Development of a Chatbot Using Speech Recognition using Machine Learning

CS584 Machine Learning Spring 2023

Basani Priyanka (A20513566) pbasani@hawk.iit.edu

Depala Rajeswari (A20526535) rdepala@hawk.iit.edu

Vinitha Inaganti (A20514310) vinaganti@hawk.iit.edu

Abstract

Since a long time ago, academics have been fascinated with human-computer communication. Interaction between humans and machines can be carried out in a wide range of methods. Chatbots are a common method for conducting these kinds of conversations. A chatbot is a computer software that facilitates engaging and simple conversation. Modern artificial intelligence techniques perform poorly even when responding to user queries in most suitable manner. As a result, sectors currently favour rule based Chatbot systems. We give a thorough analysis of the implementation of rule based Chatbot systems in this article. The parameters for measuring chatbot system success are covered in the article. New speech recognition methods and technologies will be used more frequently in daily life in the future because they greatly cut down on interaction time by replacing messaging with voice/audio.

Key words: natural language processing, speech-to-text, Python, Flask, chatbot,tensorflow, NLTK, Speech Recognition , PyAudio , Keras , Tkinter.

Introduction

The development of software that resembles human speech is still important currently. The collection of requests and replies serves as the simplest depiction of a conversation. The description of the knowledge base and the operation of the interpreter software are issues in this instance. The knowledge base's markup language may include question patterns, matching answer patterns, the context of the conversations that led up to them, and the title of the relevant communication subject. Further obligations that a chatbot can carry out include audio search, image search, fact search, calculator, weather forecast, and exchange rate display. The majority of these features have online implementations and are accessible via external APIs. The chatbot answer is mechanically generated by a program that analyses and parses the user's text. This algorithm considers the text's morphology and relationship subjects.

Fig 1: Illustration of Speech to Text transformation

Live support staff can manage challenging inquiries that call for a personal touch thanks to chatbots. The user is instantly satisfied by receiving an all the time response to their inquiry, which is more essential. As a result, we employ Machine Learning methods (also known as Natural Language Processing) to understand client inquiries through voice recognition. Text recognition technology is used by chatbots to understand visible data and react appropriately. These chatbots are commonly used in apps today that only take text input, which is a common feature. However, programs like ChatGPT do not allow speech input.

Data Collection:

A set of cases used to teach and assess the performance of chatbots is known as a chatbot dataset. The chatbot should be able to react to the variety of inputs and related outputs that constitute the dataset. Natural language text, voice, and other types of data that the chatbot is programmed to detect can all be used as inputs. The chatbot has been configured to generate a variety of outcomes, including writing, voice, and other output types.

However, we'll be utilizing the Kaggle dataset which consists of text samples labelled with their corresponding intents. The collection includes instances using English and have been created for purpose detection in chatbots. It is made up of a single JSON file called "intents.json," which includes a series of dictionaries. A variety of keywords are found in each dictionary, each of which symbolizes an intent:

One array in the **"intents.json"** file, for instance, might resemble this:

{ "intent": "greeting", "examples": ["Hello!", "Hi there.", "Good morning."] }

Three text samples that are related to the "greeting" intent are included in this dictionary, which depicts the intent. There are a total of 23 intents in the collection, including hellos, goodbyes, thank you, and inquiries about the weather, meals, and entertainment. Machine learning models for chatbot purpose detection can be trained using this dataset. The models are capable of learning to discern the user's purpose when providing input and can then adapt their responses accordingly. To be clear, this dataset does not contain answers or context for the intents, so it might not be enough on its own to build a completely working chatbot.

Every sample in the collection is a text string that symbolizes user input and matches a specific purpose. For instance, an illustration from the information related to the "weather" purpose might resemble this:

Is today going to be rainy?

