SOCIALLY RELEVANT PROJECT REPORT

Submitted by

AAKAASH KB (711620243001)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

KATHIR COLLEGE OF ENGINEERIING

“WISDOM TREE”, NEELAMBUR, COIMBATORE – 641 062

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report
"MULTIMODAL REPRESENTATION LEARNING"
is the bonafide work of
"AAKAASH KB (711620243001)"

who carried out this project under my supervision.

SIGNATURE
Dr. M.RAJESH BABU, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Professor and Head
Department of
Artificial Intelligence and Data Science
Kathir College of Engineering
Coimbatore – 641 062

SIGNATURE
Dr.AYYAVOO MITHILA,
ASSISTANT PROFESSOR
Assistant Professor
Department of
Artificial Intelligence and Data Science
Kathir College of Engineering
Coimbatore – 641 062

Project viva voce held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our immense gratitude to **Thiru E.S. KATHIR, Chairman, Kathir Institutions**, Coimbatore for giving us an opportunity to study in their prestigious institution and to take up the project in Partial Fulfillment of the Regulation for the B.E Program.

We would like to express our deepest gratitude to **Thirumathi LAVANYA KATHIR, Secretary, Kathir Institutions**, Coimbatore for the soul support in our studies.

We would bound to express our gratitude to **Dr. G. DORAISAMY, CEO and Dr. R.UDAIYAKUMAR, M.E.,Ph.D., Principal, Kathir College of Engineering**, Coimbatore for their permission and constant encouragement throughout our course.

It is a great pleasure to express our sincere and wholehearted gratitude to Professor **Dr. Rajesh Babu,M.E.,Ph.D., Head of the Artificial Intelligence and Data Science**, for their constant suggestion and encouragement in the project work.

We also express our heartfelt thanks to **Dr.Mithila A, M.E.,(Ph.D.,), Assistant Professor and Project Guide, Department of Artificial Intelligence and Data Science** for being supportive throughout the tenture of our project.

We also thank all our Faculty Members and Non Teaching Staff Members of Department of Artificial Intelligence and Data Science and our Lovable Parents and Friends who contribute many suitable ways for achieving final results.

ABSTRACT

Multimodal representation learning, which involves capturing and understanding information from multiple modalities, has gained significant attention in various fields such as natural language processing, computer vision, and audio processing. The goal is to learn a joint representation that captures the inherent relationships and correlations between different modalities, enabling effective fusion and integration of information from diverse sources.

Multimodal representation learning plays a crucial role in various applications, including multimedia analysis, natural language processing, robotics, and human-computer interaction.

The challenges in multimodal representation learning include dealing with the heterogeneity and variability of different modalities, handling missing or incomplete modalities, and ensuring the interpretability and explainability of the learned representations. Additionally, scalability and efficiency are important considerations when dealing with large-scale multimodal datasets.

Text and audio are two important modalities in multimodal representation learning. Text is a structured form of communication that conveys meaning through written language, while audio represents information in the form of sound waves. These modalities provide complementary information, and combining them can enhance the understanding and analysis of various applications, such as speech recognition, natural language processing, and audio-visual scene understanding.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation of the Project.

Multimodal representation learning using text and audio stems from the increasing availability and importance of multimodal data in various domains. With the proliferation of digital content, we now have access to vast amounts of text and audio data, such as social media posts, podcast episodes, news articles, and more.

By combining text and audio modalities, we can leverage the complementary information present in both to enhance the understanding and processing of multimodal data. Text provides rich semantic and contextual information, while audio conveys additional cues like tone, intonation, and emotional aspects. By jointly modelling text and audio, we can capture a more comprehensive representation of the underlying content.

Multimodal representation learning is the increasing prevalence of voice-controlled systems and devices. With the rise of virtual assistants like Siri, Alexa, and Google Assistant, there is a growing demand for models that can effectively process and understand both textual and spoken commands.

Multimodal representation learning can enable more accurate and robust interactions with these systems by leveraging the combined power of text and audio understanding.

Multimodal representation learning using text and audio lies in the potential to unlock new levels of understanding and utilization of multimodal data in various domains, leading to improved performance and enhanced user experiences in applications ranging from natural language processing to audio processing and beyond.

1.2 Objective of the Project

The objective of the project "Multimodal Representation Learning using Text and Audio" is to develop a model that can effectively capture and leverage the information present in both textual and audio modalities. The goal is to learn a joint representation that can capture the semantic and contextual relationships between textual and audio data, enabling better understanding and analysis of multimodal content.

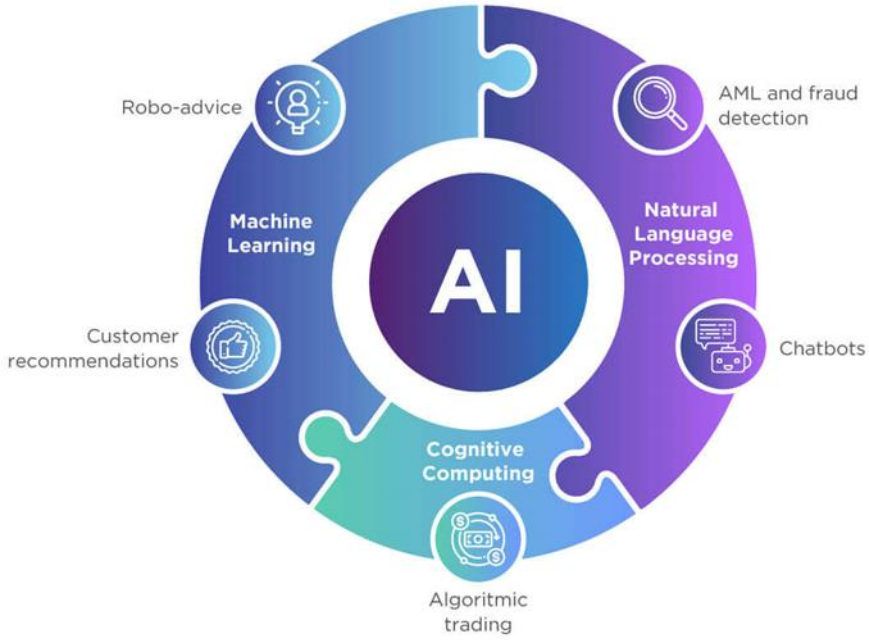
By combining information from text and audio, the project aims to address tasks that require a multimodal understanding, such as speech recognition, audio classification, sentiment analysis of spoken content, audio-based question answering, and more. The model should be capable of effectively fusing textual and audio features to provide enhanced performance and insights compared to using each modality individually.

