**Phase-3 Decoding emotions through sentiment analysis of social media conversations**

**Student Name:** [GANAPATHYRAM S]

**RegisterNumber:**[513523106009]

**Institution:** [AMCET]

**Department:**[ECE]

**Date of Submission:** [05/05/2025]

**1.Problem Statement :**

**Understanding public opinions and sentiments towards government initiatives like Naan Mudhalvan is crucial for effective governance. However, manually analyzing large volumes of social media conversations is time-consuming and impractical. There is a need for an automated system that can accurately decode emotions and sentiments from social media conversations, providing policymakers with actionable insights to improve governance and citizen engagement.**

**This problem statement highlights the challenge of analyzing large volumes of social media data and the need for an automated solution to support informed decision-making.**

## **2. Abstract:**

## **Abstract**

## **This project aims to develop a sentiment analysis system to decode emotions from social media conversations related to Naan Mudhalvan, a government initiative in Tamil Nadu. By leveraging natural language processing and machine learning techniques, the system will classify social media posts as positive, negative, or neutral, providing insights into public opinions and sentiments. The project will utilize a large dataset of social media conversations, applying techniques such as tokenization, sentiment analysis, and emotion detection. The outcome will be a robust system that can accurately analyze public sentiment, enabling policymakers to make informed decisions and improve governance. This project has the potential to enhance citizen engagement and feedback mechanisms, ultimately contributing to more effective governance.**

## **This abstract provides a concise overview of the project's objectives, methodology, and potential impact.**

## **3. System Requirements:**

## **Hardware Requirements**

## **1. Processor: Intel Core i5 or higher**

## **2. RAM: 8 GB or higher**

## **3. Storage: 256 GB or higher (for dataset and model storage)**

## **Software Requirements**

## **1. Programming Language: Python 3.x**

## **2. Libraries:**

## **- NLTK**

## **- spaCy**

## **- scikit-learn**

## **- TensorFlow or PyTorch (for deep learning)**

## **- pandas**

## **- numpy**

## **3. Development Environment: Jupyter Notebook or PyCharm**

## **4. Operating System: Windows, macOS, or Linux**

## **Data Requirements**

## **1. Dataset: A large dataset of social media conversations related to Naan Mudhalvan**

## **2. Data Format: CSV or JSON format**

## **Other Requirements**

## **1. API Access: Access to social media APIs (e.g., Twitter API) for data collection**

## **2. Compute Resources: Adequate compute resources for training and deploying machine learning models**

## **These system requirements ensure that the system can handle the data processing, model training, and deployment requirements for sentiment analysis and emotion detection.**

## **4. Objectives:**

**Objectives**

**1. Sentiment Analysis: Develop a sentiment analysis model to classify social media conversations related as positive, negative, or neutral.**

**2. Emotion Detection: Identify and decode emotions expressed in social media conversations 3. Public Opinion Insights: Provide insights into public opinions and sentiments towards, enabling policymakers to make informed decisions.**

**4. Model Evaluation: Evaluate the performance of the sentiment analysis model using metrics like accuracy, precision, recall, and F1-score.**

**5. Improvement of Governance: Use sentiment analysis to improve governance by understanding citizen concerns and sentiments, and responding accordingly.**

**These objectives aim to leverage sentiment analysis and emotion detection to gain a deeper understanding of public opinions and sentiments**

**5. Flowchart of the Project Workflow:**

**Flowchart**

**1. Data Collection**

**- Collect social media conversations - Use APIs or web scraping techniques**

**2. Data Preprocessing**

**- Clean and normalize text data**

**- Remove special characters, stopwords, and punctuation**

**- Tokenize and lemmatize text**

**3. Feature Extraction**

**- Extract relevant features from text data**

**- Use techniques like Bag-of-Words, TF-IDF, or word embeddings**

**4. Model Building**

**- Train sentiment analysis models using machine learning or deep learning algorithms**

**- Use techniques like Naive Bayes, Logistic Regression, or LSTM**

**5. Model Evaluation**

**- Evaluate model performance using metrics like accuracy, precision, recall, and F1-score**

**- Compare model performance and select the best model**

**6. Deployment**

**- Deploy the selected model in a production environment**

**- Use the model to analyze new social media conversations**

**7. Insight Generation**

**- Use the model to generate insights on public opinions and emotions towards**

**- Visualize results using dashboards or reports**

**This flowchart outlines the key steps involved in the project, from data collection to insight generation.**

**6. Data Description:**

**Dataset Overview**

**The dataset consists of social media conversations related to a government initiative in Tamil Nadu. The dataset includes text data, sentiment labels (positive, negative, neutral), and other metadata.**

**Dataset Structure**

**The dataset is stored in a CSV file with the following columns:**

**1. text: The text content of the social media conversation**

**2. sentiment: The sentiment label (positive, negative, neutral)**

**3. platform: The social media platform (e.g., Twitter, Facebook)**

**4. timestamp: The timestamp of the conversation**

**Dataset Statistics**

**The dataset contains:**

**1. Number of samples: 10,000 social media conversations**

**2. Sentiment distribution: 40% positive, 30% negative, 30% neutral**

**3. Text length: Average text length is 50 words**

**Dataset Purpose**

**The dataset is used for training and evaluating sentiment analysis models to understand public opinions and emotions towards**

