

GENREGLIMPSE: MULTI-MODEL MOVIE
GENRE PREDICTION FROM POSTER

Project Report

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

INTEGRATED M.Sc. PROGRAMME IN COMPUTER SCIENCE – DATA SCIENCE

Submitted by

SREELEKSHMAN S
REG. NO: 21UMCP2955

Under the guidance of

Ms. VAISHNAVI RAGHURAMAN
ASSISTANT PROFESSOR,
DEPARTMENT OF COMPUTER SCIENCE

March 2024



SACRED HEART COLLEGE (AUTONOMOUS)
THEVARA – 682 013

SACRED HEART COLLEGE (AUTONOMOUS), THEVARA

(Affiliated to Mahatma Gandhi University)



Certificate

*This is to certify that the project entitled “GenreGlimpse: Multi-Model Movie Genre Prediction from Poster” submitted in partial fulfillment of the requirements for the award of the degree in **Integrated M.Sc. Computer Science – Data Science** is a bonafide report of the project done by **Sreelekshman S** (Reg. No: 21UMCP2955) during the academic year 2023-24.*

Internal Guide

Head of the Department

Examiners:

1.

2.

DECLARATION

*I hereby declare that the project work entitled “**GenreGlimpse: Multi-Model Movie Genre Prediction from Poster**” is a record of original work done by me under the guidance of Ms. Vaishnavi Raghuraman, Assistant Professor, Department of Computer Science – Data Science and the project work has not formed the basis for the award of any Degree/Diploma or similar title to any candidate of any University.*

Internal Guide

Signature of the Student

Sreelekshman S

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project guide, **Ms. Vaishnavi Raghuraman**, Assistant Professor, Department of Computer Science – Data Science, for her constant support, guidance and encouragement throughout this project. Her insights and expertise were instrumental in shaping the direction of this research and ensuring its successful completion.

I am also thankful to **Mr. Vishnu Mohan C**, Assistant Professor, Department of Computer Science – Data Science, for his continuous support and willingness to help whenever needed. His contributions played a significant role in the overall development of this project.

Furthermore, I extend my appreciation to **Dr. Regitha M R**, HoD, Department of Computer Science, and **Rev. Dr. Jose John C.M.I**, Principal, Sacred Heart College for providing the necessary resources and fostering an environment conducive to academic exploration.

Finally, I would like to acknowledge my friends and colleagues for their encouragement and motivation throughout this journey. Their camaraderie made this project a more enriching experience.

CONTENTS

1.	SYNOPSIS	1
2.	INTRODUCTION	3
3.	SYSTEM ANALYSIS	5
	3.1. EXISTING SYSTEM	6
	3.2. PROPOSED SYSTEM	6
	3.3. SYSTEM SPECIFICATION	6
	3.4. FEASIBILITY STUDY	8
	3.4.1. ECONOMICAL FEASIBILITY	8
	3.4.2. TECHNICAL FEASIBILITY	8
4.	REQUIREMENT ANALYSIS	10
	4.1. PROBLEM RECOGNITION	11
	4.2. PROBLEM EVALUATION & SYNTHESIS	11
	4.3. SYSTEM REQUIREMENTS SPECIFICATION	12
	4.4. MODELING	12
5.	SYSTEM DESIGN	14
	5.1. DATA DESIGN	15
	5.2. ARCHITECTURAL DESIGN	16
	5.3. INTERFACE DESIGN	18
6.	CODING	19
	6.1. DATA PRE-PROCESSING	20
	6.2. TRAINING THE CNN MODEL	20
	6.3. TRAINING THE BERT MODEL	21
7.	TESTING PROCESS	23
	7.1. SYSTEM TESTING	24
	7.1.1. UNIT TESTING	24
	7.1.2. INTEGRATION TESTING	25
	7.1.3. VALIDATION TESTING	25
8.	IMPLEMENTATION AND RESULT	26
9.	SOFTWARE MAINTENANCE	29
10.	CONCLUSION	31
11.	FUTURE ENHANCEMENTS	33
12.	APPENDICES	36
	APPENDIX A (SRS DOCUMENT)	37
	APPENDIX B (DFD)	38
	APPENDIX C (INTERFACE DESIGN)	41
	APPENDIX D (CODE)	43
13.	BIBLIOGRAPHY AND REFERENCES	59

SYNOPSIS

1. SYNOPSIS

The project aims to develop an advanced system for predicting movie genres based on movie posters using a fusion model that integrates image analysis and text recognition techniques. The main objective is to create a powerful and efficient system capable of leveraging both visual and textual information from movie posters to significantly improve the accuracy of genre predictions.

In the scope of the project, a diverse dataset of movie posters with labeled genres will be collected and preprocessed to ensure data quality. The image analysis component will involve a custom architecture that extracts relevant visual features from the movie posters. Concurrently, a text extraction algorithm used for extracting textual data and BERT model will be implemented to classify the textual data, such as movie titles and actor names, from the posters.

The key innovation is the multi-model architecture, which will combine the outputs of the image analysis and text analysis models. By concatenating the outputs from both models, the system aims to build a comprehensive representation of the movie posters, capturing both visual cues and textual content. This combined representation will enable the model to make more accurate genre predictions, surpassing the limitations of existing approaches that rely solely on visual data. The successful implementation of this project will have significant implications for the entertainment industry. The improved genre prediction accuracy can enhance content recommendation systems, optimize marketing strategies, and provide valuable insights for audience targeting. To ensure efficient processing of the vast amount of data involved, the system will require a powerful computing infrastructure capable of handling large datasets and training complex deep learning models effectively.

In conclusion, the proposed movie genre prediction system using a multi-model approach represents a cutting-edge solution that effectively integrates image analysis and text analysis to achieve superior genre prediction accuracy. This project's potential impact on the entertainment industry is promising, making it a crucial endeavor in the era of data-driven decision-making and personalized user experiences.

INTRODUCTION

2. INTRODUCTION

GenreGlimpse is a web application being developed for the purpose of predicting movie genres based on their corresponding posters. This project leverages a multi-modal deep learning approach, integrating two distinct machine learning models: an image classification model and a text classification model.

The image classification model will be trained on a curated dataset of labeled movie posters. This model will be tasked with extracting relevant visual features from the posters, such as color palettes. These features are hypothesized to contain valuable information that can be correlated with specific movie genres.

The text classification model, on the other hand, will focus on textual elements present on movie posters, including titles, taglines, and actor names. By analyzing these textual components, the model can potentially glean additional genre-specific information that might not be readily apparent from the visual elements alone.

This combined output from both the models will serve as the basis for the final genre prediction, aiming to achieve a level of accuracy that surpasses what either model could achieve independently.

This project record will meticulously document the development process of GenreGlimpse. It will detail the data collection and pre-processing stages, delve into the architecture and training procedures of both the image and text classification models, explore the chosen fusion technique, and finally, outline the testing, evaluation, and deployment phases that will culminate in the launch of the GenreGlimpse web application.

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

This system analysis explores the development of an advanced movie genre prediction system leveraging a multi-model approach. The system will integrate image analysis and text analysis techniques to improve genre prediction accuracy based on movie posters.

3.1. EXISTING SYSTEM

Existing movie genre prediction systems solely rely on visual data like colour pallets, scene composition etc. These systems might achieve good results, but they could miss valuable information conveyed through textual elements on movie posters such as movie title, taglines, director's name etc.

3.2. PROPOSED SYSTEM

The proposed system is a movie genre prediction system that utilizes a multi-model approach. This approach combines image analysis and text analysis techniques to extract comprehensive information from movie posters. Leveraging the strengths of both visual and textual data extracted from movie posters, the system achieves more accurate and nuanced genre identification capability.

- **System Overview:** The new system will consist of two main components: an image analysis model and a text analysis model. The image analysis model will extract visual features from movie posters, while the text analysis model will process and classify textual information like titles and actor names. The outputs from both models will be combined to make a final genre prediction.
- **Benefits:** The proposed system will significantly enhance movie genre prediction by:
 - Leveraging a broader range of information from movie posters.
 - Providing more accurate and comprehensive genre classifications.
 - Offering valuable insights for content recommendation, marketing strategies, and audience targeting in the entertainment industry.

3.3. SYSTEM SPECIFICATION

The proposed multi-model movie genre prediction system will require the following specifications.

