

# CHAT BOT

# Title:

Chatbot Response for Questions about Hugging Face using nlp.

- Dr.Esraa

Eng.Moustafa

- Team member

Abdullah 205023

Fagr Mohamed 205050

Nourhan salah 205075

# Content:

1. Problem Statement
2. Approach
3. Features and Functionality
4. Results and Benefits
5. Conclusion
6. Code

# 1. Problem Statement:

The primary goal of our project is to develop a chatbot that can understand and respond to user questions related to Hugging Face.

We want the chatbot to provide informative and relevant answers, enabling users to enhance their knowledge and make the most of the Hugging Face platform.

## 2. Approach:

- To accomplish this, we employed several key techniques and components:
- **a. Natural Language Processing (NLP):** We leveraged NLP techniques to process and analyze user input. This involved tokenizing the input, performing lemmatization, and removing punctuation and stopwords. These preprocessing steps ensure that the chatbot can understand user queries effectively.
- **b. TF-IDF (Term Frequency-Inverse Document Frequency):** We utilized the TF-IDF technique to represent and analyze the significance of words in user queries and existing Hugging Face-related sentences. TF-IDF helps identify the most relevant sentence that matches the user's query by measuring the importance of each word in the context of the entire corpus.
- **c. Cosine Similarity:** By calculating the cosine similarity between the user query and existing Hugging Face-related sentences, we determined the most similar sentence to generate an appropriate response. Cosine similarity measures the similarity of two vectors and enables us to find the most relevant sentence for the given query.
- **d. Data Visualization:** We incorporated data visualization using Matplotlib to illustrate the cosine similarity scores. This allows users to visually understand the relevance and similarity of different sentences to their queries.

### 3. Features and Functionality:

- **a. Greeting Recognition:** The chatbot can recognize and respond to various greetings such as "hello," "hi," or "hey." It provides appropriate responses to create a conversational and friendly experience for users.
- **b. Response Generation:** Based on the user's query, the chatbot generates a response by finding the most similar sentence from the Hugging Face-related sentences. This response aims to address the user's question and provide relevant information.
- **c. Gratitude Acknowledgment:** The chatbot acknowledges expressions of gratitude such as "thanks" or "thank you" and responds with a corresponding message.
- **d. Exit Option:** Users have the option to exit the conversation by typing "bye." Upon receiving this input, the chatbot offers a farewell message, signaling the end of the interaction.

## 4. Results and Benefits:

- **a. Quick and Accurate Responses:** Users can obtain quick and accurate responses to their questions about Hugging Face. The chatbot leverages TF-IDF and cosine similarity to provide the most relevant information, enabling users to find the answers they need efficiently.
- **b. Improved Understanding of Hugging Face:** By interacting with the chatbot, users can enhance their understanding of Hugging Face, including its functionalities, libraries, and resources. The chatbot acts as a knowledgeable assistant, guiding users through their queries and helping them explore the platform effectively.
- **c. User-Friendly Interface:** The chatbot offers a user-friendly interface, simulating a conversation and providing a seamless experience. Users can ask questions naturally, receive informative responses, and engage

# 5.Conclusion:

- The code represents a chatbot implemented using Python and the Natural Language Toolkit (NLTK) library.
- The chatbot is designed to provide responses to user queries about Hugging Face, a popular natural language processing (NLP) library.
- The chatbot utilizes techniques such as tokenization, lemmatization, and TF-IDF (Term Frequency-Inverse Document Frequency) to process user input and generate appropriate responses.
- The chatbot begins by greeting the user and asking for their queries. It handles various types of user interactions, including greetings, expressions of gratitude, and general questions.
- The chatbot employs cosine similarity to determine the most relevant response from a pre-defined set of sentences. It calculates the similarity scores between the user's input and the available sentences and selects the response with the highest similarity score.
- The code also includes functionality to visualize the chatbot responses using Matplotlib.
- The user's input and corresponding chatbot responses are stored in a DataFrame and a bar plot is generated to display the chatbot responses over the course of the conversation.



