REPUTATION MANAGEMENT IN SOCIAL MEDIA

A PROJECT REPORT

Submitted by

AAKAASH KB	**(711620243001)**
JENIFER ROHINI R	**(711620243004)**
PRIYADHARSHINI B	**(711620243308)**

in partial fulfilment

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

KATHIR COLLEGE OF ENGINEERING

"WISDOM TREE", NEELAMBUR, COIMBATORE – 641 062

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report
"REPUTATION MANAGEMENT IN SOCIAL MEDIA"
is the bonafide work of

"AAKAASH KB	**(711620243001)**
JENIFER ROHINI R	**(711620243004)**
PRIYADHARSHINI B	**(711620243308)"**

who carried out this project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. M.RAJESH BABU, M.E., Ph.D.,	**Dr.AYYAVOO MITHILA,**
HEAD OF THE DEPARTMENT	**ASSOCIATE PROFESSOR**
Professor and Head	Associate Professor
Department of	Department of
Artificial Intelligence and Data Science	Artificial Intelligence and Data Science
Kathir College of Engineering	Kathir College of Engineering
Coimbatore – 641 062	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.Tech 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. R.UDAIYAKUMAR, 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. M.RAJESH BABU, M.E., Ph.D.,.,** Head of the Artificial Intelligence and Data Science Department, and **Dr.A.MITHILA, M.E., Ph.D.,** Project Guide and Project coordinator of Artificial Intelligence and Data Science Department, for their constant suggestion and encouragement in the project work.

We also thank all our Faculty Members and Non-Teaching Staff Members of the Department of Computer Science and Engineering and our Lovable Parents and Friends who contribute many suitable ways to achieving final results.

TABLE OF CONTENTS

ABSTRACT

The problem of dealing with reputation management in social media is proposed with the solution of building an AI enabled sentiment analysis tool that interprets emotions in social media posts, allowing people and organisations to successfully manage their online reputation. The proposed solution is built with the concept of multimodal sentimental analysis that deals with the process of analysing and understanding sentiment (emotions, attitudes, opinions) expressed in content that involves multiple modes or types of data. Unlike the traditional method which only analyses and depend purely on the textual data to understand the emotions, this approach tends to give a more comprehensive analysis of sentiment in diverse and rich multimedia information. As a valuable resource for market research, this tool keeps aware of industry trends, client preferences, and new challenges. Thus, this helps in empowering individuals and organizations to manage their online reputation and perception effectively.

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.No	Acronym	Abbreviation
1	LSTM	Long Short Term Memory
2	SC - NLM	Structure Content Neural Language Model
3	MDREA	Multimodal dual recurrent encoder with Attention (MDREA)
4	VAE	Variational Autoencoder
5	OCR	Optical Character Recognition
6	ViT	Vision Transformer
7	GPT	Generative Pretrained Transformer

CHAPTER 1
INTRODUCTION

1.1. Motivation of the Project

Maintaining a strong social media presence is critical in today's society for a variety of reasons. First off, it has a big influence on career prospects because recruiters and companies frequently evaluate applicants based on their internet presence. A good reputation can improve networking opportunities and job chances.

Secondly, social media allows and helps individuals to promote and improve their brand and industry. In a competitive landscape, a good reputation can differentiate you from others in similar roles or industries. It can also help you stand out positively when compared to your competitors.

Social media has become an effective tool in recent years for influencing public opinion, allowing for rapid contact, and promoting the broad distribution of information. But social media's separated structure also comes with hazards since people and organisations are always open to public criticism and examination.

As a result, maintaining a positive online reputation involves navigating and adjusting to the ever-changing currents of social media trends, where each like, share, or remark contributes to the dynamic creation of a person's digital identity. Thus, the AI enabled sentiment analysis tool helps us analyse and understand the emotions underlying the context of reviews, comments etc. and helps to track the overall reputation of an individual or product effectively.

1.2. Objective of the Project

Social media has become a powerful force in the globe, influencing many elements of society, communication, and culture. Beyond serving as a source of news and information, social media platforms have evolved into hubs for social action and movements, allowing users to express their views and push for societal change.

Trends on social media platforms may quickly boost or lower one's position depending on exposure and alignment with popular themes. What is deemed popular or attractive in the virtual world may rapidly become a standard for society judgement.

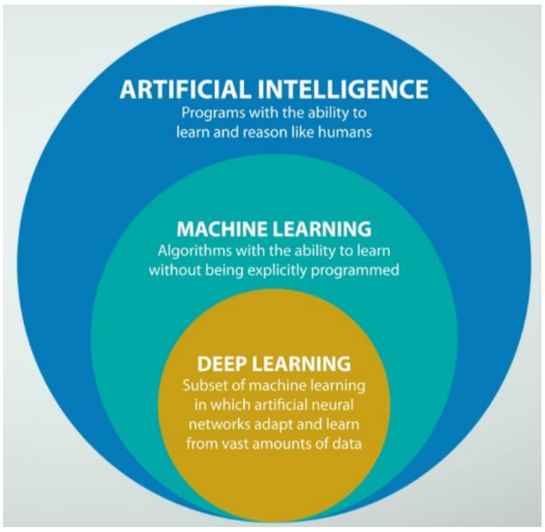
Thus, managing the online reputation optimistically has led the way to this path of innovation of developing a tool that analyses and optimizes the trends in the social media and help in managing the reputation as well as profitable marketing.

This AI enabled analytical tool developed to evaluate social media content and manage online reputation has several benefits. The sentiment analysis component assesses public opinion, analysing the trends and detects major brand references and influencers, providing useful information for focused engagement and keeps the optimized management of reputation on the web-based applications.

The proposed solution involves the process of dealing with different types of data such as image, audio, video and text. Thus, this solution introduces the concept of multimodal sentimental analysis that deals with different modalities of data and helps us analyse the overall reputation through various sources.

1.3. Deep Learning

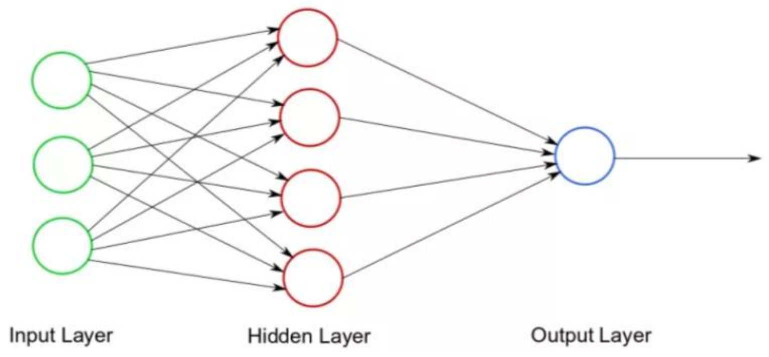
Deep learning is a branch of machine learning that focuses on using neural networks with multiple layers to learn representations of data. Inspired by the structure and function of the human brain, deep learning models consist of interconnected nodes organized into layers, where each node performs a mathematical operation on its input and passes the result to the next layer. What sets deep learning apart is its ability to handle complex tasks by learning hierarchical representations of data.


ARTIFICIAL INTELLIGENCE
Programs with the ability to learn and reason like humans
MACHINE LEARNING
Algorithms with the ability to learn without being explicitly programmed
DEEP LEARNING
Subset of machine learning in which artificial neural networks adapt and learn from vast amounts of data

1.1. Deep Learning

With deeper architectures comprising numerous layers, deep neural networks can automatically extract intricate features from raw data, enabling them to solve challenging problems in areas like computer vision, natural language processing, speech recognition, and more. The effectiveness of deep learning lies in its capacity to learn from large amounts of data, discovering intricate patterns and relationships that would be difficult to capture using traditional machine learning methods.

Deep Learning models are able to automatically learn features from the data, which makes them well-suited for tasks such as image recognition, speech recognition, and natural language processing. Network components include neurons, connections, weights, biases, propagation functions, and a learning rule. Neurons receive inputs, governed by thresholds and activation functions. Connections involve weights and biases regulating information transfer. Learning, adjusting weights and biases, occurs in three stages: input computation, output generation, and iterative refinement enhancing the network's proficiency in diverse tasks.



1.2. Basic neural network

1.4. Multimodal sentimental analysis

Multimodality is the use of several literacies within the same media. Multiple literacies or "modes" help an audience understand a piece. Multimodality is an inter-disciplinary approach that recognises communication and representation as more than just language. It has been created over the last decade to methodically examine hotly disputed concerns concerning societal developments, such as new media and technology.

