0.01 ,decay = 1e-6, momentum = 0.9, nesterov = True ) #sgd değişkenine SGD kütüphanesini kullanarak değerleri atama
model.compile(loss="categorical_crossentropy", optimizer = sgd) #model değişkenine compile kütüphanesini kullanarak değerleri atama
metrics = ["accuracy"] #metrics değişkenine accuracy değerini atama
```

Bu kod bloğu, bir yapay sinir ağı modeli oluşturur ve derler.

İlk önce, "Sequential" modelini oluşturur ve ardından "Dense" katmanları ekler. "Dense" katmanı, tüm giriş birimlerinin tüm çıkış birimlerine tamamen bağlandığı bir sinir ağı katmanıdır. İlk "Dense" katmanı 128 birim ve "relu" aktivasyon fonksiyonuna sahiptir. İkinci "Dense" katmanı 64 birime ve yine "relu" aktivasyon fonksiyonuna sahiptir. Son "Dense" katmanı, çıkış sınıflarının sayısına göre birim sayısına sahip ve "softmax" aktivasyon fonksiyonuna sahiptir.

Daha sonra, "SGD" optimizer kullanılarak model derlenir. "SGD", Stokastik Gradient Descent'in kısaltmasıdır ve sinir ağlarının eğitiminde sıklıkla kullanılan bir optimizasyon algoritmasıdır. Bu kod bloğunda, öğrenme oranı (lr) 0.01 olarak ayarlanmış, ağırlık azalımı (decay) 1e-6, momentum 0.9 ve Nesterov momentumu True olarak ayarlanmıştır.

Son olarak, modelin performansını ölçmek için "accuracy" ölçütü (metrics) kullanılmıştır.

```
#Training and saving the model #Modeli eğitme ve kaydetme
hist = model.fit(np.array(train_x),np.array(train_y), epochs = 200, batch_size =  5, verbose = 1) #hist değişkenine modeli eğitme
model.save("chatbot_model.h5",hist) #modeli chatbot_model.h5 dosyasına kaydetme
print("model is created") #model is created yazdırma
```

Bu kod bloğu, oluşturulan modelin eğitimini gerçekleştirir ve eğitim sonrasında oluşan modeli "chatbot_model.h5" adlı dosyaya kaydeder.

`model.fit()` metodu, eğitim verilerini (`train_x` ve `train_y`) kullanarak modelin eğitimini gerçekleştirir. `epochs` parametresi, tüm verilerin model tarafından kaç kez kullanılacağını belirler. `batch_size` parametresi, her eğitim adımında kaç örnek kullanılacağını belirler. `verbose` parametresi, eğitim sırasında gösterilecek loglama seviyesini belirler.

`model.save()` metodu, eğitim sonrasında oluşan modeli diskteki bir dosyaya kaydeder. Kaydedilen model, ileride farklı programlarda kullanılabilir hale gelir.

Son olarak, `print()` fonksiyonu ile modelin başarıyla oluşturulduğu kullanıcıya bildirilir.

Böylece chatbotun ana kodu oluşturulmuş olur sonrasında arayüz oluşturulur

---

Arayüz oluşturma

```
import tkinter
from tkinter import *
```

Tkinter, Python programlama dili için standart bir GUI (Graphical User Interface) kütüphanesidir. Tkinter, Tk GUI toolkit'inin Python diline uyarlanmış halidir ve bu nedenle Tkinter genellikle Python ile birlikte gelir.

Tkinter, çeşitli GUI bileşenlerini oluşturmak ve düzenlemek için kullanılabilir. Bu bileşenler arasında düğmeler, metin kutuları, etiketler, menüler, liste kutuları ve çerçeveler gibi birçok standart bileşen bulunur. Tkinter ayrıca çizimler, grafikler ve animasyonlar gibi daha gelişmiş özellikler için de kullanılabilir.

Tkinter, kullanımı kolay ve öğrenmesi kolaydır. Başlangıç seviyesindeki kullanıcılar için idealdir, ancak gelişmiş kullanıcılar tarafından da kullanılabilir. Tkinter, çok sayıda örnek uygulama ve dokümantasyon içerir, bu da kullanıcıların Tkinter'ı kullanarak kendi uygulamalarını oluşturmalarını kolaylaştırır.

Tkinter ayrıca çok platformlu bir kütüphanedir. Bu, Tkinter kullanılarak geliştirilen uygulamaların Windows, Linux ve Mac gibi farklı işletim sistemlerinde çalışabileceği anlamına gelir

ython'da Tkinter GUI kütüphanesindeki tüm sınıfları ve işlevleri kodunuzda doğrudan kullanılabilir hale getirir.

"from tkinter" ifadesi, tkinter kütüphanesini kodunuza dahil etmek için kullanılır. "import *" ifadesi, tkinter kütüphanesindeki tüm sınıfların ve fonksiyonların kodunuzda doğrudan kullanılabilir hale getirilmesini sağlar.

