

MOVIE RECOMMENDATION BASED ON EMOTION USING DEEP LEARNING

Team Members

REMINISETTI ALEKHYA	(20481A05J9)
THAMMARAPU ANIL KUMAR	(20481A05M9)
ROSHAN RISHI SRAVANAM	(20481A05K2)
TANURI GOPICHAND	(20481A05M4)

1. INTRODUCTION

1.1 Overview

Emotion-based movie recommendation systems aim to personalize movie suggestions for users by considering their emotional preferences. Traditional recommendation systems often rely on collaborative filtering or content-based methods, where movies are recommended based on user ratings or movie attributes, respectively. Emotion-based recommendation systems add another dimension by considering the emotional responses of users to movies.

Here's a hypothetical overview of how such a system might be designed:

1. **Data Collection:** The first step is to collect data, including movie reviews or ratings from users along with their emotional responses to the movies. Emotions can be collected in various ways, such as asking users to rate how a movie made them feel (e.g., happy, sad, excited) or by using sentiment analysis techniques to extract emotions from textual reviews.
2. **Feature Extraction:** Next, the system would extract relevant features from the movie data and user emotional responses. These features could include movie genres, actors, directors, as well as numerical representations of emotions.
3. **Deep Learning Model:** A deep learning model would be employed to learn patterns and relationships between movie features and user emotions. Popular deep learning models like neural networks or recurrent neural networks (RNNs) might be used for this purpose. The model will be trained on the collected data, trying to predict user emotions based on movie features.
4. **Recommendation Generation:** Once the deep learning model is trained, it can be used to predict how a user might emotionally respond to movies they haven't seen. The system then recommends movies that are expected to evoke the desired emotional responses in the user.
5. **Fine-Tuning:** The system can continuously improve its recommendations by gathering feedback from users on the recommended movies. User feedback allows the model to be fine-tuned and adapt its recommendations based on user preferences and emotional responses.

1.2 Purpose

The purpose of the "Movie Recommendation Based on Emotion Using Deep Learning" project is to develop a personalized and emotionally-aware movie recommendation system. The primary goal is to enhance the user experience by suggesting movies that are more likely to evoke the desired emotional responses in individual users.

Traditional movie recommendation systems often focus on collaborative filtering or content-based approaches, which may not consider the emotional preferences of users. Emotion-based movie recommendation systems aim to fill this gap by incorporating emotions as an additional dimension in the recommendation process.

The specific purposes and benefits of such a project may include:

1. **Personalization**: By considering user emotions, the recommendation system can provide more personalized movie suggestions. Different users may have different emotional preferences, and a one-size-fits-all approach may not be effective in capturing these individual differences.
2. **Improved User Engagement**: Emotions play a crucial role in how users connect with movies. By suggesting movies that align with a user's emotional preferences, the recommendation system can potentially increase user engagement and satisfaction.
3. **Novel Movie Discovery**: Emotion-based recommendations can introduce users to movies they might not have considered otherwise. Users might discover hidden gems or niche films that resonate with their emotional inclinations.
4. **Enhanced Movie Watching Experience**: When users watch movies that align with their emotions, they are more likely to have a fulfilling and enjoyable movie-watching experience.
5. **Increased User Retention**: Providing relevant and emotionally appealing movie recommendations can lead to higher user retention rates, as users are more likely to return to the platform for future movie suggestions.

2. LITERATURE SURVEY

2.1 Existing problem

"Movie Recommendation Based on Emotion Using Deep Learning" was an emerging area of research, and there were several challenges and existing problems associated with this approach. Some of these problems include:

1. ****Subjectivity of Emotions****: Emotions are subjective and can vary significantly from person to person. Different individuals might interpret and express emotions differently when watching the same movie. Capturing and representing such subjective emotional experiences accurately in a recommendation system is challenging.
2. ****Limited and Biased Emotional Data****: Collecting emotional data from users can be difficult and may result in limited and biased samples. Emotional responses could be influenced by various factors, such as cultural background, mood at the time of rating, or individual preferences, making the emotional dataset less diverse and potentially biased.
3. ****Emotion Classification Difficulty****: Automatically classifying emotions from textual reviews or ratings can be challenging. Emotions are complex, and sentiment analysis algorithms might not always accurately capture the nuanced emotional aspects of movie reviews.
4. ****Cold Start Problem****: Emotion-based recommendation systems might face a cold start problem, especially for new users who have not provided enough emotional feedback or ratings. Without sufficient emotional data, it becomes challenging to make accurate movie recommendations based on emotions.
5. ****Context Dependency****: Emotions are context-dependent, and a user's emotional response to a movie might change depending on the movie's genre, storyline, or their current life circumstances. The recommendation system needs to consider such context dependencies to make relevant and timely suggestions.