# Code:

## #1

```
[21] import nltk # Importing the Natural Language Toolkit library
import io # Importing the io module for input/output operations
import numpy as np # Importing the NumPy library for array operations and computations
import random # Importing the random module for generating random numbers and making random selections
import string # Importing the string module for string manipulation functions and constants
import warnings # Importing the warnings module for controlling warning messages
import matplotlib.pyplot as plt
import pandas as pd
warnings.filterwarnings("ignore") # Ignoring any warning messages

▶ f = open('/content/huggingface.txt', 'r', errors='ignore') # Open the file 'huggingface.txt' in read mode, ignoring any decoding errors
raw = f.read() # Read the contents of the file into the variable 'raw'
raw = raw.lower() # Convert the contents of 'raw' to lowercase
nltk.download('punkt') # Download the necessary resources for tokenization
nltk.download('wordnet') # Download the necessary resources for lemmatization
sent_tokens = nltk.sent_tokenize(raw) # Tokenize 'raw' into a list of sentences using NLTK's sentence tokenizer
word_tokens = nltk.word_tokenize(raw) # Tokenize 'raw' into a list of words using NLTK's word tokenizer

[23] word_tokens[:2] # Retrieve the first two elements from the list 'word_tokens'

[24] sent_tokens[:2] # Access the first two elements of the list 'sent_tokens'

▶ lemmmer = nltk.stem.WordNetLemmatizer() # Create an instance of the WordNet lemmatizer from NLTK
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]
    # Lemmatize each token in the given list of tokens using WordNet lemmatizer

remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
    # Create a dictionary to map punctuation characters to None using dictionary comprehension

def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
    # Normalize the given text by converting it to lowercase, removing punctuation,
    # tokenizing it into words, and then lemmatizing each word using WordNet lemmatizer
```

# #2

```
[43] GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up", "hey") # List of common greeting inputs
      GREETING_RESPONSES = ["hi", "hey", "hi there", "hello", "I am glad! You are talking to me"] # List of corresponding greeting responses
```

```
def greeting(sentence):
    for word in sentence.split(): # Split the sentence into individual words
        if word.lower() in GREETING_INPUTS: # Check if any word in the sentence matches a greeting input
            return random.choice(GREETING_RESPONSES) # If a greeting is detected, return a random greeting response
```

```
[27] from sklearn.feature_extraction.text import TfidfVectorizer # Importing TfidfVectorizer from scikit-learn library

      from sklearn.metrics.pairwise import cosine_similarity # Importing cosine_similarity from scikit-learn library
```

```
[28] def response(user_response):
      chatbot_response = ' ' # Initialize an empty string to store the chatbot's response
      sent_tokens.append(user_response) # Append the user's response to the list of sentence tokens
      TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words="english") # Create a TfidfVectorizer object with a tokenizer and stop words
      tfidf = TfidfVec.fit_transform(sent_tokens) # Compute the TF-IDF matrix for the sentence tokens
      vals = cosine_similarity(tfidf[-1], tfidf) # Compute the cosine similarity between the last sentence token (user's response) and all other sentence tokens
      idx = vals.argsort()[0][-2] # Get the index of the most similar sentence token (excluding the user's response)
      flat = vals.flatten() # Flatten the cosine similarity values into a 1D array
      flat.sort() # Sort the cosine similarity values in ascending order
      req_tfidf = flat[-2] # Get the second highest cosine similarity value
      if req_tfidf == 0:
          chatbot_response = chatbot_response + "I am sorry! I don't understand you" # If the similarity is 0, the chatbot doesn't understand the user's input
          return chatbot_response
      else:
          chatbot_response = sent_tokens[idx] # Retrieve the most similar sentence token as the chatbot's response
          return chatbot_response
```

```
[49] # Create empty lists to store user responses and chatbot responses
      user_responses = []
      chatbot_responses = []

      flag = True
      print("Chatbot: Hi, my name is Chatbot. I will answer your queries about Hugging face. If you want to exit, type 'Bye!'")

      while flag:
          user_response = input()
          user_response = user_response.lower()

          if user_response in ["bye", "goodbye"]:
              flag = False
              print("Chatbot: Bye! Take care.")
```

# #3

```
[49] print(chatbot, user_response)
else:
    if user_response in ["thanks", "thank you"]:
        flag = False
        print("Chatbot: You're welcome.")
    else:
        greeting_result = greeting(user_response)
        if greeting_result is not None:
            print("Chatbot: " + greeting_result)
        else:
            print("Chatbot: ", end="")
            chatbot_response = response(user_response)
            print(chatbot_response)
            sent_tokens.remove(user_response)

        # Append the user and chatbot responses to the respective lists
        user_responses.append(user_response)
        chatbot_responses.append(chatbot_response)

# Create a DataFrame using the collected responses
data = {'user_response': user_responses, 'chatbot_response': chatbot_responses}
df = pd.DataFrame(data)

# Plot the data
plt.figure(figsize=(8, 6))
plt.bar(range(len(df)), df['chatbot_response'])
plt.xlabel('Interaction')
plt.ylabel('Chatbot Response')
plt.title('Chatbot Responses')

# Display the plot
plt.show()
```