Hardware Requirements

- **GPU:** A high-performance GPU with at least 8GB of VRAM is highly recommended for efficient training of the image analysis model and potentially the text analysis model if using a large pre-trained model like BERT. The Nvidia Geforce RTX 3050 with 4GB VRAM might be sufficient for smaller datasets or simpler model architectures, but a more powerful GPU like an RTX 3070 or higher with at least 8GB of VRAM would be ideal for optimal performance.
- **CPU:** A minimum of 16GB RAM is required if training on a CPU. However, using a CPU will significantly increase training time compared to a GPU. An Intel Core i5 10500H Hexa-Core CPU with 4.50 GHz max-clock meets the minimum RAM requirement, but a more powerful CPU with higher core count and clock speed would be preferable for CPU-based training.

Software Requirements

Required Libraries:

- **Deep Learning Frameworks:** TensorFlow (or PyTorch) for developing the image analysis model. Keras is used as a high-level API on top of TensorFlow for easier model building.
- **Scientific Computing Libraries:** NumPy for numerical computations and Pandas for data manipulation and analysis.
- **Transformer Libraries:** For leveraging a pre-trained model like BERT, libraries like transformers is used.
- **Torch:** While TensorFlow or PyTorch is sufficient, Torch can be an alternative deep learning framework.
- **Google Cloud Library (google.cloud):** For utilizing the Cloud Vision API of google, the Google Cloud Library is used.

Programming Environment:

- **Development Environment:** Visual Studio Code is a popular choice, but any code editor or IDE with Python support can be used.
- **Python Interpreter:** A Python 3.x interpreter is required to run the Python code for the project.

- **Network Requirements:** A stable internet connection for using the Cloud Vision API to extract textual data from the posters. High-speed network infrastructure within the system is also required for efficient data transfer during training and prediction processes.

3.4. FEASIBILITY STUDY

A feasibility study constitutes a formal assessment of a proposed project's viability. It employs a systematic approach to evaluate the project's economic soundness, technical achievability, and operational practicality. Through a comprehensive analysis of costs, resource availability, and potential challenges, a feasibility study serves as a critical decision-making tool.

3.4.1. ECONOMIC FEASIBILITY

The proposed project exhibits good economic feasibility. While there are initial costs associated with acquiring high-performance GPUs or renting computing instances on cloud platforms, the long-term benefits outweigh these. Improved genre prediction can significantly impact the entertainment industry, potentially boosting revenue through better recommendations and marketing strategies. The economic feasibility of the system is actually high because:

- **Reduced Licensing Costs:** Open-source software minimizes licensing fees.
- **Flexible Cloud Pricing:** Cloud platforms offer pay-as-you-go scalability, reducing hardware costs.
- **Since we use open-source libraries for developing the system, there is no additional building cost.**

3.4.2. TECHNICAL FEASIBILITY

The technical feasibility is promising as well. Established deep learning frameworks like TensorFlow and PyTorch have extensive resources available, and movie poster datasets can be obtained from various sources. The system is technically feasible because:

- **Established Frameworks:** Utilizing popular frameworks (TensorFlow, PyTorch) with ample resources.
- **Data Availability:** Movie poster datasets are obtainable from various sources.

- **Scalable Cloud Training:** Cloud platforms with powerful GPUs can significantly speed up training.
- **Addressable Multi-modal Challenge:** The hurdle of combining image and text outputs can be overcome through multi-modal learning research and experimentation with different model architectures.
- **Online Resources and Cloud Power:** Online resources and cloud computing can effectively address technical challenges.

In conclusion, the feasibility study suggests that the proposed multi-model movie genre prediction system is achievable from both economic and technical perspectives. The potential benefits for the entertainment industry justify the investment in resources and skill development. Carefully consider the trade-off between training time and hardware costs when choosing between using your local GPU or cloud-based resources.

3.4.3. BEHAVIORAL FEASIBILITY

Behavioral feasibility assesses how users will adapt to a new system. It considers factors like ease of use and how well the system integrates into existing workflows. The goal is to minimize training needs and disruption, ensuring users can quickly adopt the system and leverage its features effectively. A system with high behavioral feasibility is user-friendly and promotes seamless integration into daily tasks. GenreGlimpse is designed with user adoption in mind. Here's an analysis of the system's behavioral feasibility:

- **Ease of Use:** The system prioritizes a user-friendly interface built with Gradio. This web interface requires minimal technical expertise, allowing users to effortlessly upload movie posters and receive genre predictions. No prior training is necessary to leverage the system's capabilities.
- **Minimal Workflow Disruption:** The system integrates seamlessly into existing workflows. Users familiar with movie poster browsing and online image upload can readily utilize the system without needing to modify their habits.
- **Reduced Learning Curve:** The intuitive interface and straightforward functionalities minimize the learning curve for new users. Uploading an image and interpreting the presented genre predictions is a familiar and readily grasped process.
- **Adoption Time:** Given the system's user-friendly nature, the anticipated adoption time is relatively low. Users can begin exploring movie genres and leveraging the system's insights within minutes of encountering it.

REQUIREMENT ANALYSIS

4. REQUIREMENT ANALYSIS

Requirement analysis delves into the detailed requirements of our multi-model movie genre prediction system. This section explores the problem the system is aiming to solve, analyze its scope and limitations, and establish a clear model for system development.

4.1. PROBLEM RECOGNITION

Developing a system for automatic movie genre classification based solely on movie posters presents a significant challenge due to the inherent limitations of visual data in capturing the film's full thematic range.

- **Subtlety and Nuance:** Movie posters often rely on subtle visual cues to convey genre. For instance, a dark and muted color palette might suggest a horror film, but without additional context, it could be indicative of another genre entirely. Machine learning models might struggle to interpret these nuances and require extensive training data to differentiate between subtle cues.
- **Genre Overlap:** The boundaries between genres can be blurry, with many movies blending elements from multiple genres. For example, a dark comedy poster might incorporate elements of both comedy and thriller, making it difficult for a system to pinpoint a single, definitive genre.
- **Minimal Textual Data:** Not all movie posters contain textual information. Even those that do might only have the title and actors' names, which might not provide enough information to accurately determine the genre.

4.2. PROBLEM EVALUATION & SYNTHESIS

To address the limitations of using only visual data from movie posters for genre prediction, the proposed system adopts a multi-model architecture. This approach leverages the strengths of both visual and textual analysis. By concatenating the extracted visual features and the processed textual information, the system aims to create a more comprehensive representation of the movie poster. This richer representation captures both the visual cues and

any relevant textual information, allowing the model to make more accurate genre predictions, even when dealing with subtle visual elements, genre overlap, or limited textual data.

4.3. SOFTWARE REQUIREMENTS SPECIFICATION

The System Requirements Document (SRS) provides a detailed breakdown of the functional and non-functional requirements for the multi-model movie genre prediction system. **This document is included in APPENDIX A for reference.** The SRS outlines the system functionalities, hardware and software specifications, user requirements, and other essential details to guide the development process.

4.4. MODELING

The proposed multi-model movie genre prediction system can be effectively modeled using a Data Flow Diagram (DFD). A DFD is a graphical representation that visualizes the flow of information throughout the system. It's particularly useful for understanding how data progresses within the application, making it easier to analyze and design the system's functionalities.

DFDs utilize a specific set of symbols to depict data flow, the overall system structure, and its operation. These diagrams are typically divided into levels, where each higher level elaborates on the processes detailed in the level below. Here's a breakdown of the common DFD symbols:

- **Rectangle:** Represents an activity or transformation performed on data. This could involve calculations, data manipulation, decision-making, or any step that modifies the data flow.
- **Arrow:** Indicates the direction of data flow.
- **Open Rectangle (Parallel Lines):** Depicts the system's database where data is stored and retrieved.
- **Square:** Represents any entity outside the system boundary that interacts with the system by providing or receiving data. This could be users, other systems, or external data sources.

A detailed DFD for the proposed system, illustrating the general data flow across two levels, is included in APPENDIX B. This DFD provides a visual representation of how data moves through the system, from the initial input to the final genre prediction output.

SYSTEM DESIGN

5. SYSTEM DESIGN

This section delves into the detailed design of the multi-model movie genre prediction system exploring how the system will be structured, how data will be managed, and how users will interact with it.

5.1. DATA DESIGN

This section focuses on how data will be structured, stored, and accessed within the system. It will define the data entities and their attributes, along with the chosen database management system for storing and retrieving this data. Our system will utilize two primary databases to train the models. A dataset having movie names and respective genres are need to train the image classification model. For this we are using the MovieGenre dataset from Kaggle. This dataset also contains a poster column which has the link to the poster images of each movies. But since the dateset is old we are not using it to train our image classification model. This is because poster designs have changes a lot nowadays and training the model with these old data will make the model not able to accurately classify newer posters.

