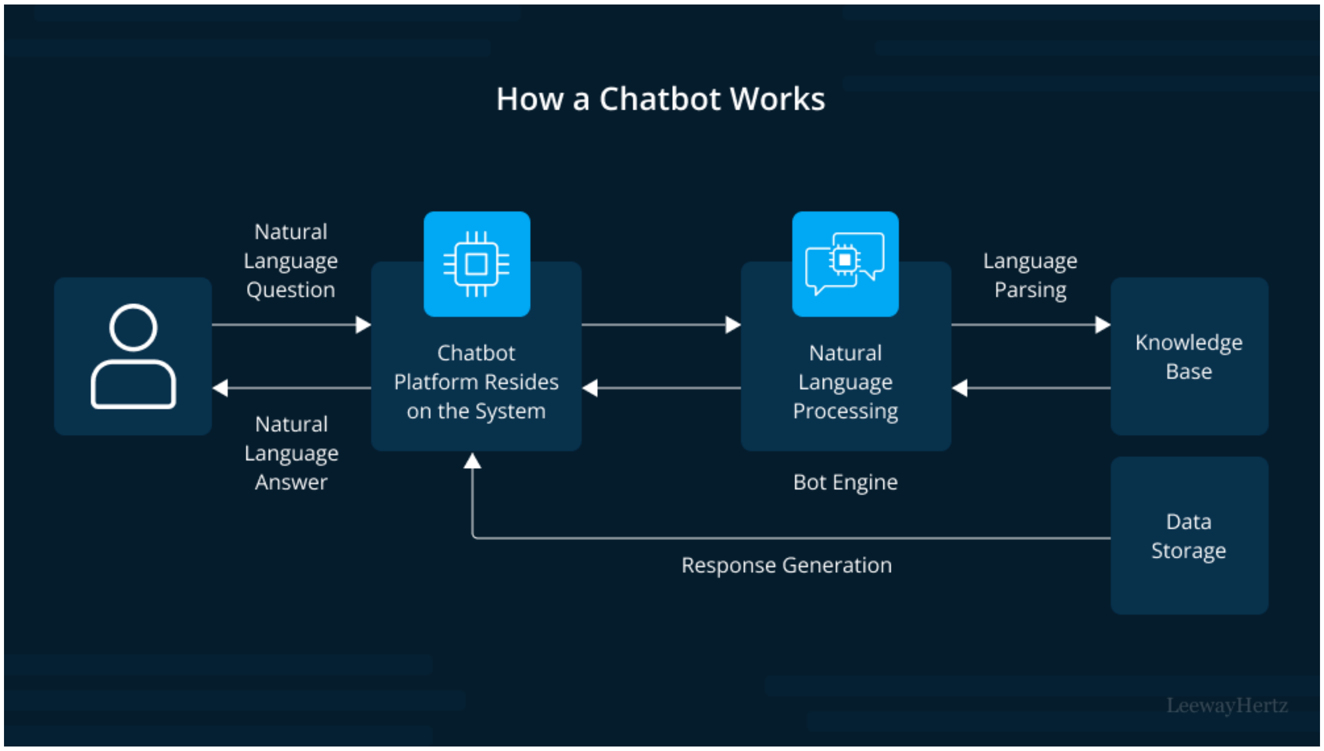
How to build an AI-powered chatbot?

leewayhertz.com/ai-chatbots



In today's fast-paced world, where time is a precious commodity, texting has emerged as one of the most common forms of communication. Hence, chatbots are becoming a crucial part of businesses' operations, regardless of their size or domain. The concept of chatbots can be traced back to the idea of intelligent robots introduced by Alan Turing in the 1950s. And ELIZA was the first chatbot developed by MIT professor Joseph Weizenbaum in the 1960s. Since then, AI-based chatbots have been a major talking point and a valuable tool for businesses to ensure effective customer interactions. According to Demand Sage, the chatbot market is expected to earn about $137.6 million in revenue by 2023. Moreover, it is projected that chatbot sales will reach approximately $454.8 million by the year 2027.

With all the hype surrounding chatbots, it's essential to understand their fundamental nature. So, what are chatbots? Chatbots are computer programs designed to simulate human conversation. They achieve this by generating automated responses and engaging in interactions, typically through text or voice interfaces. They are programmed to carry out a wide range of functions, from responding to frequently asked questions and offering customer service to automating sales and marketing procedures, and may be deployed on various platforms, including websites, social media, and messaging apps.

Businesses use these virtual assistants to perform simple tasks in business-to-business (B2B) and business-to-consumer (B2C) situations. Chatbot assistants allow businesses to provide customer care when live agents aren't available, cut overhead costs, and use staff time better. According to the Demand Sage report cited above, an average customer service agent deals with 17 interactions a day, which means adopting chatbots in enterprises can prevent up to 2.5 billion labor hours.