Researchers and developers in this field are actively working to address these problems, but it's essential to acknowledge and overcome these challenges to create effective and user-friendly emotion-based movie recommendation systems.

2.2 Proposed solution

- User interacts with the UI (User Interface) to enter the emotion
- Entered emotion is analyzed and the movie titles can be scraped from the IMDb list to recommend to the user
- The list of movies is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

- Necessary Steps
- Build HTML Pages
- Build the python flask app.
 - Import Necessary Libraries
 - Creating our flask application
 - Routing to the html Page
 - Classify the emotion associated with Genre of Movie
 - Scraping the Web Page
 - Main Function
- Run the application

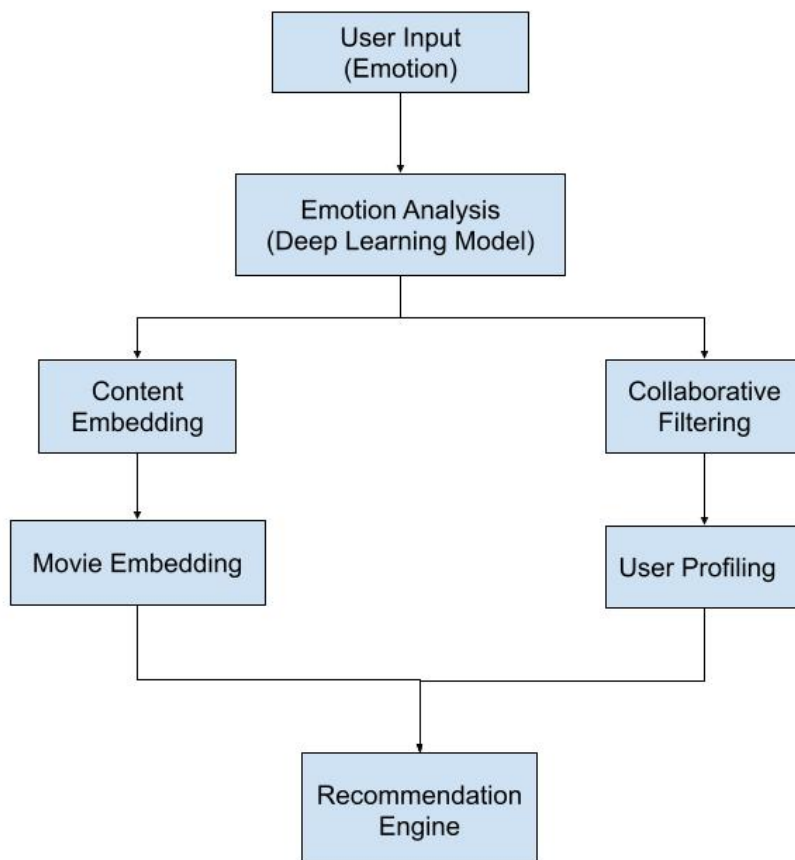
Necessary Steps

The necessary steps for accomplishing the project objective is

1. One of the fundamental focuses of movies is to bring out feelings in their watchers. So we have IMDb that offers a variety of movies for all genres or kind. In this manner, the movie titles can be scratched from the IMDb rundown to recommend to the user.
2. Presently IMDb doesn't have an API for getting data on motion pictures and TV Series. Accordingly, we need to perform scratching or scraping.
3. Scraping is utilized for getting to data from a site that is normally finished with APIs i.e utilizing BeautifulSoup which helps in scratching or scraping all the movie titles of the class relating to the information feeling and rundown to the user.
4. In the wake of Scraping, the user has to choose the emotion and based on the emotion, it will suggest the best and top movies of that genre would be recommended to the user.
5. Now the user can enjoy the movie according to their mood and emotions.

3. THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Hardware / Software designing

Hardware Requirements:

1. GPU (Graphics Processing Unit): Deep learning models, especially large ones, benefit significantly from GPU acceleration. A powerful GPU can speed up model training and inference, leading to faster and more efficient recommendation generation.
2. CPU (Central Processing Unit): A capable CPU is necessary for general system tasks, data preprocessing, and model deployment. Multi-core CPUs can help handle parallel processing tasks effectively.
3. Memory (RAM): Sufficient RAM is required to store data, load models, and perform computations during training and inference. Deep learning models can be memory-intensive, so having enough RAM is essential.
4. Storage: Adequate storage is needed to store the dataset, models, and other relevant files. SSDs (Solid State Drives) are preferred for faster data access and retrieval.
5. Network: A stable and fast internet connection may be necessary if the system involves real-time or online recommendations, especially when deploying the recommendation system on cloud-based servers.