# Result

## #1

Chatbot: Hi, my name is Chatbot. I will answer your queries about Hugging face. If you want to exit, type 'Bye!'

hi

Chatbot: hi

hugging face

Chatbot: other libraries

in addition to transformers and the hugging face hub, the hugging face ecosystem contains libraries for other tasks, such as dataset processing ("datasets")

other libraries

Chatbot: [11][12]

services and technologies

transformers library

the transformers library is a python package that contains open-source implementations of transformer models for text, image, and audio tasks.

who is the founder of the company

Chatbot: history

the company was founded in 2016 by french entrepreneurs clément delangue, julien chaumond, and thomas wolf originally as a company that developed a chatbot

year

Chatbot: find sources: "hugging face" - news · newspapers · books · scholar · jstor (february 2023) (learn how and when to remove this template message)

hugging face, inc.

hugging face logo.png

type private

industry artificial intelligence, machine learning, software development

founded 2016; 7 years ago in new york city

headquarters new york city, u.s.

area served worldwide

key people

clément delangue (ceo)

julien chaumond (cto)

thomas wolf (cso)

products transformers, datasets, spaces

website huggingface.co

hugging face, inc. is an american company that develops tools for building applications using machine learning.

thank you

Chatbot: You're welcome.

# Result

## #2

Chatbot: [11][12]

services and technologies  
transformers library  
the transformers library is a python package that contains open-source implementations of transformer models for text, image, and audio tasks.  
who is the founder of the company  
Chatbot: history  
the company was founded in 2016 by french entrepreneurs clément delangue, julien chaumond, and thomas wolf originally as a company that developed a chatbot app targeted at teenagers.  
year  
Chatbot: find sources: "hugging face" - news · newspapers · books · scholar · jstor (february 2023) (learn how and when to remove this template message)  
hugging face, inc.  
hugging face logo.png  
type private  
industry artificial intelligence, machine learning, software development  
founded 2016; 7 years ago in new york city  
headquarters new york city, u.s.  
area served worldwide  
key people  
clément delangue (ceo)  
julien chaumond (cto)  
thomas wolf (cso)  
products transformers, datasets, spaces  
website huggingface.co  
hugging face, inc. is an american company that develops tools for building applications using machine learning.  
thank you  
Chatbot: You're welcome.

find sources: "hugging face" - news · newspapers · books · scholar · jstor (february 2023) (learn how and when to remove this template message)  
hugging face, inc.  
hugging face logo.png  
type private  
industry artificial intelligence, machine learning, software development  
founded 2016; 7 years ago in new york city  
headquarters new york city, u.s.  
area served worldwide  
key people  
clément delangue (ceo)  
julien chaumond (cto)  
thomas wolf (cso)  
products transformers, datasets, spaces  
website huggingface.co  
hugging face, inc. is an american company that develops tools for building applications using machine learning.

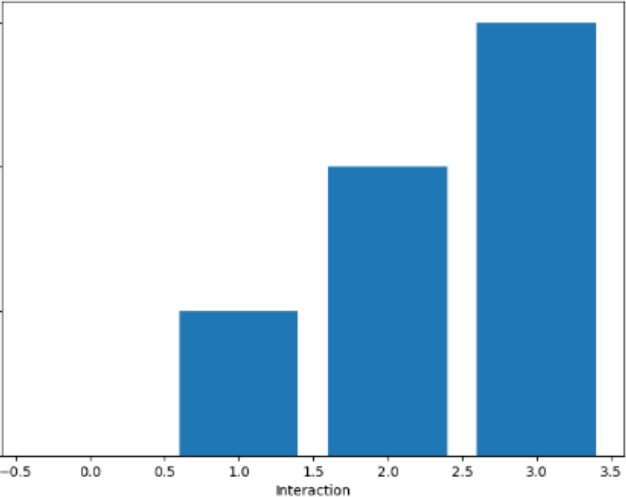
the company was founded in 2016 by french entrepreneurs clément delangue, julien chaumond, and thomas wolf originally as a company that developed a chatbot app targeted at teenagers.

[11][12]

services and technologies  
transformers library  
the transformers library is a python package that contains open-source implementations of transformer models for text, image, and audio tasks.

in addition to transformers and the hugging face hub, the hugging face ecosystem contains libraries for other tasks, such as dataset processing ("datasets"), model evaluation ("evaluate"), simulation ("simulate"), machine learning demos ("gradio").

Chatbot Responses



Chatbot Response

Thank you