Multimodal sentiment analysis refers to the process of analysing and understanding sentiment (emotions, attitudes, opinions) expressed in content that involves multiple modes or types of data. Unlike the traditional method which only analyses and depend purely on the textual data to understand the emotions, this approach tends to give a more comprehensive analysis of sentiment in diverse and rich multimedia information.

The information that are in various modes of data are combined and represented in the form of multimodal representations. This is an area of artificial intelligence (AI) and machine learning that focuses on developing models capable of learning representations from data across multiple modalities. Now the various modalities can be stated as textual data, image data, video data and audio data. Thus, combining all these forms of data that is available in the social media are extracted and final multimodal sentiment analysis model is developed.

CHAPTER 2

LITERATURE SURVEY

2.1. Introduction

This chapter gives an overall view about the survey conducted regarding the development of the multimodal sentiment analysis tool. A literature survey in a project is a thorough examination of academic materials on a given topic. It provides an overview of current knowledge, helping to find applicable ideas, methodologies, and gaps in previous research. Here, analysing the emotions in different modalities is discussed.

2.2. Textual Data

Various ideologies were proposed regarding the sentiment analysis of textual data. The conversion from one modality to another seems to be the crucial part of the analysis. If the possibility of conversion is high, then it would be better to analyse the underlying emotions present in the textual form. The other forms of data can be converted in textual form and can be used for analysis. The basic idea of implementing this structure comes from creating a base LSTM model.

There are various proposals on using a LSTM model for the conversion of modalities. The model focuses on building an encoder-decoder pipeline that builds the joint multimodal space with the text and image modalities and the decoder extracts the representations from this space [9]. The encoder allows one to rank images and sentences while the decoder can generate novel descriptions from scratch [9]. For encoder, here the LSTM (Long Short-Term Memory) recurrent neural networks is used to encode words and train a hybrid image-sentence embedding algorithm[9]. For decoder, a new neural language model known as structure-content neural language model (SC-NLM) is introduced that disentangles the structure of a sentence to its content, conditioned on distributed representations produced by the encoder and this approach effectively unifies image-text embedding models (encoder phase) with multimodal neural language models (decoder phase)[9]. This algorithm proved better than the static models.

Generating labels also proves to be one of the best ways of handling visual data and converting them to readable format[4,9]. One of the ways in creating a model that generates labels in text based on the visual and textual input and this approach follows on building an encoder-decoder pipeline that analyses the data segments and provides the report[4]. The model introduces VL-T5 and VL-BART based on two pretrained transformer language models: T5Base and BARTBase[4]. The bidirectional multimodal encoder is used for classifying the data and the autoregressive text decoder is used for producing the output[4]. We can infer the technique of pre – training the modalities and this approach is useful as we pre train them individually and combine them in the final model.

2.3. Audio Data

Usually, the textual and the audio data are dealt with a combine model or conversion is highly preferred. The ideology is implemented through the model called Multimodal dual recurrent encoder with Attention (MDREA) proposes a novel deep dual recurrent encoder model that utilizes text data and audio signals simultaneously to obtain a better understanding of speech data[7]. As emotional dialogue is composed of sound and spoken content, this model encodes the information from audio and text sequences using dual recurrent neural networks (RNNs) and then combines the information from these sources to predict the emotion class[7]. This is done by first building the recurrent encoder model for the audio and text modalities individually and then a multimodal approach is proposed that encodes both audio and textual information simultaneously via a dual recurrent encoder[7].

2.4. Image Data

Numerous ideologies are there in dealing with the visual data. Usually, they are created as individual models that uses advanced techniques in order to understand priorly the meaning underlying and then the emotion. There are various models that were inferred for this project such as Imagebind, BLIP, Ofa and SimVLM. These models try to combine different modalities and helps in visualizing the concepts not only in a theoretical way but also as a practical conceptual idea.

The first ideological approach Imagebind is an approach to learn a joint embedding across six different modalities - images, text, audio, depth, thermal, and IMU data[1]. This deals on proving that not all combinations of paired data are necessary to train such a joint embedding, and only image-paired data is sufficient to bind the modalities together[1]. This model learns a single shared representation space by leveraging multiple types of image-paired data. As conveyed, this approach lags in real world applications. But this provides an idea of leveraging the binding property of images and show that just aligning each modality's embedding to image embeddings leads to an emergent alignment across all of the modalities[1].

Another model approach is BLIP(Bootstrapping language-image pre-training for unified vision-language understanding and generation)[3]. This is a new VLP (Vision Language Pre-training) framework which transfers flexibly to both vision-language understanding and generation tasks[3]. Through the process of bootstrapping, in which a captioner creates artificial captions and a filter eliminates the noisy ones, BLIP efficiently makes use of the noisy online data. In order to increase the model's efficiency, it is required to bootstrapping multiple rounds of dataset along with increasing the training process of the model. This infers the idea of having a single machine that understands as well as generates the captions for the user with its VLP framework[3].

The next approach is referred as Ofa[2]. This works on providing a Task-Agnostic and Modality-Agnostic framework that unifies a diverse set of cross-modal and unimodal tasks in a simple sequence-to-sequence learning framework[2]. This model follows the instruction-based learning in both pretraining and finetuning stages, requiring no extra task-specific layers for downstream tasks and this task and modality agnostic model achieves competitive performance in zero-shot learning[2]. Also, it can transfer to unseen tasks with new task instructions and adapt to out - of domain information without finetuning[2]. This approach tries to provide the framework with generating images using text but the requirements for the algorithm needs more attention[2]. The multimodal pre-training is the concept that we have inferred from this approach which involves of dealing with multiple modalities in a large form and still combining them to the implementation at the process of training[2].

2.5. Sentimental Analysis

As the model is dealt with three modalities such as text, audio and image, the multimodal sentimental analysis should be inferred. One of the best ways is described by developing a FUSION model that uses audio, visual and textual modalities as sources of information. Both feature- and decision-level FUSION methods is used to merge affective information extracted from multiple modalities[8]. First, they present an empirical method used for extracting the key features from visual and textual data for sentiment analysis and then, they describe a FUSION method employed to fuse the extracted features for automatically identifying the overall sentiment expressed by a video[8]. Similarly, the audio and textual features were also extracted from each segment of the audio signal and text transcription of the video clip, respectively. Next, the audio, visual and textual feature vectors to form a final feature vector which contained the information of both audio, visual and textual data is used. Later, a supervised classifier is employed on the fused feature vector to identify the overall polarity of each segment of the video clip[8]. The results states that the accuracy improves dramatically when audio, visual and textual modalities are used together.

Then the next approach would be of developing the similar FUSION model but with a variational autoencoder along with the attention factor[7]. The model first eliminates noise interference in textual data and extracts more important image features[7]. Then, in the feature-FUSION part based on the attention mechanism, the text and images learn the internal features from each other through symmetry[7]. Then the FUSION features are applied to sentiment classification tasks[7]. Here, they use a denoising autoencoder and improve the variational auto-encoder combined with an attention mechanism (VAE-ATT) to extract text features and image features, respectively, to extract more accurate features representing the original data[7]. The essence of VAE is to extract the hidden features of data and build a model from hidden features to the generated targets. This model eliminated noise interference in the textual data and extracted the more important image features. The feature-FUSION module based on a fine-grained attention mechanism is used to learn the modal FUSION features of the image and text information interactively. Therefore, this FUSION approach obtained better experimental results than CoMN(Co Memory Network).

2.6. Conclusion

Thus, we conclude the survey by apprising that various resources of dealing with text, audio and image were discussed and multiple ideologies were inferred from the resources. On a whole, we intend to develop the FUSION model by combining the individual models.

CHAPTER 3

SYSTEM ANALYSIS

3.1. Introduction

This chapter deals on discussions about the problem definition to clearly explain the problem statement and the ideology explored, existing system to define about the pre – existing models of the same ideology and the proposed system that elaborates the ideology of the system that is to be developed.

3.2. Problem Definition

The problem statement is “Sentiment Analysis on social media platforms to manage online reputation”. Thus, the requirement of this statement requires an AI-powered sentiment analysis tool that interprets emotions in social media posts, allowing people and organisations to successfully manage their online reputation. Therefore, this ideology is to work and develop the required model that encompasses on the main vision of multimodal sentimental analysis.