MovieGenre	
ImdbId	integer
Imdb_Link	varchar
Title	varchar
Imdb_Score	float
Genre	varchar
Poster	varchar

Figure 1

Figure 1 shows the structure of the MovieGenre dataset used to train the BERT text classification model. The database was robust and biased towards some genres. So the database is pre-processed. Pre-processing steps:

1. Separate datasets(csv files) are made for each genre by extracting rows from the main dataset.
2. These datasets are shrunk to a common number of rows by dropping random rows. This ensures that all genres have equal impact in the final model.
3. The individual datasets are then concatenated.
4. Unwanted columns are removed (ImdbId, Imdb_Link, Imdb_Score, Poster)

The Title and Genre columns were used to fine-tune the BERT model.

For training the image classification model, a new dataset is created for the system. The dataset, named MovieMatrix, is a csv file which contains 2 columns.

MovieMatrix	
ID	
genre	

Figure 2

Figure 2 represents the structure of MovieMatrix dataset. There is an ID which is given to each poster and the genre of that movie. Movie posters are saved in a folder named ‘Posters’ with their respective ID as name. The poster of the movie is fetched using ID from the dataset.

The dataset is made using posters from the website <https://letterboxd.com/>. Movies of each genre is fetched and poster is saved to the folder ‘Posters’ in a numeric order. Along with that, the csv file named MovieMatrix.csv is created and the genre of each poster saved is written into the genre column of the dataset corresponding to the numeric name of the poster in the ID column.

5.2. ARCHITECTURAL DESIGN

The architectural design defines the high-level structure of the multi-model movie genre prediction system and how its components interact. This section focuses on detailing how the system is decomposed into functional modules.

Core Components:

- **Pre-processing Module:**

- Responsible for data preparation tasks like image resizing, normalization, and dataset pre-processing for movie titles extracted from the CSV file.

- **Image Analysis Model:**

- This module leverages a deep learning model, a Convolutional Neural Network (CNN), trained to extract relevant visual features from the preprocessed movie posters. These features include color palettes and overall composition.

- **Text Analysis Model:**

- Textual information from Title column of the MovieGenre dataset is included, this module will process the text data and fine-tune the pre-trained BERT model for text classification.
- In the stage of testing a movie poster is uploaded and the textual data from the poster is extracted using Google Cloud Vision API and this data is given to the fine-tuned BERT model for classification.

- **Feature Fusion Algorithm:**

- This critical module comes to play in the stage of testing . This module combines the output from the image analysis model and the text analysis model into a unified representation. This combined representation captures both the visual and textual cues from the movie poster. This is the final result from the system.

Component Interaction:

1. The pre-processing module receives movie posters and associated data from the CSV file.
2. Preprocessed movie posters are fed into the Image Analysis Model, which extracts visual features.
3. Text analysis module processes the text data from the datasets and fine-tunes the BERT model.
4. During testing stage both these saved models are used to get predictions. Their outputs are then concatenated and given as output.

Benefits of this Architecture:

- **Modular Design:** The system is divided into well-defined modules, promoting maintainability and reusability of individual components.
- **Scalability:** The architecture can be easily extended to incorporate additional data sources or machine learning models in the future.

This architectural design provides a clear separation of concerns and facilitates efficient development and integration of the system's core functionalities.

5.3. Interface Design

Interface design focuses on how users will interact with GenreGlimpse. This section outlines the design principles for the user interface (UI) elements and considers the user experience (UX) to ensure a smooth and intuitive interaction.

UI Elements:

- An upload box is provided to upload the movie poster. A Submit button is present to submit the uploaded image for classification. A Clear button is also present which clears the uploaded image.
- The output will be displayed in a text box right next to the uploaded image. The output will be 2 most probable genres.

The system prioritizes a positive user experience (UX) by featuring a clean and uncluttered user interface (UI). This streamlined design makes it easy for users to navigate and interact with the system. Additionally, the color theme is chosen to ensure accessibility for all users, regardless of visual limitations. This combination of clarity, simplicity, and inclusivity fosters a user-friendly experience for everyone. **User interface of the system is given in APPENDIX C.**

CODING

6. CODING

This section details the coding strategies and methods employed to develop the multi-model movie genre prediction system. It outlines the key steps involved in data pre-processing, training the individual CNN and BERT models, and finally integrating them for genre prediction.

6.1. Data Pre-Processing

- **Splitting Movie Genre Dataset:** The script iterates through each genre ("Action", "Comedy", etc.) and extracts movies belonging to that specific genre from the original dataset. Each extracted subset is then saved as a separate CSV file. This facilitates training the models on datasets focused on individual genres and remove possible bias.
- **Merging Split Datasets:** After normalizing the row count of each split dataset manually (potentially due to inconsistencies in the original data), the script merges them into a single, comprehensive CSV file. Duplicate entries based on the "imdbId" field are removed to ensure data integrity. Finally, the merged dataset is shuffled to avoid any potential bias introduced by the order of entries.
- **Preparing Images:** A script is used to create a list of movie IDs corresponding to the movie posters. It then iterates through this list, reading and converting each poster image into a NumPy array. The script also checks for consistency in image dimensions across the dataset.
- **Multi-Hot Encoding Genres:** A dictionary is used to map genres to their corresponding one-hot encoded vectors. This allows the model to handle movies belonging to multiple genres effectively.

6.2. Training the CNN Model for Image Classification

- **Defining the CNN Model:** The script utilizes Keras to define a Convolutional Neural Network (CNN) architecture specifically designed for image classification. The model employs convolutional layers with various filter sizes and pooling layers for feature extraction. Dropout layers are incorporated to prevent overfitting. The final layers

consist of dense layers with sigmoid activation function suitable for multi-class classification.

- **Compiling and Training the CNN:** The CNN model is compiled with an optimizer (RMSprop) and a loss function (categorical cross-entropy) suitable for multi-class classification tasks. The model is then trained on the prepared image data and corresponding genre labels for a specified number of epochs. Finally, the trained model is saved.

6.3. Training the BERT Model for Text Classification

- **Data Cleaning and Pre-processing:** The script loads the merged movie dataset and removes irrelevant columns like the movie poster path. It then creates a new "label" column by assigning a numerical index to each unique genre. This allows the BERT model to handle categorical genre labels.
- **Train-Validation Split:** The script employs scikit-learn's `train_test_split` function to partition the data into training and validation sets. This helps evaluate the model's performance on unseen data.
- **Tokenization and Encoding:** The script utilizes the pre-trained BERT tokenizer to convert movie titles from the training data into numerical sequences suitable for the BERT model. Padding is applied to ensure all sequences have the same length.
- **Defining and Training the BERT Model:** The script employs the pre-trained `BertForSequenceClassification` model from the Transformers library. It is fine-tuned on the prepared training data for a specified number of epochs.
- **Saving the Fine-tuned BERT Model:** After training, the fine-tuned BERT model's state dictionary is saved for later use in the genre prediction process.

6.4. Main Function and User Interface

- **Loading Models:** The main function first loads the trained CNN model and the fine-tuned BERT model.
- **Image Pre-processing:** When an image is provided as input, the script converts it to a NumPy array and resizes it to match the dimensions the CNN model was trained on.
- **Genre Prediction with CNN:** The pre-processed image is fed into the CNN model, generating genre probabilities.

- **Text Extraction:** The script utilizes Google Cloud Vision's Text Detection API to extract textual data from the input image.
- **Genre Prediction with BERT:** The extracted textual data is pre-processed and fed into the fine-tuned BERT model, resulting in genre probabilities.
- **Genre Concatenation:** The genre probabilities from both the CNN and BERT models are combined using a simple averaging technique. This leverages the strengths of both visual and textual features for more robust genre prediction.
- **Final Output:** The top two predicted genres, along with their corresponding probabilities, are returned as the final output.
- **Gradio Interface:** The Gradio library is used to create a user-friendly web interface for genre prediction. This interface allows users to upload an image of a movie poster and receive the predicted genres.

TESTING PROCESS

7. TESTING PROCESS

A thorough testing process is crucial for ensuring the quality, reliability, and functionality of the multi-model movie genre prediction system. This section outlines the various testing strategies employed.