Clearly, chatbots are one of the most valuable and well-known use cases of artificial intelligence becoming increasingly popular across industries.

Let's delve deeper into chatbots and gain insights into their types, key components, benefits, and a comprehensive guide on the process of constructing one from scratch.

- What is a chatbot?
- Types of chatbots
- Architectural components of AI-powered chatbots and their operational mechanics
- What are the advantages of using AI-powered chatbots?
- How to build an AI-powered chatbot?
- The smartest AI chatbots used across different industries

What is a chatbot?

A chatbot is an Artificial Intelligence (AI) program that simulates human conversation by interacting with people via text or speech. Chatbots use Natural Language Processing (NLP) and machine learning algorithms to comprehend user input and deliver pertinent responses. Chatbots can be integrated into various platforms, including messaging programs, websites, and mobile applications, to provide immediate responses to user queries, automate tedious processes, and increase user engagement. While some chatbots are task-oriented and offer particular responses to predefined questions, others closely mimic human communication. Computer scientist Michael Mauldin first used the term "chatterbot" in 1994 to to describe what later became recognized as the chatbot. The biggest reason chatbots are gaining popularity is that they give organizations a practical approach to enhancing customer service and streamlining processes without making huge investments.

Chatbots use dialogue systems to efficiently handle tasks related to retrieving information, directing inquiries to the appropriate channels, and delivering customer support services. Some chatbots utilize advanced natural language processing and word categorization techniques to understand and interpret user inputs. These chatbots can comprehend the context and nuances of the conversation, allowing for more accurate and detailed responses. On the other hand, some chatbots rely on a simpler method of scanning for general keywords and constructing responses based on pre-defined expressions stored in a library or database. The primary methods through which chatbots can be accessed online are virtual assistants and website popups. Virtual assistants, such as voice-activated chatbots, provide interactive conversational experiences through devices like smartphones or smart speakers. Website popups, on the other hand, are chatbot interfaces that appear on websites, allowing users to engage in text-based conversations. These two contact methods cater to various utilization areas, including business (such as e-commerce support), learning, entertainment, finance, health, news, and productivity.

Types of chatbots

Chatbots can be classified in various ways based on different criteria. Here are some typical chatbot classifications:

Rule-based chatbots: They operate according to a set of rules or scripts and hence, are known as “rule-based” chatbots. They use predetermined answers based on phrases or patterns found in user inputs. Rule-based chatbots are straightforward and have limited functionality because they can only answer a limited set of predefined questions.

Machine learning-based chatbots: Artificial intelligence (AI) and natural language processing (NLP) techniques are used by machine learning-based chatbots to understand and respond to user input. Based on user interactions, they can gradually improve their responses as they learn from the data. Machine learning-based chatbots are more sophisticated and capable of handling a larger range of questions than rule-based chatbots.

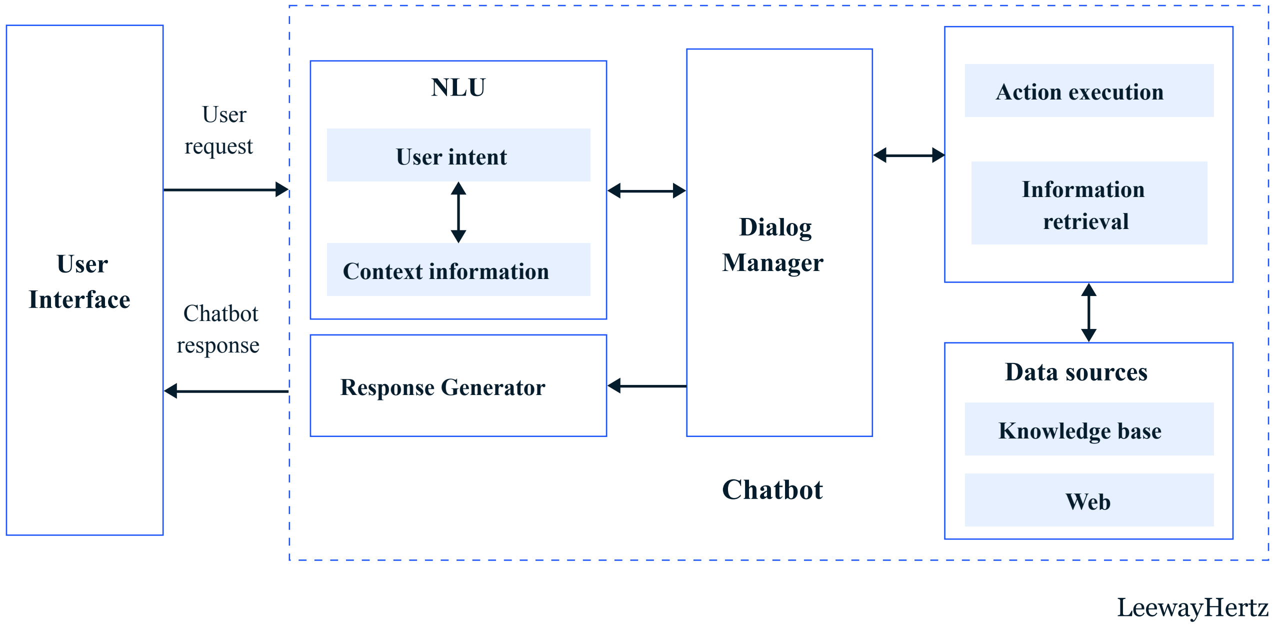
Retrieval-based chatbots: Chatbots that rely on retrieval create responses based on predefined responses kept in their database. They don’t produce new responses but choose the most appropriate response based on user input. Retrieval-based chatbots are frequently employed for particular tasks or domains where predefined responses are possible.