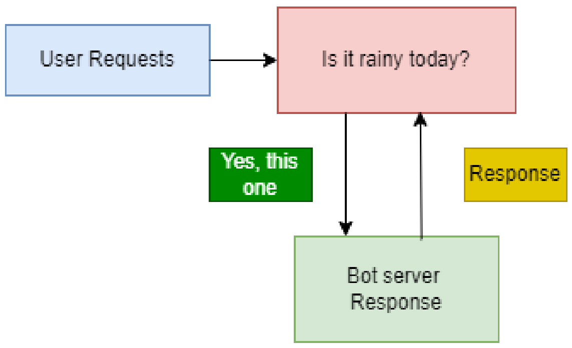

Fig 2: Simplified Illustration of user and bot communication

This illustration depicts a customer input that inquiries about the weather.

A machine learning model can be trained using the information to determine the purpose of user inputs in a chatbot. Natural language processing (NLP) and machine learning methods can be used to teach the model. Once trained, the model can be used to categorize fresh user inputs and produce pertinent answers based on the purpose. It is significant to observe that neither the dataset nor the intended setting will contain any responses to the questions. As a result, it might not be enough by itself to build a chatbot that is completely effective. The dataset is a helpful place to start when developing a chatbot, but additional knowledge and advanced techniques will probably be required in order to build one that can process user input and react correct.

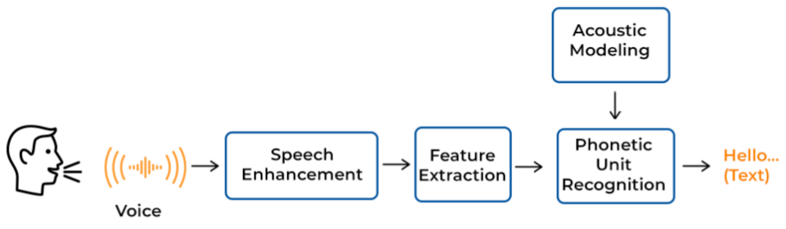

Fig 3: Speech Recognition Process

Load Libraries

Natural Language Toolkit (NLTK) - The text is preprocessed using NLTK, which tokenizes words, eliminates stop words, and lemmatizes words.

- **Tokenization:** Tokenization is the process of separating a document into tokens, which are individual phrases. Word_tokenize, sent_tokenize, and regexp_tokenize are just a few of the tokenizers offered by NLTK. Here is a usage case for word_tokenize:

```
from nltk.tokenize import word_tokenize
text = "This is a sample sentence."
tokens = word_tokenize(text)
print(tokens)
# Output: ['This', 'is', 'a', 'sample', 'sentence', '.']
```

- **Stop word removal:** Stop words are frequent words that are typically eliminated from text data because they don't contribute any additional sense to the text. A collection of stop phrases for various languages is provided by NLTK. Here is an illustration using stop phrases in English:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stop_words = set(stopwords.words('english'))
text = "This is a sample sentence with some stop words."
tokens = word_tokenize(text)
filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
print(filtered_tokens)
# Output: ['sample', 'sentence', 'stop', 'words', '.']
```

- **Lemmatization:** Lemmatization is the method of stripping words down to their lexicon or most basic shape. (lemma). A WordNetLemmatizer class from NLTK is available for lemmatizing phrases. Here's an illustration:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
word = "running"
lemma = lemmatizer.lemmatize(word, pos='v')
print(lemma)
# Output: 'run'
```

- **Speech Recognition**: The speech recognition function is used to recognize speech from the audio source and convert it to text.

```
import speech_recognition as sr
def recognize_speech():
r = sr.Recognizer()
with sr.Microphone() as source:
audio = r.listen(source)
return r.recognize_google(audio)
```

- **NaiveBayesClassifier** : NaiveBayesClassifier is used to train a model that can recognize the intent of user input based on previously labelled examples.

Train the Data

The Naive Bayes Classifier is a well-known machine learning method for text categorization problems, such as chatbots. The purpose of a chatbot is to detect the intent of user input using previously tagged samples.

The Naive Bayes Classifier method calculates the likelihood of a specific class (intent) given the input text. This is accomplished by computing the conditional probability of each feature (word or combination of words) in the input text in relation to the class (intent).

Labelled samples of user input and their accompanying intentions are used to train a Naive Bayes Classifier for a chatbot. The labelled examples are used to determine each feature's conditional probability for each intent. The program learns which characteristics are most predictive of each intent throughout the training phase.