The project may involve various steps, including data collection and preprocessing, designing and training a multimodal learning architecture, developing techniques for feature extraction and fusion, evaluating the model's performance on different multimodal tasks, and potentially fine-tuning or optimizing the model based on the specific objectives and requirements.

1.3 Artificial Intelligence

Artificial intelligence (AI), the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Artificial Intelligence is composed of two words Artificial and Intelligence, where Artificial defines "man-made," and intelligence defines "thinking power", hence AI means "a man-made thinking power."

Artificial Intelligence exists when a machine can have human based skills such as learning, reasoning, and solving problems. With Artificial Intelligence you do not need to preprogram a machine to do some work, despite that you can create a machine with programmed algorithms which can work with own intelligence, and that is the awesomeness of AI.

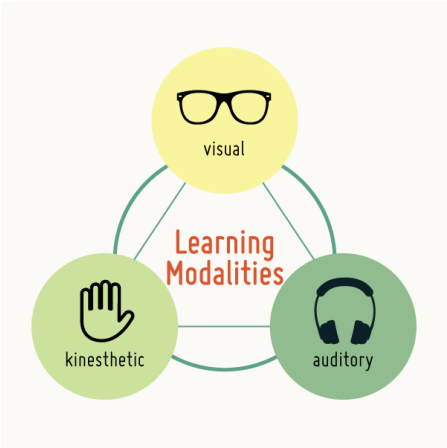


1.1.Artificial Intelligence

1.4 Multimodality

Multimodality is the application of multiple literacies within one medium. Multiple literacies or "modes" contribute to an audience's understanding of a composition. Everything from the placement of images to the organization of the content to the method of delivery creates meaning.

Multimodality is an inter-disciplinary approach that understands communication and representation to be more than about language. It has been developed over the past decade to systematically address much-debated questions about changes in society, for instance in relation to new media and technologies.

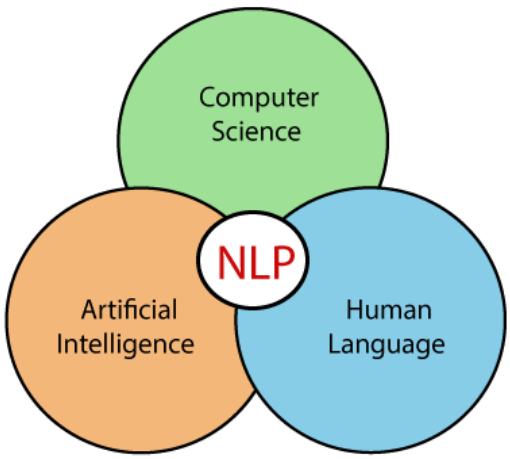


1.2.Multimodality

1.5 Natural Language Processing.

NLP stands for Natural Language Processing, which is a part of Computer Science, Human language, and Artificial Intelligence. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation.

Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language.



1.3.Natural Language Processing

1.6.Literature Survey

a) Zhang, C., Yang, Z., He, X., & Deng, L. (2020). Multimodal intelligence: Representation learning, information fusion, and applications. IEEE Journal of Selected Topics in Signal Processing, 14(3), 478-493.

Deep learning technologies have transformed speech recognition, picture recognition, and natural language processing. Input signals for each of these tasks are of a single modality. Many applications in the field of artificial intelligence, however, include many modalities. As a result, studying the more difficult and complicated topic of modelling and learning across various modalities is of broad interest. We present a technical overview of various models and learning approaches for multimodal intelligence in this study. The merging of vision and natural language modalities is the major subject of this study, which has become a significant issue in both the computer vision and natural language processing research groups. This study examines contemporary multimodal deep learning research from three perspectives: learning multimodal representations, integrating multimodal data at multiple levels, and multimodal applications. We discuss the essential ideas of embedding in multimodal representation learning, which integrate multimodal signals into a single vector space and so enable cross-modality signal processing. We also look at the features of various embeddings that are built and learnt for typical downstream tasks. This paper focuses on unique designs for the integration of representations of unimodal inputs for a specific purpose in multimodal fusion. In terms of applications, the present literature covers a few areas of general interest, such as image-to-text caption production, text-to-image generation, and visual question answering.

b) T. Baltrušaitis, C. Ahuja and L. -P. Morency, "Multimodal Machine Learning: A Survey and Taxonomy," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 2, pp. 423-443, 1 Feb. 2019, doi: 10.1109/TPAMI.2018.2798607.