This will be helpful in various aspects like managing the reputation, influencer marketing, and content strategy optimisation, allowing organisations to communicate with their audiences in real time, modify communication plans, and maintain compliance with industry rules.

3.3. 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 combining the three modalities of text, audio and image together. The existing models work on the pairs of modalities such as text-audio, text-image, audio-text and so on. The developing model would be helpful in combining the three dimensions and analysing its features.

3.4. Proposed System

The proposed solution here involves the concept of multimodal sentimental analysis. Multimodal sentiment analysis refers to the process of analysing and understanding sentiment that involves multiple modes or types of data. The information that are in various modes of data are combined and represented in the form of multimodal representations. This is an area of artificial intelligence (AI) and machine learning that focuses on developing models capable of learning representations from data across multiple modalities.

Now the various modalities can be stated as textual data, image data and audio data. Thus, combining all these forms of data that is available in the social media are extracted and final multimodal sentiment analysis model is developed. For processing the textual data, the LSTM model (Long Short Term Memory) is implemented along with the natural language processing (NLP). For the audio data, the model is basically implemented by converting the audio data into textual data using speech recognition package and train a separate model. This is because upon further reference, we inferred that the semantic understanding of the audio is required rather than the emotional/expressional meaning of the audio.

For processing the image data This can be implemented by dealing with two crucial components such as recognizing text in image and image captioning.

By image captioning, the visual data can be converted into textual data and the LSTM model can be implemented in a separate process.

Different modalities of data are converted to a base modality. Here, Individual models are created for each modality in the base modality with the weighted approach (depending on popularity of the modality and input size). Therefore, An FUSION model is created which learns from three individual learners, along with boosting.

3.5. Conclusion

As we conclude, we have discussed the definition of the problem statement and the approach proposed for the statement is described and the efficiency is discussed elaborately.

CHAPTER 4

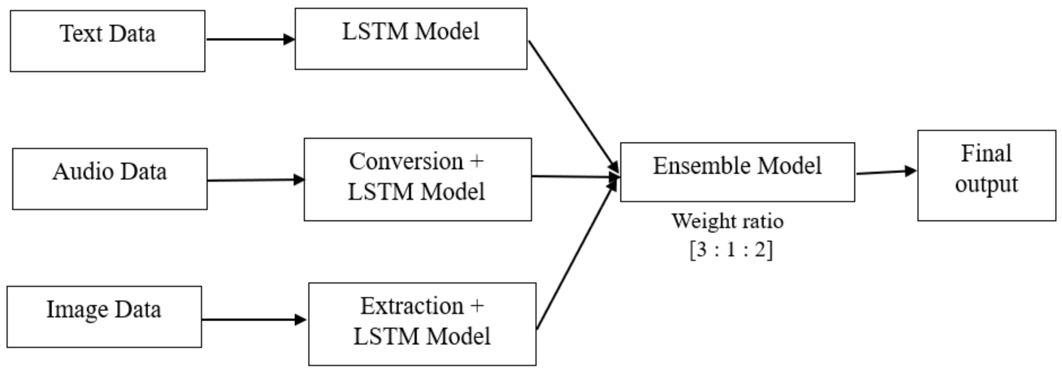
SYSTEM DESIGN

4.1. Introduction

Designing a system is similar to creating a plan for a big structure, with each aspect carefully created to complement the overall. In technology, system design is the art and science of building scalable, dependable, and efficient software structures to fulfil the different demands of both consumers and organisations. This chapter deals on dealing with various features such as system architecture, use case diagram and the data flow diagram.

4.2. 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 fulfil the system's requirements and objectives.

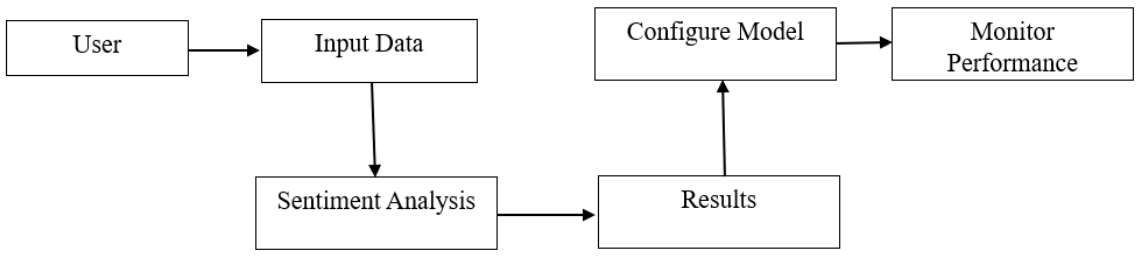


4.1. System Architecture

The basic architecture of the model begins with collecting the input from different modalities individually such as text data, audio data and image data and the data follows the respective model namely text data follows the LSTM model, audio data is converted into text using Speech Recognition and passed to its individual LSTM model and the image data is made into its text form either by EasyOCR or ViT – GPT and transmitted to the LSTM model. The outputs from individual model form an ensemble model which produces the final output based on the weighted approach.

4.3. Use case diagram

A use case diagram is a visual representation of the interactions and relationships within a system.

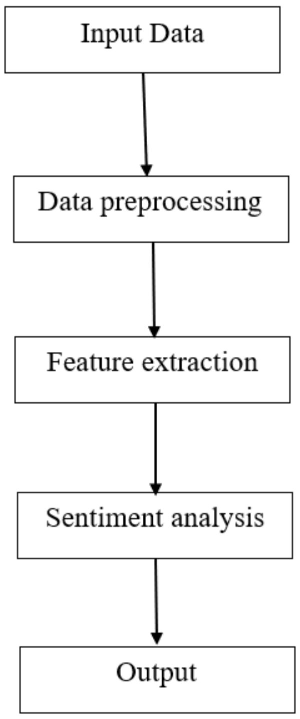


4.2. Use case diagram

The use case diagram depicted here gives the flow of the model in its individual pace. The same pattern follows for all the individual models implemented here such as text data model, audio data model and the image data model. The data is obtained from the user and the sentiment analysis is performed with the basic architecture flow and with the results obtained, the model can be configured and the performance can be monitored.

4.4. 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.



4.3. Data Flow Diagram

The data flow follows the same pattern of the whole model process. The data is obtained from the user and the basic preprocessing steps for different types of data is carried out. The features required for analysing the emotions is identified and the sentiment analysis is carried out by the model. Thus, the output is produced based on the weighted approach which is given based on popularity of the model and the input size.

4.5. Conclusion

Thus, the overall system architecture which describes the functioning of the model, the use diagram which employs the routing of the model and the data flow diagram which deals with the overall flow of the information passed is discussed and further developed.

CHAPTER 5

SYSTEM DESCRIPTION

5.1. Introduction

A system description is a section of a technical document or report that provides an overview of the system, its structure and components, and explains how it works. It may also provide information about related systems and technologies used in conjunction with the main system. This chapter describes the various packages and models that is used in implementing the model and enhancing its features.

5.2. PyTorch

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs. 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.

5.1.PyTorch

PyTorch is different from other deep learning frameworks in that it uses dynamic computation graphs. While static computational graphs are defined prior to runtime, dynamic graphs are defined "on the fly" via the forward computation. In other words, the graph is rebuilt from scratch on every iteration.

5.3. NLTK(NATURAL LANGUAGE TOOLKIT)

Natural language processing (NLP) is a field that focuses on 12 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.

5.4. 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.

5.5. 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.

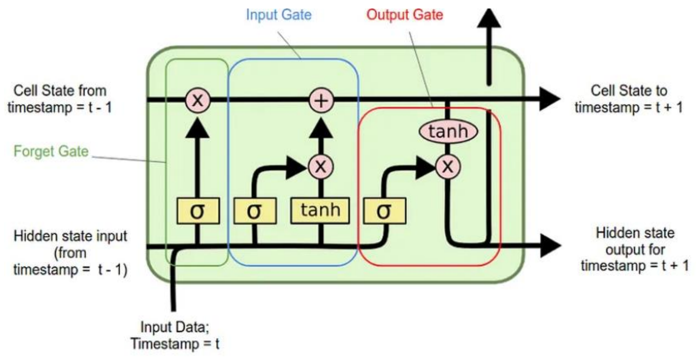
5.6. Torchtext

TorchText is a library in PyTorch specifically designed to simplify the process of handling text data and preparing it for use with deep learning models. It provides a set of easy-to-use functionalities to perform common tasks such as tokenization, vocabulary management, batching, and dataset handling for natural language processing (NLP) tasks.

