Vivekanand Education Society's

Institute of Technology

(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)

Department of Information Technology

AIDS - 2 Lab

Experiment - 5

Aim: To build a Cognitive based application to acquire knowledge through audio/video files for a Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government etc.

Roll No.	70
Name	MAYURI SHRIDATTA YERANDE
Class	D20B
Subject	AIDS - 2
Grade:	

EXPERIMENT - 5

AIM: To build a Cognitive based application to acquire knowledge through audio/video files for a Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government etc.

THEORY:

A cognitive-based application, also known as a cognitive application or cognitive computing application, is a type of software or system that leverages artificial intelligence (AI) and machine learning techniques to mimic human cognitive functions such as reasoning, problem-solving, learning, and decision-making. These applications are designed to process and analyze vast amounts of data, understand natural language, and adapt to changing information and circumstances.

An audio-based cognitive system is a type of artificial intelligence (AI) system that is primarily designed to process and understand audio data using cognitive computing techniques. These systems are capable of mimicking human cognitive functions to analyze and derive insights from audio information.

1. **Define Objectives and Use Cases:**
   Determine the specific objectives of your audio-based cognitive system.
   Identify the use cases and applications for which you want to use the system.

2. **Data Collection:**
   Collect a diverse and representative dataset of audio recordings relevant to your objectives.
   Ensure the data is labeled or annotated appropriately for supervised learning if necessary.

3. **Data Preprocessing:**
   Clean and preprocess the audio data, which may include noise reduction, resampling, and audio segmentation (if working with longer recordings).
   Extract relevant features from the audio data, such as Mel-frequency cepstral coefficients (MFCCs) or spectrograms.

4. **Speech Recognition (ASR):**
   If your system involves transcribing spoken words, you may need to develop or use automatic speech recognition (ASR) technology to convert audio to text.
   Pretrained ASR models like those available in libraries like Hugging Face Transformers can be used.

5. **Natural Language Processing (NLP):**
   If you're working with transcribed text, apply NLP techniques to understand the meaning and context of the text.
   This may involve tasks like sentiment analysis, keyword extraction, and entity recognition.

IMPLEMENTATION:

1. Conversion of Audio to Text:-

- Install Speech Recognition Library

```
!pip install SpeechRecognition
```

```
Collecting SpeechRecognition
  Downloading SpeechRecognition-3.10.0-py2.py3-none-any.whl (32.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 32.8/32.8 MB 34.1 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.10/dist-packages (from SpeechRecognition) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->SpeechRecognition) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->SpeechRecognition) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->SpeechRecognition) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->SpeechRecognition) (2023.7.22)
Installing collected packages: SpeechRecognition
Successfully installed SpeechRecognition-3.10.0
```

- This Python code uses the `speech_recognition` library to transcribe audio from the file 'harvard.wav' into text using Google's Speech Recognition service.
- It then prints the transcribed text, handling potential recognition errors.
- Audio:-https://www.kaggle.com/datasets/pavanelisetty/sample-audio-files-for-speech-recognition?resource=download

```
import speech_recognition as sr

recognizer = sr.Recognizer()

audio_file_path = 'harvard.wav'

with sr.AudioFile(audio_file_path) as source:
    audio_data = recognizer.record(source)

try:
    text = recognizer.recognize_google(audio_data)
    print("Transcript:", text)
except sr.UnknownValueError:
    print("Speech recognition could not understand the audio.")
except sr.RequestError as e:
    print(f"Could not request results from Google Speech Recognition service; {e}")
```

Output:-

```
Transcript: the stale smell of old beer lingers it takes heat to
bring out the odor a cold dip restores health and zest a salt
pickle taste fine with ham tacos al pastor are my favorite a
zestful food is the hot cross bun
```

```
Transcript: the stale smell of old beer lingers it takes heat to
```

2. Performing Cognitive analysis on transcript text:-

- This Python code uses the `TextBlob` library to analyze sentiment (polarity and subjectivity) and `spaCy` for named entity recognition (NER) in a given text.
- It then prints the text's sentiment scores and identifies named entities, providing insights into its emotional tone and relevant entities.

```
[25] from textblob import TextBlob
     import spacy
     blob = TextBlob(text)
     sentiment = blob.sentiment

     nlp = spacy.load("en_core_web_sm")

     doc = nlp(text)

     entities = [(ent.text, ent.label_) for ent in doc.ents]

     print("Sentiment Polarity:", sentiment.polarity)
     print("Sentiment Subjectivity:", sentiment.subjectivity)
     print("Named Entities:", entities)
```

```
Sentiment Polarity: 0.02380952380952382
Sentiment Subjectivity: 0.5785714285714286
Named Entities: [('al', 'ORG')]
```

- Install Summarization Library

```
pip install bert-extractive-summarizer
```

```
Collecting bert-extractive-summarizer
  Downloading bert_extractive_summarizer-0.10.1-py3-none-any.whl (25 kB)
Collecting transformers (from bert-extractive-summarizer)
  Downloading transformers-4.33.1-py3-none-any.whl (7.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 19.1 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from bert-extractive-summarizer) (1.2.2)
Requirement already satisfied: spacy in /usr/local/lib/python3.10/dist-packages (from bert-extractive-summarizer) (3.6.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->bert-extractive-summari
```

- This code leverages the `bert-extractive-summarizer` library to perform extractive text summarization on the provided `text`.
- It then prints the generated summary, condensing the key information from the original text.

```
[23] from summarizer import Summarizer
     text = text
     model = Summarizer()
     summary = model(text)
     print(summary)
```

```
the stale smell of old beer lingers it takes heat to bring out the odor a cold dip restores health a
```

```
[24] print(summary)
```

```
the stale smell of old beer lingers it takes heat to bring out the odor a cold dip restores health a
```

CONCLUSION: We have created an application that converts spoken audio content into written text (transcript) and then provides a condensed version (summary) of that transcript. There are numerous commercial and open-source audio transcript and summary applications available, ranging from simple tools to advanced AI-driven solutions. The choice of an application depends on your specific needs, such as the volume of audio content, desired accuracy, and customization options.