```
def send(): #mesaj gönderme fonksiyonu
    msg = EntryBox.get("1.0","end-1c").strip() #EntryBox deki değerleri msg değişkenine atama
    EntryBox.delete("0.0",END) #EntryBox deki değerleri silme

    if msg !="": #  msg değişkeninde değer varsa
        ChatBox.config(state= DISABLED) #ChatBox deki değerleri devre dışı bırakma
        ChatBox.inters(END,"You: "+msg+ "\n\n") #ChatBox deki değerleri ekleme
        ChatBox.config(foreground="#446665",font = ("Verdana",12)) #ChatBox deki yazı tipini ve yazı rengini değiştirme
        ints = predict_class(msg) #predict_class fonksiyonunu msg değişkeni ile çağırma
        res = getResponse(ints,intents) #getResponse fonksiyonunu ints ve intents değişkenleri ile çağırma

        ChatBox.insert(END, "Bot: "+ res + "\n\n") #ChatBox deki değerleri ekleme
        ChatBox.config(state=DISABLED)  #ChatBox deki değerleri devre dışı bırakma
        ChatBox.yview(END) #ChatBox deki değerleri sona götürme
```

Bu kod, bir sohbet botunun kullanıcının mesajını alıp yanıtlamasını sağlayan bir fonksiyondur.

Fonksiyonun adı "send()" ve hiçbir parametre almaz. Fonksiyon, kullanıcının girdiği mesajı alır ve bu mesajı "msg" adlı bir değişkene atar. Daha sonra, mesajın girildiği metin kutusunu temizler.

Eğer mesaj boş değilse, "if msg != "":", kod bloğuna girer. ChatBox adlı bir metin kutusu devre dışı bırakılır ve ardından kullanıcının mesajını içeren bir etiket oluşturulur ve ChatBox'a eklenir. Etiketin yazı rengi "#446665" ve yazı tipi "Verdana" ve boyutu 12'dir.

Sonra, kullanıcının mesajını anlamlandırmak için "predict_class()" fonksiyonu çağrılır ve ardından botun yanıtı almak için "getResponse()" fonksiyonu çağrılır. Bu yanıt ChatBox'a eklenir ve tekrar devre dışı bırakılır. ChatBox, son eklenen mesajı görüntülemek için en aşağı kaydırılır.

Bu kod, bir sohbet botu uygulamasında kullanılabilir ve kullanıcının mesajlarını alıp yanıtlarını göstermek için Tkinter kullanır.

```
root =Tk() #Tkinter kütüphanesini çağırma
root.title("Chatbot") #Pencere başlığını değiştirme
root.geometry("400x500")   #boyutunu değiştirme
root.resizable(width=FALSE,height=FALSE) #frame boyutunu değiştirme
```

Bu kod, bir Tkinter penceresi oluşturur ve bazı ayarları yapar.

İlk olarak, "Tk()" fonksiyonu çağrılarak bir Tkinter nesnesi oluşturulur ve "root" adlı bir değişkene atanır. Bu nesne, Tkinter uygulamasında ana pencereyi temsil eder.

Ardından, "title()" yöntemi çağrılarak pencere başlığı "Chatbot" olarak ayarlanır.

"geometry()" yöntemi, pencerenin boyutunu "400x500" piksel olarak ayarlar.

"resizable()" yöntemi, pencere boyutunun sabit kalmasını sağlar. width=FALSE, genişliğin değiştirilemeyeceğini, height=FALSE ise yüksekliğin değiştirilemeyeceğini belirtir. Bu yöntem, pencere boyutunun kullanıcı tarafından değiştirilmesini önlemek için kullanılır.

Bu kodlar, Tkinter kullanarak bir sohbet botu uygulaması için bir pencere oluşturmak için kullanılabilir.

```
#cREATE CHAT WİNDOW
ChatBox = Text(root, bd= 0, bg = "white",height= "8", width = "50", font = "Arial",) #ChatBox değişkenine Text fonksiyonunu çağırma
ChatBox.config(state = DISABLED) #ChatBox deki değerleri devre dışı bırakma
```

Bu kod, bir metin kutusu oluşturur ve bazı ayarları yapar.

"Text()" fonksiyonu, Tkinter'da metin kutusu oluşturmak için kullanılır. Fonksiyon, metin kutusunun bulunacağı ana pencereyi belirtmek için "root" adlı bir parametre alır.

"bd" parametresi, metin kutusunun kenarlık genişliğini belirler. Bu örnekte "bd=0" olarak ayarlandığı için kenarlık yoktur.

"bg" parametresi, metin kutusunun arka plan rengini belirler. Bu örnekte "bg = "white"" olarak ayarlandığı için arka plan rengi beyazdır.

"height" ve "width" parametreleri, metin kutusunun boyutunu belirler. Bu örnekte "height=8" ve "width=50" olarak ayarlandığı için metin kutusu 8 satır ve 50 karakter genişliğinde olacaktır.

"font" parametresi, metin kutusunun yazı tipini belirler. Bu örnekte "Arial" olarak ayarlandığı için yazı tipi Arial'dır.

"config()" yöntemi, belirli özellikleri ayarlamak için kullanılır. "state" parametresi, metin kutusunun kullanıcı tarafından düzenlenebilir olup olmayacağını belirler. Bu örnekte "DISABLED" olarak ayarlandığı için, metin kutusundaki veriler yalnızca program tarafından değiştirilebilir ve kullanıcı tarafından değiştirilemez.

Bu kodlar, bir sohbet botu uygulamasında kullanılan metin kutusunu oluşturmak için kullanılabilir.

```
#Bind scrollbar to chat window
scrollbar = Scrollbar(root, command = ChatBox.yview, cursor = "heart")
ChatBox["yscrollcommand"] = scrollbar.set
```

Bu kod, bir kaydırma çubuğu oluşturur ve metin kutusuna bağlar.

"Scrollbar()" fonksiyonu, Tkinter'da kaydırma çubuğu oluşturmak için kullanılır. Fonksiyon, kaydırma çubuğunun bulunacağı ana pencereyi belirtmek için "root" adlı bir parametre alır.

"command" parametresi, kaydırma çubuğunun yönünü belirler. Bu örnekte "ChatBox.yview" olarak ayarlandığı için, kaydırma çubuğu metin kutusunun yönünü takip edecektir.

"cursor" parametresi, kaydırma çubuğunun işaretçi şeklini belirler. Bu örnekte "cursor = "heart"" olarak ayarlandığı için kaydırma çubuğu işaretçi şekli kalp şeklindedir.

"yscrollcommand" özelliği, metin kutusundaki kaydırmayı kontrol etmek için kullanılır. Bu örnekte "ChatBox["yscrollcommand"] = scrollbar.set" satırı, metin kutusundaki kaydırmayı kaydırma çubuğuna bağlar.

Bu kodlar, bir sohbet botu uygulamasında kaydırma çubuğunu oluşturmak ve metin kutusuyla bağlamak için kullanılabilir.