We have a multimodal perception of the environment in which

we see things, hear noises, feel texture, smell odours, and taste flavours. A study topic is described as multimodal when it involves numerous such modalities. Modality refers to the way something happens or is experienced. To make progress in comprehending the world around us, Artificial Intelligence must be able to process such multimodal inputs simultaneously. The goal of multimodal machine learning is to create models that can interpret and connect data from various modalities. It is a thriving multidisciplinary field that is growing in prominence and has tremendous potential. Rather than focusing on specific multimodal applications, this study analyses recent achievements in multimodal machine learning and organises them into a taxonomy. We go beyond the conventional early and late fusion classification to highlight larger multimodal machine learning issues, such as representation, translation, alignment, fusion, and co-learning. This new taxonomy will help scholars better grasp the state of the discipline and suggest future research areas.

c)Aishwarya Jayagopal, "Multimodal Representation Learning With Text and Images "arXiv preprint arXiv:2205.00142v1

This project leverages multimodal AI and matrix factorization techniques for representation learning, on text and image data simultaneously, thereby employing the widely used techniques of Natural Language Processing (NLP) and Computer Vision. The learnt representations are evaluated using downstream classification and regression tasks. The methodology adopted can be extended beyond the scope of this project as it uses Auto-Encoders for unsupervised representation learning.

CHAPTER 2
SYSTEM ANALYSIS

2.1.Existing System

Multimodal representation learning is a technique of learning, that is used to embed information from different modalities and understand the similarity between them. There are not much existing efficient systems as of in the field of audio - text processing. There are multiple modals for audio – video – text transformers like ViLBERT and LXMERT that combines visual, text and audio inputs for better understanding. These transformers are being developed by higher organisations. The developing audio text model would help in analysing the audio-lingual data and providing for the similarity between them.

2.2.Proposed System

Here we have proposed a base algorithm for the audio text processing which could later be developed into transformers. As of now, we have created about 4 modules which could analyse the relationship between the audio and the text data. The four basic approaches can be given as

1. Audio Text Coherence.
2. Audio Audio Coherence.
3. Summarizing text data to audio.
4. Summarizing audio to text.

1.Audio Text Coherence

The audio text coherence can be defined as the level of semantic and contextual agreement between audio material and its matching literary representation. This model analyses the relationship between the audio and the text and ciphers the similarity between them. Here we use the speech recognition package and convert the audio files

to text and by using the PyTorch and BERT transformers for natural language processing(NLP) and then we deal with the similarity part. This proves to be an efficient to analyse two different forms of data.

2.Audio Audio Coherence

The audio audio coherence can be given as the analysing level between two audio files and matching literary representation. When given two audio files, both are converted into text files and the natural language processing is done to find the similarity in them. The coherence is generally a concept of examining the relation between data signals or textual data.

3.Summarizing text data to audio

The textual data when given in a larger level is summarized using abstractive method in NLP and the summarized text is represented as an audio output which proves to be useful in terms of understanding and reporting the summarized form.

4.Summarizing audio to text

In this approach , the larger audio file is extracted into a text data using a speech recognition package in python and the textual file is then summarized into another short text by using the abstractive method. We even have a PyTorch approach in which we can convert the live audio file into text.

2.2.1.Advantages of the proposed system

There are various advantages of these modules in audio-text multimodal representation learning. They are

- Upgraded performance: Enhances performance on a variety of multimodal tasks, including audio captioning, audio-to-text alignment, audio-based question answering, and multimodal retrieval. Multimodal models can do these tasks more effectively

than individual modalities alone by concurrently learning representations from audio and text.

- Improved comprehension: Multimodal algorithms can extract complimentary information from both the audio and the text modalities. While writing supplies semantic and contextual information, audio provides a wealth of acoustic clues including intonation, tone, and background noise. The material is better understood overall when the two modalities are combined, which results in representations that are more thorough and accurate.

- Resistance to modality-specific limitations: Text and audio each have their own drawbacks. Audio, for instance, could include background noise, accents, or transcription problems, whereas text might be unclear or lacking in specifics. Multimodal models can overcome these constraints and increase resilience in comprehension and interpretation by using both modalities.

CHAPTER 3
SYSTEM REQUIREMENTS SPECIFICATION

3.1.Software Requirements

We would require Flask for creating an interface and the program is efficiently using VSCode. Microsoft created VSCode, a free and open-source source code editor. It is frequently used by developers and is compatible with a broad number of programming languages and systems. VSCode is well-known for its lightweight and quick speed, broad customization possibilities, and a large ecosystem of extensions that extend its capability.

3.1.Visual Studio Code

3.2.Hardware Requirements

There is no specific hardware requirements involved in this process of developing an algorithm. However we would required about 4 GB RAM for better GPU.

CHAPTER 4
SYSTEM DESCRIPTION

4.1.Frontend Description

a)Flask

Flask is a Python web framework that allows you to rapidly and simply construct online applications. It is basic, minimalistic, and straightforward to grasp, making it a popular choice for constructing small to medium-sized web apps and APIs. Flask includes the tools and functionality required to handle HTTP requests, routing, and URL mapping. It also has an integrated development server and a templating engine for producing HTML templates. Some of the key features include templating, flask extensions, routing, request handling and being lightweighted and flexible.