Generative chatbots: Chatbots that develop responses on their own, rather than relying on predefined ones, are called generative chatbots. They can produce fresh, interactive replies in response to user input. Compared to retrieval-based chatbots, generative chatbots are more adaptable and imaginative in their responses, but they can also be more difficult to build and train.

Task-oriented chatbots: Chatbots built to do certain jobs or duties, such as scheduling appointments, offering customer assistance, or making reservations, are known as task-oriented chatbots. They have a limited functional range and are concentrated on carrying out a specific activity.

Conversational Chatbots: Chatbots built to have open-ended interactions with users are called conversational chatbots. They may not have clear tasks or responsibilities because they are primarily concerned with facilitating conversational interactions. Conversational chatbots are frequently employed for amusement, customer engagement, and social connections.

Architectural components of AI-powered chatbots and their operational mechanics



Chatbots are similar to a messaging interface where bots respond to users' queries instead of human beings. They look like other apps. But their UI layer works differently. Machine learning algorithms power the conversation between a human being and a chatbot.

ML algorithms break down your queries or messages into human-understandable natural languages with NLP techniques and send a response similar to what you expect from the other side. Let us share an example of how a chatbot works.

Suppose you have a smart AI-based conversational chatbot app on your phone or computer and want to travel from LA to New York. You can open the chatbot app and write a message:

"Book a flight from LA to New York."

You may get a response like this:

"How many people are traveling with you?

Once you send a response, the bot will respond with all relevant flight details in seconds. Sounds amazing. Right?

The generated response from the chatbot exhibits a remarkable level of naturalness, resembling that of genuine human interaction. However, it is essential to recognize the extensive efforts undertaken to deliver such an immersive experience.

First of all, a bot has to understand what input has been provided by a human being. Chatbots achieve this understanding via architectural components like artificial neural networks, text classifiers, and natural language understanding.

Let’s understand the architectural components in depth.

Text classifiers

In chatbot development, text classification is a typical technique where the chatbot is educated to comprehend the intent of the user’s input and reply appropriately. Text classifiers examine the incoming text and group it into intended categories after analysis. Certain intentions may be predefined based on the chatbot’s use case or domain. Text classification can be accomplished using deep learning models like Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN), as well as machine learning methods like naive bayes and Support Vector Machines (SVM). Since the input text is linked to specified categories or intentions, these models are trained on labeled data, and to forecast the intent of the incoming text input, the model learns the patterns and features in the text data.

Once the intent of the text input has been determined, the chatbot can produce a response or carry out the appropriate activities in accordance with the programmed responses or actions related to that intent. For instance, if the user wants to book a flight, the chatbot can request essential details, such as the destination, time of travel, and the number of passengers, before booking the flight as necessary.

Text classifiers are a crucial part of chatbot architecture because they allow the chatbot to comprehend user input and react accordingly based on the user’s intent, resulting in a more tailored and meaningful conversation experience.

Natural Language Processing (NLP)

Natural Language Processing (NLP) is a subfield of artificial intelligence that enable computers to understand, interpret, and respond to human language. Applications for NLP include chatbots, virtual assistants, sentiment analysis, language translation, and many more.

The processing of human language by NLP engines frequently relies on libraries and frameworks that offer pre-built tools and algorithms. Popular libraries like NLTK (Natural Language Toolkit), spaCy, and Stanford NLP may be among them. These libraries assist with tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis, which are crucial for obtaining relevant data from user input.

The preprocessing of the incoming text is the initial stage in NLP. Tokenization separates the text into individual words or phrases (tokens), eliminating superfluous features like punctuation, special characters, and additional whitespace. To reduce noise in the text data, stopwords, which are frequent words like “and,” “the,” and “is,” can be safely eliminated.