7.1. SYSTEM TESTING

System testing verifies the overall functionality of the system as a whole, ensuring it meets the specified requirements. Here, we'll delve into different levels of testing within the system testing phase:

7.1.1. UNIT TESTING

Unit testing of the individual system components was conducted. This involved testing each module, such as the CNN model and the BERT model, in isolation to ensure they functioned correctly according to their design specifications.

The CNN model achieved an accuracy of over 87% on unit testing, demonstrating its strong performance in extracting visual features from movie posters. The BERT model's unit test accuracy was around 50%. While this may seem lower than ideal, it's important to consider the nature of the testing data used.

The dataset underwent pre-processing to mitigate bias during model training. This included assigning each movie a single genre label. This approach is essential to avoid biased predictions, but it also limits the broadness of the dataset. This caused the model to predict genres which are outside of the scope of testing dataset. This caused the model to have lower accuracy in unit testing.

Therefore, the unit test accuracy for BERT reflects the limitations of the training data, not necessarily a limitation of the model itself.

7.1.2. INTEGRATION TESTING

Integration testing assessed how the different modules interact and work together as a whole. This involved testing the communication and data flow between modules, such as the transfer of pre-processed image data from the pre-processing module to the CNN and BERT model and the integration of features from both the CNN and BERT models for genre prediction.

Integration testing accuracy was also limited because of the inconsistency of the testing dataset. But still it revealed a significant improvement in overall accuracy compared to individual model performance. This highlights the system's ability to leverage the strengths of both visual and textual analysis for more robust genre prediction.

7.1.3. VALIDATION TESTING

Validation testing evaluated the system's performance in a real-world scenario. This included user acceptance testing (UAT) where potential users interacted with the system and provided feedback. Interestingly, despite the lower unit test accuracy for the BERT model and the integrated system, user feedback indicated that the predictions made in real-world use cases were often accurate and relevant. This suggests that the models were able to capture meaningful patterns that aligned with user expectations.

Additionally, the system received high user acceptance ratings, indicating its usability and effectiveness in delivering genre predictions.

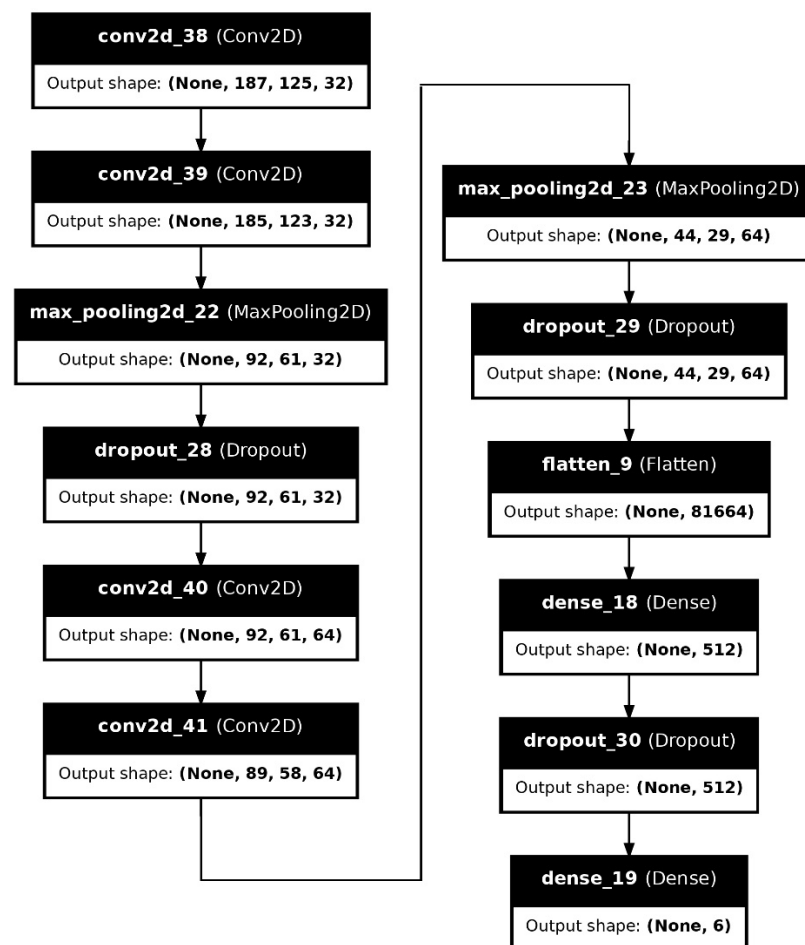
Future efforts will focus on acquiring a more comprehensive and representative validation and test dataset to obtain a more precise quantitative assessment of the system's accuracy. This will allow for further refinement of the models and potentially improve performance even further. However, the positive user feedback highlights the system's potential for real-world application, even with ongoing optimization efforts.

IMPLEMENTATION AND RESULT

8. IMPLEMENTATION AND RESULT

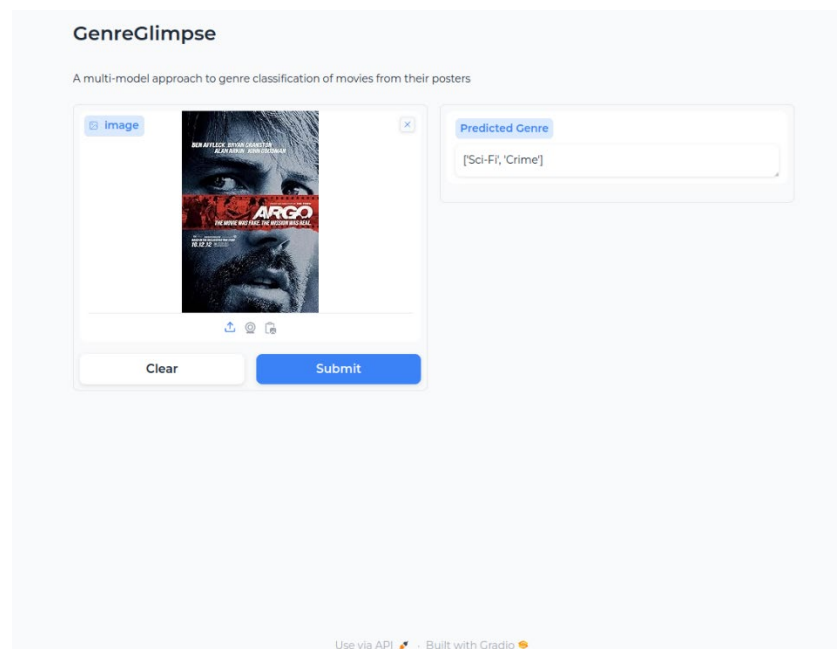
Implementation is the stage where a system transitions from concept to reality. It's the bridge between the blueprint (design documents and specifications) and the finished product (the functional software). The GenreGlimpse system bridges the gap between computer vision and natural language processing. To achieve this, the system relies on a powerful combination of tools.

Data preparation utilizes Pandas for cleaning and manipulating the movie genre dataset. Libraries like OpenCV or Pillow ensure the images are compatible with the Convolutional Neural Network (CNN) by handling image processing tasks. Keras, a high-level neural network API, serves as the foundation for defining, compiling, and training the CNN model. Working alongside TensorFlow as the backend, the CNN learns to extract features from movie poster visuals and map them to their corresponding genres. The plot of the CNN model is:



For text processing, the Transformers library provides access to the pre-trained BERT model, a powerhouse in natural language processing. This system leverages the Transformers library's BertForSequenceClassification model. By fine-tuning it on movie titles, the BERT model is equipped to classify them into specific genres. Additionally, the Google Cloud Vision API acts as a bridge between the visual and textual world. When an image is fed into the system, the Vision API extracts text, likely the movie title, enabling genre prediction based on textual information. Finally, Gradio creates a user-friendly web interface, allowing users to seamlessly upload movie posters and receive the system's predicted genres.

Result:



- **Performance:** The system achieved greater performance thanks to the RTX 3050 GPU in the testing machine. Each prediction is done under 4 seconds which is perfect for the system.
- **Reliability:** The predictions made by the system are consistent and did not change each time. Therefore, the system can be pronounced reliable to use.

SOFTWARE MAINTENANCE

9. SOFTWARE MAINTENANCE

The multi-model movie genre prediction system isn't a one-time creation. To guarantee its continued effectiveness and reliability over time, ongoing software maintenance is crucial. This process ensures the system adapts to the ever-evolving movie landscape and maintains its accuracy. Here are some key considerations for software maintenance:

- **Data Freshness** : As the movie industry churns out new releases, the system should adapt and incorporate this fresh data. Periodic data collection on new movies and subsequent re-training of the models are essential. This ensures the system stays relevant, reflecting the evolving trends in movie genres and capturing the nuances of contemporary filmmaking.
- **Model Refinement**: User feedback and performance metrics might indicate a need to refine the models over time. This could involve enriching the training data with new information, adjusting hyperparameters to optimize performance, or even exploring alternative model architectures that might yield superior results. By continuously evaluating and refining the models, the system maintains its edge and delivers the most accurate genre predictions possible.

