



Dissertation on

“Artificial Radio Agent using Stochastic Models”

Submitted in partial fulfillment of the requirements for the award of degree of

**Bachelor of Technology
in
Computer Science & Engineering**

UE21CS461A – Capstone Project Phase - 2

Submitted by:

Chandrachud Sarath	PES1UG21CS150
Charitha K Reddy	PES1UG21CS151
Darsh Agarwal	PES1UG21CS166
Suvan Ashwin	PES1UG21CS651
Hiremath	

Under the guidance of

Dr. Ramamoorthy Srinath
Professor

August – Dec, 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100 Feet Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

‘Artificial Radio Agent using Stochastic Models’

is a bonafide work carried out by

Chandrachud Sarath

PES1UG21CS150

Charitha K Reddy

PES1UG21CS151

Darsh Agarwal

PES1UG21CS166

Suvan Ashwin Hiremath

PES1UG21CS651

in partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE21CS461A) in the Program of Study - **Bachelor of Technology in Computer Science and Engineering** under rules and regulations of PES University, Bengaluru during the period Aug 2024 – Dec 2024. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature
Dr. Ramamoorthy Srinath
Professor

Signature
Dr. Mamatha H R
Chairperson

Signature
Dr. B K Keshavan
Dean of Faculty

External Viva

Name of the Examiners

Signature with Date

1. _____

2. _____

Declaration

We hereby declare that the Capstone Project Phase - 2 entitled “Artificial Radio Agent using Stochastic Models” has been carried out by us under the guidance of Dr. Ramamoorthy Srinath, Professor, submitted in partial fulfillment of the completion of the seventh semester of Bachelor of Technology in Computer Science and Engineering of PES University, Bengaluru during the academic semester August - December, 2024. The matter embodied in this report has not been submitted to any other University or Institution for the award of any degree.

Chandrachud Sarath	PES1UG21CS150
Charitha K Reddy	PES1UG21CS151
Darsh Agarwal	PES1UG21CS166
Suvan Ashwin	PES1UG21CS651
Hiremath	

Acknowledgment

We would like to express our gratitude to Dr. Ramamoorthy Srinath, Professor, Department of Computer Science and Engineering, PES University, for his continuous guidance, assistance, and encouragement throughout the development of this UE19CS390A – Capstone Project Phase – 1.

We are grateful to the project coordinator, Dr. Priyanka H, all the panel members & the supporting staff for organizing, managing, and helping the entire process.

We take this opportunity to thank Dr. Mamatha H R, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support we have received from her.

We are grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro-Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, Dr. B.K. Keshavan, Dean of Faculty, PES University for providing us with various opportunities and enlightenment during every step of the way.

Finally, this project could not have been completed without the continual support and encouragement we have received from our family and friends.

Abstract

Artificial intelligence (AI) technologies are becoming widely used as a result of their unprecedented growth in a variety of industries, including business, entertainment, and medicine. This phenomenon also applies to the radio broadcasting industry, where AI applications are becoming more and more popular among professionals in the field, indicating a trend toward their integration into different aspects of radio production and operations. Conversational agents, such as chatbots, are useful in a variety of settings that require audience interaction, such as FAQs, auto tellers, and so on. However, developing communication agents that can mimic human behavior is a formidable challenge. In alignment with this sentiment, we endeavor to develop an Online AI radio jockey for infotainment purposes. Our objective is to develop a basic communication agent imbued with a compelling personality to stimulate interactions with the audience. The system would be capable of emulating a Radio Jockey including the ability to engage in general commentary. This initiative is just one of many potential developments in this direction.

TABLE OF CONTENTS

Declaration	2
Acknowledgement	3
Abstract	4
List Of Figures	6
Introduction	7
Problem Statement	9
Literature Review	10
Playlist Recommendation Based on Reinforcement Learning [1]	10
A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization [2]	10
Prompted LLMs as Chatbot Modules for Long Open-domain Conversation [3]	11
Hidden Markov model approach to skill learning and its application to telerobotics [4]	11
Markov chain Hebbian learning algorithm with ternary synaptic units [5]	12
JayBot – Aiding University Students and Admission with an LLM-based Chatbot [6]	12
AI IN RADIO: THE GAME CHANGER YOU DID NOT HEAR COMING [7]	13
System Requirements Specifications	14
1. Overall Description	14
1.1. Product Functions	14
1.2. User Classes and Characteristics	15
1.3. Design and Implementation Constraints	16
1.4. Assumptions and Dependencies	16
2. Specific Requirements	17
2.1. Functional Requirements	17
2.2. Non-Functional Requirements	18
2.2.1. Performance Requirements	18
2.2.2. Usability Requirements	18
3. External Interface Requirements	19
3.1. Software Interfaces	19
4. System Features	19
4.1. Music Recommendation	19
4.2. Weather and News Updates	20
4.3. Radio script building	20
4.4. Audience Interaction	21
4.5. Radio Commentary (Voice Integration)	22
System Design	23
5. Introduction	23
5.1. Scope	23
5.2. Objectives	23
6. Architectural Overview	23
6.1. System Components	23

6.2. Interactions	25
6.3. Diagrams	26
7. System Components	29
7.1. Streaming software	29
7.2. Content Management System	29
7.3. LM Chatbot	29
7.4. Music Recommendation System	29
8. Interface Design	30
8.1. News interface	30
8.2. Weather Interface	30
9. Methodology, Implementation, and pseudocode	31
9.1. Overview	31
9.2. Initial analysis	32
9.3. Set-up and implementation of Modules	34
9.3.1. Simplified Music Recommendation Module	34
9.3.2. Talk Module.	35
9.3.3. News Summarizer Module	37
9.3.4. Weather Summarizer Module	38
9.3.5. Markov Decision Process (MDP)	39
9.3.6. Integration Framework	40
10. Results and Discussions	41
10.1. Results	41
10.1.1. Enhanced Content Generation	41
10.2. Discussion	42
11. Conclusion and Future Work	44
11.1. Future Work	44
References	46

List Of Figures

Figure No.	Title	Page No.
Figure 1	Block diagram of the system	24
Figure 2	Data Flow Diagram	25
Figure 3	Master Class Diagram	26
Figure 4	Interface Diagram	28

Introduction

Advancements in digital technology have brought an age of unprecedented accessibility and variety in information; especially entertainment.

Streaming services such as Netflix, Hulu, and Amazon Prime Video provide an ever-expanding catalogue of movies, TV series, and documentaries that consumers may watch at their convenience, eliminating the need for regular scheduled television.

Similarly, music streaming services such as Spotify, Apple Music, and YouTube Music offer millions of songs from diverse artists and genres, as well as personalized playlists and curated radio stations.

In addition to the above, social media platforms such as Facebook, Instagram, YouTube, Twitter, and TikTok serve as hubs for content creation, sharing, and discovery, allowing users to interact with friends and influencers while consuming a wide range of multimedia material personalized to their interests and preferences.

All this is extensively further enabled by the development of digital gadgets such as smartphones, tablets, and smart TVs, which have made entertainment media immensely more accessible.

We believe that it is only logical to conclude that the digital revolution has definitely democratised access to entertainment and information, allowing people to discover and experience a whole new world of content catered to their specific likes and inclinations.

We are now in a new era of personalised entertainment experiences, with a different manner of interacting with multiple mediums.

Despite the proliferation of digital media outlets, a certain legacy medium is still ever popular.

Radio broadcasting remains a reliable source of news, music, and community participation. Radio's capacity to cross geographical barriers and establish real-time connections with listeners makes it an essential medium for communication and cultural exchange. Radio stations, whether through news bulletins, music playlists, talk shows, or live event coverage, play an important role in moulding public conversation and cultivating a feeling of community among listeners. It is important to emphasise the importance of radio jockeys, who are responsible for shaping these experiences.