Further, lemmatization and stemming are methods for condensing words to their root or fundamental form. While stemming entails truncating words to their root form, lemmatization reduces words to their basic form (lemma). For instance, “running” would be stemmed and lemmatized to “run.” Part of Speech (POS) tagging is then employed, which entails classifying each word in the text according to its grammatical type, such as a noun, verb, adjective, etc. Understanding the grammatical structure of the text and gleaning relevant data is made easier with this information.

Named Entity Recognition (NER) is a crucial NLP task that involves locating and extracting specified data from user input, including names of individuals, groups, places, dates, and other pertinent entities. The chatbot or other NLP programs can use this information to interpret the user’s purpose, deliver suitable responses, and take pertinent actions.

The chatbot responds based on the input message, intent, entities, sentiment, and dialogue context. This can involve selecting pre-defined responses from a set of templates, dynamically producing responses using text generation techniques like language models, or combining pre-defined templates with dynamic text generation to provide tailored responses. Natural language generation is the next step for converting the generated response into grammatical and human-readable natural language prose. This process may include putting together pre-defined text snippets, replacing dynamic material with entity values or system-generated data, and assuring the resultant text is cohesive. The chatbot replies with the produced response, displayed on the chat interface for the user to read and respond to.

The chatbot may continue to converse with the user back and forth, going through the above-said steps for each input and producing pertinent responses based on the context of the current conversation.

Knowledge base

A knowledge base is a collection of data that a chatbot utilizes to generate answers to user questions. It acts as a repository of knowledge and data for the chatbot to deliver precise and accurate answers to user inquiries.

Installing the chatbot will determine the exact format of the knowledge base. It could be a database with organized data like facts, Frequently Asked Questions (FAQs), product details, or other pertinent data. The chatbot can run a query against this database to get the relevant data and utilize it to generate responses. A knowledge base could also include a library of pre-written responses or templates that the chatbot can utilize to create responses on the go. These pre-written replies can be created to address frequent user scenarios and queries. The following steps are often required for a chatbot’s knowledge base to function:

Collecting essential data is the first stage in creating a knowledge base. Text files, databases, webpages, or other information sources create the knowledge base for the chatbot. After the data has been gathered, it must be transformed into a form the chatbot can understand. Tasks like cleaning, normalizing, and structuring may be necessary to ensure the data is searchable and retrievable.

The knowledge base's content must be structured so the chatbot can easily access it to obtain information. To do this, it may be necessary to organize the data using techniques like taxonomies or ontologies, natural language processing (NLP), text mining, or data mining.

The knowledge base must be indexed to facilitate a speedy and effective search. Various methods, including keyword-based, semantic, and vector-based indexing, are employed to improve search performance. The collected data may subsequently be graded according to relevance, accuracy, or other factors to give the user the most pertinent information. The chatbot explores the knowledge base to find relevant information when it receives a user inquiry. After retrieving the required data, the chatbot creates an answer based on the information found.

A knowledge base must be updated frequently to stay informed because it is not static. Chatbots can continuously increase the knowledge base by utilizing machine learning, data analytics, and user feedback. To keep the knowledge base updated and accurate, new data can be added, and old data can be removed. The knowledge base is connected with the chatbot's dialogue management module to facilitate seamless user engagement. The dialogue management component can direct questions to the knowledge base, retrieve data, and provide answers using the data.

A chatbot knowledge base generally functions by gathering, processing, organizing, and expressing information to facilitate effective search, retrieval, and response creation. It is an essential element that allows chatbots to offer users accurate and relevant information and continuously enhance their performance through continuous learning.

Dialogue management

A crucial part of a chatbot is dialogue management which controls the direction and context of the user's interaction. Dialogue management is responsible for managing the conversation flow and context of the conversation. It keeps track of the conversation history, manages user requests, and maintains the state of the conversation. Dialogue management determines which responses to generate based on the conversation context and user input. Let's explore the technicalities of how dialogue management functions in a chatbot.

Input processing: The chatbot accepts inputs from the user, which may be speech or text. The task of retrieving pertinent data from the input, including intent, entities, and context, falls into the input processing module. Variables, such as dates, names, or locations, are the specific bits of information provided as the input, whereas intent represents the user's intention or purpose behind the input. Context is used to describe pertinent details from the conversation's earlier turns, which aids in preserving the dialogue's flow.

Intent recognition: Inferring the user's intention based on their input is a crucial step in dialogue management for chatbots. Various methods can be employed to achieve this, including rule-based strategies, statistical approaches, and machine learning algorithms such as natural language understanding (NLU) models. NLU models utilize techniques like deep learning, recurrent neural networks, or transformers to process and understand the input text, enabling them to identify the intent behind the user's message.

Dialogue state tracking: The chatbot updates its internal dialogue state, which depicts the current context and stage of the interaction, as soon as the intent is identified. The dialogue state contains any important context that must be retained to preserve discussion consistency, such as user choices, system answers, and other information. Dialogue state tracking is essential for managing the dynamic nature of discussions and delivering suitable responses based on the current state of the dialogue.