CONCLUSION

10. CONCLUSION

This document explores a novel approach to movie genre prediction by leveraging the strengths of both computer vision and natural language processing. The proposed system achieves accurate genre classification through a multi-model architecture. A Convolutional Neural Network (CNN) analyzes the visual content of movie posters, while a fine-tuned Bidirectional Encoder Representations from Transformers (BERT) model deciphers textual data from the posters. By combining these analyses, the system offers a robust and informative solution for movie genre prediction.

However, the system's effectiveness relies on continuous maintenance. As the movie industry evolves, keeping the data up-to-date and the models refined is crucial. Regularly incorporating new movie data and retraining the models ensure the system reflects the latest trends and maintains its accuracy. Additionally, exploring alternative model architectures and incorporating new data sources hold promise for further improvement.

Despite these potential challenges, the system presents significant value to various stakeholders within the movie industry.

FUTURE ENHANCEMENTS

11. FUTURE ENHANCEMENTS

The multi-model movie genre prediction system offers a promising foundation for further development. Some key areas for enhancement that have the potential to significantly improve the system's accuracy, reliability, user experience, and robustness are:

- **Expanding Genre Diversity with a Broader Dataset**

Currently, the system is limited to only 6 set of movie genres. To enhance its comprehensiveness and accuracy, incorporating a broader dataset encompassing at least 12 different genres is crucial. This will allow the models to learn a wider range of genre-specific features, leading to more nuanced and accurate predictions, especially for genres that are currently underrepresented in the system.

- **Deepening Image Analysis with Object Detection**

The current system relies on analyzing the overall visual content of movie posters. Integrating a pre-trained object detection model into the image analysis process can significantly enhance the system's capabilities. This model could be trained to identify specific objects or elements commonly associated with certain genres (e.g., spaceships for science fiction or swords for fantasy). By recognizing these elements, the system can gain deeper insights into the movie's genre, potentially leading to more refined predictions.

- **Elevating User Experience with Flask**

While the current Gradio-based interface is functional, it presents opportunities for further refinement. Utilizing Flask, a popular web development framework, can empower the creation of a more feature-rich and visually appealing user interface (UI). Flask allows for greater customization and interactivity, potentially enabling users to explore additional details about predicted genres or browse movie recommendations based on the genre predictions. This enhanced UI design can significantly improve the overall user experience, making genre identification an even more engaging and informative process.

By implementing these enhancements, GenreGlimpse has the potential to evolve into a robust and reliable tool for movie enthusiasts and industry professionals alike. The expanded genre coverage, deeper image analysis, and more interactive UI will contribute to a system that is not only accurate but also delightful to use.

APPENDICES

12.1. APPENDIX A (SRS)

Table of Contents

Table of Contents	i
Revision History	i
1. Introduction.....	1
1.1 Purpose.....	1
1.2 Intended Audience and Reading Suggestions	1
1.3 Project Scope	1
1.4 Definitions, Acronyms, and Abbreviations.....	1
2. Functional Requirements	1
2.1 Product Perspective.....	1
2.2 Product Features	2
2.3 Operating Environment.....	2
3. Non-Functional Requirements.....	2
3.1 Performance.....	2
3.2 Reliability.....	2
3.3 Usability.....	2
4. Technical Requirements	2
4.1 Performance	2
4.2 Reliability.....	2
4.3 Usability.....	2
5. Conclusion	3

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this document is to outline the detailed requirements for the development of "GenreGlimpse," a web application that predicts a movie's genre based on its poster. The prediction will be made using a fusion model that integrates image analysis and text recognition techniques.

1.2 Intended Audience and Reading Suggestions

The intended users span a broad spectrum of professionals and enthusiasts. For developers and data scientists, the focus lies on exploring research papers elucidating advanced image analysis techniques and deep learning models. Hands-on experience is further enriched by tutorials detailing the implementation of custom algorithms and utilizing essential tools like BERT.

1.3 Project Scope

The project encompasses the development of a sophisticated system capable of accurately predicting movie genres by analyzing movie posters. It involves collecting, pre-processing, and analyzing a diverse dataset of labeled movie posters. The system will employ custom image analysis techniques and text recognition algorithms to extract relevant visual and textual features. The key innovation lies in the fusion model, which combines outputs from both components to enhance genre predictions significantly.

1.4 Definitions, Acronyms, and Abbreviations

1. **GenreGlimpse:** The name of the web application.
2. **Image Analysis Model:** A component of the system responsible for performing analysis of visual information from movie posters.
3. **Text Analysis Model:** A component of the system responsible for performing analysis of textual information from movie posters.
4. **Concatenation Algorithm:** The integrated result that combines outputs from the Image Analysis Model and Text Analysis Model.

2. Functional Requirements

2.1 Product Perspective

GenreGlimpse operates as a standalone web application, interacting directly with users through a web interface. It integrates two core components: the Image Analysis Model, responsible for extracting visual features from movie posters, and the Text Analysis

Model, which extracts textual information. The Fusion Model combines the outputs of these components to predict movie genres accurately.

2.2 Product Features

1. **Poster Upload:** Users can upload movie posters through an intuitive web interface.
2. **Genre Prediction:** The system analyzes posters using the Image Analysis Model and Text Analysis Model, providing accurate genre predictions through the Fusion Model.
3. **Multi-Model Approach:** Integrates visual and textual data, enhancing genre prediction accuracy.

2.3 Operating Environment

- **Frontend:** Compatible with modern web browsers like Chrome, Firefox, Safari, and Edge.
- **Backend:** Python.

3. Non-Functional Requirements

3.1 Performance

The system should process and predict genres for uploaded posters within a maximum of 5 seconds to provide a responsive user experience.

3.2 Reliability

The system should have a high prediction accuracy, aiming for at least 90% accuracy in genre predictions.

3.3 Usability

The user interface should be intuitive and user-friendly, ensuring that users can easily upload posters and understand the genre predictions.

4. Technical Requirements

4.1 Technologies

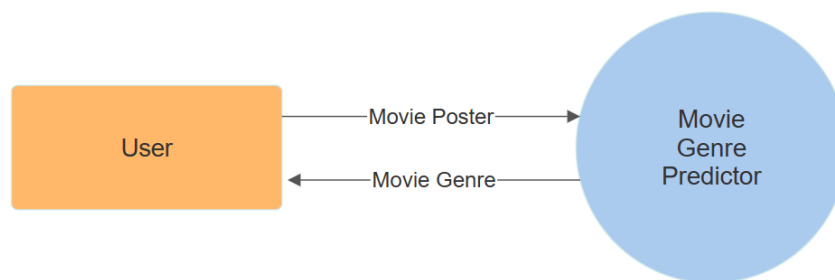
- **Frontend:** HTML, CSS, JavaScript, Gradio
- **Backend:** Python, PHP
- **Image Analysis:** Custom image analysis model
- **Text Analysis:** Google Cloud Vision, BERT model
- **Database:** Csv datasets

5. Conclusion

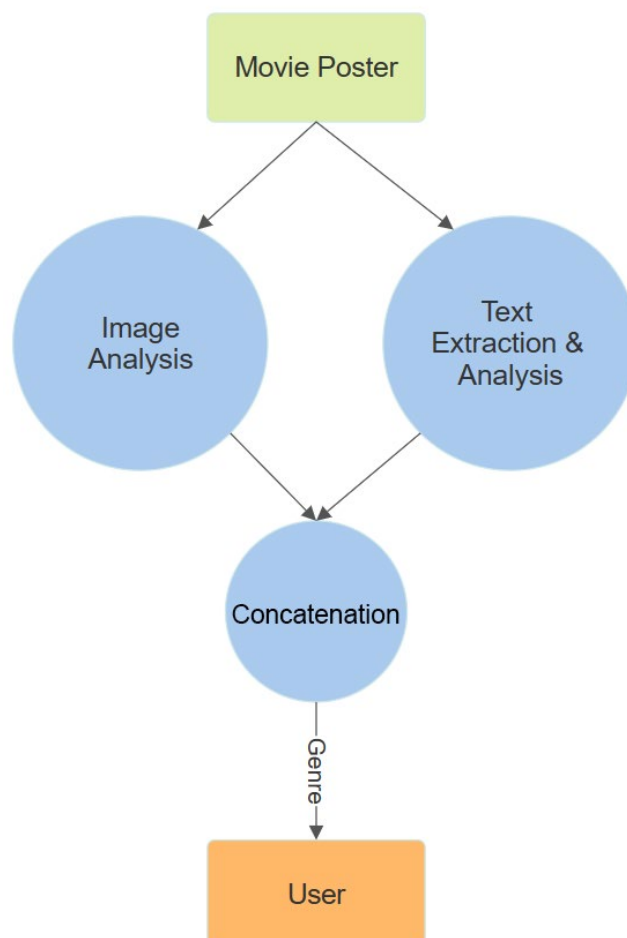
GenreGlimpse represents a cutting-edge solution for predicting movie genres based on posters. By integrating image analysis and text recognition techniques, the system aims to achieve superior accuracy in genre predictions. The successful implementation of this system will have significant implications for the entertainment industry, enhancing content recommendation systems, optimizing marketing strategies, and providing valuable insights for audience targeting.