That is not to imply that it is without obstacles, since conventional radio finds it hard to adapt to the changing tastes and habits of modern consumers. Radio stations must adapt and innovate in order to

remain relevant as listeners' preferences for personalized experiences and on-demand material grow more and more pronounced on digital platforms. Audience fragmentation, competition from digital platforms, and shifting consumption patterns pose significant challenges to traditional radio's sustainability and growth.

In response to these issues, the incorporation of artificial intelligence (AI) into radio broadcasting emerges as a game-changing solution with the potential to alter the medium.

In an increasingly competitive media market, radio stations can establish deeper relationships and loyalty by employing AI-driven algorithms and data analytics to offer personalized content experiences that resonate with individual listeners.

However the primary concern that emerges in this situation is the rigid and pattern oriented nature of generative AI.

The apparent lack of organicity in the creative field of entertainment, is a massive obstacle to overcome in order to achieve a suitable rendition for an agent to fulfill this task.

Problem Statement

Creating an “Artificially intelligent Radio Jockeying system” (henceforth referred to as AI RJ) is a watershed event in the advancement of entertainment technology. However, realising its full potential requires overcoming a number of technological, ethical, and user-centric problems.

Ensuring the ethical and responsible operation of AI RJ is a major consideration in its development. As it interacts with people in real-time, the AI-powered entity AI RJ must abide by strict ethical guidelines to prevent the proliferation of prejudicial, discriminatory, offensive material. There is the utmost imperative to ensure the balance between acting ethically and creating engaging content for both developers and stakeholders.

AI RJ needs to interface with a variety of external sources seamlessly in order to deliver a rich, interesting and engaging user experience. These could include news feeds, traffic data providers, music streaming services, and commentary databases, among others. Smooth and efficient integration demands the creation of robust APIs, data processing pipelines, and interoperability with diverse data formats and protocols. Moreover, new technical challenges arise in assuring the timeliness and reliability of data from these sources.

In the era of personalized content distribution, AI RJ must also consider the diverse interests and preferences of every user. To ensure this, complex algorithms for preference analysis, user profiling, and real-time content customization have to be developed. AI RJ can provide personalized content recommendations, improve user engagement, and create a stronger bond with listeners by analyzing user behavior and preferences. Finding the ideal balance between privacy protection and customisation is still a major obstacle, though.

Through this problem statement, we hope to make progress that can help the overall paradigm of AI-integration in our daily lives, in superset of this particular application.

Literature Review

Playlist Recommendation Based on Reinforcement Learning [1]

Binbin Hu, Chuan Shi, and Jian Liu's study "Playlist Recommendation Based on Reinforcement Learning" takes a novel approach to music playlist suggestion by modeling it as a Markov Decision Process (MDP) and applying reinforcement learning techniques. The research has suggested modeling the recommendation task as an MDP and using reinforcement learning approaches to overcome the difficulties in music recommendation. This method takes into account user opinions and song influence in order to take advantage of the best playlist creation strategy.

This paper presents RLWRec, a reinforcement learning-based playlist recommendation mechanism. Using user preferences and interaction history, RLWRec estimates the best music recommendation strategy using Q-learning combined with an ϵ -greedy approach. To reduce state space dimensionality and boost training efficiency, the RLWRec model employs collaborative filtering-based state compression. It uses ϵ -greedy strategy to balance exploration and exploitation and Q-learning to estimate Q-values for actions in each state. RLWRec chooses music clusters and makes song recommendations to users using a recommendation approach based on a tree structure. This method enhances suggestion coverage and accuracy by combining user preference scores with Q-value probability.

Experiments conducted on real music datasets demonstrate the effectiveness of the RLWRec model in generating personalized playlists. The paper also analyzes the influence of user listening frequency and window size on the model's performance.

A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization [2]

A survey of scientific literature on text summarization specifically abstractive summarization based on neural networks. It goes through a conceptual framework when reviewing literature particularly the encoder decoder architecture employed , mechanisms used , datasets and evaluation metrics.

The authors have also presented a concept matrix to help identify common trends in the design of neural abstractive summarization systems.

Prompted LLMs as Chatbot Modules for Long Open-domain Conversation [3]

The paper discusses the importance of modular prompting in enhancing reasoning abilities and improving performance across various benchmarks.

This paper gives a novel approach to building chatbots with a working memory using existing pre-trained LLMs without much fine-tuning using techniques like “In Context Learning” and “Chain of Thoughts”.

The paper introduces Modular Prompted chatbots, made up of 4 components each having its own LLM. It consists of a Utterance Classifier, Memory processor, Utterance generator and dialogue summarizer. Each module has a specific role and together they can help build a working memory for the chatbot and capability of learning new tasks based only on text prompting.

However splitting the system into multiple modules introduces overhead and leads to accumulation of errors. Best Language modules for each module needs further testing , and the maximum memory that can be set up will depend on the maximum token limit of the underlying Language Model.

Hidden Markov model approach to skill learning and its application to telerobotics [4]

The research study presents a sophisticated technique to skill acquisition using Hidden Markov Models (HMM) and applies it to telerobotics. It suggests the development of a multidimensional HMM framework for skill acquisition challenges by modeling both observable actions and unobservable mental states. Using the concept of "most likely performance," the system determines optimal action sequences based on previously recorded data, allowing for continual real-time updates and tweaks to improve learning results. HMM's adaptability in supporting skill acquisition

and improvement in dynamic situations is highlighted, with a focus on its applicability in telerobotics applications where adaptive learning processes are critical to operational success.

Markov chain Hebbian learning algorithm with ternary synaptic units [5]

The research paper talks about the “Markov chain Hebbian learning” method, which is intended to facilitate real-time learning by using “ternary synaptic units” in complement of a Markov chain to memory training efficiency.

The ternary synaptic refers to a 3 state approach of [-1,0,1] instead of the binary approach of [0,1]. The method's utility in real-world applications was confirmed by evaluating it on tasks like multiplication table memorization and handwritten digit identification. By utilizing the humane approach of memory-based multiplication arithmetic. Through this application, the method provides insights into human-like mental arithmetic and factorization processes. The usage of ternary elements in the weight matrix, as well as the Markov chain property of consideration of last state, contribute to the algorithm's ability to optimize parameters without the requirement for previous weight memory, solving computational complexity concerns in machine learning.

JayBot – Aiding University Students and Admission with an LLM-based Chatbot [6]

This paper talks about using generative AI like ChatGPT to address customer support issues and inquiries, with special focus on University Admissions.

To tackle issues like Hallucination or out of context replies, they use an embedding model to embed any data to store in a vector database. Any queries to these data would have similar embeddings in the vector space and can be received.

In the paper they built a LangChain for the same, using vector embeddings and vector databases to store out of model information (Domain specific) like university courses and accommodations. They have emphasized focus on Prompt engineering as the base language model (GPT-3.5 Turbo)

can handle a variety of tasks, so to build a query system using them can be easily done using a few well written prompts. They use RAG and vector store to handle user inquiries.

AI IN RADIO: THE GAME CHANGER YOU DID NOT HEAR

COMING [7]

The paper examines how artificial intelligence (AI) is developing both globally and domestically in Slovakia's radio broadcasting sector. It charts the evolution of AI integration in radio, its history, emphasizing significant turning points like the arrival of AI DJs and news presenters in different nations. The paper explores the methods utilized to examine the applications of AI in radio, including literature reviews and interviews with industry professionals.

The study examines the application of AI in radio from the perspective of radio professionals, for its use in news generation and broadcasting. It discusses how artificial intelligence (AI) technology makes automated journalism easier by creating news stories, summarizing data, and even creating broadcast-quality voice overs. However, it also recognizes some of the challenges associated with deploying AI, such as linguistic barriers and the perception that information generated by AI lacks authenticity.