```
# Create button to send message

SendButton = Button(root,font=("Verdana",12,"bold"), text = "Send", width = "12", height= "5", bd = 0, bg= "#f9a602",
                    activebackground = "#3c9d9b",fg="#000000",
                    command = send)
```

Bu kod, "Send" adlı bir düğme oluşturur.

"Button()" fonksiyonu, Tkinter'da düğme oluşturmak için kullanılır. Fonksiyon, düğmenin bulunacağı ana pencereyi belirtmek için "root" adlı bir parametre alır.

"font" parametresi, düğmenin yazı tipini, boyutunu ve kalınlığını belirler.

"text" parametresi, düğmenin üzerindeki metni belirtir.

"width" ve "height" parametreleri, düğmenin genişliği ve yüksekliğini belirtir.

"bd" (border) parametresi, düğmenin kenarlık kalınlığını belirler.

"bg" (background) parametresi, düğmenin arka plan rengini belirler.

"activebackground" parametresi, fare düğmeye tıkladığında arka plan rengini değiştirir.

"fg" (foreground) parametresi, düğme metninin rengini belirler.

"command" parametresi, düğmeye tıklandığında hangi işlevin çağrılacağını belirtir. Bu örnekte, "send" adlı işlev çağrılır.

Bu kodlar, sohbet botu uygulamasındaki "Send" düğmesini oluşturmak için kullanılabilir.

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

Bu kod, Tkinter'da bir metin kutusu oluşturmak için "Text()" fonksiyonunu kullanır.

"EntryBox" adlı bir değişkene, "Text()" fonksiyonunun döndürdüğü nesne atanır.

"root" adlı parametre, metin kutusunun ana penceresini belirtir.

"bd" (border) parametresi, metin kutusunun kenarlık kalınlığını belirler.

"bg" (background) parametresi, metin kutusunun arka plan rengini belirler.

"width" ve "height" parametreleri, metin kutusunun genişliği ve yüksekliğini belirler.

"font" parametresi, metin kutusundaki yazı tipini, boyutunu ve kalınlığını belirler.

Bu kodlar, sohbet botu uygulamasındaki metin kutusunu oluşturmak için kullanılabilir.

```
#pencereye yerleştirme
scrollbar.place(x = 376, y = 6, height = 386) #scrollbar konumunu değiştirme
ChatBox.place(x= 6, y =6, height= 90, width = 265 ) #ChatBox konumunu değiştirme
EntryBox.place(x=128,y = 401, height = 90,width = 265) #EntryBox konumunu değiştirme
SendButton.place(x=6,y=401,height=90) #SendButton konumunu değiştirme

root.mainloop() #Pencereyi açma
```

Bu kodlar, oluşturulan nesnelerin pencerede nasıl konumlandırılacağını belirler.

"place()" fonksiyonu, bir nesnenin x ve y koordinatlarını, yüksekliğini ve genişliğini belirler.

"scrollbar.place()" kodu, scrollbar nesnesinin konumunu, boyutunu ve yerleşimini belirler.

"ChatBox.place()" kodu, ChatBox nesnesinin konumunu, boyutunu ve yerleşimini belirler.

"EntryBox.place()" kodu, EntryBox nesnesinin konumunu, boyutunu ve yerleşimini belirler.

"SendButton.place()" kodu, SendButton nesnesinin konumunu, boyutunu ve yerleşimini belirler.

"root.mainloop()" fonksiyonu, pencereyi açar ve uygulamanın kullanıcı tarafından etkileşime girmesini sağlar.

Dosyaların bütün hali

Merhaba bugün Makine öğrenmesinden faydalanarak basit bir chatbot yapmayı analiz edeceğiz.

Chatbotlar doğal dil anlama ve doğal dil oluşturma methotlarını kullanır.

Öncelikle gerekli kütüphaneleri yüklüyoruz.

Kütüphane yükleme

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle
nltk.download("punkt")
```

Yüklenen kütüphaneleri anlama

- **Numpy**

Numpy kütüphanesi bize diziler, matrisler rastgele sayılar oluşturma ve çeşitli sayısal konularda yardımcı olur.

- **Keras**

Keras pyton kaynaklı bir sinir ağı kütüphanesidir.

- **NLTK**

NLTK doğal dil işleme için bir araç kitidir. bu kit kelimeleri ayırma (Tokenization) , kök bulma (Sremming) gibi işlemleri içerir.

WordNetLemmatizer, doğal dil işleme uygulamalarında sıklıkla kullanılan bir lemmatizasyon aracıdır. Lemmatizasyon, kelimenin kökünü bulmak için kullanılan bir yöntemdir.

WordNetLemmatizer, Python dilindeki nltk (Natural Language Toolkit) kütüphanesinde bulunur. Bu araç, bir kelimenin kökünü bulmak için WordNet adlı bir İngilizce sözlük kullanır. Kelimenin eş anlamlıları ve ilişkili kelimeler de WordNet tarafından sağlanır. WordNetLemmatizer, kelimenin eş anlamlılarına göre kelime köklerini bulmak için kullanılan bir yöntem olan morfolojik analiz yapar.

Örneğin, "running" kelimesi için WordNetLemmatizer, "run" kelimesini kök olarak bulacaktır. Bu, metin tabanlı uygulamalarda kelime sınıflandırması, anlamsal analiz, kelime benzerliği gibi çeşitli görevler için önemlidir.