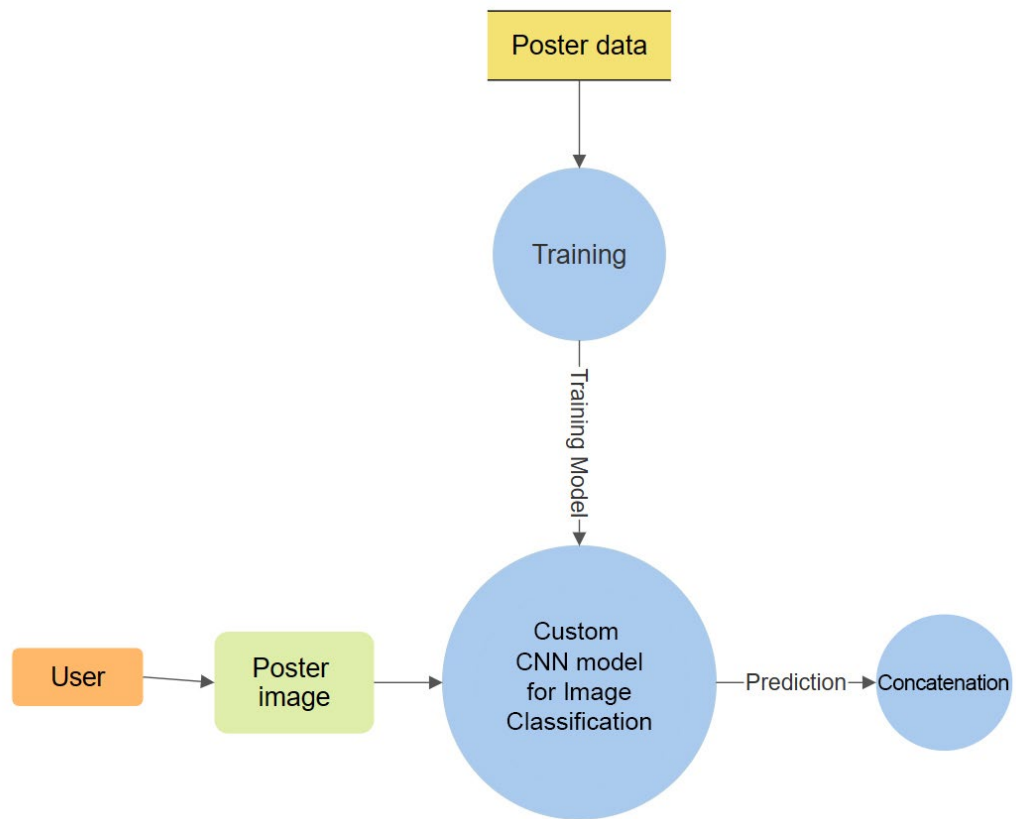
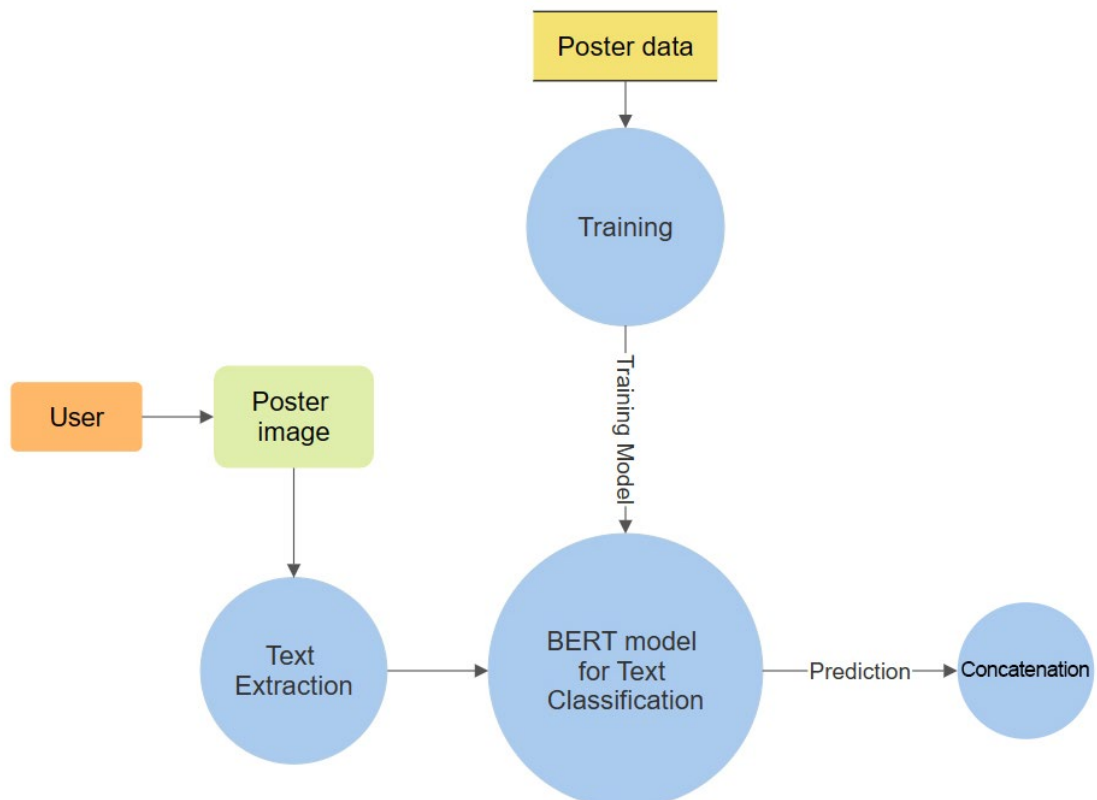
12.2. APPENDIX B (DFD)

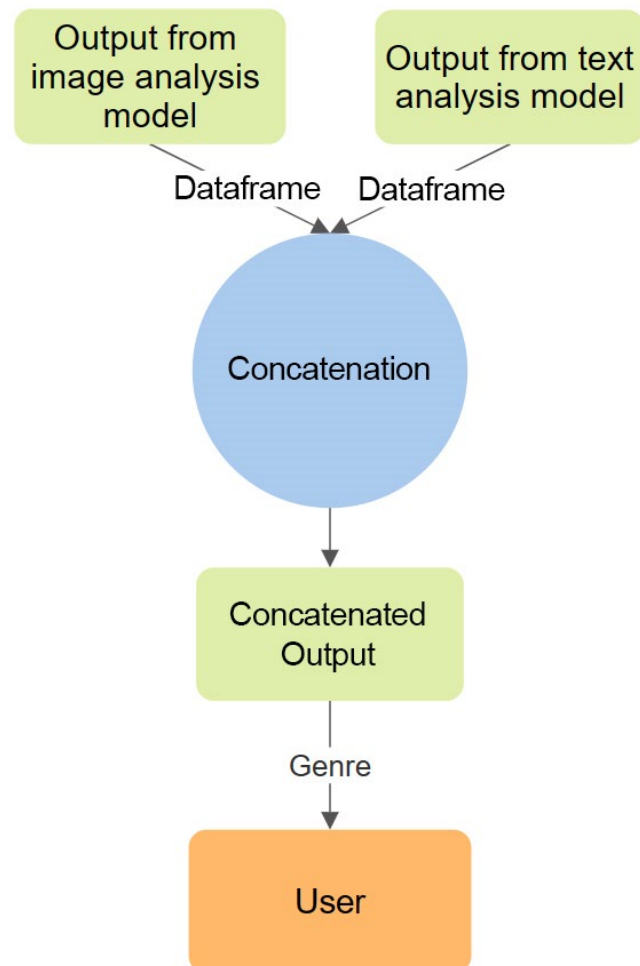
Level 0



Level 1



Level 2 : (Image Analysis Model)**Level 2 : (Text Analysis Model)**


Level 3




12.3. APPENDIX C (INTERFACE DESIGN)

GenreGlimpse

A multi-model approach to genre classification of movies from their posters

image




Drop Image Here
- or -
Click to Upload



Clear

Submit


Predicted Genre


Use via API  · Built with Gradio 




GenreGlimpse

A multi-model approach to genre classification of movies from their posters

image









Clear

Submit

Predicted Genre

['Sci-Fi', 'Crime']