The report offers observations obtained from speaking with officials of Slovak radio stations, which exposed a cautious but positive strategy for integrating AI. Even though artificial intelligence (AI) has great promise and efficiency for content development and production, human monitoring is still necessary to guarantee accuracy and keep viewers interested.

The paper concludes by highlighting the important role AI is playing in transforming radio broadcasting and providing chances for improved content production and faster procedures. To satisfy the needs and preferences of radio listeners, it emphasizes the significance of striking a balance between artificial intelligence's capabilities and human creativity and authenticity.

System Requirements Specifications

1. Overall Description

1.1. Product Functions

Main functions that will be available in RJ-AI.

1.1.1. Reliable News Updates

The RJ must be capable of providing reliable news updates, with relevant details and provide it in a way that is engaging to the audience.

1.1.2. Provide Weather Updates

Relevant weather updates are useful for daily commuters, the chief users of the Radio.

1.1.3. Provide Popular music

Main purpose of audio mass communication is to share and spread music.

1.1.4. Add commentary

A witty radio host is able to engage the audience in the show. A smooth transition between segments of the radio show is a must.

1.1.5. Setup proper radio events

Radio shows have specific events, like music-request, add-free music, quiz and many more.

1.1.6. Advertise products / Feature advertisements

Chief source of income for a radio station is their advertisements. Having dedicated sections without very long sections of advertisements is necessary to maintain audience retention.

1.2. User Classes and Characteristics

The main User classes and their characteristics are as follows.

1.2.1. Listeners

Listeners are people who tune in to the radio broadcast to enjoy music, stay updated with news and events, or listen to commentary and discussions. They like different kinds of content and have different musical interests. Listening in, choosing stations, messaging song requests, and even joining in on call-in programs or contests are how they engage with the radio. Can be chiefly classified into commuters, music listeners and news listeners.

1.2.2. Advertisers

Advertisers are businesses or organizations that use radio advertisements to market their goods, services, or events to radio listeners. They work with radio stations in order to produce and air commercials during sponsored segments or commercial breaks.

1.2.3. Radio administrator

Radio administrators are in charge of managing and supervising the radio network or station. Radio administrators oversee a variety of administrative responsibilities, such as managing personnel and resources, scheduling programming, making sure that rules are followed, and keeping an eye on ratings and comments from the audience.

1.2.4. Script reviewer

Before radio scripts are broadcast, they must be reviewed and approved by script reviewers to ensure coherence. By spotting and fixing mistakes, contradictions, or offensive information in scripts, script reviewers work to preserve the caliber and integrity of radio programming.

1.3. Design and Implementation Constraints

1. Technology Constraints:

Availability and compatibility of AI frameworks and libraries suitable for natural language processing (NLP).

Limitations imposed by hardware resources, such as processing power and memory.

2. Time Constraints:

Project deadlines and milestones, including the time required for development, and testing.

Time constraints for training and fine-tuning machine learning models for sentiment analysis, and content curation

3. Regulatory Constraints:

Compliance with regulations and guidelines related to broadcasting standards, privacy, and data protection.

Adherence to copyright laws and licensing agreements for broadcasting music and news content.

1.4. Assumptions and Dependencies

1.4.1. Availability of music to play on the radio station with permission from music licensors. IP is of chief concern.

1.4.2. Availability of medium to reach out audience. Frequency band sanctioning or website hosting (internet radio).

1.4.3. The end product will be acceptable to the user audience.

-
- 1.4.4. Dependence on external APIs or data sources for retrieving real-time news updates, weather forecasts, and music streaming services.

2. Specific Requirements

2.1. Functional Requirements

2.1.1. Music Recommendation

The radio show must be able to build music playlists, with a mix of trending and famous songs to appeal to a wider audience of music listeners. Ensure a song is not repeated too many times during the show and introduce the audience to new songs whenever possible.

2.1.2. Music Player

A dedicated system to enable the playing of songs in the environment, with relevant permissions from relevant authorities.

2.1.3. Information Retrieval

Be able to retrieve important details, especially news and weather updates from the web, learn new details as early as possible, ensuring correctness of the details.

2.1.4. News Summarization

Should be able to summarize the news collected into quick and easily digestible parts for the audience. Necessary details are never omitted and the summary must be complete.

2.1.5. Context based Commentary

The Radio host must be able to give certain comments and details, be it for the music or news wherever applicable.

2.1.6. Transition through segments

Allow for seamless movement between segments of the radio show.

2.2. Non-Functional Requirements

2.2.1. Performance Requirements

2.2.1.1. Low Latency

The radio host must ensure low latency with output and be ready for the next segment quickly to ensure a smooth radio show, without any breaks in it.

2.2.1.2. Modular

Ensure all the necessary components of the show are modularized, so that any updates, or fixes are independent of each other, so that if any component fails the others are available to take over.

2.2.1.3. Appropriate Content

The content produced must meet strict requirements and ensure that harmful and insulting content is not present.

2.2.2. Usability Requirements

2.2.2.1. Content Variability

Allow the content to have different themes, to ensure novelty and not strict monotonous behaviour.

2.2.2.2. Documentation

Documentation on the working and details to make changes will be provided to ensure smooth working for the users.

3. External Interface Requirements

3.1. Software Interfaces

3.1.1. API to retrieve news.

News stories are collected via World News APIs for news updates.

3.1.2. Weather updates API.

Get weather conditions from Open-Meteo weather API to provide summarized updates to the audience.

3.1.3. Database interface to obtain and play music

During building of playlist, choose appropriate music from the databases

4. System Features

4.1. Music Recommendation

4.1.1. Description and Priority:

4.1.1.1. Description: The model should provide music recommendations to users based on their preferences or a specified theme.

4.1.1.2. Priority: High

4.1.2. Stimulus/Response Sequences:

Stimulus: Music recommendation segment.

Response: Play the songs suggested by the recommender.

4.1.3. Functional Requirements:

4.1.3.1. The system should be able to gather popular and trending songs.

4.1.3.2. Generate recommendations based on recent trends and current theme

4.1.3.3. Retrieve relevant music from a database or music streaming service.

4.2. Weather and News Updates

4.2.1. Description and Priority:

- 4.2.1.1. Feature: Real-time updates on weather conditions, news, and traffic.
- 4.2.1.2. Description: RJ AI must provide users with real-time updates on weather conditions, news, and traffic.
- 4.2.1.3. Priority: Medium

4.2.2. Stimulus/Response Sequences:

Stimulus: Weather/News update segment.

Response: Provide appropriate updates relevant to the location

4.2.3. Functional Requirements:

- 4.2.3.1. Retrieve weather data: Obtain real-time weather data from a reliable source, such as a weather API for a specific location.
- 4.2.3.2. Retrieve news data: Fetch the latest news articles or headlines from trusted news sources.
- 4.2.3.3. Retrieve traffic data: Access up-to-date traffic information from relevant sources, such as traffic APIs or transportation agencies for a specific location.
- 4.2.3.4. Send updates: Deliver weather, news, and traffic updates to the user in a timely manner.

4.3. Radio script building

4.3.1. Description and Priority:

- 4.3.1.1. Feature: Radio Show Script Building
- 4.3.1.2. Description: AI model should build the script for the radio show, including thematic music, news updates, and other necessary parts of a radio show.
- 4.3.1.3. Priority: High

4.3.2. Stimulus/Response Sequences:

Stimulus: Today's theme is given by the Radio Station

Response: Build a script on the given theme

4.3.3. Functional Requirements:

- 4.3.3.1. The model will build a script based on the theme given.
- 4.3.3.2. Will provide proper segments for advertisements and news.
- 4.3.3.3. Provide context for the music and news wherever applicable.
- 4.3.3.4. Add cutoffs and jokes as needed.