5.2. TorchText

5.7. LSTM Model

An LSTM (Long Short-Term Memory) model is a type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem and capture long-range dependencies in sequential data.


Input Gate
Output Gate
Cell State from timestamp = t - 1
Forget Gate
Hidden state input (from timestamp = t - 1)
Input Data; Timestamp = t
Cell State to timestamp = t + 1
Hidden state output for timestamp = t + 1
σ
σ
tanh
σ
tanh
X
+
X
X

5.3. A single LSTM cell

LSTMs are particularly effective for tasks involving sequential or time-series data, such as natural language processing (NLP), speech recognition, and time series forecasting.

5.8. Conclusion

Thus, the various toolkits used in implementation such as Pytorch, Torchtext, Scikit-learn, NLTK and so on are described and their functionality is provided. A simple LSTM model which is the key part of the overall model is also explained.

CHAPTER 6

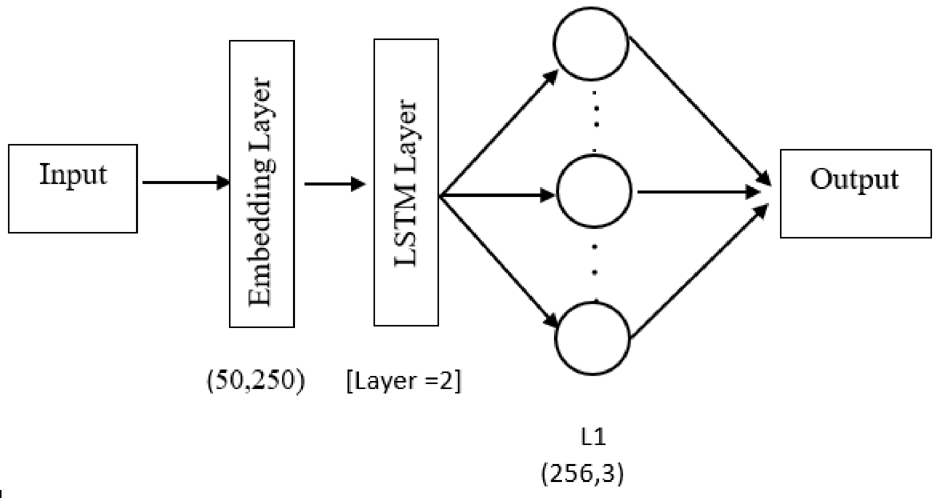
TEXT MODEL

6.1. Introduction

Textual data is information that is stored and written in a text format. It can be anything from emails to blog posts to social media posts and online forum comments. This model is implemented to analyse the emotions present in the textual data. This chapter discusses about the basic architecture, preprocessing and the final implementation of the model handling the text modality.

6.2. Basic Architecture

The textual data is pre-processed, split into training and testing datasets and then the architecture follows itself towards the LSTM model. Here, the implemented LSTM model consists of an embedding layer, 2 LSTM layers and a fully connected network (with 1 linear layer). This can be depicted as follows:



6.1. Basic Text LSTM Model

6.3. Preprocessing

Preprocessing text data is a crucial step in natural language processing (NLP) tasks, especially when preparing data for deep learning models like LSTMs. Proper preprocessing ensures that the text data is clean, normalized, and transformed into a format suitable for model training. The preprocessing steps that are carried out here can be listed as follows:

1. Text Cleaning
2. Tokenization
3. Removing Stop Words
4. Normalization
5. Handling Numbers and Special Characters
6. Handling Out-of-Vocabulary (OOV) Tokens
7. Padding and Truncation
8. Encoding Text
9. Data Splitting

6.4. Implementation

An LSTM (Long Short-Term Memory) model for textual data sentiment analysis is a type of deep learning architecture specifically designed to analyze and classify the sentiment expressed in text data. Sentiment analysis aims to determine the emotional tone or polarity (positive, negative, neutral) conveyed by a piece of text.

The input to the LSTM model consists of sequences of words or tokens representing the textual data. Each word or token is typically converted into a numerical representation and then proceeded to the embedding layer. This embedding layer converts each word/token into dense vectors of fixed size. And captures semantic relationships between words and represents them in a continuous vector space.

The core of the LSTM model consists of multiple LSTM layers. Each LSTM layer contains memory cells that can maintain information over time and learn long-range dependencies in the input sequences. Here, 2 LSTM layers are used. The LSTM layers process the embedded sequences of tokens, retaining relevant contextual information and discarding irrelevant details. The output of each LSTM layer at each time step is the hidden state, which captures the current state of the input sequence.

The final hidden state output from the last LSTM layer is typically used as input to a fully connected (dense) output layer. In this model, the fully connected network consists of 1 linear layer. The output layer uses a softmax activation function to compute the probability distribution over the predefined sentiment classes (e.g., positive, negative, neutral). The sentiment class with the highest probability score is predicted as the sentiment label for the input text.

6.5. Conclusion

Thus, the model for handling the textual data is implemented using the LSTM model that is properly processed and the emotions underlying the textual data is derived from the model.

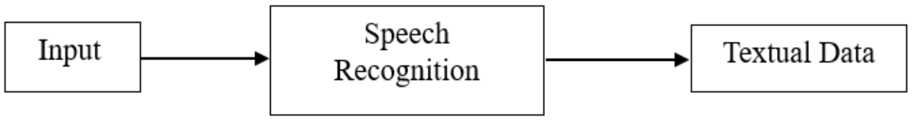
CHAPTER 7
AUDIO MODEL

7.1. Introduction

Audio data refers to the representation of sound as a set of electrical impulses or digital data. It involves the process of converting sound waves into an electrical signal that can be stored, transferred, or processed electronically. This chapter discusses about the basic architecture of the model handling the audio modality, preprocessing and the final implementation of the model.

7.2. Basic Architecture

The basic architecture involves dealing with the conversion of audio data into textual data and the same LSTM model is followed. The sentiment underlying the textual form of the data is provided.



7.1. Basic Audio Architecture

7.3. Preprocessing

The preprocessing refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task. The basic preprocessing for audio such as removing noise, filtering, noise reduction and feature extraction is carried out before the implementation of the model.

7.4. Implementation

Upon further inference, we infer that the semantic understanding of the audio is required rather than the emotional/expressional meaning of the audio. For instance, a person might provide negative criticism for a product but not in a negative emotion. For this reason, we decided to convert the audio data into textual data using speech recognition package and train a separate model. This is done using the Speech Recognition package. The Speech Recognition module, often referred to as SpeechRecognition, is a library that allows Python developers to convert spoken language into text by utilizing various speech recognition engines and APIs. Thus, the audio data is converted into textual data and the LSTM model is used to provide the sentiment analysis of the data.

7.5. Conclusion

Thus, the model for dealing with the audio data is implemented by converting the audio form into textual form and the sentiment analysis is derived from the model effectively.

CHAPTER 8

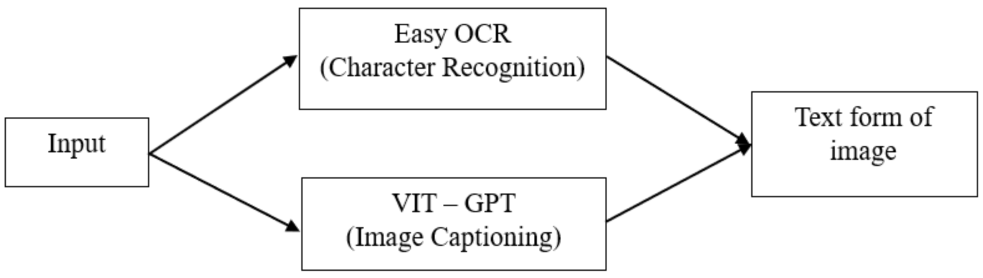
IMAGE MODEL

8.1. Introduction

Image data refers to digital information that represents images. These images can be photographs, graphics, scanned documents, or any other visual content captured or created in digital format. This chapter provides about the overall view regarding the basic architecture, preprocessing and the implementation of the image model.

8.2. Basic Architecture

The architecture of the model that is used to deal with the image data and perform the sentimental analysis consists of two main components: Easy OCR which is used for character recognition and VIT – GPT for image captioning. Thus, this allows us to convert the image into text as well infer the text present in the image. The structure can be depicted as follows:


Input
Easy OCR (Character Recognition)
VIT – GPT (Image Captioning)
Text form of image

8.1. Image Model

8.3. Preprocessing

The basic preprocessing can be carried out if the image is not clear or blurred out such as sharpening out the image, smoothening and extracting the features from the image.