Python'da `import json`
komutu, JSON (JavaScript Object
Notation) veri formatını işlemek için kullanılan bir kütüphaneyi yükler.
JSON, veri değiş tokuşu ve depolama için yaygın olarak kullanılan bir
formattır.

- **Pickle**

Pickle, Python'da bulunan bir modül ve veri serileştirme işlemi için kullanılan bir protokoldür. Serileştirme, verilerin bir dosyaya yazılması veya ağ üzerinden başka bir makineye gönderilmesi gibi işlemlerde kullanılır.

Pickle modülü, Python nesnelerini, yani listeleri, sözlükleri, sınıfları ve hatta fonksiyonları da dahil olmak üzere her türlü nesneyi serialize etmek ve deserialize etmek için kullanılabilir. Serialize etmek, bir nesneyi bir bayt dizesi gibi bir biçimde temsil etmektir. Deserialize etmek ise, bu bayt dizisini tekrar bir Python nesnesine dönüştürmektir.

Pickle, kullanımı kolay ve esnek bir serileştirme aracıdır. Özellikle büyük veri setleri ile çalışırken veya verileri ağ üzerinden gönderirken kullanışlıdır. Ancak, dikkatli kullanılmadığında güvenlik açıkları doğurabilir. Bu nedenle, güvenilir kaynaklardan gelen verileri pickle etmek ve depickle etmek önemlidir.

İntent.json dosyasında belirli cevaplar ve yanıtları sıralanmış

```
intents_file = open("intents.json").read() #intents.json dosyasını okuma
intents = json.loads(intents_file) #json dosyasını okuma
```

bazı boş listeler ve görmezden gelinmesi gereken noktalam işaretleri için atamalar

```
words = [] #kelimeleri tutmak için boş liste
classes = [] #sınıfları tutmak için boş liste
documents = [] #dokümanları tutmak için boş liste
ignore_letters = ["!","?",",","."] #ignore_letters değişkenine !,?,.,, değerlerini atama
```

daha sonra doğal dil işleme projesinde kullanılan eğitim verilerini hazırlanır.

Veriler, bir JSON dosyasından yüklenir ve ardından her bir niyetin (intent) ve her bir niyetin farklı örneklerinin (patterns) altındaki cümleleri işlenir.

Her bir cümle, nltk.word_tokenize() fonksiyonu kullanılarak ayrıştırılır ve ardından elde edilen kelimeler, `words` adlı bir liste içinde toplanır. Bu kelimeler, ayrı bir liste içinde `documents` adlı bir liste içinde saklanır. Her bir örnek, kelimeleri ve niyeti (`intent["tag"]`) birleştirerek bir tuple olarak `documents` listesine eklenir. Ayrıca, her niyetin bir listesi de `classes` adlı bir liste içinde toplanır.

Son olarak, `documents` değişkeni yazdırılır, yani her bir örnek (örneğin, bir kullanıcının sormuş olabileceği bir soru) ve onun hangi niyetle ilişkili olduğu görüntülenir. Bu veriler daha sonra makine öğrenimi modellerinin eğitiminde kullanılabilir.

```
for intent in intents["intents"]: #intents["intents"] değişkenindeki değerleri intent değişkenine atama
    for pattern in intent["patterns"]: #    intent["patterns"] değişkenindeki değerleri pattern değişkenine atama

        word = nltk.word_tokenize(pattern)  #word değişkenine pattern değişkenini tokenize etme
        words.extend(word)#words değişkenine word değişkenini ekleme
        documents.append((word,intent["tag"])) #documents değişkenine word ve intent["tag"] değerlerini ekleme
        # sınıflarımızı listeye ekleme
        if intent["tag"] not in classes: #intent["tag"] değişkenindeki değerler classes değişkeninde yoksa
            classes.append(intent["tag"]) #classes değişkenine intent["tag"] değişkenini ekleme
print(documents) #documents değişkenini yazdırma
```

bazı hataları gidermek için

```
#nlt wordnet error
nltk.download("wordnet") #wordnet kütüphanesini indirme

#nltk nltk.download('omw-1.4') error
nltk.download('omw-1.4')
```

```
## lemmatize and lower each word and remove duplicates #lemmatize ve her kelimeyi küçültme ve tekrar edenleri kaldırma
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_letters] #words değişkenine lemmatizer kütüphanesini kullanarak
words = sorted(list(set(words))) #words değişkenine sıralama ve tekrar edenleri kaldırma
#sort classes
classes = sorted(list(set(classes))) #classes değişkenine sıralama ve tekrar edenleri kaldırma
#documents = combination between patterns and intents #documents = patterns ve intents arasındaki kombinasyon
print(len(documents),"documents") #documents değişkeninin uzunluğunu yazdırma
#classes = intents
print(len(classes),"classes",classes) #classes değişkeninin uzunluğunu yazdırma
#words = all words, vocabulary
print(len(words),"unique lemmatized words", words) #words değişkeninin uzunluğunu yazdırma
```

```
pickle.dump(words,open("words.pkl","wb")) #words değişkenini words.pkl dosyasına yazma
pickle.dump(classes,open("classes.pkl","wb"))
```

Bu kod bloğu, doğal dil işleme projelerinde kullanılan bir terim "lemmatization" işlemi ile ilgili işlemleri gerçekleştirir. Burada, projede kullanılacak olan veri kümesindeki tüm kelimelerin lemmatize edilmesi, küçük harfe dönüştürülmesi ve tekrar eden kelimelerin kaldırılması işlemleri yapılır.

Öncelikle, nltk kütüphanesi içinde yer alan `WordNetLemmatizer()` sınıfı kullanılarak, `words` listesi içindeki tüm kelimeler lemmatize edilir ve `ignore_letters` adlı bir listede yer alan harfler dikkate alınmaz. Lemmatize edilmiş kelimelerin küçük harfe çevrilmesi, `lower()` fonksiyonu kullanılarak gerçekleştirilir. Bu işlem, kelime köklerini elde etmek için yapılır.