4.4. Audience Interaction

4.4.1. Description and Priority:

- 4.4.1.1. Feature: Interaction with the audience
- 4.4.1.2. Description: Making the radio show engaging by hosting quizzes, game shows, playing user-requested music, and song dedications
- 4.4.1.3. Priority: Low

4.4.2. Stimulus/Response Sequences:

Stimulus: Segment of radio show dedicated to audience interactions

response: Facilitating interaction with the audience via text messages or voice calls.

4.4.3. Functional Requirements:

- 4.4.3.1 The model will prepare quizzes and game show contents.
- 4.4.3.2 The model will host the game and will coordinate audience input with it.
- 4.4.3.3 The model will take in song requests and choose the appropriate song to play. If a song is dedicated it will be mentioned to the audience.

4.5. Radio Commentary (Voice Integration)

4.5.1. Description and Priority:

4.5.1.1. Feature: Radio Voice

4.5.1.2. Description: Use Text to speech models

4.5.1.3. Priority: Low

4.5.2. Stimulus/Response Sequences:

Stimulus: Radio show script.

Response: Give voice to the script

4.5.3. Functional Requirements:

4.5.3.1. A text-to-speech software is given the script.

System Design

5. Introduction

5.1. Scope

RJ-AI is an AI model capable of building the script to host a radio show, complete with segments for music, News, updates and more. This document covers the core features and architectural components of the system, including Script building, Music recommendation and audience interaction.

5.2. Objectives

- 5.2.1. Able to write a cohesive script for a radio show.
- 5.2.2. Make seamless transitions between segments of the radio show.
- 5.2.3. Manage reasonable audience interaction like song requests.
- 5.2.4. Be able to find relevant updates and present the same in a condensed and engaging manner to the listener.

6. Architectural Overview

6.1. System Components

6.1.1. Module for Recommending Music:

Recommender: Makes music suggestions according to the time of day, the weather, and the user's current mood. A score algorithm directs the selection, which is then input into a cooldown system to avoid song recurrence.

Song Cooldown: Reduces duplication by using a fixed-size queue to track recent tracks.

Scoring: Sorts music based on mood and other contextual elements using a straightforward score-based recommendation system.

6.1.2. News API Module and summarizer:

Retrieves news stories from the World News API that are location-based and include sentiment scores. Depending on the needs of the presentation, the sentiment score is utilized to choose either neutral or polarizing articles.

Transforms news stories into succinct, objective reports fit for a radio audience by applying a Llama 3 model. This module keeps news parts short and informative.

6.1.3. Talk Module:

Responsible for generating talk segments with a radio host persona. The talk segment comprises of commentary on the previously played segment i.e news or music segment. Additionally if there is no previously played segment, weather commentary will be played to start off the radio program. We will be utilizing the llama 3.2 3B model. The model is fine tuned on radio program transcripts using the hugging face transformers library. The radio program chosen for this task is "Talk of the nation" starring "Neal Conan". The aim of fine-tuning would be to give the talk segment a radio host feeling and will be trained on this objective thereby giving the generated script a radio host tone and character.

6.1.4. Argument Collection:

Gathers and maintains contextual information about time, weather, mood, and news sentiment. The show's content decisions are based on this data.

6.1.5. Voice Module:

Utilizes speech technology from ElevenLabs to create realistic audio for news and chat programs. The lifelikeness of the synthetic voice is guaranteed by this module, which enhances the listening experience.

6.1.6. HMM based Scheduler

Serves as the underlying framework for generating the show's sequence of segments (music, talk, news) based on efficiency and flow requirements. The HMM ensures a smooth broadcast by striking a balance between effective control and natural transitions.

6.1.7. Iterator for programs:

Based on control inputs from other modules, this decision-making component iterates over the content, guiding the transition between chat, music, and news parts.

6.1.8. Controls for Output:

Creates a dynamic, organized audio output that mimics a live radio experience by routing the final outputs (music, talk, and news) to the appropriate channels.

6.2. Interactions

6.2.1. Scheduling

6.2.1.1. Use the stochastic scheduler (HMM) to build a probabilistic schedule for each of the main segments in a radio show. Wherever applicable ensure proper transitions and proper scheduling to ensure engagement of the audience.

6.2.2. Script Building

6.2.2.1. Use an LLM based chatbot, with a working memory to build a script for the radio show, with relevant focus on the engagement, and proper transitions between segments. The produced script must be free from all False information and not have toxicity.

6.2.3. Data Collection

6.2.3.1. Use APIs to collect relevant data like weather updates, recent news relevant to locations and use data summarization tools and LM to build a narrative on them for the audience

6.2.4. Music Recommendation

6.2.4.1. Use a well built music recommendation system, with focus on engagement to build a playlist for radio segments focused on music. Build a playlist with relevant themes as well.