Conversation policy: The dialogue policy decides the chatbot's action or response based on the current dialogue state. The conversation policy may be rule-based, in which case previously established rules determine the response, or it may be machine-learned using deep learning or reinforcement learning. The chatbot is trained using reinforcement learning to discover the best course of action based on a reward signal that denotes the effectiveness of the action taken. Large amounts of training data can be utilized to learn the dialogue policy using deep learning techniques like neural networks.

Response generation: The chatbot generates a response to the user once the dialogue policy decides the proper action or reaction. Natural Language Generation (NLG) techniques or predefined templates can generate the response, with placeholders filled in with the relevant data from the dialogue state. The goal of NLG techniques is to produce text that sounds like human speech based on the state of the conversation and the desired response. These techniques can use template-based generation, rule-based generation, or machine learning-based generation using tools like recurrent neural networks or transformers.

System response: The system responds by sending the generated response back to the user. Depending on the chatbot's capabilities and the user's desires, the system's answer may be in text, speech, or other modalities.

User interaction and feedback: The chatbot keeps up the conversation with the user, taking in new information and changing the dialogue's status in response to the user's responses. The chatbot may also gather user feedback to improve over time by enhancing responses.

Error handling and fall-back mechanisms: Conversation management also entails handling mistakes and fall-back procedures if the chatbot cannot discern intent, extract entities, or produce suitable responses. The error handling strategies may include error correction, re-prompting, or clarifying inquiries. Fall-back techniques can involve redistributing the workload or offering substitute replies.

Machine learning models

The components of the chatbot architecture heavily rely on machine learning models to comprehend user input, retrieve pertinent data, produce responses, and enhance the user experience.

Collecting data is the initial step in creating an ML-based chatbot. A wide variety of inputs and outputs, including text dialogues, user questions, and related answers, can be included in this data. The ML algorithms are trained using this data to recognize and respond to various inputs and query types.
After data collection is preprocessed to eliminate noise and irrelevant information, the data is extracted with the required features or patterns from the text data. Tokenization, a technique that separates text into individual words or phrases and generates numerical representations of these words or phrases, such as word embeddings or bag-of-words representations, may be used in this situation. These features operate as inputs to the ML algorithms, assisting them in interpreting the meaning of the text.

The ML model needs to be trained after the data has been preprocessed and the features are extracted. Chatbots can use a variety of machine learning algorithms, including rule-based algorithms, traditional ML algorithms like naive bayes and decision trees, and more sophisticated ML algorithms like recurrent neural networks, convolutional neural networks, and transformer models. The model builds a mapping between user queries and responses during training by learning from the input attributes and associated outputs.

The ML model must be tested after training to gauge its effectiveness. A valid set of data—which was not used during training—is often used to accomplish this. The model's performance can be assessed using various criteria, including accuracy, precision, and recall. Additional tuning or retraining may be necessary if the model is not up to the mark. Once trained and assessed, the ML model can be used in a production context as a chatbot. Based on the trained ML model, the chatbot can converse with people, comprehend their questions, and produce pertinent responses. For a more engaging and dynamic conversation experience, the chatbot can contain extra functions like natural language processing for intent identification, sentiment analysis, and dialogue management.

After deployment, an iterative procedure can continuously enhance the chatbot. The ML model can be further trained and improved by gathering user input and interactions. The chatbot can continuously learn from and adjust to human behavior thanks to this feedback loop, which helps it perform better over time.

Machine learning plays a crucial role in the technical operation of chatbots by enabling them to comprehend user inputs, produce suitable responses, and enhance their performance over time through continuous learning and adaptation.

Backend services

A chatbot’s integration and functionality depend heavily on backend services. The backend handles the underlying business logic, manages data storage, and interacts with external systems to ensure the chatbot runs smoothly. In a chatbot architecture, backend services have the following critical functions:

Business logic: The underlying code or algorithms that implement the particular functionality or features of the chatbot are referred to as the chatbot’s business logic. Based on specified rules, scripts, or machine learning models, it dictates how the chatbot understands user inputs, creates appropriate responses, and handles different jobs or activities.

Integration with external systems: To retrieve or store data, backend services frequently need to integrate with external systems like databases, APIs, or other services. This can involve getting user data and information from external APIs or updating databases with relevant data from user interactions.

API integrations: Backend services can integrate with external APIs to obtain information or features from other services, such as payment processors, third-party databases, weather APIs, or weather forecasting tools. As a result, the chatbot can utilize other resources to give users access to more thorough and accurate information and services.

Backend services are essential for the overall operation and integration of a chatbot. They manage the underlying processes and interactions that power the chatbot’s functioning and ensure efficiency.