8.4. Implementation

The image data is critical when dealing and processing with the models. Thus, the individual model is developed in order to convert the image data into text and then the LSTM model can be followed. This is done because processing the image directly and developing the model needs more hardware requirements and increases the time complexity. Therefore, the ideology of converting the data and bringing it to the uniformity is proposed.

This consists of two components: One to recognize the characters present in the image and other one to understand the content of the images. For recognizing the characters, the EasyOCR (Optical Character Recognition) is used. EasyOCR is a Python library for Optical Character Recognition (OCR) that allows you to easily extract text from images and scanned documents. And for understanding the content, VIL – GPT is used. The Vision Transformer (ViT) with a Generative Pretrained Transformer (GPT) model is designed to understand the content of images and generate descriptive captions using the power of computer vision and natural language processing. The ViT part of the model is responsible for processing and encoding visual information from images, while the GPT-2 part generates human-like text based on that visual information.

After obtaining the textual form of the image, the LSTM model is used in the same way but it is trained individually. The implemented LSTM model consists of a single embedding layer, 2 LSTM layers and a fully connected network (with 1 linear layer) that helps in understand the underlying emotions of the textual data.

8.5. Conclusion

Thus, the individual model for dealing with the image data is developing by using the ViT – GPT and Easy OCR to obtain the textual form from the image and the LSTM model is used to understand the underlying emotions in the textual data efficiently.

CHAPTER 9

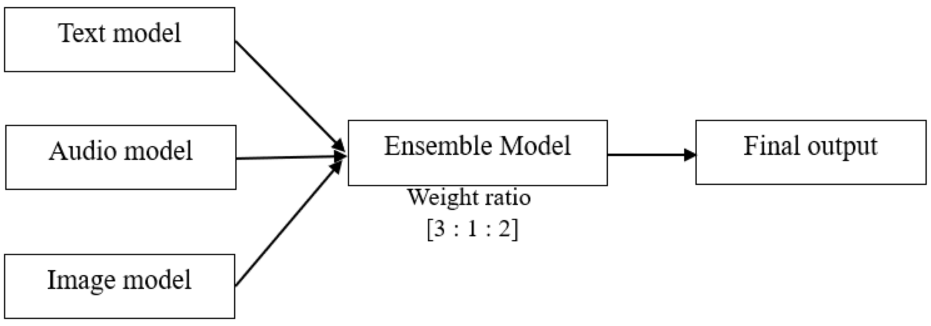
ENSEMBLE MODEL

9.1. Introduction

The ensemble model can be described as the combination of multiple models like the models used here such as text model, audio model and image model. All these models are combined and gives raise to the uniform output. This chapter provides the overview about the architecture and the implementation of the ensemble model.

9.2. Basic Architecture

The basic architecture involves combining the individual text, audio and image models and combining them in order to provide the final output. This architecture can be depicted as follows:



9.1. Ensemble Model Architecture

9.3. Implementation

First of all, the inputs from different modalities are collected individually. The pure LSTM model along with the linear layers is used to understand and analysing the sentiment underlying in the textual form. The audio data is converted using Speech Recognition Package and the LSTM model is trained individually for the audio data. For image data, the EasyOCR and the ViT – GPT is used for recovering the text from the image and the LSTM model is trained separately for the image data.

Now, different modalities have been converted to a base modality. Here, Individual models are created for each modality in the base modality with the weighted approach (depending on popularity of the modality and input size). Therefore, An Ensemble model is created which learns from three individual learners, along with boosting.

9.4. Conclusion

Thus, the ensemble model is created by combining the individual learner models and provides the output with the weighted approach which depends on the hike of the modality and the size of the input. Therefore, the final model is created and the output is obtained in an effective way.

CHAPTER 10
PERFORMANCE AND EVALUATION

Now, the created multimodal analysis tool using PyTorch provides about 84% accuracy in the results. The overall performance is good and the model is not too slow to provide the output. The training requires a lot of time but the test validation is in a much better pace. This accuracy can also be improved by training the model in better GPU (Graphics Processing Units) systems. With the hardware limitations, this much better result is achieved. The performance can also be improved by using the concept of transformers. Transformers are a type of neural network architecture that has revolutionized the field of deep learning, particularly in natural language processing (NLP). Transformers are a powerful tool in deep learning due to their ability to efficiently process and learn from sequential data, their scalability, and their capacity for parallel computation, which collectively contribute to their effectiveness and widespread adoption in various applications. Thus, the performance of the multimodal sentiment analysis model is evaluated, validated and the results reflect the perfect working of the model.

CHAPTER 11
CONCLUSION AND FUTURE WORK

Thus, this AI enabled analytical tool to interpret social media content to manage online reputation helps in analysing and optimizing the trends in the social media and helps in managing the reputation as well as profitable marketing. This AI enabled analytical tool developed to evaluate social media content and manage online reputation has several benefits. The sentiment analysis component assesses public opinion, analysing the trends and detects major brand references and influencers, providing information for focused engagement and keeps the optimized management of reputation on the web-based applications. Regarding the future work, the LSTM models can be replaced with that of transformers that would help in improving the accuracy and performance of the model. A transformer is a deep learning model that uses the attention mechanism to differentiate the importance of each part of the input data. Thus, the progress of the model could be noted drastically.

BIBLIOGRAPHY

1. Girdhar, R., El-Nouby, A., Liu, Z., Singh, M., Alwala, K. V., Joulin, A., & Misra, I. (2023). Imagebind: One embedding space to bind them all. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 15180-15190).

2. Wang, P., Yang, A., Men, R., Lin, J., Bai, S., Li, Z., ... & Yang, H. (2022, June). Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In International Conference on Machine Learning (pp. 23318-23340). PMLR.

3. Li, J., Li, D., Xiong, C., & Hoi, S. (2022, June). Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In International Conference on Machine Learning (pp. 12888-12900). PMLR.

4. Cho, J., Lei, J., Tan, H., & Bansal, M. (2021, July). Unifying vision-and-language tasks via text generation. In International Conference on Machine Learning (pp. 1931-1942). PMLR

5. Wang, Z., Yu, J., Yu, A. W., Dai, Z., Tsvetkov, Y., & Cao, Y. (2021). Simvlm: Simple visual language model pretraining with weak supervision. arXiv preprint arXiv:2108.10904.

6. Zhang, K., Geng, Y., Zhao, J., Liu, J., & Li, W. (2020). Sentiment analysis of social media via multimodal feature FUSION. Symmetry, 12(12), 2010.

7. Yoon, S., Byun, S., & Jung, K. (2018, December). Multimodal speech emotion recognition using audio and text. In 2018 IEEE Spoken Language Technology Workshop (SLT) (pp. 112-118). IEEE.

8. Poria, S., Cambria, E., Howard, N., Huang, G. B., & Hussain, A. (2016). Fusing audio, visual and textual clues for sentiment analysis from multimodal content. Neurocomputing, 174, 50-59.

9. Kiros, R., Salakhutdinov, R., & Zemel, R. S. (2014). Unifying visual-semantic embeddings with multimodal neural language models. arXiv preprint arXiv:1411.2539.