4.1.Flask

4.2.Backend Description

a)PyAudio

The PyAudio library is a Python package that provides a cross-platform audio input and output interface. It enables developers to quickly access audio devices and execute audio-related operations such as microphone recording, playing audio files, and processing real-time audio streams. PyAudio is built on top of the PortAudio library, which provides a low-level audio interface that works with a variety of operating systems, including Windows, macOS, and Linux.

b)Speech Recognition

The Python speech recognition module is a library that allows you to execute automated speech recognition (ASR) operations. It enables developers to use multiple voice recognition engines and APIs to convert spoken language into written text. The main Python library for voice recognition is named "SpeechRecognition." It functions as a wrapper for many voice recognition engines, allowing you to simply switch between them depending on your needs.

c)Wave

In Python, the "wave" package is a built-in module that allows you to read and write WAV (Waveform Audio File Format) files. WAV is a widely used audio file format for storing uncompressed audio data.

d)Torch

The Python package "torch" alludes to PyTorch, a famous open-source machine learning framework created by Facebook's AI Research group. PyTorch has a comprehensive set of tools and features for developing and training deep neural networks.It provides fast tensor operations, automated differentiation for gradient calculations, and a dynamic computational network, making it a popular option among artificial intelligence and machine learning researchers and practitioners.

e)Transformers

In Python, the "transformers" package refers to the Hugging Face Transformers library. It is a well-known open-source library with cutting-edge natural language processing (NLP) capabilities, notably in the field of transformer-based models.

f) NLTK(NATURAL LANGUAGE TOOLKIT)

Natural language processing (NLP) is a field that focuses on

making natural human language usable by computer programs. NLTK, or Natural Language Toolkit, is a Python package that you can use for NLP. NLTK (Natural Language Toolkit) Library is a suite that contains libraries and programs for statistical language processing. It is one of the most powerful NLP libraries, which contains packages to make machines understand human language and reply to it with an appropriate response.

g) Spacy

spaCy, a free and open-source library with a lot of built-in capabilities. It's becoming increasingly popular for processing and analyzing data in the field of NLP.

spaCy is designed specifically for production use and helps you build applications that process and "understand" large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning.

h) re in Python

A Regular Expressions (RegEx) is a special sequence of characters that uses a search pattern to find a string or set of strings. It can detect the presence or absence of a text by matching it with a particular pattern, and also can split a pattern into one or more sub-patterns. Python provides a re module that supports the use of regex in Python. Its primary function is to offer a search, where it takes a regular expression and a string. Here, it either returns the first match or else none.

i)pyttsx3 in Python

Pyttsx3 is a Python library that provides cross-platform support for text-to-speech (TTS) synthesis. pyttsx3 allows us to convert written text into spoken words by utilizing the text-to-speech engines available on our operating system.

j)os

The os library in Python is a built-in module that allows you to communicate with the operating system. It provides functions and methods for performing numerous operating system-related activities like as file and directory operations, process management, environment variables, and more.

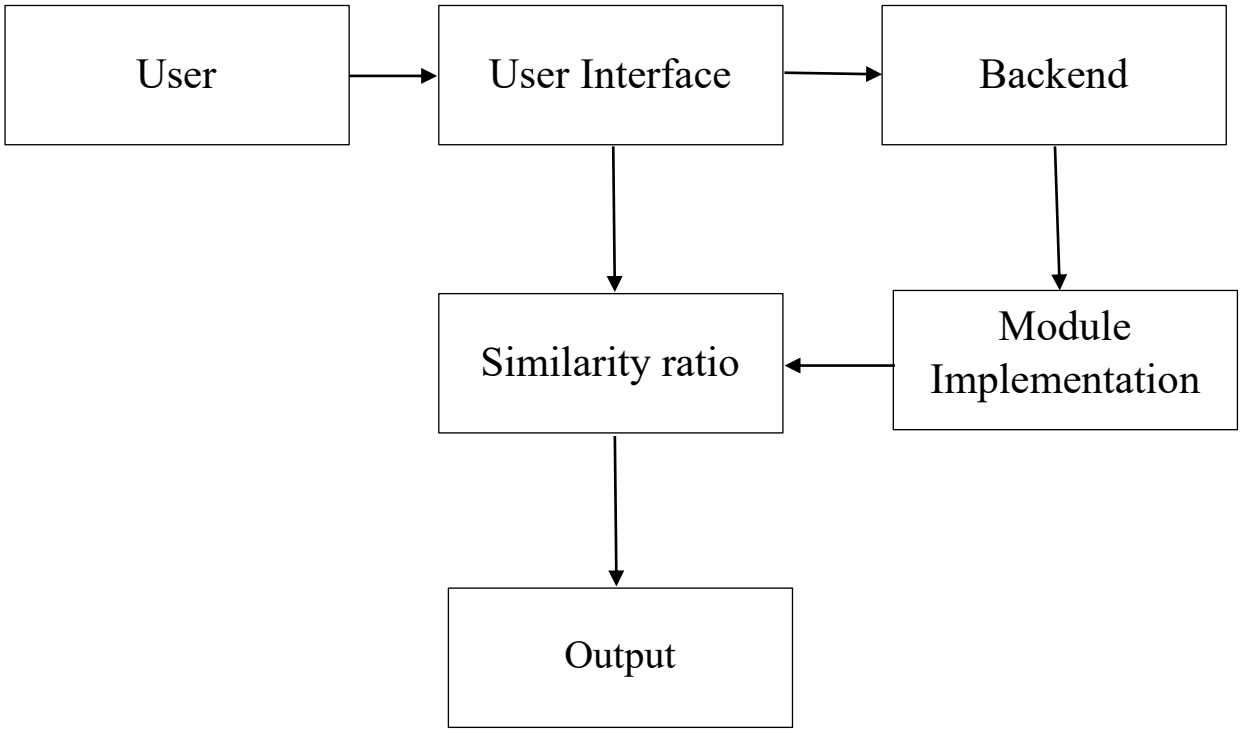
k)sklearn

Sklearn is an acronym for scikit-learn, a popular Python machine learning package. It offers a complete range of tools and functions for diverse machine learning tasks such as classification, regression, clustering, dimensionality reduction, model selection, and preprocessing.