6.3. Diagrams

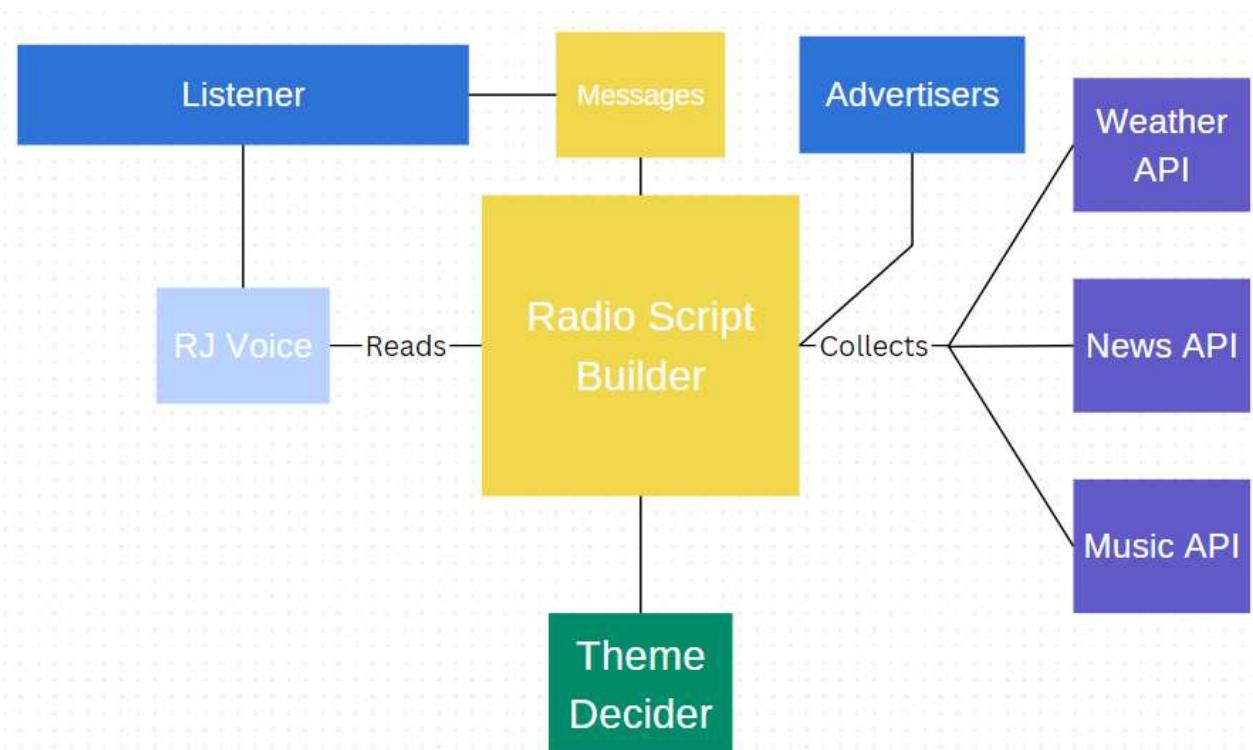


Figure 1: Block Diagram of the system

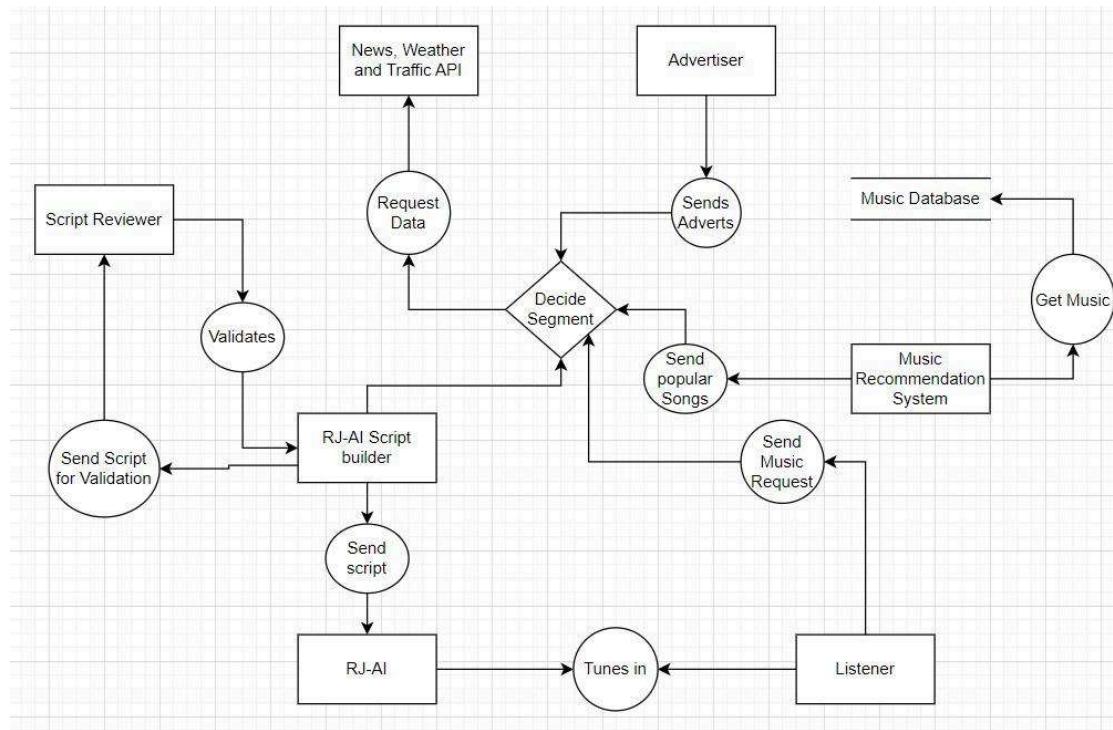


Figure 2: Data Flow Diagram of the system

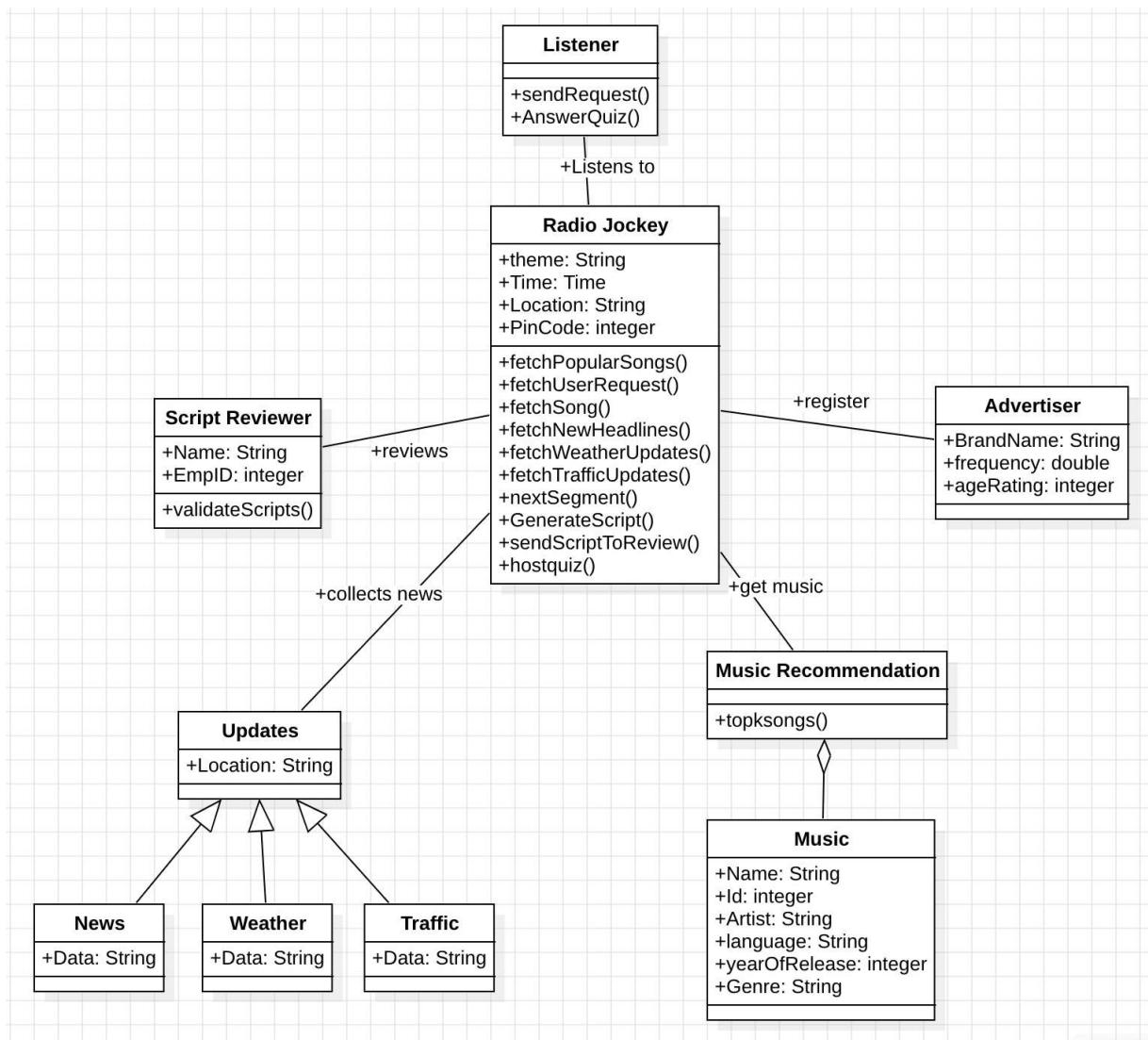


Figure 3: Master Class Diagram

7. System Components

7.1. Streaming software

Streaming software enables online streaming of the radio broadcast, allowing listeners to tune in via the internet.

7.2. Content Management System

It helps organize and manage the radio station's content, including show schedules, playlists, and archived recordings.

7.3. LM Chatbot

Will help build the script and subsequent communication of the RJ, with proper segmentation to ensure working of the radio host. This will be made using LangChain to facilitate multiple models to focus on specific parts (text summarization, Quiz generation and continuity builder) of the system.

7.4. Music Recommendation System

Will build a suitable playlist with a theme if necessary, for the radio show.

8. Interface Design

8.1. News interface

Connect to News outlets via their APIs to get latest and local news.

8.2. Weather Interface

Weather APIs are used to connect and receive local weather updates.