Daha sonra, `words` listesi sıralanır ve `set()` fonksiyonu ile tekrar eden kelimeler kaldırılır. Elde edilen kelimeler, `words` listesi içinde saklanır. Aynı işlem `classes` listesi için de uygulanır.

`documents` adlı bir liste, `patterns` ve `intents` verilerinin birleştirilmesi ile oluşturulur. Bu veriler, daha sonra makine öğrenimi modellerinin eğitimi için kullanılacak olan eğitim verileri olarak kullanılabilir.

Son olarak, `documents`, `classes` ve `words` listelerinin uzunluğu ekrana yazdırılır ve ayrıca `words` ve `classes` listeleri, `pickle` kütüphanesi kullanılarak `words.pkl` ve `classes.pkl` dosyalarına yazdırılır. Bu dosyalar, daha sonra modellerin eğitimi sırasında kullanılabilir.

```
#create the trainin data
training = [] #boş liste oluşturma
#create empty array for the output #çıktı için boş dizi oluşturma
output_empty = [0]*len(classes) #output_empty değişkenine classes değişkeninin uzunluğu kadar 0 değeri atama
#trainin set, bag of words for every sentence #trainin set, her cümle için bag of words
for doc in documents: #documents değişkenindeki değerleri doc değişkenine atama
    #initializing bag of words
    bag = [] #bag değişkenine boş liste atama
    #list of tokenized words for the pattern
    word_patterns = doc[0] #word_patterns değişkenine doc[0] değişkenini atama
    #lemmatize each word - create base word, in attempt to represent related words #her kelimeyi lemmatize etme - temel kelimeyi oluşturma,
    word_patterns = [lemmatizer.lemmatize(word.lower()) for word in word_patterns] #word_patterns değişkenine lemmatizer kütüphanesini kull
    # create the bag of words array with 1, şf words şs found in current pattern #1 ile bag of words dizisini oluşturma, bulunan kelimeler
    for word in words: #words değişkenindeki değerleri word değişkenine atama
        bag.append(1) if word in word_patterns else bag.append(0) #bag değişkenine word değişkenindeki değerlerin word_patterns değişkenind

    #output is a "0 for each tag and "1" for current tag (for each pattern)
    output_row = list(output_empty) #output_row değişkenine output_empty değişkenini atama
    output_row[classes.index(doc[1])] = 1 #output_row değişkenine doc[1] değişkenindeki değerin indexini atama
    training.append([bag,output_row]) #training değişkenine bag ve output_row değerlerini ekleme
#shuffle the features and make numpy array
random.shuffle(training) #training değişkenindeki değerleri karıştırma
training = np.array(training) #training değişkenini numpy dizisine çevirme
#create training and testing lists. X- patterns, Y - intents
train_x = list(training[:,0]) #train_x değişkenine training değişkenindeki 0. değerleri atama
train_y = list(training[:,1]) #train_y değişkenine training değişkenindeki 1. değerleri atama
print("Trainin data is created") #trainin data is created yazdırma
```

Bu kod bloğu, bir chatbot için eğitim verileri oluşturmak için kullanılır.

Öncelikle, "documents" adlı bir liste oluşturulur ve bu listede her bir öğe, bir cümle ve bu cümlenin hangi amaçla kullanıldığını belirten bir etiket içerir.

Daha sonra, "classes" adlı bir liste oluşturulur ve bu liste, chatbot'un anlamaya çalışacağı farklı amaçları temsil eden etiketler içerir.

Eğitim verileri oluşturmak için bir döngü kullanılır. Her bir belge ("doc"), "bag" adlı bir boş liste oluşturulur ve cümlenin kelimeleri üzerinde dolaşarak her bir kelimenin lemmatize edilmiş hali "word_patterns" adlı bir liste oluşturulur. Daha sonra, bir iç içe döngü kullanılarak, her bir kelime "words" listesinde dolaşılır ve bu kelime, "word_patterns" listesinde varsa "1" olarak "bag" listesine eklenir, aksi takdirde "0" olarak eklenir. Bu işlem, belge için bir "bag of words" (kelimelerin torbası) modeli oluşturur.

"output_row" adlı bir boş liste oluşturulur ve bu liste, "classes" listesinin uzunluğu kadar sıfır içerir. Daha sonra, belgenin etiketi, "classes" listesindeki indeksine karşılık gelen "1" değeri olarak "output_row" listesine eklenir.

Son olarak, "training" adlı bir liste oluşturulur ve bu liste, her bir belgenin "bag" listesi ve "output_row" listesi olarak bir öğe içerir. "training" listesi karıştırılır ve "train_x" ve "train_y" listeleri oluşturulur. Bu iki liste, sırasıyla, "bag" listeleri ve "output_row" listelerini içerir ve chatbot'un eğitiminde kullanılır.

```
#deep neural networds model #derin sinir ağı modeli
model = Sequential() #model değişkenine Sequential kütüphanesini atama
model.add(Dense(128,input_shape=(len(train_x[0]),), activation="relu")) #model değişkenine Dense kütüphanesini kullanarak 128 değerini atam
model.add(Dropout(0.5)) #model değişkenine Dropout kütüphanesini kullanarak 0.5 değerini atama
model.add(Dense(64,activation="relu")) #model değişkenine Dense kütüphanesini kullanarak 64 değerini atama
model.add(Dropout(0.5)) #model değişkenine Dropout kütüphanesini kullanarak 0.5 değerini atama
model.add(Dense(len(train_y[0]), activation="softmax")) #model değişkenine Dense kütüphanesini kullanarak len(train_y[0]) değerini atama
```

Bu kod bloğunda, bir yapay sinir ağı modeli oluşturuluyor. İlk olarak, Sequential() fonksiyonu kullanılarak bir model nesnesi oluşturuluyor. Daha sonra, add() fonksiyonu ile bu model nesnesine sırayla katmanlar ekleniyor.