APPENDIX

a) SOURCE CODE

Textmodel.ipynb

```
import pandas as pd
import codecs
import torch
import torch.nn.functional as F
import torchtext
import time
import random
import pandas as pd

torch.backends.cudnn.deterministic = True

with codecs.open('/Users/aakaash_kb/Developer/Final
Project/Dataset/yelp_review_full_csv/train.csv', 'r', encoding='ISO-8859-1')
as f:
    df_train = pd.read_csv(f)
with codecs.open('/Users/aakaash_kb/Developer/Final
Project/Dataset/yelp_review_full_csv/test.csv', 'r', encoding='ISO-8859-1') as
f:
    df_test = pd.read_csv(f)


df_train.columns = ["Label", "Text"]
df_train = df_train[["Text", "Label"]].copy()

df_test.columns = ["Label", "Text"]
df_test = df_test[["Text", "Label"]].copy()

df_train = df_train[["Text", "Label"]].copy()
df_test = df_test[["Text", "Label"]].copy()

df_train = df_train.loc[df_train.Label.isin([1, 3, 5])]
df_all_train = df_train.copy()
df_train = df_train[:30000]
df_train = df_train.replace([1,3,5],[0,1,2])
print(df_train['Label'].value_counts())

df_test = df_test.loc[df_test.Label.isin([1, 3, 5])]
df_test = df_test
df_test = df_test.replace([1,3,5],[0,1,2])
print(df_test['Label'].value_counts())
```

```
X_train, y_train = df_train.Text, df_train.Label
X_test, y_test = df_test.Text, df_test.Label

from string import punctuation

from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

def pre_process(df):
    all_reviews=list()
    for text in df.Text:
        text = text.lower()
        text = "".join([ch for ch in text if ch not in punctuation])
        text = " ".join([c for c in text.split() if c not in stop_words and
c.isalpha()])
        all_reviews.append(text)
    all_text = " ".join(all_reviews)
    all_words = all_text.split()
    return all_reviews, all_words


def create_vocab(words):
    from collections import Counter
    # Count all the words using Counter Method
    count_words = Counter(words)
    total_words=len(words)
    sorted_words=count_words.most_common(total_words)
    print(type(sorted_words))
    print("Top ten occuring words : ",sorted_words[:10])

    vocab_to_int={w:i+1 for i,(w,c) in enumerate(sorted_words)}
    print(len(vocab_to_int))

    return vocab_to_int

def encode_features(reviews, vocab):
    encoded_reviews=list()
    for review in reviews:
      encoded_review=list()
      for word in review.split():
        if word not in vocab.keys():
          #if word is not available in vocab_to_int put 0 in that place
          encoded_review.append(0)
        else:
          encoded_review.append(vocab[word])
      encoded_reviews.append(encoded_review)
    return encoded_reviews
```

```
import numpy as np
def pad_features(encoded_reviews):
    sequence_length=150
    features=np.zeros((len(encoded_reviews), sequence_length), dtype=int)
    for i, review in enumerate(encoded_reviews):
        review_len=len(review)
        if (review_len<=sequence_length):
            zeros=list(np.zeros(sequence_length-review_len))
            new=zeros+review
        else:
            new=review[:sequence_length]
        features[i,:]=np.array(new)
    return features

reviews, words = pre_process(df_all_train)
train_reviews, train_words = pre_process(df_train)
train_vocab = create_vocab(words)
train_vocab = dict(list(train_vocab.items())[:20000])
#print(train_vocab)
train_encoded_features = encode_features(train_reviews, train_vocab)
train_features = pad_features(train_encoded_features)


from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(train_features,
df_train.Label, test_size=0.2, shuffle=True)


test_reviews, test_words = pre_process(df_test)
test_encoded_features = encode_features(test_reviews, train_vocab)
test_features = pad_features(test_encoded_features)
X_test, y_test = test_features[:29950], df_test.Label[:29950]
print(len(X_test), len(y_test))
print(X_test.shape, y_test.shape)


import torch
from torch.utils.data import DataLoader, TensorDataset


#create Tensor Dataset
train_data=TensorDataset(torch.FloatTensor(X_train),
torch.FloatTensor(list(y_train)))
valid_data=TensorDataset(torch.FloatTensor(X_val),
torch.FloatTensor(list(y_val)))
test_data=TensorDataset(torch.FloatTensor(X_test),
torch.FloatTensor(list(y_test)))
```

```
#dataloader
batch_size=50
train_loader=DataLoader(train_data, batch_size=batch_size, shuffle=True)
valid_loader=DataLoader(valid_data, batch_size=batch_size, shuffle=True)
test_loader=DataLoader(test_data, batch_size=batch_size, shuffle=True)


dataiter = iter(train_loader)
sample_x, sample_y = next(dataiter)


import torch.nn as nn

class LSTM0(nn.Module):
    def __init__(self, vocab_size, output_size, embedding_dim, hidden_dim,
n_layers):
        super().__init__()
        self.output_size=output_size
        self.n_layers=n_layers
        self.hidden_dim=hidden_dim

        #Embedding and LSTM layers
        self.embedding=nn.Embedding(vocab_size, embedding_dim)
        self.lstm=nn.LSTM(embedding_dim, hidden_dim, n_layers,
batch_first=True)

        #Linear and sigmoid layer
        self.fc1=nn.Linear(hidden_dim, output_size)


    def forward(self, x, hidden):
        batch_size=x.size()

        #Embadding and LSTM output
        x = x.long()
        embedd=self.embedding(x)
        lstm_out, hidden=self.lstm(embedd, hidden)

        #stack up the lstm output
        lstm_out=lstm_out.contiguous().view(-1, self.hidden_dim)

        #dropout and fully connected layers
        out=self.fc1(lstm_out)
        soft_out = out.reshape(list(batch_size)+[3])
        soft_out = soft_out[:, -1, :]
        return soft_out, hidden

    def init_hidden(self, batch_size, device):
        """Initialize Hidden STATE"""
        # Create two new tensors with sizes n_layers x batch_size x
hidden_dim,
        # initialized to zero, for hidden state and cell state of LSTM
        weight = next(self.parameters()).data


        hidden = (weight.new(self.n_layers, batch_size,
self.hidden_dim).zero_().to(device),
                  weight.new(self.n_layers, batch_size,
self.hidden_dim).zero_().to(device))

        return hidden

torch.manual_seed(42)
vocab_size = len(train_vocab)+1 # +1 for the 0 padding
output_size = 3
embedding_dim = 400
hidden_dim = 256
n_layers = 2

model = LSTM0(vocab_size, output_size, embedding_dim, hidden_dim, n_layers)
print(model)

# loss and optimization functions
lr=0.001

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=lr)

# check if CUDA is available
# device = "mps" if torch.backends.mps.is_built() else "cpu"
device = "cpu"

# training params

epochs = 3 # 3-4 is approx where I noticed the validation loss stop decreasing

counter = 0
print_every = 100
clip=5 # gradient clipping

# move model to GPU, if available
model.to(device)

model.train()
# train for some number of epochs
for e in range(epochs):
    # initialize hidden state
    h = model.init_hidden(batch_size=batch_size, device=device)

    # batch loop
    for inputs, labels in train_loader:
        counter += 1
        print(counter, end=",")

        inputs=inputs.to(device)
        labels=labels.to(device)

        # Creating new variables for the hidden state, otherwise
        # we'd backprop through the entire training history
        h = tuple([each.data for each in h])

        # zero accumulated gradients
        model.zero_grad()

        # get the output from the model
        output, h = model(inputs, h)

        # calculate the loss and perform backprop
        loss = criterion(output.squeeze(), labels.long())
        loss.backward()

        # `clip_grad_norm` helps prevent the exploding gradient problem in
RNNs / LSTMs.
        nn.utils.clip_grad_norm_(model.parameters(), clip)
        optimizer.step()

        # loss stats
        if counter % print_every == 0:
            # Get validation loss
            val_h = model.init_hidden(batch_size, device)
            val_losses = []
            model.eval()
            for inputs, labels in valid_loader:

                # Creating new variables for the hidden state, otherwise
                # we'd backprop through the entire training history
                val_h = tuple([each.data for each in val_h])

                inputs, labels = inputs.to(device), labels.to(device)
                output, val_h = model(inputs, val_h)
                val_loss = criterion(output.squeeze(), labels.long())

                val_losses.append(val_loss.item())
            model.train()
            print("\n\nEpoch: {}/{}...".format(e+1, epochs),
                  "Step: {}...".format(counter),
                  "Loss: {:.6f}...".format(loss.item()),
                  "Val Loss: {:.6f}".format(np.mean(val_losses)))


model.load_state_dict(torch.load("/Users/aakaash_kb/Developer/Final
Project/demo/updated_model/lstm_04.pth"))


test_losses = [] # track loss
num_correct = 0

# init hidden state
h = model.init_hidden(batch_size, device)

model.eval()
# iterate over test data
for inputs, labels in test_loader:

    # Creating new variables for the hidden state, otherwise
    # we'd backprop through the entire training history
    h = tuple([each.data for each in h])

    inputs, labels = inputs.to(device), labels.to(device)


    output, h = model(inputs, h)

    # calculate loss
    test_loss = criterion(output.squeeze(), labels.long())
    test_losses.append(test_loss.item())

    # convert output probabilities to predicted class (0 or 1)
    pred = output.argmax(dim=1)  # rounds to the nearest integer

    # compare predictions to true label
    correct_tensor = pred.eq(labels.float().view_as(pred))
    correct = np.squeeze(correct_tensor.cpu().numpy())
    num_correct += np.sum(correct)


# -- stats! -- ##
# avg test loss
print("Test loss: {:.3f}".format(np.mean(test_losses)))

# accuracy over all test data
test_acc = num_correct/len(test_loader.dataset)
```

```
print("Test accuracy: {:.3f}".format(test_acc))

print("Test loss: {:.3f}".format(np.mean(test_losses)))

# accuracy over all test data
test_acc = num_correct/len(test_loader.dataset)
print("Test accuracy: {:.3f}".format(test_acc))
```

demo.ipynb

```
import pandas as pd
import codecs
import torch
import torch.nn.functional as F
import torchtext
import time
import random
import pandas as pd
from string import punctuation
import numpy as np
import torch.nn as nn
from torch.utils.data import DataLoader, TensorDataset
import speech_recognition as sr
import os
from transformers import VisionEncoderDecoderModel, ViTFeatureExtractor,
AutoTokenizer
from PIL import Image
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
import easyocr

torch.backends.cudnn.deterministic = True
```