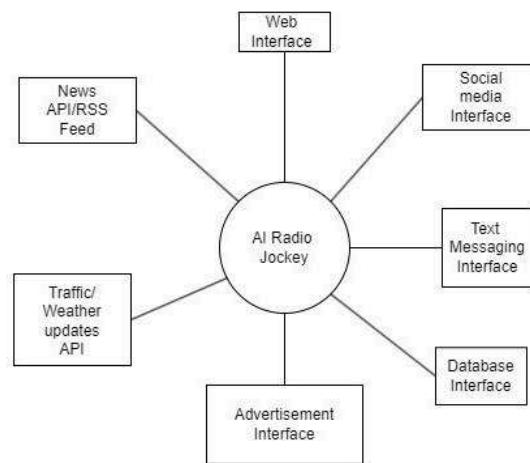


Figure 4: Interface Diagram

9. Methodology, Implementation, and Pseudocode

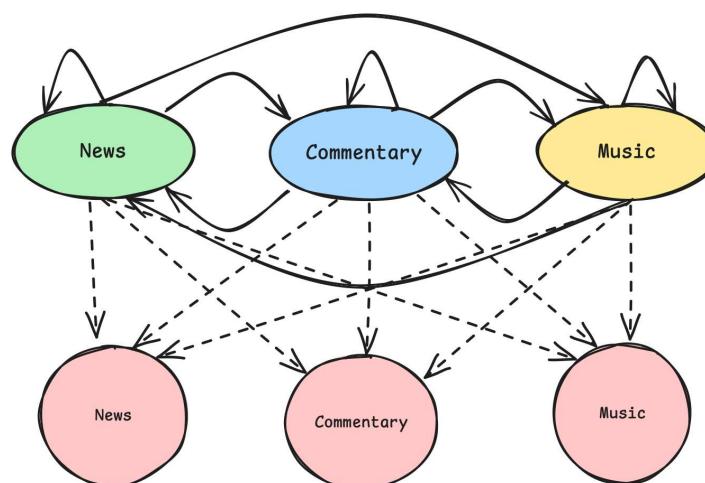
9.1. Overview

Markov models are mathematical tools used to represent systems in which state transitions are dependent only on the current state and not on the previous course of events. In radio broadcasting, these "states" stand for various content pieces, including COMMENTARY, NEWS, and MUSIC. The primary objective of employing a Markov model is to predict optimal segment transitions to maximize engagement and in our case to introduce variability in the organization of segments.

Here, we enhance the Markov-based system by integrating a simplified Music Recommendation System, a News Summarizer, and a Weather Summarizer. These components, coupled with a Markov Decision Process (MDP), ensure each segment transitions seamlessly, balancing listener engagement and content relevance.

Markov models are a robust approach to optimizing sequential decision-making processes, especially in dynamic environments like radio programming.

The states in the MDP represent the radio segments: MUSIC, NEWS and COMMENTARY. The actions invoke the relevant function: music recommendation, news summarization, or weather forecasting, guided by the probabilities.



Quasi-HMM: Hidden Markov model with Stochastic transition to self -emissions.

9.2. Initial analysis

The process begins with analyzing the components individually and integrating them into the MDP framework:

9.2.1. Music Recommendation System Analysis

A premade music dataset is used, containing:

Title: The song's name.

Singer: The performing artist.

Time: Best-suited time slot (e.g., morning, afternoon, evening, late night).

Day: Categorization for weekday or weekend.

Mood: Emotional tone of the song (e.g., Very Happy, Happy, Neutral, Sad, Very Sad).

Weather Suitability: Tracks matched with weather conditions (e.g., Warm and Sunny, Rainy and Overcast, Cold and Windy).

Dataset built using LLM tagging from available Kaggle dataset[12].

9.2.2. News Summarizer Analysis

9.2.2.1. News headlines and key points are extracted via a summarizer module. The input data comes from “World News API”.

9.2.2.2. Summarization ensures brevity while maintaining informational value, ideal for radio delivery. An average sentiment of collected news articles is used to decide mood for the day.

9.2.3. Weather Summarizer Analysis

9.2.3.1. Weather data is fetched using APIs (Open-Meteo).

9.2.3.2. The module condenses weather updates to listener-relevant summaries, including current conditions and short-term forecasts.

9.2.3.3. Weather conditions used to decide weather suitability of songs.

9.2.4. Talk Module

- 9.2.4.1. Use fine-tuned Llama 3.2 3B parameter model to provide script for the talk segment of the radio show when provided with context.
- 9.2.4.2. Used for the commentary parts during the radio show and the weather segment.

9.2.5. State Definitions

- 9.2.5.1. MUSIC
- 9.2.5.2. NEWS
- 9.2.5.3. COMMENTARY

9.2.6. Transition Probability Matrix

	News	Music	Commentary
News	0.1	0.6	0.3
Music	0.4	0.35	0.25
Commentary	0.35	0.55	0.1

9.2.7. Emission Probability Matrix

	News	Music	Commentary
News	0.2	0.2	0.6
Music	0.1	0.8	0.1
Commentary	0.7	0.2	0.1

9.3. Set-up and implementation of Modules

9.3.1. Simplified Music Recommendation Module

Input: Time, day, mood, and weather parameters.

Process: Filter songs from the dataset based on matching criteria.

Output: A curated playlist for the segment.

Example:

If it's a rainy weekend evening, the system might suggest "Rainy and Overcast Songs" with a "Sad" or "Neutral" mood tone.

```
def recommend_songs(songs_df, cur_mood, temp, prec, cloud, num_recommendations=5):
    """Recommend songs based on current mood and manage cooldown."""
    cooldown_manager = CooldownManager()
    recommendations = []

    # Calculate scores for each song
    songs_df['Score'] = songs_df.apply(lambda row: assign_score(row, cur_mood, temp, prec, cloud), axis=1)

    # Sort songs by score in descending order
    sorted_songs = songs_df.sort_values(by='Score', ascending=False)

    # Recommend songs while respecting cooldown
    for _, row in sorted_songs.iterrows():
        song = row["Title"]
        if cooldown_manager.add_song(song):
            recommendations.append((song+" by "+row["Singer"]))
        if len(recommendations) == num_recommendations:
            break

    return recommendations
```

Use a queue to implement a cooldown system to ensure the same song will not get repeated multiple times during playlist building. Use a score based approach for sorting the songs, and take the top few songs for the playlist. Scores are allocated based on current mood, weather parameters as well as time details.

9.3.2. Talk Module.

The dataset selected for the fine-tuning objective was [INTERVIEW: NPR media dialog transcripts](#)[13]. It consists of radio transcripts for various NPR radio shows broadcasted in America. We selected one show called "Talk of the nation" and took a subset of that show. Furthermore, the dataset contained transcripts of radio hosts and guests. We fine-tuned on the transcripts of the main radio host called "Neal Conan".

Parameter-efficient fine-tuning (PEFT) method called LoRA (Low-Rank Adaptation of Large Language Models) was used to fine-tune llama 3.2 3B model efficiently. The rank selected for LoRA was 16 considering higher ranks give negligible gain.

LoRA added rank decomposition matrices to 28 QKV layers, 28 MLP layers and 28 output projection layers (O layers).

```

model = FastLanguageModel.get_peft_model(
    model,
    r = 16, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = "none",    # Supports any, but = "none" is optimized
    use_gradient_checkpointing = "unsloth",
    random_state = 3407,
    use_rslora = False,
    loftq_config = None,
)

```

→ Unsloth 2024.10.7 patched 28 layers with 28 QKV layers, 28 O layers and 28 MLP layers.

Fine tuning details

Hugging face transformers library was the ML framework chosen for fine-tuning llama 3.2. Fine-tuning was done in 200 steps with the linear learning rate scheduler with 0.5e-5 as the learning rate and batch size as 2.