İlk katmanda, input_shape parametresi ile giriş boyutu belirtiliyor. Burada, train_x dizisindeki her bir öğenin boyutu len(train_x[0]) olarak belirtiliyor. Bu değer, eğitim verilerinin boyutunu temsil ediyor. Dense() fonksiyonu kullanılarak, 128 nöronlu bir katman oluşturuluyor. Activation parametresi "relu" olarak belirtiliyor. Bu, ReLU (Rectified Linear Unit) aktivasyon fonksiyonunun kullanılacağı anlamına geliyor.

İkinci katman, Dropout() fonksiyonu ile bir dropout katmanıdır. Dropout, overfittingi önlemek için kullanılan bir tekniktir. 0.5 değeri, her bir nöronun %50'sinin rastgele olarak bırakılacağı anlamına gelir.

Üçüncü katman, yine Dense() fonksiyonu kullanılarak, 64 nöronlu bir katmandır. ReLU aktivasyon fonksiyonu kullanılır.

Dördüncü katman, tekrar Dropout() fonksiyonu ile bir dropout katmanıdır.

Son katmanda, Dense() fonksiyonu ile çıkış boyutu belirtilir. Burada, len(train_y[0]) olarak belirtilir. Bu, çıkış katmanındaki nöron sayısını temsil eder. Activation parametresi "softmax" olarak belirtilir. Bu, çıkışın sınıflandırma için kullanılacağını gösterir.

Bu şekilde, bir yapay sinir ağı modeli oluşturulmuş olur.

```
#Compliling mode. SGD with Nesterov accelerated gradient gives good result for this model #Compliling mode. SGD with Nesterov hızlandırılmı
sgd = SGD(lr=0.01 ,decay = 1e-6, momentum = 0.9, nesterov = True ) #sgd değişkenine SGD kütüphanesini kullanarak değerleri atama
model.compile(loss="categorical_crossentropy", optimizer = sgd) #model değişkenine compile kütüphanesini kullanarak değerleri atama
metrics = ["accuracy"] #metrics değişkenine accuracy değerini atama
```

Bu kod bloğu, bir yapay sinir ağı modeli oluşturur ve derler.

İlk önce, "Sequential" modelini oluşturur ve ardından "Dense" katmanları ekler. "Dense" katmanı, tüm giriş birimlerinin tüm çıkış birimlerine tamamen bağlandığı bir sinir ağı katmanıdır. İlk "Dense" katmanı 128 birim ve "relu" aktivasyon fonksiyonuna sahiptir. İkinci "Dense" katmanı 64 birime ve yine "relu" aktivasyon fonksiyonuna sahiptir. Son "Dense" katmanı, çıkış sınıflarının sayısına göre birim sayısına sahip ve "softmax" aktivasyon fonksiyonuna sahiptir.

Daha sonra, "SGD" optimizer kullanılarak model derlenir. "SGD", Stokastik Gradient Descent'in kısaltmasıdır ve sinir ağlarının eğitiminde sıklıkla kullanılan bir optimizasyon algoritmasıdır. Bu kod bloğunda, öğrenme oranı (lr) 0.01 olarak ayarlanmış, ağırlık azalımı (decay) 1e-6, momentum 0.9 ve Nesterov momentumu True olarak ayarlanmıştır.

Son olarak, modelin performansını ölçmek için "accuracy" ölçütü (metrics) kullanılmıştır.

```
#Training and saving the model #Modeli eğitme ve kaydetme
hist = model.fit(np.array(train_x),np.array(train_y), epochs = 200, batch_size =  5, verbose = 1) #hist değişkenine modeli eğitme
model.save("chatbot_model.h5",hist) #modeli chatbot_model.h5 dosyasına kaydetme
print("model is created") #model is created yazdırma
```

Bu kod bloğu, oluşturulan modelin eğitimini gerçekleştirir ve eğitim sonrasında oluşan modeli "chatbot_model.h5" adlı dosyaya kaydeder.

`model.fit()` metodu, eğitim verilerini (`train_x` ve `train_y`) kullanarak modelin eğitimini gerçekleştirir. `epochs` parametresi, tüm verilerin model tarafından kaç kez kullanılacağını belirler. `batch_size` parametresi, her eğitim adımında kaç örnek kullanılacağını belirler. `verbose` parametresi, eğitim sırasında gösterilecek loglama seviyesini belirler.

`model.save()` metodu, eğitim sonrasında oluşan modeli diskteki bir dosyaya kaydeder. Kaydedilen model, ileride farklı programlarda kullanılabilir hale gelir.

Son olarak, `print()` fonksiyonu ile modelin başarıyla oluşturulduğu kullanıcıya bildirilir.

Böylece chatbotun ana kodu oluşturulmuş olur sonrasında arayüz oluşturulur

---

Arayüz oluşturma

```
import tkinter
from tkinter import *
```

Tkinter, Python programlama dili için standart bir GUI (Graphical User Interface) kütüphanesidir. Tkinter, Tk GUI toolkit'inin Python diline uyarlanmış halidir ve bu nedenle Tkinter genellikle Python ile birlikte gelir.

Tkinter, çeşitli GUI bileşenlerini oluşturmak ve düzenlemek için kullanılabilir. Bu bileşenler arasında düğmeler, metin kutuları, etiketler, menüler, liste kutuları ve çerçeveler gibi birçok standart bileşen bulunur. Tkinter ayrıca çizimler, grafikler ve animasyonlar gibi daha gelişmiş özellikler için de kullanılabilir.