User Interface

The user interface in a chatbot serves as the bridge between the chatbot and consumers, enabling communication through a message interface like an online chat window or messaging app. This component plays a crucial role in delivering a seamless and intuitive experience. A well-designed UI incorporates various elements such as text input/output, buttons, menus, and visual cues that facilitate a smooth flow of conversation. The UI must be simple, ensuring users can easily understand and navigate the chatbot’s capabilities and available options. Users can effortlessly ask questions, receive responses, and accomplish their desired tasks through an intuitive interface, enhancing their overall engagement and satisfaction with the chatbot.

Analytics and monitoring

A chatbot architecture must have analytics and monitoring components since they allow tracking and analyzing the chatbot’s usage and performance. They allow for recording relevant data, offering insights into user interactions, response accuracy, and overall chatbot efficacy. The performance and capabilities of the chatbot enhance over time with the use of this data.

Monitoring performance metrics such as availability, response times, and error rates is one-way analytics, and monitoring components prove helpful. This information assists in locating any performance problems or bottlenecks that might affect the user experience. These insights can also help optimize and adjust the chatbot’s performance.

User interaction analysis is essential for comprehending user trends, preferences, and behavior. Analytics and monitoring components offer insights into how users interact with the chatbot by collecting data on user queries, intentions, entities, and responses. This data can be utilized to spot trends, frequently asked questions by users, and areas where the chatbot interpretations and response capabilities should be strengthened.

What are the advantages of using AI-powered chatbots?

Here are some key advantages of deploying AI chatbots in various business processes:

Improved end-user experience: Chatbots provide end-user support on a real-time basis in any setting, whether in a retail sales store, product support center/website or a business front or back office. Because these interfaces are readily available to end-users, there is no specific wait time. This means customers or end-users can readily have answers to their queries, which significantly enhances the user experience. Based on the query, chatbots can present users with rich content with documentation, videos, and so on to help resolve queries. Furthermore, chatbots can provide 24/7 assistance and support to end-users. They can be programmed to immediately provide automated answers to common queries and forward the request to a real person when a more comprehensive action is required. This has a significant positive impact on customer and user experience.

Increased face time with customers: Businesses can use chatbots to increase their face time. Modern consumers anticipate a personalized relationship with their preferred brands through longer interactions and more tailored communication channels. Chatbots facilitate this by facilitating quicker and easier access. Moreover, chatbots are easily integrated into well-known platforms like Facebook or Instagram, allowing for a seamless user experience.

Analytics and insights: Chatbots are a great communication channel and a medium to gather insights into customer preferences and behavior. Businesses can collect instant feedback from customers and end-users through chatbots and then analyze the data to gather insights into their habits and preferences. Besides, chatbots can also be leveraged to identify purchasing patterns and consumer behavior. It can help businesses make critical decisions around product marketing and launch strategies.

Lead generation and conversion: Chatbots can be developed to offer users personalized guidance and assistance throughout their shopping experience. Chatbots can recommend goods or services that are probably interesting to the user by gathering information on their preferences, behavior, and previous purchases. Moreover, chatbots can employ persuasive language and techniques to influence user judgment and persuade them to purchase.

For instance, a chatbot on an e-commerce website can inquire about the user’s tastes and spending limit before making product recommendations that match those parameters. To persuade the user to buy anything, the chatbot can also provide social evidence, such as testimonials and ratings from other consumers. Chatbots can occasionally offer users special discounts or promotions to entice them to buy. Businesses may boost conversion rates and customer satisfaction by introducing chatbots to help consumers through shopping. Chatbots can make users’ buying experiences more personalized and interesting, enhancing customer retention and brand loyalty.

Cost savings and scalability: Developing and implementing a fully functional chatbot is faster and cheaper than developing a cross-platform app or hiring employees to handle many incoming queries. Thus, businesses can significantly save hiring, training and payroll costs. A typical chatbot would only involve the initial development cost and a nominal runtime cost, potentially less than the costs spent on actual human resources.

Furthermore, multi-lingual chatbots can scale up businesses in new geographies and linguistic areas relatively faster.

Let’s explore the process of building an AI-powered chatbot using Python.

Step-1: Import necessary libraries

You can import the libraries using the following code:

```
#import necessary libraries

import io

import random

import string # to process standard python strings

import warnings

import numpy as np
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity

import warnings

warnings.filterwarnings('ignore')
```

Step-2: Download and install NLTK

A popular toolkit for creating Python applications that interact with human language data is NLTK (Natural Language Toolkit). It offers user-friendly interfaces to more than 50 corpora and lexical resources, including WordNet, and a collection of text-processing libraries for categorization, tokenization, stemming, tagging, parsing, and semantic reasoning, which act as wrappers for powerful NLP libraries.

Python’s Natural Language Processing offers a useful introduction to language processing programming.