CHAPTER 5
SYSTEM DESIGN

5.1.System Architecture

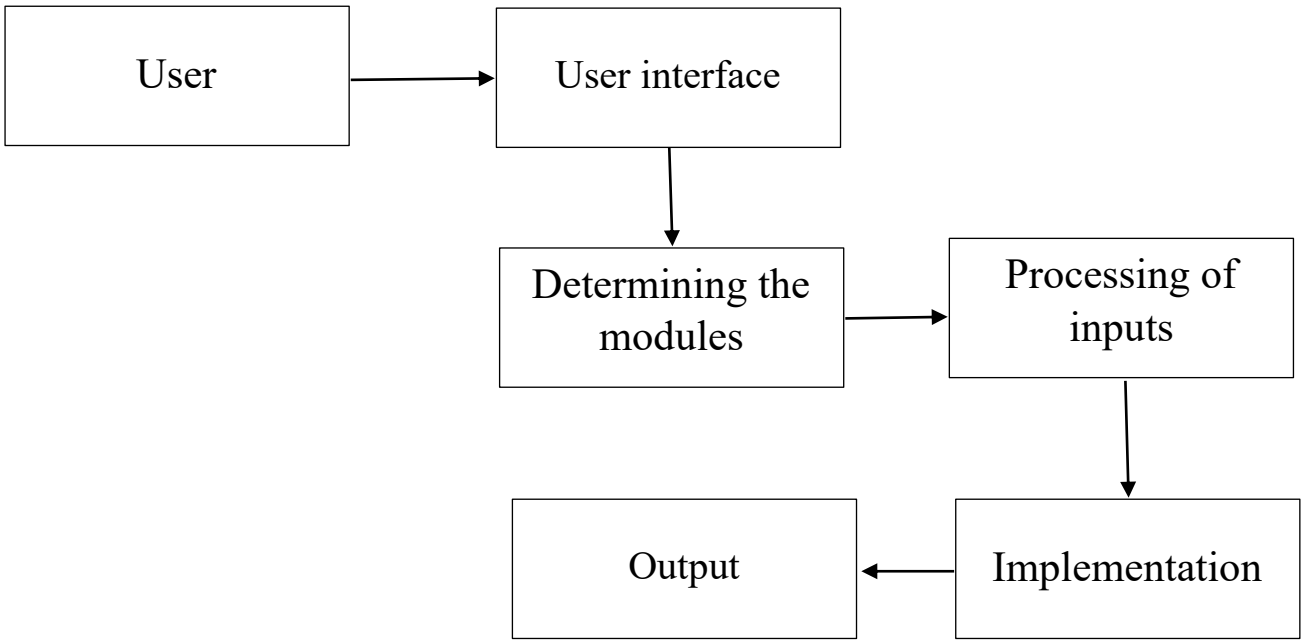
System architecture refers to the conceptual structure and design of a software or hardware system. It defines how the various components of a system are organized, how they interact with each other, and how they collectively fulfill the system's requirements and objectives.



5.1.System Implementation

5.2.Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of how data flows within a system. It illustrates the movement of data between various processes, data stores, and external entities. DFDs are commonly used in system analysis and design to depict the logical structure of a system and understand the flow of information.

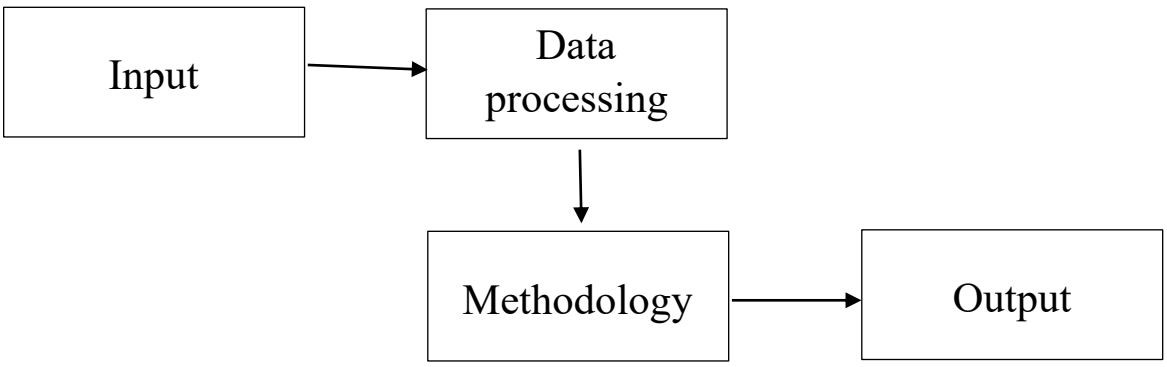


5.2.Data Flow Diagram

5.3.UML Diagram

UML (Unified Modeling Language) is a standardized visual modeling language used in software engineering for designing, documenting, and communicating software systems. UML diagrams provide a graphical representation of different aspects of a system, including its structure, behavior, and interactions.

The UML Diagram differs for each module and thus the general structure can be just given as a static view.