Tkinter, kullanımı kolay ve öğrenmesi kolaydır. Başlangıç seviyesindeki kullanıcılar için idealdir, ancak gelişmiş kullanıcılar tarafından da kullanılabilir. Tkinter, çok sayıda örnek uygulama ve dokümantasyon içerir, bu da kullanıcıların Tkinter'ı kullanarak kendi uygulamalarını oluşturmalarını kolaylaştırır.

Tkinter ayrıca çok platformlu bir kütüphanedir. Bu, Tkinter kullanılarak geliştirilen uygulamaların Windows, Linux ve Mac gibi farklı işletim sistemlerinde çalışabileceği anlamına gelir

ython'da Tkinter GUI kütüphanesindeki tüm sınıfları ve işlevleri kodunuzda doğrudan kullanılabilir hale getirir.

"from tkinter" ifadesi, tkinter kütüphanesini kodunuza dahil etmek için kullanılır. "import *" ifadesi, tkinter kütüphanesindeki tüm sınıfların ve fonksiyonların kodunuzda doğrudan kullanılabilir hale getirilmesini sağlar.

```
def send(): #mesaj gönderme fonksiyonu
    msg = EntryBox.get("1.0","end-1c").strip() #EntryBox deki değerleri msg değişkenine atama
    EntryBox.delete("0.0",END) #EntryBox deki değerleri silme

    if msg !="": #  msg değişkeninde değer varsa
        ChatBox.config(state= DISABLED) #ChatBox deki değerleri devre dışı bırakma
        ChatBox.inters(END,"You: "+msg+ "\n\n") #ChatBox deki değerleri ekleme
        ChatBox.config(foreground="#446665",font = ("Verdana",12)) #ChatBox deki yazı tipini ve yazı rengini değiştirme
        ints = predict_class(msg) #predict_class fonksiyonunu msg değişkeni ile çağırma
        res = getResponse(ints,intents) #getResponse fonksiyonunu ints ve intents değişkenleri ile çağırma

        ChatBox.insert(END, "Bot: "+ res + "\n\n") #ChatBox deki değerleri ekleme
        ChatBox.config(state=DISABLED)  #ChatBox deki değerleri devre dışı bırakma
        ChatBox.yview(END) #ChatBox deki değerleri sona götürme
```

Bu kod, bir sohbet botunun kullanıcının mesajını alıp yanıtlamasını sağlayan bir fonksiyondur.

Fonksiyonun adı "send()" ve hiçbir parametre almaz. Fonksiyon, kullanıcının girdiği mesajı alır ve bu mesajı "msg" adlı bir değişkene atar. Daha sonra, mesajın girildiği metin kutusunu temizler.

Eğer mesaj boş değilse, "if msg != "":", kod bloğuna girer. ChatBox adlı bir metin kutusu devre dışı bırakılır ve ardından kullanıcının mesajını içeren bir etiket oluşturulur ve ChatBox'a eklenir. Etiketin yazı rengi "#446665" ve yazı tipi "Verdana" ve boyutu 12'dir.

Sonra, kullanıcının mesajını anlamlandırmak için "predict_class()" fonksiyonu çağrılır ve ardından botun yanıtı almak için "getResponse()" fonksiyonu çağrılır. Bu yanıt ChatBox'a eklenir ve tekrar devre dışı bırakılır. ChatBox, son eklenen mesajı görüntülemek için en aşağı kaydırılır.

Bu kod, bir sohbet botu uygulamasında kullanılabilir ve kullanıcının mesajlarını alıp yanıtlarını göstermek için Tkinter kullanır.

```
root =Tk() #Tkinter kütüphanesini çağırma
root.title("Chatbot") #Pencere başlığını değiştirme
root.geometry("400x500")   #boyutunu değiştirme
root.resizable(width=FALSE,height=FALSE) #frame boyutunu değiştirme
```

Bu kod, bir Tkinter penceresi oluşturur ve bazı ayarları yapar.

İlk olarak, "Tk()" fonksiyonu çağrılarak bir Tkinter nesnesi oluşturulur ve "root" adlı bir değişkene atanır. Bu nesne, Tkinter uygulamasında ana pencereyi temsil eder.

Ardından, "title()" yöntemi çağrılarak pencere başlığı "Chatbot" olarak ayarlanır.

"geometry()" yöntemi, pencerenin boyutunu "400x500" piksel olarak ayarlar.

"resizable()" yöntemi, pencere boyutunun sabit kalmasını sağlar. width=FALSE, genişliğin değiştirilemeyeceğini, height=FALSE ise yüksekliğin değiştirilemeyeceğini belirtir. Bu yöntem, pencere boyutunun kullanıcı tarafından değiştirilmesini önlemek için kullanılır.

Bu kodlar, Tkinter kullanarak bir sohbet botu uygulaması için bir pencere oluşturmak için kullanılabilir.

```
#cREATE CHAT WİNDOW
ChatBox = Text(root, bd= 0, bg = "white",height= "8", width = "50", font = "Arial",) #ChatBox değişkenine Text fonksiyonunu çağırma
ChatBox.config(state = DISABLED) #ChatBox deki değerleri devre dışı bırakma
```

Bu kod, bir metin kutusu oluşturur ve bazı ayarları yapar.

"Text()" fonksiyonu, Tkinter'da metin kutusu oluşturmak için kullanılır. Fonksiyon, metin kutusunun bulunacağı ana pencereyi belirtmek için "root" adlı bir parametre alır.

"bd" parametresi, metin kutusunun kenarlık genişliğini belirler. Bu örnekte "bd=0" olarak ayarlandığı için kenarlık yoktur.

"bg" parametresi, metin kutusunun arka plan rengini belirler. Bu örnekte "bg = "white"" olarak ayarlandığı için arka plan rengi beyazdır.