Software Requirements:

1. Python: Most deep learning frameworks and libraries are implemented in Python. Python is the standard programming language used in the deep learning community, offering a wide range of libraries for data preprocessing, model development, and evaluation.
2. Deep Learning Frameworks: Choose a deep learning framework such as TensorFlow, PyTorch, or Keras to build, train, and deploy the deep learning models. These frameworks provide efficient GPU support, neural network building blocks, and pre-trained models.
3. NumPy: NumPy is essential for numerical computations and array operations, commonly used in data preprocessing and manipulation.
4. Scikit-learn: Scikit-learn is a popular machine learning library that offers various algorithms and tools for data preprocessing, feature selection, and model evaluation.
5. Natural Language Processing (NLP) Libraries: If the system involves processing textual data (e.g., emotion from user reviews), NLP libraries like NLTK (Natural Language Toolkit) or spaCy can be helpful for text processing and sentiment analysis.

4. EXPERIMENTAL INVESTIGATIONS

Flask is a web application framework written in python, let us build our web application which will be running in our local browser with a user interface.

In the flask application, whenever the user interacts with UI and selects emoji, it will suggest the best and top movies of that genre to the user.

Import Necessary Libraries

The first step is usually importing the libraries that will be needed in the program.

The required libraries to be imported to Python script are:

Numpy:

NumPy is a Python library used for working with arrays. In Python, we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

LXML

Python lxml is the most feature-rich and simple to utilize library for processing XML and HTML data. Python contents are composed to perform numerous errands like Web scraping or scratching and parsing XML.

BeautifulSoup

It is a library of python which is utilized to pull the data from the web pages i.e HTML and XML files. It works with your preferred parser to give colloquial methods for exploring, looking, and changing the parse tree.

Regular Expression:

A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern.

Requests:

requests are a Python HTTP library.

The different types of HTTP requests are

GET request is the most common method and is used to obtain the requested data from the specific server.

POST is the most common request method used to send data mostly through 'form' to the server for creating/updating in the server.

