CST3130 Advanced Web Development with Big Data

Waseemah Moedeen

16/04/2024

Silas Keru Kahumbu
M00854840

1 CONTENTS

2 INTRODUCTION

This report presents the design and implementation of a movie genre prediction website. The website utilizes the TMDB (The Movie Database) API to retrieve movie data, which is then stored in DynamoDB. It features functionality for displaying predictions of movie genres, which are augmented with synthetic data and sentiment analysis to enhance accuracy and user experience.

Linked to the website: http://cst3130-m00854840.s3-website-us-east-1.amazonaws.com/
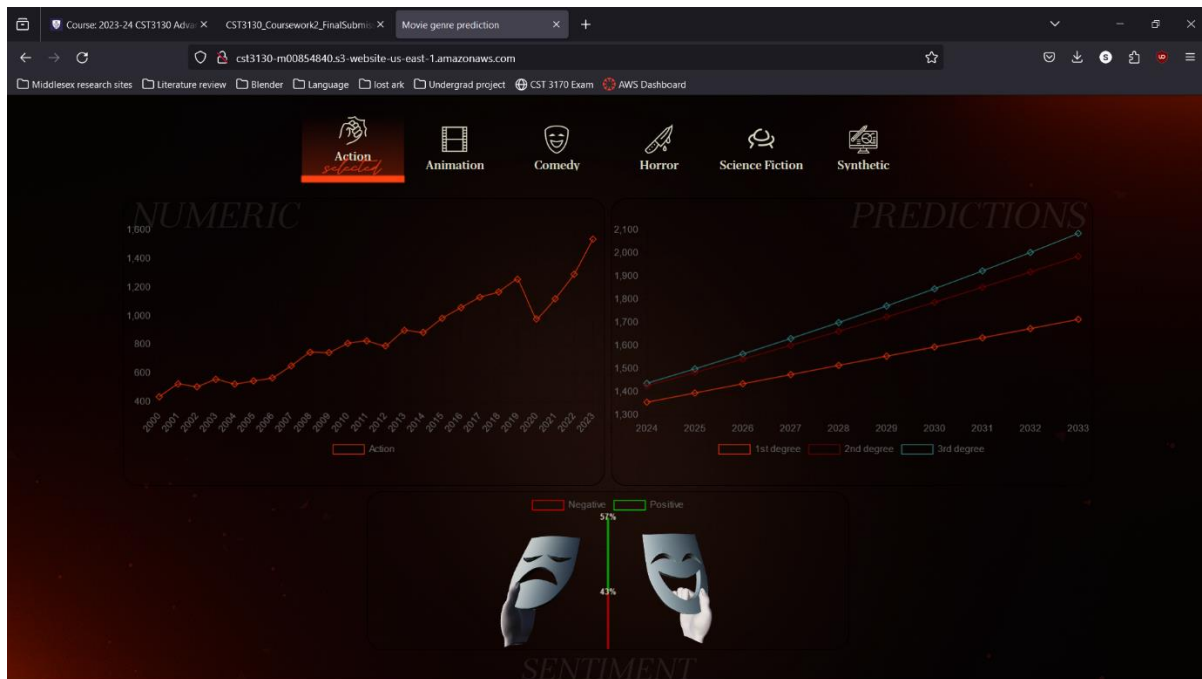
3 FRONT-END SCREENSHOTS
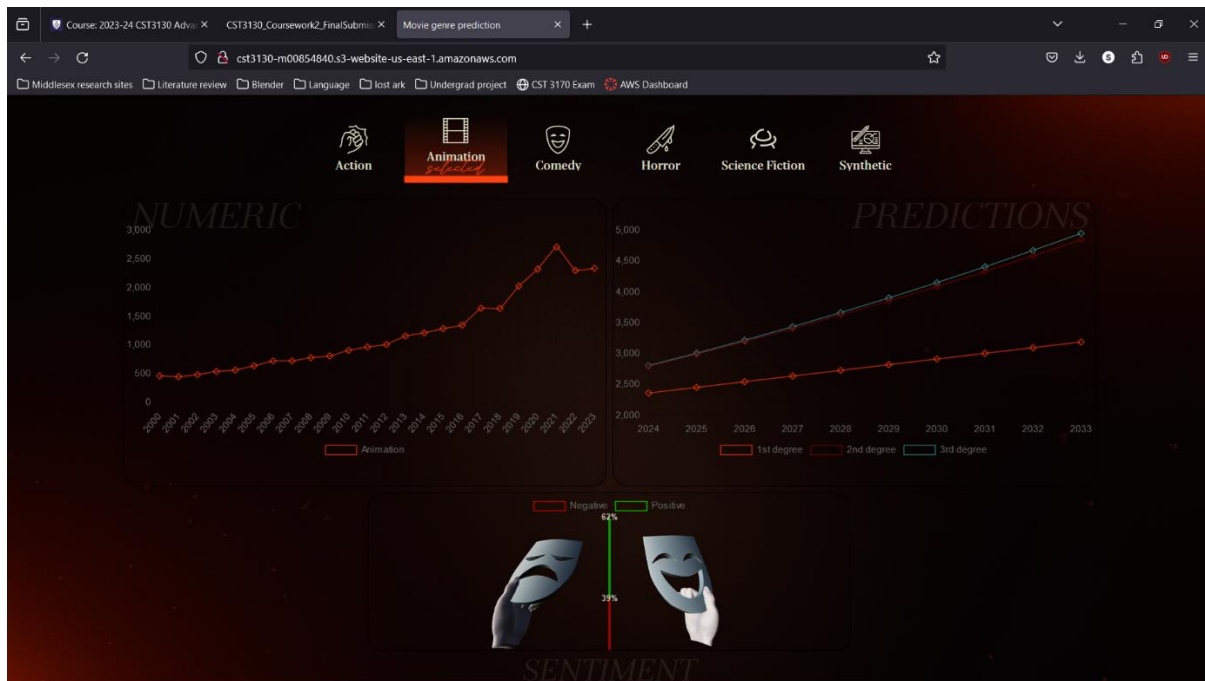