Training and evaluation loss details

```

▶ trainer_stats = trainer.train()

→ ==((=====))== Unsloth - 2x faster free finetuning | Num GPUs = 1
    \| /| Num examples = 243,853 | Num Epochs = 1
    0^0/ \_ / Batch size per device = 2 | Gradient Accumulation steps = 4
    \_ / Total batch size = 8 | Total steps = 200
    "—" Number of trainable parameters = 24,313,856
[200/200 47:38, Epoch 0/1]

Step Training Loss Validation Loss
50      3.800400      3.727782
100     3.499800      3.397751
150     3.291300      3.305167
200     3.380700      3.282981

```

The fine-tuned model was quantized in 8 bit format and ran locally using Ollama.

```

def talk(choice,content):
    if choice==1:
        prompt='News Prompt.'
        content="News articles:" +". ".join(content)
    elif choice==2:
        prompt='Music Prompt.'
        content="Songs:" +", ".join(content)
    elif choice==0:
        prompt='Weather prompt'
        content="Current weather is :" +content[0]+ " forecasted weather is :" +content[1]

    response = ollama.chat(
        model='hostllama',
        messages=[
            {'role': 'system', 'content': 'You are a radio host. Your job is to interactively speak to the audience.'},
            {'role': 'assistant', 'content':prompt} ,
            {'role': 'user', 'content': content}
        ]
    )
    k=response['message'][ 'content']
    print(response['message'][ 'content'])

```

9.3.3. News Summarizer Module

Input: Raw news articles fetched via API.

Process: Summarize key stories using llama 3.1 LLM model.

Output: Concise news bulletins tailored for the audience as well as overall sentiment for the day

```
def news_data(data):
    result = []
    sentiment = 0
    count = 0
    client = GPTClient(
        api_key="")
    if data:
        for article in data['news']:
            sentiment += article['sentiment']
            count += 1
            response = client.chat.completions.create(
                messages=[
                    {
                        "role": "system",
                        "content": "You are given a the text from a news article. Generate a summary(within 80 words)"
                    },
                    {
                        "role": "user",
                        "content": f"Here are the news article in the form of an array\n{article['text']}"
                    }
                ],
                model="llama-3.1-8b-instant",
            )
            result.append(response.choices[0].message.content)

    return result,sentiment/count
```

Collect news articles from “World News API” and send the articles to llama 3.1 LLM for summarization of the articles. Collects sentiment of each article, and calculate an average sentiment for the day. Sentiment is used by the music recommender to predict which songs to play.

Audiences get a glimpse of the day by the summaries generated from the news summarization module.

9.3.4. Weather Summarizer Module

Input: Real-time weather data fetched via API.

Process: Extract and summarize temperature, conditions, and forecasts.

Output: A clear weather report as well as Value for Song recommendation.

```
def weather():
    url = "https://api.open-meteo.com/v1/forecast"
    params = {
        "latitude": 12.9719,
        "longitude": 77.5937,
        "hourly": ["temperature_2m", "relative_humidity_2m", "apparent_temperature", "precipitation_probability", "rain"],
        "timezone": "auto",
        "forecast_hours": 12,
        "forecast_days": 1
    }
    response = openmeteo.weather_api(url, params=params)

    # Process hourly data. The order of variables needs to be the same as requested.
    hourly = response.Hourly()
    hourly_data = {"date": pd.date_range(
        start = pd.to_datetime(hourly.Time)+response.UtcOffsetSeconds(), unit = "s", utc = True),
        end = pd.to_datetime(hourly.TimeEnd)+response.UtcOffsetSeconds(), unit = "s", utc = True),
        freq = pd.Timedelta(seconds = hourly.Interval()),
        inclusive = "left"
    )
    hourly_data["temperature_2m"] = hourly.temperature_2m
    hourly_data["relative_humidity_2m"] = hourly.relative_humidity_2m
    hourly_data["apparent_temperature"] = hourly.apparent_temperature
    hourly_data["precipitation_probability"] = hourly.precipitation_probability
    hourly_data["rain"] = hourly.rain
    hourly_dataframe = pd.DataFrame(data = hourly_data)

    current_weather=(hourly_dataframe.iloc[0:1])
    current_weather=current_weather.drop(columns=[ precipitation_probability'])
    forecast_hourly=hourly_dataframe.iloc[1:]

    current_weather=current_weather.to_json(orient='records',date_format='iso')
    forecast_hourly=forecast_hourly.to_json(orient='records',date_format='iso')

    return [current_weather,forecast_hourly]
```

Open-meteo weather api is used to fetch weather data for a particular location. Various weather parameters are fetched like temperature, current condition, weather code, humidity and forecast.

Audiences will be better informed by the weather summarization provided by the fine tuned llama 3.2 model. Listeners get an accurate and lively commentary on the weather.

Current weather conditions are also used by the simple music recommender to recommend songs appropriate to the current weather.

9.3.5. Markov Decision Process (MDP)

```

def programme():
    initial_probs= np.array([0,0,1]) #State1=News, State2=Music, State3=Idle Chat

    transition_probs= np.array([
        [0.1, 0.6, 0.3],
        [0.4, 0.35, 0.25],
        [0.35, 0.55, 0.1]
    ])

    # Emission probabilities matrix
    # For simplicity, let's assume 3 possible observations: "greeting", "song", "content"
    emission_probs = np.array([
        [0.2, 0.2, 0.6], # State 1
        [0.1, 0.8, 0.1], # State 2
        [0.7, 0.2, 0.1] # State 3
    ])

    states = ["Talk", "song", "News"]

    # Define the model
    model = MultinomialHMM(n_components=len(states),n_trials=10)

    # Set model parameters
    model.startprob_ = initial_probs
    model.transmat_ = transition_probs
    model.emissionprob_ = emission_probs

    # Simulating a radio show sequence
    # Number of steps in the sequence
    n_steps = 10

    # Generate a sequence
    logprob, sequence = model.sample(n_samples=n_steps)

    # Decode the sequence of states
    states_sequence = model.predict(logprob)

    observations = ["News", "song", "Talk"]
    return [observations[int(obs)] for obs in sequence.flatten()]

```

Implementing a “Quasi HMM” approach where the emission states are the same as the hidden states.

This gives the decision process more organicity and unpredictability.

9.3.6. Integration Framework

```

weather_today=weather()
tw=json.loads(weather_today[0])
temp=tw[0].get('temperature_2m')
prec=tw[0].get('precipitation')
cloud=tw[0].get('cloud_cover')
recommendations = recommend_songs(songs_df, sentiment, temp, prec, cloud, today_programme.count("song")*3)
m_pointer=0
n_pointer=0
def showrunner(programme):
    global m_pointer
    global n_pointer
    for i in range(len(programme)):
        if programme[i]=="News":
            playsound("./nt.mp3")
            print(news_snippet[n_pointer:n_pointer+2])
            n_pointer=n_pointer+2
        elif programme[i]=="song":
            playsound("./mt.mp3")
            print(recommendations[m_pointer:m_pointer+5])
            m_pointer+=5
        elif programme[i]=="Talk":
            playsound("./rt.mp3")
            if i==0:
                talk(0,weather_today)
            elif programme[i-1]=="News":
                talk(1,news_snippet[n_pointer-2:n_pointer])
            elif programme[i-1]=="song":
                talk(2,recommendations[m_pointer-5:m_pointer])
showrunner(today_programme)

```

The radio show will have a morning segment, afternoon segment and evening segment. Segment length can be decided before running the radio program. Each segment may begin with a weather report by the fine tuned llama 3.2 LLM. The next segment is decided by MDP. A queue and sliding windows are used to fetch and play news and music relevant to the current context (i.e. sentiment and weather).

10. Results and Discussions

10.1. Results

The fine-tuning of the Large Language Model (LLM) integrated with a simplified Music Recommender System, News Summarizer, and a Markov Decision Process (MDP) for segment decision-making yielded significant insights and measurable improvements.