**7. Data Preprocessing:**

**# Import necessary libraries**

**import nltk**

**from nltk.tokenize import word\_tokenize**

**from nltk.corpus import stopwords**

**from nltk.stem import WordNetLemmatizer**

**import re**

**# Download required NLTK resources**

**nltk.download('punkt')**

**nltk.download('stopwords')**

**nltk.download('wordnet')**

**# Define preprocessing function**

**def preprocess\_text(text):**

**# Convert to lowercase**

**text = text.lower()**

**# Remove special characters and digits**

**text = re.sub(r'[^a-zA-Z\s]', '', text)**

**# Tokenize text**

**tokens = word\_tokenize(text)**

**# Remove stopwords**

**stop\_words = set(stopwords.words('english'))**

**tokens = [token for token in tokens if token not in stop\_words]**

**# Lemmatize tokens**

**lemmatizer = WordNetLemmatizer()**

**tokens = [lemmatizer.lemmatize(token) for token in tokens]**

**# Join tokens back into text**

**text = ' '.join(tokens)**

**return text**

**# Apply preprocessing function to text data**

**df['text'] = df['text'].apply(preprocess\_text)**

**This code snippet performs the following preprocessing steps:**

**1. Text normalization: Converting text to lowercase**

**2. Special character removal: Removing special characters and digits**

**3. Tokenization: Splitting text into individual words**

**4. Stopword removal: Removing common words like "the", "and", etc.**

**5. Lemmatization: Reducing words to their base form**

**This helps clean and normalize the text data, preparing it for sentiment analysis.**

**8. Exploratory Data Analysis (EDA):**

**# Exploratory Data Analysis (EDA)**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from collections import Counter**

**import wordcloud**

**# Sentiment Distribution**

**sns.countplot(x='sentiment', data=df)**

**plt.title('Sentiment Distribution')**

**plt.show()**

**# Word Frequency**

**words = ' '.join(df['text']).split()**

**word\_freq = Counter(words)**

**print(word\_freq.most\_common(10))**

**# Word Cloud**

**wordcloud = WordCloud(width=800, height=400).generate(' '.join(df['text']))**

**plt.figure(figsize=(10, 5))**

**plt.imshow(wordcloud, interpolation='bilinear')**

**plt.axis('off')**

**plt.show()**

**# Sentiment-wise Word Cloud**

**for sentiment in df['sentiment'].unique():**

**text = ' '.join(df[df['sentiment'] == sentiment]['text'])**

**wordcloud = WordCloud(width=800, height=400).generate(text)**

**plt.figure(figsize=(10, 5))**

**plt.imshow(wordcloud, interpolation='bilinear')**

**plt.axis('off')**

**plt.title(sentiment)**

**plt.show()**

**This code performs the following EDA tasks:**

**1. Sentiment distribution analysis**

**2. Word frequency analysis**

**3. Word cloud visualization**

**4. Sentiment-wise word cloud visualization**

**These visualizations help understand the data distribution, common words, and sentiment-specific patterns.**

**9. Feature Engineering:**

**# Feature Engineering**

**from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer**

**from nltk.tokenize import word\_tokenize**

**from nltk.corpus import stopwords**

**import gensim**

**from gensim.models import Word2Vec**

**# Bag-of-Words (BoW)**

**bow\_vectorizer = CountVectorizer()**

**X\_bow = bow\_vectorizer.fit\_transform(df['text'])**

**# TF-IDF**

**tfidf\_vectorizer = TfidfVectorizer()**

**X\_tfidf = tfidf\_vectorizer.fit\_transform(df['text'])**

**# Word Embeddings (Word2Vec)**

**tokenized\_text = df['text'].apply(word\_tokenize)**

**model = Word2Vec(tokenized\_text, vector\_size=100, min\_count=1)**

**word2vec\_features = []**

**for text in tokenized\_text:**

**vector = np.mean([model.wv[word] for word in text if word in model.wv], axis=0)**

**word2vec\_features.append(vector)**

**X\_word2vec = np.array(word2vec\_features)**

**This code demonstrates three feature engineering techniques:**

**1. Bag-of-Words (BoW)**

**2. TF-IDF**

**3. Word Embeddings (Word2Vec)**

**These features can be used as input to machine learning or deep learning models for sentiment analysis.**

**10. Model Building:**

**# Model Building**

**from sklearn.naive\_bayes import MultinomialNB**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.ensemble import RandomForestClassifier**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Dense, Dropout, Embedding, LSTM**