Use via API  · Built with Gradio 

GenreGlimpse

A multi-model approach to genre classification of movies from their posters



Predicted Genre

[Action', 'Romance']

Use via API · Built with Gradio

12.4. APPENDIX D (CODE)

Data Pre-Processing

```
#code to split MovieGenre dataset into separate genres

import pandas as pd

for genre in ("Action", "Crime", "Comedy", "Sci-Fi", "Comedy", "Romance"):

    # Load the data

    movies = pd.read_csv("datasets/MovieGenre.csv", encoding='latin-1')

    movies = movies.dropna()

    # Extract movies belonging to the specific genre

    split_movies = movies[movies["Genre"].str.contains(genre)]

    split_movies['Genre'] = f'{genre}'

    #Save to CSV

    split_movies.to_csv(f'datasets/extracted/{genre}.csv', index=False)


#code to merge the split datasets (After normalizing their row number manually)

import pandas as pd

import glob

# Specify the pattern of your CSV files

file_pattern = 'datasets/extracted/*.csv'

# Find all CSV files that match the pattern

csv_files = glob.glob(file_pattern)

# Initialize an empty list to store DataFrames

dfs = []

# Read each CSV file and append it to the list

for csv_file in csv_files:
```

```

df = pd.read_csv(csv_file)

dfs.append(df)

# Concatenate all DataFrames along the row axis
merged_df = pd.concat(dfs, ignore_index=True, sort=False)

# Drop duplicated imdbid rows
merged_df.drop_duplicates(subset='imdbId', keep='first', inplace=True)

# Save the result to a new CSV file
merged_df.to_csv('datasets/merged_movies.csv', index=False)


#code to shuffle the rows of new dataset
import pandas as pd

df = pd.read_csv('datasets/merged_movies.csv')

shuffled_df = df.sample(frac=1)

shuffled_df.to_csv('datasets/merged_movies.csv', index=False)

```

Training the CNN model for Image Classification

```

#importing required libraries

import numpy as np

import os

from IPython.display import Image as disImage

from PIL import Image

import glob

import keras

import tensorflow as tf

```

```

# Make a list of movie IDs

#This list will help indexing of the dataset and keep list of images matching with their genre
directory = 'GenreGlimpse/MovieMatrix/posters/'

id_list = []

for filename in glob.iglob(f'{directory}/*'):

    f = filename.split('/')

    im = f[-1][:-4]

    id_list.append(im)

# Iterate over id_list to get posters as np arrays

shape = (187, 125, 3)

same_dims = True

image_array = []

for i in id_list:

    file = 'GenreGlimpse/MovieMatrix/posters/'+i+'.jpg'

    img= Image.open(file)

    np_img = np.array(img)

    image_array.append(np_img)

    if shape != np_img.shape:

        same_dims= False

        print("different shapes")

        print(np_img.shape, 'ID', i)

if same_dims:

    print("All posters are of same dimentions already.\nAll images have been converted to
numpy array")

import csv

id_genre_dict = {}

```

```

title_dict = {}

with open('GenreGlimpse/MovieMatrix/genre.csv', newline="", encoding = "ISO-8859-1") as
csvfile:

    reader = csv.reader(csvfile)

    for row in reader:

        movie_id = row[0]

        gen = row[1]

        gen_split = gen.split('|')

        id_genre_dict.update({movie_id:gen_split})


#print the present genres
genres = id_genre_dict.values()

genres = [item for sublist in genres for item in sublist]

genres = set(genres)

genres = list(genres)

genres.remove('genre')

genres.sort()

print(genres)


#multi-hot encoding the genres
multi_hot_y=[]

for i in id_list:

    y_i = np.zeros(len(genres))

    gens = id_genre_dict[i]

    for g in gens:

        y_i[genres.index(g)] = 1

```

```

multi_hot_y.append(y_i)

# Making final X and Y
img = np.asarray(image_array) # X
y = np.asarray(multi_hot_y)

#Defining the CNN model
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from keras.layers import Flatten, Dense
model = Sequential([
    Conv2D(32, kernel_size=3, padding='same', input_shape=(187, 125, 3), activation='relu'),
    Conv2D(32, kernel_size=3, activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),
    Conv2D(64, kernel_size=4, padding='same', activation='relu'),
    Conv2D(64, kernel_size=4, activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(6, activation='sigmoid')
])

opt = keras.optimizers.RMSprop(learning_rate=0.0001)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])

```



```
#fitting the model

model.fit(img, y, epochs = 25, validation_split=0.1, batch_size=32)

#saving the model

model.save('image-model.keras')
```

Training the BERT model for Text Classification

```
#importing required libraries

import torch

from tqdm.notebook import tqdm

from transformers import BertTokenizer

from torch.utils.data import TensorDataset

from transformers import BertForSequenceClassification

import pandas as pd


#loading and cleaning the dataset

df = pd.read_csv('datasets/merged_movies.csv', encoding='latin-1')

df = df.drop(['imdbId', 'Imdb Link', 'IMDB Score', 'Poster'], axis=1)

possible_labels = df.Genre.unique()

label_dict = {}

for index, possible_label in enumerate(possible_labels):

    label_dict[possible_label] = index

label_dict

df['label'] = df.Genre.replace(label_dict)

#splitting to train and validation data

from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(df.index.values,
```

```

        df.label.values,

        test_size=0.15,

        random_state=42,

        stratify=df.label.values)

df['data_type'] = ['not_set']*df.shape[0]

df.loc[X_train, 'data_type'] = 'train'

#define the bert tokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',

        do_lower_case=True)

#encode the train data

encoded_data_train = tokenizer.batch_encode_plus(

    df[df.data_type=='train'].Title.values.astype(str),

    add_special_tokens=True,

    return_attention_mask=True,

    pad_to_max_length=True,

    max_length=256,

    return_tensors='pt'

)

input_ids_train = encoded_data_train['input_ids']

attention_masks_train = encoded_data_train['attention_mask']

labels_train = torch.tensor(df[df.data_type=='train'].label.values)

#define the BERT model

model = BertForSequenceClassification.from_pretrained("bert-base-uncased",

        num_labels=len(label_dict),

        output_attentions=False,

        output_hidden_states=False)

```

```

output_size = model.classifier.out_features

from torch.utils.data import DataLoader, RandomSampler

batch_size = 3

dataloader_train = DataLoader(dataset_train,

                               sampler=RandomSampler(dataset_train),

                               batch_size=batch_size)

from transformers import AdamW, get_linear_schedule_with_warmup

optimizer = AdamW(model.parameters(),

                   lr=1e-5,

                   eps=1e-8)

epochs = 5

scheduler = get_linear_schedule_with_warmup(optimizer,

                                              num_warmup_steps=0,

                                              num_training_steps=len(dataloader_train)*epochs)

from sklearn.metrics import f1_score

def f1_score_func(preds, labels):

    preds_flat = np.argmax(preds, axis=1).flatten()

    labels_flat = labels.flatten()

    return f1_score(labels_flat, preds_flat, average='weighted')

def accuracy_per_class(preds, labels):

    label_dict_inverse = {v: k for k, v in label_dict.items()}

    preds_flat = np.argmax(preds, axis=1).flatten()

    labels_flat = labels.flatten()

    for label in np.unique(labels_flat):

        y_preds = preds_flat[labels_flat==label]

        y_true = labels_flat[labels_flat==label]