5.3.UML Diagram

CHAPTER 6
SYSTEM IMPLEMENTATION

a)Input File:

Now when the interactive webpage is created , the user can provide the input file which can be an audio or text and these inputs can be converted into required forms and the similarity can be provided. Thus the input is obtained from the user with the user interface developed by using Flask.

b)Processing the input:

From the input , if the input is an audio file, then the audio is converted into text using PyTorch and BERT transformers and the two files are processed in order to find the similarity between them.

c)Audio text Coherence

Here we use the speech recognition package and convert the audio files to text and by using the PyTorch and BERT transformers for natural language processing(NLP) and then we deal with the similarity part. This proves to be an efficient to analyse two different forms of data.

d)Audio Audio Coherence

When given two audio files, both are converted into text files and the natural language processing is done to find the similarity in them. The coherence is generally a concept of examining the relation between data signals or textual data.

e)Summarizing text data to audio

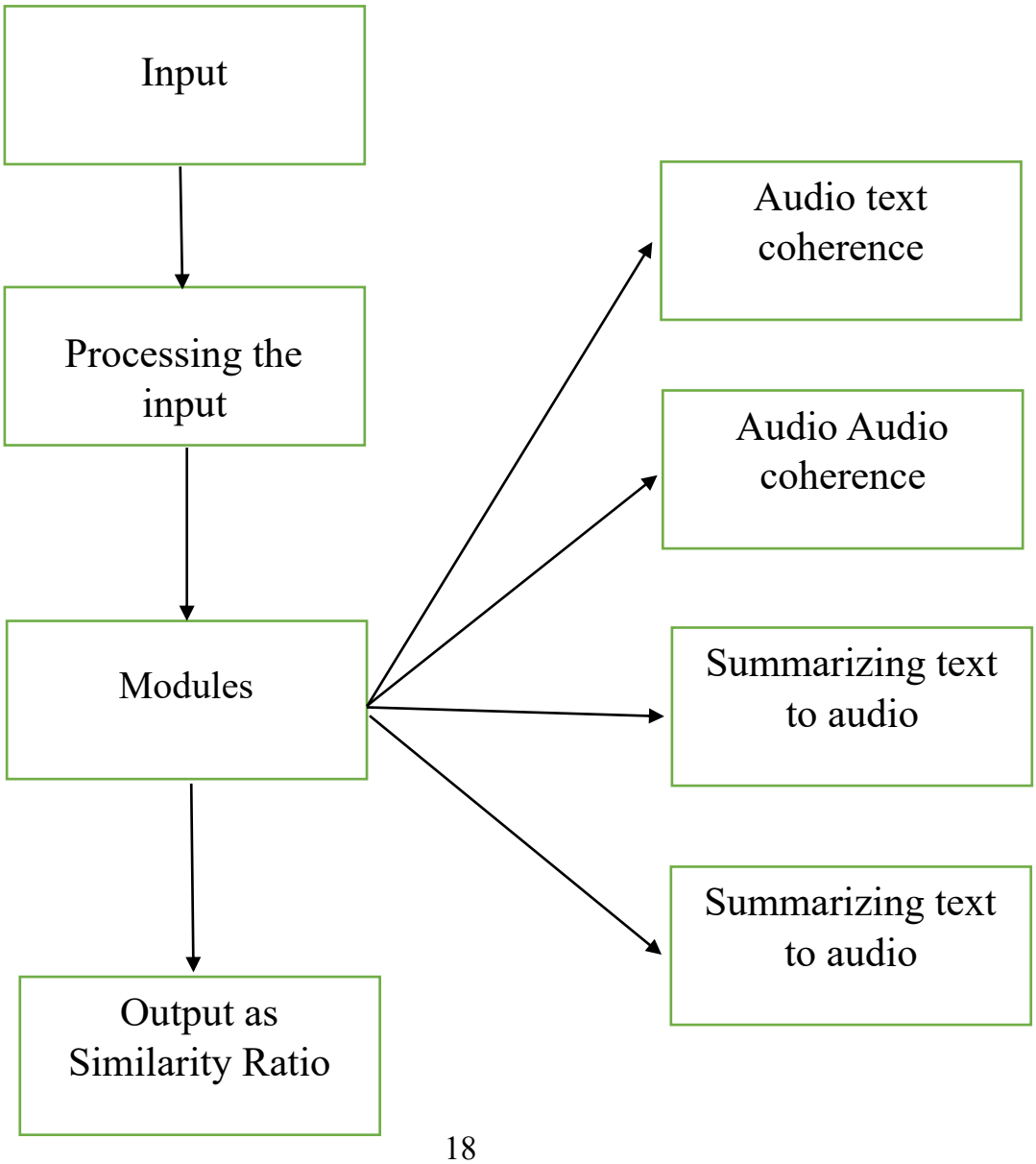
The textual data when given in a larger level is summarized using abstractive method in NLP and the summarized text is represented as an audio output which proves to be useful in terms of understanding and reporting the summarized form.

f)Summarizing audio to text

In this approach , the larger audio file is extracted into a text data using a speech recognition package in python and the textual file is then summarized into another short text by using the abstractive method. We even have a PyTorch approach in which we can convert the live audio file into text.

g)Output

The output is displayed as the similarity ratio between the audio and the text file and this helps us to know the relationship between two different modes of data.



CHAPTER 7
TESTING AND VALIDATION

Thus the testing is done with the user interface which makes it easier to analyze the program. When the main file is run , we get a UI with options to perform the task such as Audio Text Coherence, Audio audio coherence, Summarizing audio to text and summarizing text to audio. Thus the user can choose the way he wants to proceed. And the user can provide the input file. We can even live record the audio file and trasnsform it into text with seconds. Thus with each method taken, We get the exact similarity ratio we required. Thus the model is tested and validated with full efficiency produced. We get an accurate model with being 100% proper and this can be further developed into neural networks as well as trasnformers.