Text Data

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))


def pre_process(df):
    all_reviews=list()
    for text in df:
        text = text.lower()
        text = "".join([ch for ch in text if ch not in punctuation])
        text = " ".join([c for c in text.split() if c not in stop_words and
c.isalpha()])
        all_reviews.append(text)
    all_text = " ".join(all_reviews)
    all_words = all_text.split()
    return all_reviews, all_words

def create_vocab(words):
    from collections import Counter
    # Count all the words using Counter Method
    count_words = Counter(words)
    total_words=len(words)
    sorted_words=count_words.most_common(total_words)
    print("Top ten occuring words : ",sorted_words[:10])

    vocab_to_int={w:i+1 for i,(w,c) in enumerate(sorted_words)}
    print(len(vocab_to_int))

    return vocab_to_int

def encode_features(reviews, vocab):
    encoded_reviews=list()
    for review in reviews:
      encoded_review=list()
      for word in review.split():
        if word not in vocab.keys():
          #if word is not available in vocab_to_int put 0 in that place
          encoded_review.append(0)
        else:
          encoded_review.append(vocab[word])
      encoded_reviews.append(encoded_review)
    return encoded_reviews


def pad_features(encoded_reviews):
    sequence_length=150
    features=np.zeros((len(encoded_reviews), sequence_length), dtype=int)
    for i, review in enumerate(encoded_reviews):
        review_len=len(review)
        if (review_len<=sequence_length):
            zeros=list(np.zeros(sequence_length-review_len))
            new=zeros+review
        else:
            new=review[:sequence_length]
        features[i,:]=np.array(new)
    return features
```

```
with codecs.open('/Users/aakaash_kb/Developer/Final
Project/Dataset/yelp_review_full_csv/train.csv', 'r', encoding='ISO-8859-1')
as f:
    df_train = pd.read_csv(f)

df_train.columns = ["Label", "Text"]
df_train = df_train.loc[df_train.Label.isin([1, 3, 5])]
df = df_train.Text.tolist()

reviews, words = pre_process(df)
vocab = create_vocab(words)
vocab = dict(list(vocab.items())[:20000])
with codecs.open('/Users/aakaash_kb/Developer/Final Project/demo/reviews.csv',
'r', encoding='ISO-8859-1') as f:
    df_test = pd.read_csv(f)

df_test = df_test["reviews"][:50].tolist()

test_encode = encode_features(df_test, vocab)
test_features = pad_features(test_encode)


class LSTM0(nn.Module):
    def __init__(self, vocab_size, output_size, embedding_dim, hidden_dim,
n_layers):
        super().__init__()
        self.output_size=output_size
        self.n_layers=n_layers
        self.hidden_dim=hidden_dim

        #Embedding and LSTM layers
        self.embedding=nn.Embedding(vocab_size, embedding_dim)
        self.lstm=nn.LSTM(embedding_dim, hidden_dim, n_layers,
batch_first=True)

        #Linear and sigmoid layer
        self.fc1=nn.Linear(hidden_dim, output_size)


    def forward(self, x, hidden):
        batch_size=x.size()

        #Embedding and LSTM output
        x = x.long()
        embedd=self.embedding(x)
        lstm_out, hidden=self.lstm(embedd, hidden)

        #stack up the lstm output
        lstm_out=lstm_out.contiguous().view(-1, self.hidden_dim)

        #dropout and fully connected layers
        out=self.fc1(lstm_out)
        soft_out = out.reshape(list(batch_size)+[3])
        soft_out = soft_out[:, -1, :]
        return soft_out, hidden

    def init_hidden(self, batch_size, device):
        """Initialize Hidden STATE"""
        # Create two new tensors with sizes n_layers x batch_size x
hidden_dim,
        # initialized to zero, for hidden state and cell state of LSTM
        weight = next(self.parameters()).data


        hidden = (weight.new(self.n_layers, batch_size,
self.hidden_dim).zero_().to(device),
                  weight.new(self.n_layers, batch_size,
self.hidden_dim).zero_().to(device))

        return hidden


torch.manual_seed(42)
vocab_size = 20000+1 # +1 for the 0 padding
output_size = 3
embedding_dim = 400
hidden_dim = 256
n_layers = 2

model = LSTM0(vocab_size, output_size, embedding_dim, hidden_dim, n_layers)
print(model)


model.load_state_dict(torch.load("/Users/aakaash_kb/Developer/Final
Project/demo/updated_model/lstm_04.pth"))

inp_features = torch.FloatTensor(test_features)
model.eval()
device="cpu"
batch_size = 50
h = model.init_hidden(batch_size, device)
sm = torch.nn.Softmax(dim=1)
with torch.inference_mode():
    h = tuple([each.data for each in h])
    pred, h = model(inp_features, h)
    res = sm(pred).argmax(dim=1).numpy()
    print(res)

text_op = {"Negative Response":0,"Neutral Response":0,"Positive Response":0}
for i in list(res):
    if i == 0:
        text_op["Negative Response"] += 1
    elif i == 1:
        text_op["Neutral Response"] += 1
    else:
        text_op["Positive Response"] += 1
print(text_op)
```

Audio Data

```
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}")

# Path to the directory containing audio files
directory_path = "/Users/aakaash_kb/Developer/Final Project/demo/audio data"

audio_semantics = []

audio_extensions = [".mp3", ".wav", ".ogg", ".flac"]
for filename in os.listdir(directory_path):
    if any(filename.endswith(ext) for ext in audio_extensions):
        full_path = os.path.join(directory_path, filename)
        print(full_path)
        semantic = convert_audio_to_text(full_path)
        audio_semantics.append(semantic)

print(audio_semantics)


audio_encode = encode_features(audio_semantics, vocab)
audio_features = pad_features(audio_encode)
```

```
inp_features = torch.FloatTensor(audio_features)
model.eval()
device="cpu"
batch_size = 50
h = model.init_hidden(batch_size, device)
sm = torch.nn.Softmax(dim=1)
with torch.inference_mode():
    h = tuple([each.data for each in h])
    pred, h = model(inp_features, h)
    ares = sm(pred).argmax(dim=1).numpy()
    print(ares)

audio_op = {"Negative Response":0,"Neutral Response":0,"Positive Response":0}
for i in list(ares):
    if i == 0:
        audio_op["Negative Response"] += 1
    elif i == 1:
        audio_op["Neutral Response"] += 1
    else:
        audio_op["Positive Response"] += 1
print(audio_op)
```

Image Data