10.1.1. Enhanced Content Generation

10.1.1.1. Music Recommendations:

Simplified score based recommendation system helps in providing contextual and dynamic song recommendations based on the metadata (time, day, mood, weather)

```
● Recommended Songs: [('Too Sweet', 'Hozier'), ('Virtual Insanity', 'Jamiroquai'), ('Uptown Funk', 'Mark Ronson'), ('Dancing Queen', 'ABBA'), ('Blinding Lights', 'The Weeknd'), ('Uptown Funk', 'Mark Ronson'), ('Bohemian Rhapsody', 'Queen'), ('Rasputin', 'Boney M')]

○ PS C:\Users\Chandrachud\Desktop\Capstone_codes\Music_Recommendation>
```

10.1.1.2. News Summarization:

Effective Prompts can help in effectively condense lengthy articles into concise, engaging bulletins

```
1 ; Vokkaliga turf warfare escalates ahead of Channapatna bypoll. Deputy Chief Minister D.K. Shivakumar and Union Minister H.D. Kumaraswamy engage in verbal duel, accusing each other of corruption and ineffective governance. Kumaraswamy hits back at Shivakumar's claims, saying his achievements prove his developmental efforts. By polls result crucial for future state politics; constituents to decide on performance and policy initiatives.
```

```
2 ; BJP stages State-wide protest against Waqf Board, demanding amendments to the Act to protect farmers from losing land ownership. Leader R Ashok urges Congress government to make necessary amendments, accusing Chief Minister Siddaramaiah of appeasement policy and seeking his resignation amid allegations of waqf officials issuing eviction notices to farmers.
```

10.1.2. Fine-Tuning LLM Outcomes

Results of Fine-tuning LLM:

Input Prompt: You are a radio host. News articles have been played. Provide a short paragraph of commentary on the following news article.give your opinions. Do not mention anything else.

Before Fine-Tuning: It's clear that Microsoft is taking a cautious approach with their highly anticipated Recall feature, and for good reason. The company has been dealing with a flurry of criticism and concerns from security experts and insiders since its initial announcement in May, and it's evident that they're prioritizing user safety above all else. While the delayed release may be frustrating for some, it's reassuring to know that Microsoft is taking the time to refine and strengthen their feature, ensuring that it meets their high standards for security and performance.

After Fine-Tuning: Microsoft's Recall feature has been a hot topic of controversy since its initial announcement in May. It seems that concerns over security have led to yet another delay in its release. I've got to say, I'm a bit surprised by this. On one hand, Microsoft is clearly taking steps to ensure the safety and security of their users. Refining the feature to provide a more secure user experience isn't something you can rush. However, I do wonder if there's been too much delay at all. After being pushed back from June to October, it feels like we're waiting for what seems like an eternity. I'm hoping that by December, Microsoft will be able to iron out any remaining issues and get Recall into the hands of Windows Insiders program members.

As you can see, the output after Fine-Tuning has more of the characteristics of a radio host persona. The output closely resembles what a radio host would say. Furthermore, a radio host tone and character is maintained throughout the generated output.

10.2. Discussion

10.2.1. Effectiveness of Markov Models in Programming

- 10.2.1.1. The Markov model provided a data-driven framework for segment transitions, improving listener engagement.
- 10.2.1.2. By focusing on state-dependent transitions, the system ensured continuity and minimized abrupt shifts in content.

10.2.2. Synergy Between LLM and Markov Model

- 10.2.2.1. The Markov model provided a robust structure for decision-making, while the fine-tuned LLM enriched content within each segment.

10.2.2.2. The probabilistic nature of the Markov model ensured that segment transitions aligned with listener preferences, while the fine-tuned LLM ensured the content quality within each segment.

10.2.3. Challenges and Limitations

10.2.3.1. Data Quality:

The system's effectiveness heavily depended on the accuracy and completeness of the input data (e.g., listener preferences, song metadata).

10.2.3.2. Complexity in Transition Probabilities:

Some segments, such as COMMENTARY, had unpredictable transitions, requiring additional data to refine probabilities.

10.2.3.3. Real-Time Adjustments:

Sudden changes in listener behavior (e.g., during special events or holidays) posed challenges for the model's adaptability.

10.2.4. Comparative Analysis with Traditional Programming

Traditional, static programming lacked the flexibility to adapt to real-time data, resulting in lower retention rates.

The Markovian approach remedies that by introducing uncertainty in emissions from transitions, ensuring a more organic experience.

11. Conclusion and Future Work

By establishing a Markov model with music recommendations , summarised news, and weather updates, the radio show can become more organic and audience intriguing. Beyond this, the system can ensure that all segments are both timely and engaging, providing a superior listening experience. The modular design implemented allows for easy and continuous refinement and upgrading, ensuring the proposed methodology evolves with listener preferences and technological advancements.

Fine-tuning the LLM and integrating it with a Markov-based decision-making process is seen to provide significant improvements in radio programmes. The model agent developed can deliver relevant content of various purposes with smooth transitions, Effectively balancing automation and personalization. These results highlight the potential of combining LLMs with probabilistic models for adaptive, data-driven media solutions.

11.1. Future Work

- 11.1.1. A more versatile LLM/multiple-dedicated-LLMs to include multilingual content generation for a wider audience base.
- 11.1.2. Implementing advanced deep learning approaches into the music recommendation system for personalized playlists.
- 11.1.3. Expand the programming to include more variety of content in terms of segments and sub-segments, such as interviews or live calls, while maintaining state transition accuracy.
- 11.1.4. Providing social media platforms where audiences can interact and communicate their thoughts with the AI radio jockey.

References

- [1] Hu, B., Shi, C., Liu, J. (2017). Playlist Recommendation Based on Reinforcement Learning. In: Shi, Z., Goertzel, B., Feng, J. (eds) Intelligence Science I. ICIS 2017. IFIP Advances in Information and Communication Technology, vol 510. Springer, Cham.
https://doi.org/10.1007/978-3-319-68121-4_18
- [2] A. A. Syed, F. L. Gaol and T. Matsuo, "A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization," in IEEE Access, vol. 9, pp. 13248-13265, 2021, doi: 10.1109/ACCESS.2021.3052783.
- [3] G. Lee, V. Hartmann, J. Park, D. Papailiopoulos, and K. Lee, "Prompted LLMs as Chatbot Modules for Long Open-domain Conversation," *arXiv.org*, 2023.
<https://arxiv.org/abs/2305.04533>
- [4] Jie Yang, Yangsheng Xu and C. S. Chen, "Hidden Markov model approach to skill learning and its application to telerobotics," in IEEE Transactions on Robotics and Automation, vol. 10, no. 5, pp. 621-631, Oct. 1994, doi: 10.1109/70.326567.
- [5] Guhyun, Kim., Guhyun, Kim., Vladimir, Kornjicuk., Dohun, Kim., Dohun, Kim., Inho, Kim., Jaewook, Kim., Hyo, Cheon, Woo., Ji, Hun, Kim., Cheol, Seong, Hwang., Doo, Seok, Jeong. (2019). Markov chain hebbian learning algorithm with ternary synaptic units. IEEE Access, 7:10208-10223. doi: 10.1109/ACCESS.2018.2890543
- [6] J. Odede and Ingo Frommholz, "JayBot -- Aiding University Students and Admission with an LLM-based Chatbot," Mar. 2024, doi: <https://doi.org/10.1145/3627508.3638293>.
- [7] Furtáková, Lucia & Janáčková, Lubica. (2023). AI in radio: The game changer you did not hear coming. Media & Marketing Identity. 10.34135/mmidentity-2023-09.
- [8] Visual RadioGPT Radio Automation Software with AI: <https://radiogpt.ar/>
- [9] RadioGPT: Revolutionizing Radio:
<https://medium.com/ai-for-product-people/radiogpt-revolutionizing-radio-b51758578ce5>

[10]Get Down Tonight - Futuri AudioAI: <https://listen.streamon.fm/radiogpt>

[11]5 AI Tools To Help Your Radio Station:<https://radio.co/blog/ai-tools-radio>

[12] Million song dataset:

<https://www.kaggle.com/datasets/notshrirang spotify-million-song-dataset>

[13]INTERVIEW: NPR Media Dialog Transcripts

<https://www.kaggle.com/datasets/shuyangli94/interview-npr-media-dialog-transcripts>