All the above modules can be imported into our program using the below code

Model Building

```
from flask import Flask, request, render_template
import numpy as np
import re
import os
from event.pywsgi import WSGIServer
import requests as HTTP
from bs4 import BeautifulSoup as SOUP
app = Flask(__name__)
app=Flask(__name__,template_folder="templates")

@app.route('/',methods=['GET'])
def index():
    return render_template('home.html')

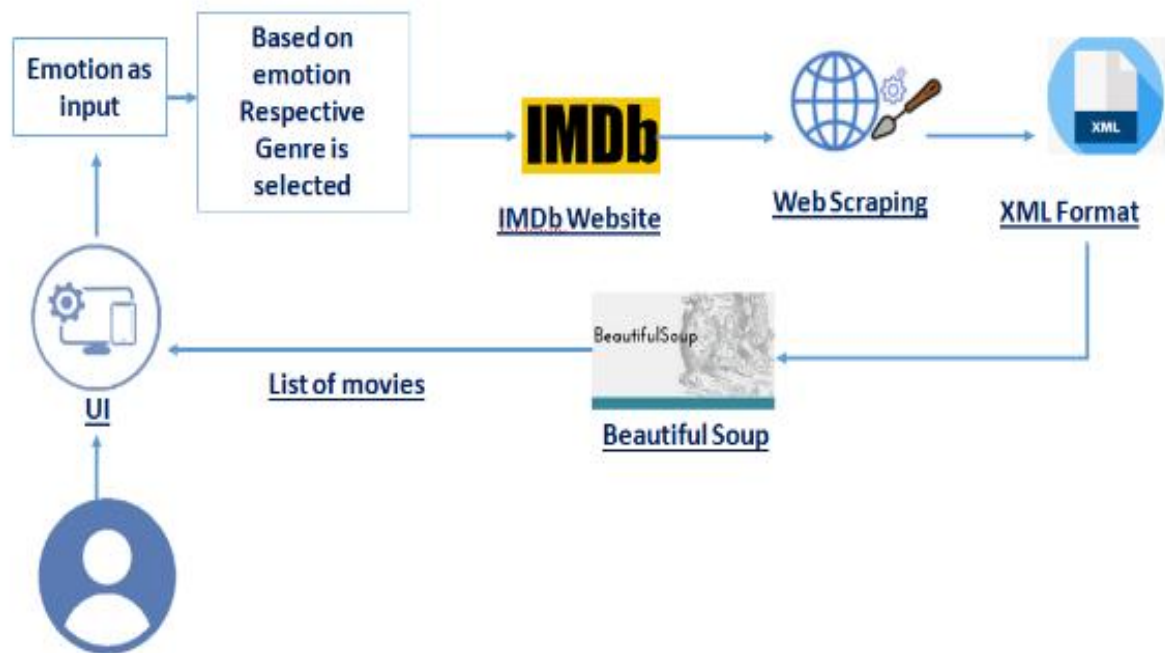
@app.route('/home', methods=['GET'])
def about():
    return render_template('home.html')

@app.route('/predict', methods=["GET","POST"])
def predict():
    #emotion=None
    #urlhere='http://www.imdb.com/search/title?genres=drama&title_type=feature&sort=metascore, asc'
    if request.method == "POST":
        emotion=request.form['emotion']
    print(emotion)
    if(emotion == "happy"):
        urlhere = 'https://www.imdb.com/search/title/?genres=comedy&languages=te'
    elif(emotion == "angry"):
        urlhere = 'https://www.imdb.com/search/title/?genres=drama&languages=te'
    elif(emotion == "disgust"):
        urlhere = 'https://www.imdb.com/search/title/?genres=horror&languages=te&sort=user_rating,desc'
    elif(emotion == "think"):
        urlhere = 'https://www.imdb.com/search/title/?genres=biography&languages=te&sort=year,asc'
    elif(emotion == "sad"):
        urlhere = 'https://www.imdb.com/search/title/?genres=thriller&languages=te&sort=metascore'
    response = HTTP.get(urlhere)
    data = response.text
    soup = SOUP(data, "lxml")
    supa = soup.find_all('h3', attrs={'class' : 'list-item-header'})
    list = []
    for header in supa:
        name = ""
        aElement_soup = header.find_all('a')
        spanElement_soup = header.find_all('span')
        spanElement = spanElement_soup[0]
        name = name + spanElement.text
        aElement = aElement_soup[0]
        name = name + "" + aElement.text
        if len(spanElement_soup)>1:
            spanElement = spanElement_soup[1]
            name = name + "\n" + spanElement.text
        list.append(name)

    return render_template('home.html',prediction_text="{}".format(emotion),data=list)

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

5. FLOWCHART



6 RESULT

- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type “python app.py” command.
- It will show the local host where your app is running on <http://127.0.0.1:5000>

```
C:\Users\remin>C:/Users/remin/AppData/Local/Microsoft/WindowsApps/python3.9.exe "c:/Users/remin/movie recom.py"

* Serving Flask app 'movie recom'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Copy that localhost URL and open that URL in the browser. It does navigate me to where you can view your web page.




Select an emotion:


☐ 😊 ☐ 😞 ☐ 😡 ☐ 😭 ☐ 🤔


Submit


Click on the Emoji's to get respective titles of the movies belonging to respective genre(or emotion)


Select an emotion:

☐ 

☐ 

☐ 

☐ 

☐ 

Submit

To display the related movies, click on submit button. The output will be displayed on home.html page.

Output:

Select an emotion:

☐ 

☐ 

☐ 

☐ 

☐ 

Submit

- 1.Kalki 2898-AD (2024)
- 2.RRR (Rise Roar Revolt) (2022)
- 3.72 Hoorain (2019)
- 4.The Kerala Story (2023)
- 5.Maaveeran (2023)
- 6.Baby (III) (2023)
- 7.The Jungle Book (2016)
- 8.Adipurush (2023)
- 9.Bāhubali: The Beginning (2015)
- 10.Rana Naidu (2023–)
- 11.Baahubali 2: The Conclusion (2017)
- 12.Mayapetika (2023)
- 13.Gadar 2 (2023)
- 14.Maamannan (2023)
- 15.Dasara (2023)

7 ADVANTAGES & DISADVANTAGES

Movie recommendation systems based on emotion using deep learning offer several advantages and disadvantages. Let's explore them:

Advantages:

1. **Personalization:** Emotion-based movie recommendation systems can provide highly personalized suggestions that align with users' emotional states. This can lead to more engaging and satisfying movie-watching experiences.
2. **Improved user engagement:** By recommending emotionally relevant movies, the system can captivate users' attention and keep them actively involved in the platform or service.
3. **Enhanced user satisfaction:** When users receive movie recommendations that resonate with their emotions, they are more likely to feel satisfied with the recommendation system, leading to increased user retention.
4. **Novelty and diversity:** Integrating emotions into the recommendation process can lead to a broader diversity of movie suggestions. Users may discover movies they wouldn't have otherwise considered, leading to a more enriched viewing experience.
5. **Emotional well-being:** In certain applications, such as mental health or therapy, emotion-based movie recommendations can contribute to emotional catharsis and help users explore and process their feelings.