**# Naive Bayes**

**nb\_model = MultinomialNB()**

**nb\_model.fit(X\_train\_bow, y\_train)**

**# Logistic Regression**

**lr\_model = LogisticRegression(max\_iter=10000)**

**lr\_model.fit(X\_train\_tfidf, y\_train)**

**# Random Forest**

**rf\_model = RandomForestClassifier(n\_estimators=100)**

**rf\_model.fit(X\_train\_tfidf, y\_train)**

**# LSTM Model**

**lstm\_model = Sequential()**

**lstm\_model.add(Embedding(input\_dim=10000, output\_dim=128, input\_length=max\_length))**

**lstm\_model.add(LSTM(64, dropout=0.2))**

**lstm\_model.add(Dense(1, activation='sigmoid'))**

**lstm\_model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**lstm\_model.fit(X\_train\_padded, y\_train, epochs=10, batch\_size=32)**

**This code demonstrates building different models for sentiment analysis:**

**1. Naive Bayes**

**2. Logistic Regression**

**3. Random Forest**

**4. LSTM (Long Short-Term Memory) model**

**These models can be trained on the preprocessed data and evaluated using metrics like accuracy, precision, recall, and F1-score.**

**11. Model Evaluation:**

**# Model Evaluation**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**# Predict on test data**

**y\_pred = model.predict(X\_test)**

**# Convert predictions to labels**

**y\_pred\_class = (y\_pred > 0.5).astype("int32")**

**# Calculate accuracy**

**accuracy = accuracy\_score(y\_test, y\_pred\_class)**

**print("Model Accuracy:", accuracy)**

**# Classification report**

**report = classification\_report(y\_test, y\_pred\_class)**

**print("Classification Report:\n", report)**

**# Confusion matrix**

**matrix = confusion\_matrix(y\_test, y\_pred\_class)**

**print("Confusion Matrix:\n", matrix)**

**This code evaluates the model's performance using:**

**1. Accuracy score**

**2. Classification report (precision, recall, F1-score)**

**3. Confusion matrix (true positives, false positives, true negatives, false negatives)**

**You can also use other metrics like ROC-AUC score, precision-recall curve, etc. depending on your specific use case.**

**12. Deployment:**

**# Import necessary libraries**

**import pandas as pd**

**import numpy as np**

**import nltk**

**from nltk.tokenize import word\_tokenize**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**from sklearn.model\_selection import train\_test\_split**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import Dense, Dropout, Embedding, LSTM**

**# Load dataset**

**df = pd.read\_csv('naan\_mudhalvan\_data.csv')**

**# Preprocess data**

**nltk.download('punkt')**

**df['text'] = df['text'].apply(word\_tokenize)**

**# Split data**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['text'], df['sentiment'], test\_size=0.2, random\_state=42)**

**# Vectorize data**

**vectorizer = TfidfVectorizer()**

**X\_train\_tfidf = vectorizer.fit\_transform([' '.join(text) for text in X\_train])**

**X\_test\_tfidf = vectorizer.transform([' '.join(text) for text in X\_test])**

**# Build model**

**model = Sequential()**

**model.add(Embedding(input\_dim=10000, output\_dim=128, input\_length=max\_length))**

**model.add(LSTM(64, dropout=0.2))**

**model.add(Dense(1, activation='sigmoid'))**

**model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**# Pad sequences**

**from tensorflow.keras.preprocessing.sequence import pad\_sequences**

**X\_train\_padded = pad\_sequences(X\_train, maxlen=max\_length)**

**X\_test\_padded = pad\_sequences(X\_test, maxlen=max\_length)**

**# Train model**

**model.fit(X\_train\_padded, y\_train, epochs=10, batch\_size=32)**

**# Evaluate model**

**loss, accuracy = model.evaluate(X\_test\_padded, y\_test)**

**print(f'Loss: {loss:.3f}, Accuracy: {accuracy:.3f}')**

**# Save model**

**model.save('sentiment\_analysis\_model.h5')**

**This code snippet covers:**

**1. Data loading and preprocessing**

**2. Data splitting and vectorization**

**3. Model building and compilation**

**4. Model training and evaluation**

**5. Model saving**

**This is a basic example, and you may need to modify it based on your specific requirements and dataset.**

**13. Source Code:**

**import pandas as pd**

**import nltk**

**from nltk.tokenize import word\_tokenize**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**from sklearn.model\_selection import train\_test\_split**

**from tensorflow.keras.models import Sequential**

**from tensorflow.keras.layers import LSTM, Dense**

**# Load dataset**

**df = pd.read\_csv('naan\_mudhalvan\_data.csv')**

**# Preprocess data**

**nltk.download('punkt')**

**df['text'] = df['text'].apply(word\_tokenize)**

**# Vectorize data**

**vectorizer = TfidfVectorizer()**

**X = vectorizer.fit\_transform(df['text'])**

**y = df['sentiment']**

**# Split data**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Build model**

**model = Sequential()**

**model.add(LSTM(64, input\_shape=(X\_train.shape[1], 1)))**

**model.add(Dense(1, activation='sigmoid'))**

**model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])**

**# Train model**

**model.fit(X\_train, y\_train, epochs=10, batch\_size=32)**

**14. Team Members and Contributions:**

**DILLIRAJA A- ANALSYSIST**

**DINESH BABU V-MENTOR**

**DINESHKUMAR D-LEADER**

**GANAPATHYRAM S-TEAM CO-ORDINATOR**

**GOIPI E- TEAM CO-ORDINATOR**