CHAPTER 8
EXPERIMENTAL ANALYSIS AND RESULTS

Now the created user interface using flask seems to be successful. And the multimodal learning is almost 98% correct through its efficiency. The performance can be even increased by working with other algorithms or a combination of them. When this is improved, the performance of the system can be improved as now it can increase the chances of high usage. We have obtained the results as planned and the system can help in various peoples. For the people who are unable to see, this model can be developed as an app that summarizes the text in audio. There are multiple uses for this model development. Thus the results are perfect with the way and the model is analysed with requirements of combination of algorithms that speed up the model. These models can be compressed and converted in the form of transformers as well as by approaching neural networks.

CHAPTER 9
CONCLUSION AND FUTURE WORKS

In conclusion, audio-text multimodal representation learning is an area of research and study that focuses on collecting and comprehending the connections and interactions between audio and text modalities. This technique intends to improve the performance of different machine learning tasks such as voice recognition, audio captioning, audio classification, and sentiment analysis by merging audio and text data.

Models can learn shared representations that capture complementing information from both audio and text inputs using multimodal representation learning. This allows for a more in-depth knowledge of the data and can lead to higher performance in jobs requiring multimodal analysis.The benefits of audio-text multimodal representation learning include improved performance in tasks involving both audio and text data, improved contextual understanding, better cross-modal generalisation, and the ability to leverage the strengths of each modality to overcome limitations and ambiguities.

Overall, audio-text multimodal representation learning has the potential to enable more robust and accurate models in areas such as speech recognition, audio analysis, and natural language understanding, opening up new opportunities in speech processing, multimedia analysis, and human-computer interaction.

The availability of large-scale multimodal datasets and advances in deep learning have fueled advancements in audio-text multimodal representation learning. To extract meaningful representations and capture the relationships between audio and text, researchers and practitioners have created a variety of models and methodologies, including multimodal transformers.

BIBLIOGRAPHY

1. Zhang, C., Yang, Z., He, X., & Deng, L. (2020). Multimodal intelligence: Representation learning, information fusion, and applications. IEEE Journal of Selected Topics in Signal Processing, 14(3), 478-493.

2. T. Baltrušaitis, C. Ahuja and L. -P. Morency, "Multimodal Machine Learning: A Survey and Taxonomy," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 2, pp. 423-443, 1 Feb. 2019, doi: 10.1109/TPAMI.2018.2798607.

3.Aishwarya Jayagopal, "Multimodal Representation Learning With Text and Images “arXiv preprint arXiv:2205.00142v1

APPENDIX

Source Code:

root.py

```
from flask import Flask, render_template, request, session, redirect, url_for
from trial import *
from trial2 import *
from trial3 import *
from trial4 import *
import os
app = Flask(__name__)


@app.route("/")
def home():
    return render_template("home.html")

@app.route("/module_1", methods = ["GET", "POST"])
def module_1():
    if request.method == "GET":
        return render_template("module_1.html")
    elif request.method == "POST":
        opp = open("recorded_content.txt", 'r')
        inn = opp.read()
        opp.close()
        sentences = [request.form['input'], inn]
        res = find(sentences)
        res = res[0][0] * 100
        return render_template("results.html", result = res)

@app.route("/module_2", methods = ["GET"])
def module_2():
    if request.method == "GET":
        res1 = convert_audio_to_text("output.wav")
        res2 = convert_audio_to_text("output1.wav")
        res = find([res1, res2])
        res = res[0][0] * 100
        return render_template("results.html", result = res)

@app.route("/module_3", methods = ["GET", "POST"])
def module_3():
    if request.method == "GET":
        return render_template("module_3.html")
    elif request.method == "POST":
        inp = request.form["text_area"]
        summarize(inp)
        return render_template("home.html")
```

```
@app.route("/module_4", methods = ["GET"])
def module_4():
    if request.method == "GET":
        res = pull()
        return render_template("results.html", result = res)

if __name__ == "__main__":
    app.run(debug=True)
```

trial1.py

```
import pyaudio
import wave
import speech_recognition as sr
import datetime

def voice_to_audio():
    FRAMES_PER_BUFFER = 3200
    FORMAT =pyaudio.paInt16
    CHANNELS = 1
    RATE = 16000

    p = pyaudio.PyAudio()

    stream = p.open(
        format = FORMAT,
        channels = CHANNELS,
        rate = RATE,
        input = True,
        frames_per_buffer = FRAMES_PER_BUFFER
    )

    print("Start Recording")
    seconds = 5
    frames = []
    for i in range(0, int(RATE/FRAMES_PER_BUFFER*seconds)):
        data = stream.read(FRAMES_PER_BUFFER)
        frames.append(data)

    stream.stop_stream()
    stream.close()
    p.terminate()

    time = str(datetime.datetime.now())
    obj = wave.open("output.wav", "wb")
    obj.setnchannels(CHANNELS)
    obj.setsampwidth(p.get_sample_size(FORMAT))
    obj.setframerate(RATE)
```

```
    obj.writeframes(b"".join(frames))
    obj.close()
    return convert_audio_to_text("output.wav")

def convert_audio_to_text(audio_file):
    recognizer = sr.Recognizer()
    with sr.AudioFile(audio_file) as source:
        audio = recognizer.record(source)  # Read the entire audio file
    try:
        text = recognizer.recognize_google(audio)
        return text
    except sr.UnknownValueError:
        print("Speech recognition could not understand audio")
    except sr.RequestError as e:
        print(f"Could not request results from Google Speech Recognition
service; {e}")
```