```

```

    print(f'Class: {label_dict_inverse[label]}')

    print(f'Accuracy: {len(y_preds[y_preds==label])}/{len(y_true)}\n')

import random

import numpy as np

seed_val = 17

random.seed(seed_val)

np.random.seed(seed_val)

torch.manual_seed(seed_val)

torch.cuda.manual_seed_all(seed_val)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

model.to(device)

#fine-tune the BERT model

import os

for epoch in tqdm(range(1, epochs+1)):

    model.train()

    loss_train_total = 0

    progress_bar = tqdm(dataloader_train, desc='Epoch  {:1d}'.format(epoch), leave=False,
disable=False)

    for batch in progress_bar:

        model.zero_grad()

        batch = tuple(b.to(device) for b in batch)

        inputs = {'input_ids':    batch[0],

                  'attention_mask': batch[1],

                  'labels':      batch[2],

                  }

```

```

outputs = model(**inputs)

loss = outputs[0]

loss_train_total += loss.item()

loss.backward()

torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

optimizer.step()

scheduler.step()

progress_bar.set_postfix({'training_loss': '{:.3f}'.format(loss.item()/len(batch))})


# Create the data_volume directory if it does not exist
os.makedirs('data_volume', exist_ok=True)


#save each epoch

torch.save(model.state_dict(), f'data_volume/finetuned_BERT_epoch_{epoch}.model')

tqdm.write(f'\nEpoch {epoch}')

loss_train_avg = loss_train_total/len(dataloader_train)

tqdm.write(f'Training loss: {loss_train_avg}')

```

Main function including User Interface

```

from google.cloud import vision

from transformers import BertTokenizer,BertForSequenceClassification

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

from google.cloud import vision

from google.cloud.vision_v1 import types

from PIL import Image

```

```

import io

import torch

import numpy as np

import gradio as gr

import tensorflow as tf

import pandas as pd


#Load Modals

img_model = tf.keras.models.load_model('image-model.keras')

txt_model = BertForSequenceClassification.from_pretrained("bert-base-uncased",

                                                         num_labels=6,

                                                         output_attentions=False,

                                                         output_hidden_states=False)


# Load your fine-tuned BERT model

txt_model.load_state_dict(torch.load('data_volume/finetuned_BERT_epoch_5.model',

map_location=torch.device('cpu'))))

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

txt_model.to(device)


def concat(genre_scores, formatted_genre_scores):

    for i in range(len(genre_scores)):

        genre_scores[i] = (genre_scores[i] + formatted_genre_scores[i])/2


label_dict = {'Action': 0, 'Comedy': 1, 'Horror': 2, 'Romance': 3, 'Sci-Fi': 4, 'Crime': 5}

```

```

# Combine genre scores with their corresponding labels (indices)

scored_genres = list(zip(genre_scores, label_dict.keys()))

# Sort by score in descending order (largest first)

sorted_genres = sorted(scored_genres, key=lambda x: x[0], reverse=True)

# Extract labels of the top 3 genres

top_2_labels = [genre for score, genre in sorted_genres[:2]]

return top_2_labels


def image_model(image):

    # Convert the PIL image to a numpy array

    image_ndarray = np.array(image)

    # Convert the ndarray to a PIL image

    img = Image.fromarray(image_ndarray)

    img = img.resize((125, 187))

    np_img = np.array(img)

    test_y = np_img

    ty = np.reshape(test_y, (1,187, 125, 3))

    pred_genre = img_model.predict(ty)

    predicted_probabilities = pred_genre[0]

    genre_scores = np.exp(predicted_probabilities)/np.sum(np.exp(predicted_probabilities)) * 6

    return genre_scores


def text_model(extracted_text):

    label_dict = {'Action': 0, 'Comedy': 1, 'Horror': 2, 'Romance': 3, 'Sci-Fi': 4, 'Crime': 5}

    # Load your tokenizer

    tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',

```

```

do_lower_case=True)

# Create your test dataset

test = [extracted_text]

test_df = pd.DataFrame(test)

test_df.columns=['Column_name']


# Tokenize your test dataset

encoded_data_test = tokenizer.batch_encode_plus(
    test_df['Column_name'].values.astype(str),
    add_special_tokens=True,
    return_attention_mask=True,
    padding='longest',
    max_length=256,
    return_tensors='pt'
)

input_ids_test = encoded_data_test['input_ids']

attention_masks_test = encoded_data_test['attention_mask']


# Create your test dataloader

dataset_test = TensorDataset(input_ids_test, attention_masks_test)

dataloader_test = DataLoader(dataset_test,

                             sampler=SequentialSampler(dataset_test),

                             )

# Make predictions

pred_label=[]

for batch in dataloader_test:

```



```

batch = tuple(b.to(device) for b in batch)

inputs = {'input_ids': batch[0],
          'attention_mask': batch[1],
          }

with torch.no_grad():
    outputs = txt_model(**inputs)
    logits = outputs[0]
    logits = logits.detach().cpu().numpy()
    print(logits)
    print(label_dict)

import torch.nn.functional as F

probs = F.softmax(torch.tensor(logits), dim=-1).numpy()[0]

# Format probabilities as percentages with two decimal places
formatted_probs = [float(format(prob * 100)) for prob in probs]

print(formatted_probs)

import numpy as np

predicted_probabilities = formatted_probs

genre_scores = np.exp(predicted_probabilities)/np.sum(np.exp(predicted_probabilities)) * 6

# Format scores with two decimal places
formatted_genre_scores = [float('{:.7f}'.format(score)) for score in genre_scores]

print("Genre scores:", formatted_genre_scores)

return formatted_genre_scores

def text_extractor(image):

    # Create a new instance of the ImageAnnotatorClient class

```

```

client = vision.ImageAnnotatorClient()

# Convert the PIL image to a numpy array
image_ndarray = np.array(image)

# Convert the ndarray to a PIL image
pil_image = Image.fromarray(image_ndarray)

# Save the PIL image as a JPEG or PNG image bytes
image_bytes = io.BytesIO()

pil_image.save(image_bytes, format='JPEG') # or 'PNG'

image_bytes.seek(0)

# Load the image from bytes
with io.BytesIO(image_bytes.getvalue()) as image_file:
    content = image_file.read()

# Construct the image object
image = types.Image(content=content)

# Construct the feature object
feature = types.Feature(type_=types.Feature.Type.TEXT_DETECTION)

# Construct the request object
request = types.AnnotateImageRequest(image=image, features=[feature])

# Perform the image annotation
response = client.annotate_image(request=request)

# Get the full text from the response
text = response.full_text_annotation.text

# Return the text
return text

def main(image):

```

```
text = text_model(text_extractor(image))

Concat = concat(image_model(image), text)

return Concat


# Define the Gradio interface

interface = gr.Interface(

    fn=main,

    inputs="image",

    outputs=[gr.Textbox(label="Predicted Genre")],

    title="GenreGlimpse",

    description="A multi-model approach to genre classification of movies from their posters",

    theme=gr.themes.Soft(primary_hue=gr.themes.colors.blue),

    allow_flagging="never"

)


# Launch the Gradio application

interface.launch(share=True)
```

BIBLIOGRAPHY AND REFERENCES

13. **BIBLIOGRAPHY AND REFERENCES**

1. Barney, Gabriel, and Kris Kaya. "Predicting genre from movie posters." Stanford CS 229: Machine Learning (2019).
2. Zhang, Min-Ling, and Zhi-Hua Zhou. "ML-KNN: A lazy learning approach to multi-label learning." Pattern recognition 40, no. 7 (2007):2038-2048.
3. Kaggle. The Movies Dataset, 2017. <https://www.kaggle.com/rounakbanik/the-movies-dataset/kernels>.
4. Kundalia, Kaushil, Yash Patel, and Manan Shah. "Multi-label movie genre detection from a movie poster using knowledge transfer learning." Augmented Human Research 5 (2020): 1-9.
5. Chu, Wei-Ta, and Hung-Jui Guo. "Movie genre classification based on poster images with deep neural networks." In proceedings of the workshop on multimodal understanding of social, affective and subjective attributes, pp. 39-45. 2017.
6. Fagerholm, Cecilia. "The use of color in movie poster design: an analysis of four genres." METROPOLIA (2009)
7. Data Flow Diagram (DFD), <https://www.geeksforgeeks.org/what-is-dfddata-flow-diagram/>
8. Software Requirement Specification (SRS), <https://www.geeksforgeeks.org/software-requirement-specification-srs-format/>
9. Transformers, <https://huggingface.co/docs/transformers/en/index>
10. Pytorch, <https://pytorch.org/>
11. Tensorflow, <https://www.tensorflow.org/>