Disadvantages:

1. **Emotion recognition challenges:** Detecting emotions accurately from user data (e.g., text, audio, video) is still a challenging task. Incorrectly identified emotions can lead to inaccurate recommendations, potentially frustrating users.
2. **Subjectivity and privacy concerns:** Recommending movies based on emotions might involve sensitive personal information. Users may feel uncomfortable sharing their emotions, raising privacy and ethical concerns.
3. **Emotional complexity:** Emotions are intricate and can change rapidly, making it difficult to accurately capture users' emotional states in real-time. Recommendations based on transient emotions may not always align with long-term preferences.
4. **Overemphasis on emotions:** Relying solely on emotions for movie recommendations might neglect other critical factors, such as a user's genre preferences, historical behavior, or external factors like time and context.

8. APPLICATIONS

A Movie recommendation system based on emotion using deep learning has several practical applications across various industries and domains. Here are some of the key applications:

1. Entertainment platforms: Emotion-based movie recommendation systems can be integrated into streaming platforms like Netflix, Amazon Prime, Hulu, or Disney+ to provide users with personalized movie suggestions based on their emotional states. This can enhance user engagement and satisfaction, leading to increased retention and loyalty.
2. Advertising and marketing: Emotion-aware movie recommendations can be utilized in advertising and marketing campaigns to target specific emotional states of the audience. For example, advertisers can recommend movies aligned with the emotional context of their product or brand message, leading to more effective ad campaigns.
3. Mental health and well-being: Emotion-based movie recommendations can be applied in mental health and well-being applications. For instance, mental health professionals can use these systems to suggest movies that align with a patient's emotional state or therapeutic needs, fostering emotional catharsis and self-expression.
4. E-learning and education: In educational platforms, emotion-aware movie recommendations can be used to supplement learning materials. By recommending emotionally relevant movies, educators can enhance the learning experience and promote emotional engagement with the subject matter.
6. Virtual reality (VR) and augmented reality (AR): Emotion-based movie recommendation systems can be integrated into VR and AR applications to create more immersive and emotionally engaging experiences. For instance, a VR cinema platform could recommend emotionally appropriate movies to users based on their real-time emotional states.

9. CONCLUSION

Movie recommendation systems based on emotion using deep learning have the potential to revolutionize user experiences and engagement. However, there are several challenges to address, including emotion recognition accuracy, privacy concerns, and interpretability. By continuously refining the models and striking a balance between emotional relevance and other movie characteristics, these systems can offer truly personalized and emotionally resonant recommendations in the future.

10. FUTURE SCOPE

The future scope for a Movie recommendation system based on emotion using deep learning is promising and opens up several exciting possibilities. Here are some key future directions:

1. Advanced emotion recognition models: Continued research and development in emotion recognition using deep learning can lead to more accurate and robust models. Emotion recognition can be expanded to include facial expressions, physiological signals, and voice analysis to capture emotions in a multimodal manner.
2. Multilingual and cross-cultural emotion analysis: Emotions are expressed differently across languages and cultures. Future systems can work on understanding and analyzing emotions in a cross-cultural context, enabling more culturally sensitive and inclusive recommendations.
3. Continuous emotion tracking: Instead of relying on a single snapshot of emotion, future systems can track users' emotional states continuously. This can lead to dynamic and real-time recommendations that adapt to users' changing emotional states as they watch movies.
4. Context-aware recommendations: Incorporating contextual information, such as the time of day, weather, location, and recent activities, can further refine the emotion-based recommendations. Context-awareness can provide a deeper understanding of users' emotional needs at a particular moment.
5. User interaction and feedback: Designing recommendation systems that actively seek and incorporate user feedback on movie recommendations can enhance the overall user experience. Feedback loops can help the system refine its understanding of emotions and improve the quality of recommendations.

11. BIBLIOGRAPHY

<https://www.mdpi.com/2227-7390/8/2/241>

APPENDIX A.

https://drive.google.com/file/d/1JheSZF3KzXWwo4dcojbfwf0kuGeiHQoer/view?usp=drive_link