trial2.py

```
from sklearn.metrics.pairwise import cosine_similarity
from transformers import AutoTokenizer, AutoModel
import torch
def find(sentences):
    tokenizer = AutoTokenizer.from_pretrained("D:\SRP\model")
    model = AutoModel.from_pretrained("D:\SRP\model")

    tokens = {'input_ids':[], 'attention_mask':[]}

    for sentence in sentences:
        new_tokens = tokenizer.encode_plus(sentence, max_length = 128,
                                           truncation = True,   padding =
'max_length',
                                           return_tensors = 'pt')
        tokens['input_ids'].append(new_tokens['input_ids'][0])
        tokens['attention_mask'].append(new_tokens['attention_mask'][0])

    tokens['input_ids'] = torch.stack(tokens['input_ids'])
    tokens['attention_mask'] = torch.stack(tokens['attention_mask'])

    outputs = model(**tokens)

    embeddings = outputs.last_hidden_state

    attention = tokens["attention_mask"]
    mask = attention.unsqueeze(-1).expand(embeddings.shape).float()

    mask_embeddings = embeddings * mask
```

```
    summed = torch.sum(mask_embeddings, 1)

    counts = torch.clamp(mask.sum(1), min=1e-9)

    mean_pooled = summed/counts
    mean_pooled = mean_pooled.detach().numpy()
    return cosine_similarity([mean_pooled[0]], mean_pooled[1:])
```

trial3.py

```
import pyttsx3
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from spacy.lang.en.stop_words import STOP_WORDS
import re


def nltk_summarizer(docx):
    stopWords = set(stopwords.words("english"))
    words = word_tokenize(docx)
    freqTable = dict()

    for word in words:
        word = word.lower()
        if word not in stopWords:
            if word in freqTable:
                freqTable[word] += 1
            else:
                freqTable[word] = 1

    sentence_list= sent_tokenize(docx)
    #sentenceValue = dict()
    max_freq = max(freqTable.values())
    for word in freqTable.keys():
        freqTable[word] = (freqTable[word]/max_freq)

    sentence_scores = {}
    for sent in sentence_list:
        for word in nltk.word_tokenize(sent.lower()):
            if word in freqTable.keys():
                if len(sent.split(' ')) < 30:
                    if sent not in sentence_scores.keys():
                        sentence_scores[sent] = freqTable[word]
                    else:
                        sentence_scores[sent] += freqTable[word]#total number
of length of words

    import heapq
```

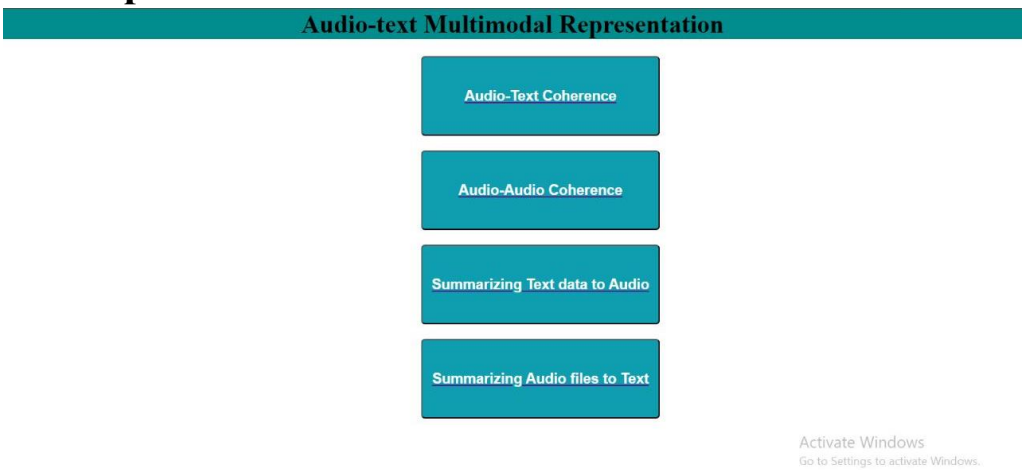
```
    summary_sentences = heapq.nlargest(8, sentence_scores,
key=sentence_scores.get)
    summary = ' '.join(summary_sentences)
    return summary

def summarize(article_text):
    article_text = re.sub(r'\\[[0-9]*\\]', ' ',article_text)
    article_text = re.sub('[^a-zA-Z.,]', ' ',article_text)
    article_text = re.sub(r"\b[a-zA-Z]\b",'',article_text)
    article_text = re.sub("[A-Z]\Z",'',article_text)
    article_text = re.sub(r'\s+', ' ', article_text)

    summary_result = nltk_summarizer(article_text)

    print(summary_result)

    text_speech = pyttsx3.init()

    text_speech.say(summary_result)
    text_speech.runAndWait()
```

trial4.py

```
import os
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from spacy.lang.en.stop_words import STOP_WORDS
import re
import speech_recognition as sr
from trial3 import *


def convert_audio_to_text(audio_file):
    recognizer = sr.Recognizer()
    with sr.AudioFile(audio_file) as source:
        audio = recognizer.record(source)  # Read the entire audio file
    try:
        text = recognizer.recognize_google(audio)
        return text
    except sr.UnknownValueError:
        print("Speech recognition could not understand audio")
    except sr.RequestError as e:
        print(f"Could not request results from Google Speech Recognition
service; {e}")
```

```
def pull():

    # Specify the path to the folder containing audio files
    folder_path = "test_data"

    article_text = []

    # Iterate over the files in the folder
    for filename in os.listdir(folder_path):
        if filename.endswith(".flac") or filename.endswith(".mp3") or
filename.endswith(".wav"):
        # Access the audio file one by one
            audio_file_path = os.path.join(folder_path, filename)
            article_text.append(convert_audio_to_text(audio_file_path))

    article = ". ".join(article_text)

    return nltk_summarizer(article)
```

Sample Output:



Result

58.527934551239014

Home