Go here for platform-specific instructions.

```
pip install nltk
```

Step-3: Install NLTK packages

Install the NLTK packages using the following codes:

```
import nltk

from nltk.stem import WordNetLemmatizer

nltk.download('popular', quiet=True) # for downloading packages

#nltk.download('punkt') # first-time use only

#nltk.download('wordnet') # first-time use only
```

Step-4: Read in the corpus

We will use the Wiki page on chatbots as our corpus for this example. Copy the page’s content and paste it into a text file called “chatbot.txt,” then save it. You are free to use any corpus, though.

```
f=open('chatbot.txt','r',errors = 'ignore')

raw=f.read()
```

raw = raw.lower()# converts to lowercase

The primary drawback of text data is that it is only available in text form (strings). Unfortunately, machine learning algorithms cannot complete the task without a numerical feature vector. So, any NLP project must be pre-processed to be ready to work before we begin. Simple text pre-processing consists of the following:

- Making all of the text capital or lowercase will prevent the algorithm from treating similar words in various cases as different.
- Tokenization refers to turning a list of tokens or words we desire from the usual text strings. You can use the Word tokenizer to find the list of words in strings and the Sentence tokenizer to get the list of sentences.

The NLTK data package includes a pre-trained Punkt tokenizer for English.

- Filtering out the noise or anything that isn’t a regular number or letter.
- Elimination of the Stop words. While choosing texts that best fit a user’s needs, some extremely popular words that would seem to be of little utility are occasionally completely omitted from the lexicon. These phrases are known as stop words.
- Stemming: Inflected (or occasionally derived) words are stemmed or reduced to their stem, base, or root form, typically a written word form. For instance, the word “stem” would arise from stemming the phrases “Stems,” “Stemming,” “Stemmed,” and “Stemtization.”
- Lemmatization: Lemmatization is a minor variation of stemming. The primary distinction between both is that lemmas are actual words, whereas stemming frequently produces fictitious words. In a dictionary, you cannot look up your root stem or the word you end up with, but you can look up a lemma instead. Lemmatization examples include the fact that “run” serves as the root for terms like “running” or “ran” or the fact that the words “better” and “good” are in the same lemma and are therefore regarded as one and the same.

Step-5: Tokenization

For tokenization, follow the codes given below:

#TOkenisation

sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences

word_tokens = nltk.word_tokenize(raw)# converts to list of words

Step-6: Pre-processing

Now, we will build a function called LemTokens, which will take the tokens as an argument and output normalized tokens.

```
# Preprocessing

lemmer = WordNetLemmatizer()

def LemTokens(tokens):

return [lemmer.lemmatize(token) for token in tokens]

remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)

def LemNormalize(text):

return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

Step-7: Keyword matching

The function for a bot's greeting will then be defined; if a user inputs a greeting, the bot will respond with a greeting. While welcoming people, ELIZA employs simple keyword matching. Now, we will apply the same idea here.

```
# Keyword Matching

GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up","hey",)

GREETING_RESPONSES = ["hi", "hey", "*nods*", "hi there", "hello", "I am glad! You are talking to me"]

def greeting(sentence):

"""If user's input is a greeting, return a greeting response"""

for word in sentence.split():

if word.lower() in GREETING_INPUTS:

return random.choice(GREETING_RESPONSES)
```

Step-8: Generating response

Bag of words

Following the preliminary preprocessing stage, text must be converted into a useful vector (or array) of numbers. The bag-of-words is a textual illustration illustrating how words appear in a document. There are two components:

- A collection of well-known words.
- A metric for the number of well-known words.

Why is it referred to as a “bag” of words? That’s because the model only cares about whether the known words are in the document, not where they appear, and any information about the order or structure of words in the document is ignored.

The underlying premise of Bag of Words is that two documents are comparable if they contain similar information. Additionally, the document’s content itself can provide some insight into the meaning of the document.

For instance, if we were to vectorize the sentence “Learning is excellent,” but our dictionary had the words “Learning, is, the, not, great,” we would get the following vector: (1, 1, 0, 0, 1).

TF-IDF approach

The Bag of Words method has the drawback of making extremely common terms predominant in the document (leading to a higher score), but the paper may not include as much “informational content.” Also, it will give lengthier documents more weight than shorter ones.

One strategy is to rescale the frequency of words according to how frequently they appear in all texts, penalizing words like “the” that are frequently used across all publications. Term Frequency-Inverse Document Frequency, or TF-IDF for short, is a method of scoring that considers:

- The frequency of a word in the present document is scored using the term “term frequency.”
- TF = (Number of occurrences of term t in a document)/ (Number of terms in the document)
  A measure of how uncommon a term is across documents is called Inverse Document Frequency.
- IDF = 1 + log(N/n), where N and n are the numbers of documents in which a term t has appeared.

Cosine similarity

The below-mentioned code implements a response generation function using the TF-IDF (Term Frequency-Inverse Document Frequency) technique and cosine similarity. The Tf-idf weight is a weight that is frequently used in text mining and information retrieval. This weight is a statistical metric to assess a word’s significance to a collection or corpus of documents.

Where d1,d2 are two non-zero vectors, cosine similarity (d1, d2) is calculated as follows: Dot product (d1, d2) / ||d1|| * ||d2||.

When a user provides input, their response is appended to a list of previously processed sentences. The TF-IDF vectorizer is used to convert these sentences into a numerical representation. Then, the cosine similarity between the user’s input and all the other sentences is computed.

The code finds the sentence with the second-highest similarity score to the user’s input. If the similarity score is zero, indicating no relevant response, the code generates a default “I am sorry! I don’t understand you” response. If the similarity score is not zero, the code retrieves the most similar sentence and uses it as the generated response. The following codes are used for the above-mentioned process:

```
# Generating response
def response(user_response):
robo_response=''
sent_tokens.append(user_response)
TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
tfidf = TfidfVec.fit_transform(sent_tokens)
vals = cosine_similarity(tfidf[-1], tfidf)
idx=vals.argsort()[0][-2]
flat = vals.flatten()
flat.sort()
req_tfidf = flat[-2]
if(req_tfidf==0):
robo_response=robo_response+"I am sorry! I don't understand you"
return robo_response
else:
robo_response = robo_response+sent_tokens[idx]
return robo_response
```

Finally, based on the user’s input, we will provide the lines we want our bot to say while beginning and concluding a conversation.

```
flag=True
```

```
print("ROBO: My name is Robo. I will answer your queries about Chatbots. If you want to exit, type Bye!")
while(flag==True):
user_response = input()
user_response=user_response.lower()
if(user_response!='bye'):
if(user_response=='thanks' or user_response=='thank you' ):
flag=False
print("ROBO: You are welcome..")
else:
if(greeting(user_response)!=None):
print("ROBO: "+greeting(user_response))
else:
print("ROBO: ",end="")
print(response(user_response))
sent_tokens.remove(user_response)
else:
flag=False
print("ROBO: Bye! take care..")
```

We at LeewayHertz build robust AI solutions to meet your specific needs. Our generative AI platform, ZBrain.ai, allows you to create a ChatGPT-like app using your own knowledge base. You only need to link your data source to our platform; the rest is on us. ZBrain supports data sources in various formats, such as PDFs, Word documents, and web pages.

The smartest AI chatbots used across industries

Here are some of the smartest AI chatbots you should explore to experience the power of AI:

- **Watson Assistant**
  Built by IBM (one of the leaders in the AI space), Watson Assistant is the most advanced AI-powered chatbot in the market. It can be pre-trained with data from your particular industry to understand your historical call logs, chat, ask customers for clarity, connect them with human representatives, search for an answer in your knowledge base and provide you with training recommendations to enhance your conversational abilities.
- **Rulai**
  Powered with deep learning-based natural language understanding and multitasking capabilities, Rulai is an AI-powered chatbot for enterprises. It can predict user behavior, analyze the conversation context, take actions, move to different tasks, ask customers for more clarity, and understand customer preferences.
- **Inbenta**
  Designed explicitly for enterprises, Inbenta leverages its own NLP (Natural Language Processing) engine and machine learning to discover the context of each customer conversation and respond to their questions accurately. It has a dialog manager that allows you to design custom conversation flows.
  When the Inbenta chatbot feels that any of your customers should talk to a human for a specific concern, it escalates the conversation to the right support agent.
- **LivePerson**
  By gathering over 20 years of messaging transcript data and feeding it to the AI Chatbot, LivePerson, it can automate messaging for every industry and integrate with messaging channels such as mobile apps, websites, text messaging, Apple business chat, Line, Whatsapp, Google, Facebook Messenger, Google AdLingo and Google Rich Business messaging.
- **Bold360**
  Bold360 patented its own NLP engine to allow brands to develop chatbots that can understand the customer's intent without the requirement of keyword matching, and it knows how to provide the most accurate answers. It can interpret complicated language, respond to customers with natural responses and remember the context of a conversation.

Endnote

Chatbots have emerged as a powerful technology that combines the strengths of artificial intelligence and natural language processing, enabling automated interactions and the simulation of human-like conversations. With their integration into the workflows of various industries, such as e-commerce, healthcare, and banking, chatbots have proven to enhance operational efficiency, streamline processes, and improve overall customer experience. As their adoption continues to grow rapidly, chatbots have the potential to fundamentally transform our interactions with technology and reshape business operations. AI-powered chatbots offer a wider audience reach and greater efficiency compared to human counterparts. Looking ahead, it is conceivable that they will evolve into valuable and indispensable tools for businesses operating across industries.

Ready to build an AI-based chatbot? Consult our LeewayHertz AI experts and enhance internal operations as well as customer experience with a robust chatbot.