```
model_dir = "./caption_model"
caption_model = VisionEncoderDecoderModel.from_pretrained(model_dir)
cfeature_extractor = ViTFeatureExtractor.from_pretrained(model_dir)
ctokenizer = AutoTokenizer.from_pretrained(model_dir)
device = torch.device("mps" if torch.cuda.is_available() else "cpu")
caption_model.to(device)
max_length = 16
num_beams = 4
gen_kwargs = {"max_length": max_length, "num_beams": num_beams}

def predict_step(image_paths):
  images = []
  for image_path in image_paths:
    i_image = Image.open(image_path)
    if i_image.mode != "RGB":
      i_image = i_image.convert(mode="RGB")

    images.append(i_image)

  pixel_values = cfeature_extractor(images=images,
return_tensors="pt").pixel_values
  pixel_values = pixel_values.to(device)
  output_ids = caption_model.generate(pixel_values, **gen_kwargs)

  preds = ctokenizer.batch_decode(output_ids, skip_special_tokens=True)
  preds = [pred.strip() for pred in preds]
  return preds


directory_path = "/Users/aakaash_kb/Developer/Final Project/demo/image data"

img_paths = []
ocrs = []
img_extensions = [".jpeg", ".jpg", ".png"]
for filename in os.listdir(directory_path):
    if any(filename.endswith(ext) for ext in img_extensions):
        full_path = os.path.join(directory_path, filename)
        reader = easyocr.Reader(['en'])
        result = reader.readtext(full_path)
        if len(result) != 0:
            ocrs.append(result[0][1])
        else:
            ocrs.append("")
        img_paths.append(full_path)
captions = predict_step(img_paths)


image_semantics = []
for i in range(len(captions)):
    iop = captions[i] + ". " + ocrs[i]
    image_semantics.append(iop)
image_semantics


image_encode = encode_features(image_semantics, vocab)
image_features = pad_features(image_encode)

inp_features = torch.FloatTensor(image_features)
model.eval()
device="cpu"
batch_size = 50
h = model.init_hidden(batch_size, device)
sm = torch.nn.Softmax(dim=1)
with torch.inference_mode():
    h = tuple([each.data for each in h])
    pred, h = model(inp_features, h)
    ires = sm(pred).argmax(dim=1).numpy()
    print(ires)
```

```
image_op = {"Negative Response":0,"Neutral Response":0,"Positive Response":0}
for i in list(ires):
    if i == 0:
        image_op["Negative Response"] += 1
    elif i == 1:
        image_op["Neutral Response"] += 1
    else:
        image_op["Positive Response"] += 1
print(image_op)
```

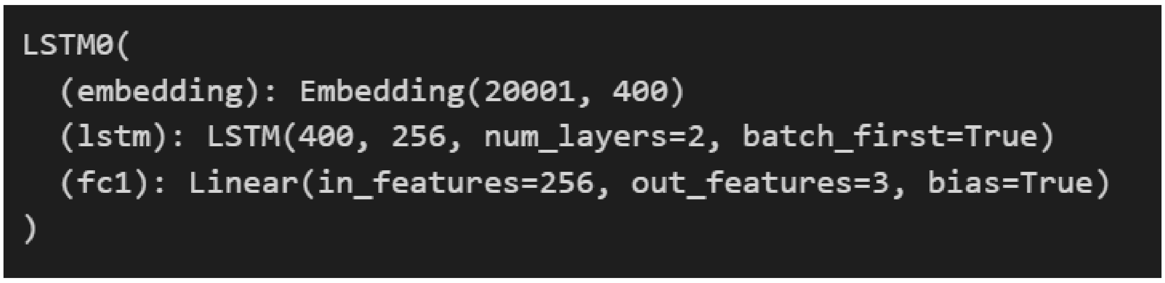
Embedded Final Result

```
def normalize_dict(dictionary):
    total = sum(dictionary.values())
    normalized_dict = {key: value / total for key, value in
dictionary.items()}
    return normalized_dict


def ensemble_dicts(text_op, audio_op, image_op, ratios):
    normalized_dict_text = normalize_dict(text_op)
    normalized_dict_audio = normalize_dict(audio_op)
    normalized_dict_image = normalize_dict(image_op)

    weighted_dict_text = {key: normalized_dict_text[key] * ratios[0] for key
in normalized_dict_text}
    weighted_dict_audio = {key: normalized_dict_audio[key] * ratios[1] for key
in normalized_dict_audio}
    weighted_dict_image = {key: normalized_dict_image[key] * ratios[2] for key
in normalized_dict_image}

    ensemble_output = {}
    for key in weighted_dict_text:
        ensemble_output[key] = weighted_dict_text[key] +
weighted_dict_audio[key] + weighted_dict_image[key]

    total_sum = sum(ensemble_output.values())
    ensemble_output_percentage = {key: round((value / total_sum) * 100, 2) for
key, value in ensemble_output.items()}

    return ensemble_output_percentage

ratios = [3, 1, 2]

ensemble_op = ensemble_dicts(text_op, audio_op, image_op, ratios)
```

```
def repr(d):
    for i in d:
        print(i+" : "+str(d[i])+"%", end="  |  ")
    print()


print("-"*90)
print("\nTextual Data Summary:")
repr(text_op)
print("-"*90)
print("\nAudio Data Summary:")
repr(audio_op)
print("-"*90)
print("\nImage Data Summary:")
repr(image_op)
print("-"*90)
print("\nFinal Report:")
repr(ensemble_op)
print("-"*90)
```

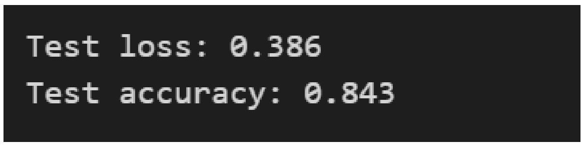
b) SCREENSHOT REFERENCES

LSTM Model:



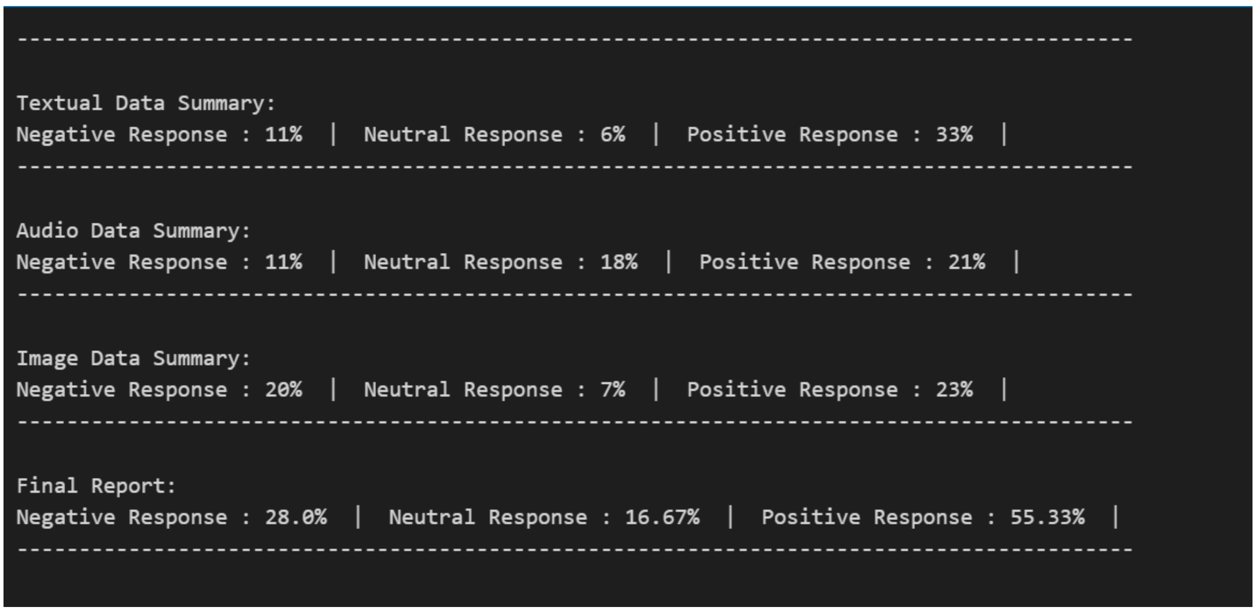
```
LSTM0(
  (embedding): Embedding(20001, 400)
  (lstm): LSTM(400, 256, num_layers=2, batch_first=True)
  (fc1): Linear(in_features=256, out_features=3, bias=True)
)
```

Model Evaluation:



```
Test loss: 0.386
Test accuracy: 0.843
```

Final Output:



```
-----------------------------------------------------------------------------------------

Textual Data Summary:
Negative Response : 11%  |  Neutral Response : 6%  |  Positive Response : 33%  |
-----------------------------------------------------------------------------------------

Audio Data Summary:
Negative Response : 11%  |  Neutral Response : 18%  |  Positive Response : 21%  |
-----------------------------------------------------------------------------------------

Image Data Summary:
Negative Response : 20%  |  Neutral Response : 7%  |  Positive Response : 23%  |
-----------------------------------------------------------------------------------------

Final Report:
Negative Response : 28.0%  |  Neutral Response : 16.67%  |  Positive Response : 55.33%  |
-----------------------------------------------------------------------------------------
```