"height" ve "width" parametreleri, metin kutusunun boyutunu belirler. Bu örnekte "height=8" ve "width=50" olarak ayarlandığı için metin kutusu 8 satır ve 50 karakter genişliğinde olacaktır.

"font" parametresi, metin kutusunun yazı tipini belirler. Bu örnekte "Arial" olarak ayarlandığı için yazı tipi Arial'dır.

"config()" yöntemi, belirli özellikleri ayarlamak için kullanılır. "state" parametresi, metin kutusunun kullanıcı tarafından düzenlenebilir olup olmayacağını belirler. Bu örnekte "DISABLED" olarak ayarlandığı için, metin kutusundaki veriler yalnızca program tarafından değiştirilebilir ve kullanıcı tarafından değiştirilemez.

Bu kodlar, bir sohbet botu uygulamasında kullanılan metin kutusunu oluşturmak için kullanılabilir.

```
#Bind scrollbar to chat window
scrollbar = Scrollbar(root, command = ChatBox.yview, cursor = "heart")
ChatBox["yscrollcommand"] = scrollbar.set
```

Bu kod, bir kaydırma çubuğu oluşturur ve metin kutusuna bağlar.

"Scrollbar()" fonksiyonu, Tkinter'da kaydırma çubuğu oluşturmak için kullanılır. Fonksiyon, kaydırma çubuğunun bulunacağı ana pencereyi belirtmek için "root" adlı bir parametre alır.

"command" parametresi, kaydırma çubuğunun yönünü belirler. Bu örnekte "ChatBox.yview" olarak ayarlandığı için, kaydırma çubuğu metin kutusunun yönünü takip edecektir.

"cursor" parametresi, kaydırma çubuğunun işaretçi şeklini belirler. Bu örnekte "cursor = "heart"" olarak ayarlandığı için kaydırma çubuğu işaretçi şekli kalp şeklindedir.

"yscrollcommand" özelliği, metin kutusundaki kaydırmayı kontrol etmek için kullanılır. Bu örnekte "ChatBox["yscrollcommand"] = scrollbar.set" satırı, metin kutusundaki kaydırmayı kaydırma çubuğuna bağlar.

Bu kodlar, bir sohbet botu uygulamasında kaydırma çubuğunu oluşturmak ve metin kutusuyla bağlamak için kullanılabilir.

```
# Create button to send message

SendButton = Button(root,font=("Verdana",12,"bold"), text = "Send", width = "12", height= "5", bd = 0, bg= "#f9a602",
                  activebackground = "#3c9d9b",fg="#000000",
                  command = send)
```

Bu kod, "Send" adlı bir düğme oluşturur.

"Button()" fonksiyonu, Tkinter'da düğme oluşturmak için kullanılır. Fonksiyon, düğmenin bulunacağı ana pencereyi belirtmek için "root" adlı bir parametre alır.

"font" parametresi, düğmenin yazı tipini, boyutunu ve kalınlığını belirler.

"text" parametresi, düğmenin üzerindeki metni belirtir.

"width" ve "height" parametreleri, düğmenin genişliği ve yüksekliğini belirtir.

"bd" (border) parametresi, düğmenin kenarlık kalınlığını belirler.

"bg" (background) parametresi, düğmenin arka plan rengini belirler.

"activebackground" parametresi, fare düğmeye tıkladığında arka plan rengini değiştirir.

"fg" (foreground) parametresi, düğme metninin rengini belirler.

"command" parametresi, düğmeye tıklandığında hangi işlevin çağrılacağını belirtir. Bu örnekte, "send" adlı işlev çağrılır.

Bu kodlar, sohbet botu uygulamasındaki "Send" düğmesini oluşturmak için kullanılabilir.

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

Bu kod, Tkinter'da bir metin kutusu oluşturmak için "Text()" fonksiyonunu kullanır.

"EntryBox" adlı bir değişkene, "Text()" fonksiyonunun döndürdüğü nesne atanır.

"root" adlı parametre, metin kutusunun ana penceresini belirtir.

"bd" (border) parametresi, metin kutusunun kenarlık kalınlığını belirler.

"bg" (background) parametresi, metin kutusunun arka plan rengini belirler.

"width" ve "height" parametreleri, metin kutusunun genişliği ve yüksekliğini belirler.

"font" parametresi, metin kutusundaki yazı tipini, boyutunu ve kalınlığını belirler.

Bu kodlar, sohbet botu uygulamasındaki metin kutusunu oluşturmak için kullanılabilir.

```
#pencereye yerleştirme
scrollbar.place(x = 376, y = 6, height = 386) #scrollbar konumunu değiştirme
ChatBox.place(x= 6, y =6, height= 90, width = 265 ) #ChatBox konumunu değiştirme
EntryBox.place(x=128,y = 401, height = 90,width = 265) #EntryBox konumunu değiştirme
SendButton.place(x=6,y=401,height=90) #SendButton konumunu değiştirme

root.mainloop() #Pencereyi açma
```

Bu kodlar, oluşturulan nesnelerin pencerede nasıl konumlandırılacağını belirler.

"place()" fonksiyonu, bir nesnenin x ve y koordinatlarını, yüksekliğini ve genişliğini belirler.

"scrollbar.place()" kodu, scrollbar nesnesinin konumunu, boyutunu ve yerleşimini belirler.

"ChatBox.place()" kodu, ChatBox nesnesinin konumunu, boyutunu ve yerleşimini belirler.

"EntryBox.place()" kodu, EntryBox nesnesinin konumunu, boyutunu ve yerleşimini belirler.

"SendButton.place()" kodu, SendButton nesnesinin konumunu, boyutunu ve yerleşimini belirler.

"root.mainloop()" fonksiyonu, pencereyi açar ve uygulamanın kullanıcı tarafından etkileşime girmesini sağlar.