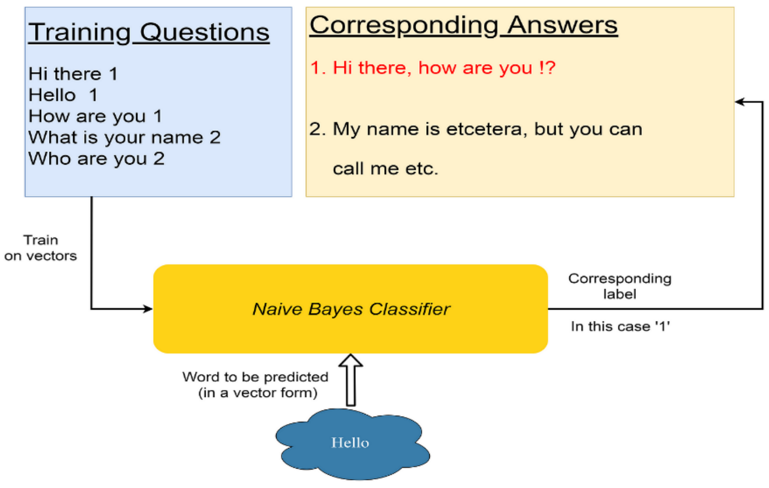

Fig 4: Illustration of queries and responses using Naïve Bayes Classifier

The Naive Bayes Classifier evaluates the likelihood of each intent given the input text when a new user input is received. The highest likelihood intent is then picked as the projected intent for the input.

Assume we have a chatbot that assists consumers in booking flights. If a user enters "I want to book a flight to New York," the Naive Bayes Classifier will assess the likelihood of each intent, such as "book flight" or "search flight information." The classifier will anticipate the most probable purpose and respond accordingly based on the determined probability.

Overall, the Naive Bayes Classifier is an effective algorithm for chatbot text classification jobs since it properly classifies user input into the right intent, allowing the chatbot to respond to user requests.

```
import nltk

from nltk.classify import NaiveBayesClassifier

# split the data into training and testing sets

train_set, test_set = featuresets[:800], featuresets[800:]

# train the Naive Bayes Classifier

classifier = NaiveBayesClassifier.train(train_set)
```

Unfinished Work

Neural Network : A neural network model using TensorFlow will be defined by the script. It will have three completely linked layers with **ReLU and softmax** activation functions. With the aid of the **Adam algorithm** and the category cross-entropy loss function, the model will be trained on the preprocessed training data.

The script will specify a function for handling user input and receiving an answer from the chatbot once the model has been trained. The function will tokenize the incoming message, stem the words, and produce a depiction in the form of a bag of words. Next, a response will be chosen from the training data using the model to anticipate the incoming message's intention.

Create a GUI interface : The Tkinter utility will be used by the software to create a GUI interface for the chatbot. A text submission box and a text box for showing the conversation record are both included in the user interface. A "Send" icon will be used to transmit the user's input to the chatbot, and the chatbot's answer will be shown in the conversation record.

The model will be compiled using the categorical cross entropy loss function, the Adam optimizer, and accuracy will be used as the evaluation metric. The model will then be trained using the fit method with the training data and output provided for a given number of epochs and batch size. Finally, the trained model will be saved to a **file named "model.h5".**

Testing: After the model is trained, it will be evaluated using a different dataset. This will give a sense of applicability and aid in evaluating how well the model performs on unknown data. Testing will help to find any problems or places for development in the functionality, usability, and user experience of the voice recognition model.

Metrics: A number of metrics, including word error rate (WER), sentence error rate (SER), and accuracy, will be used to assess the success of the **voice recognition algorithm**. These measurements will offer precise evaluations of the model's effectiveness.

Continuous Improvement: The speech recognition model will be consistently monitored and refined based on user input and performance measures, just like any other machine learning model.

Overall, assessment is an important part of a chatbot voice recognition project because it makes sure the model is correctly deciphering user input and reacting to it. These methods can be used to improve the user experience by optimizing the chatbot voice recognition algorithm for precision, speed, and usability.