Figure 1 Action section
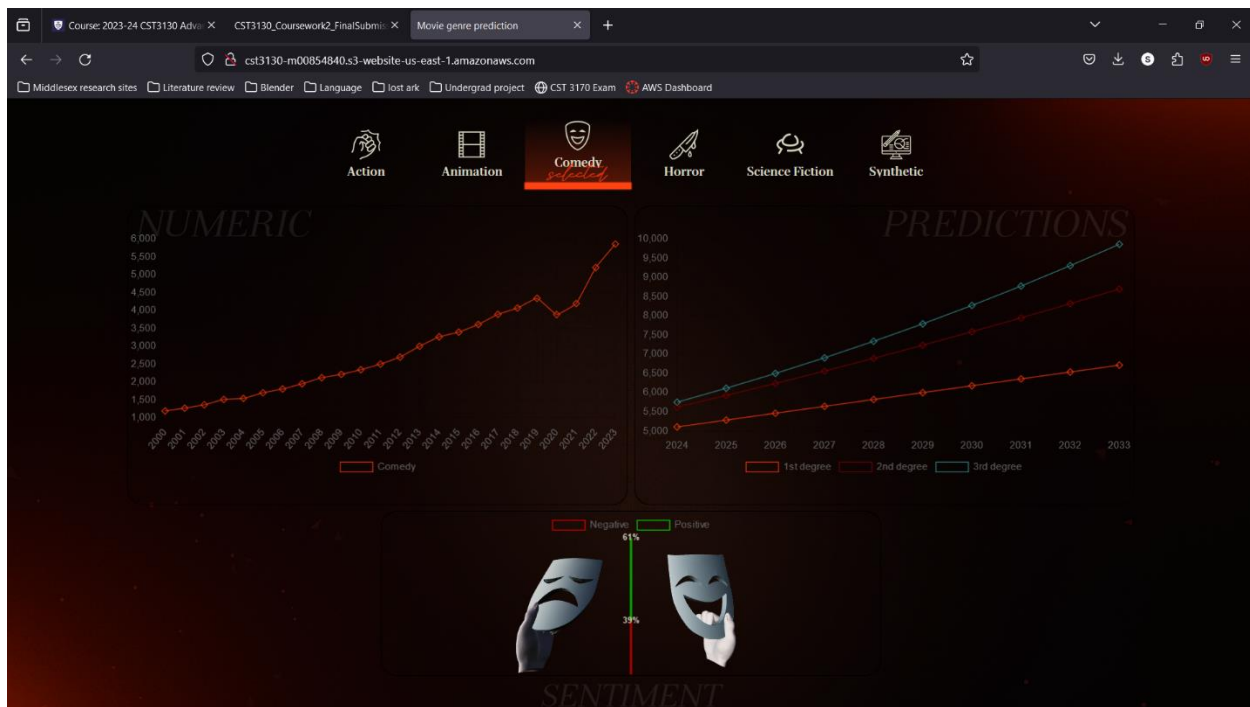
Figure 2 Animation section
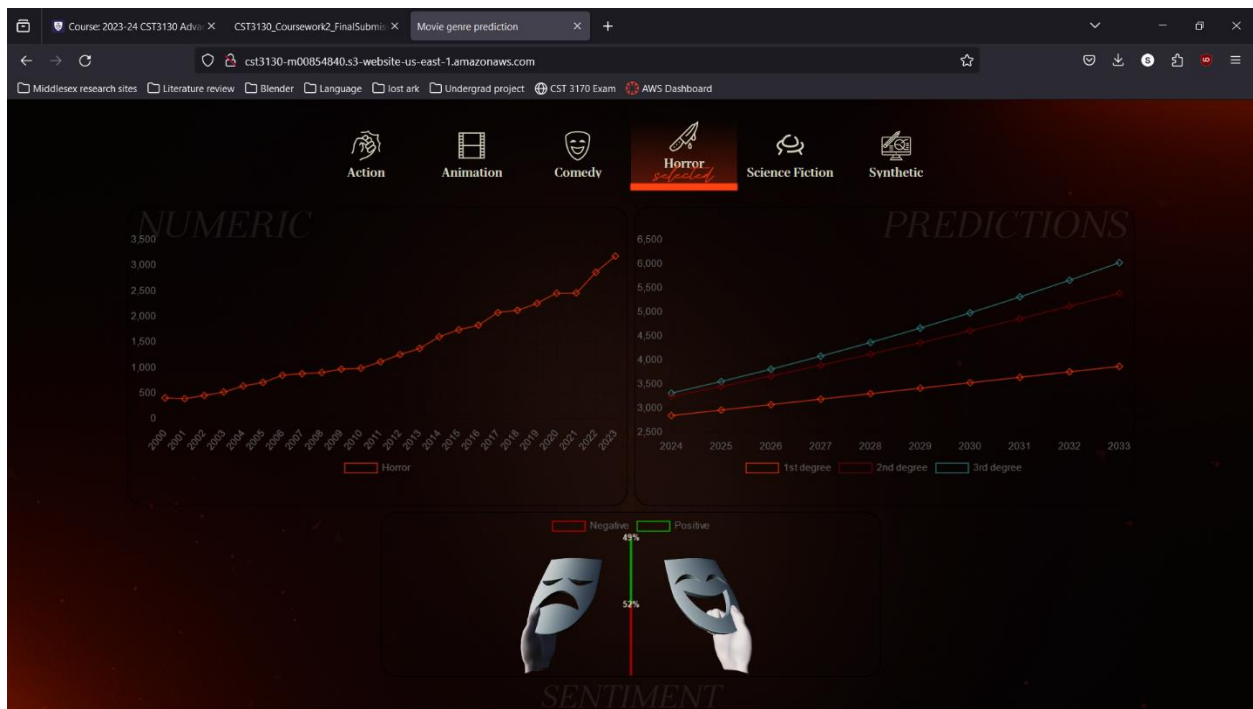

Figure 3 Comedy section
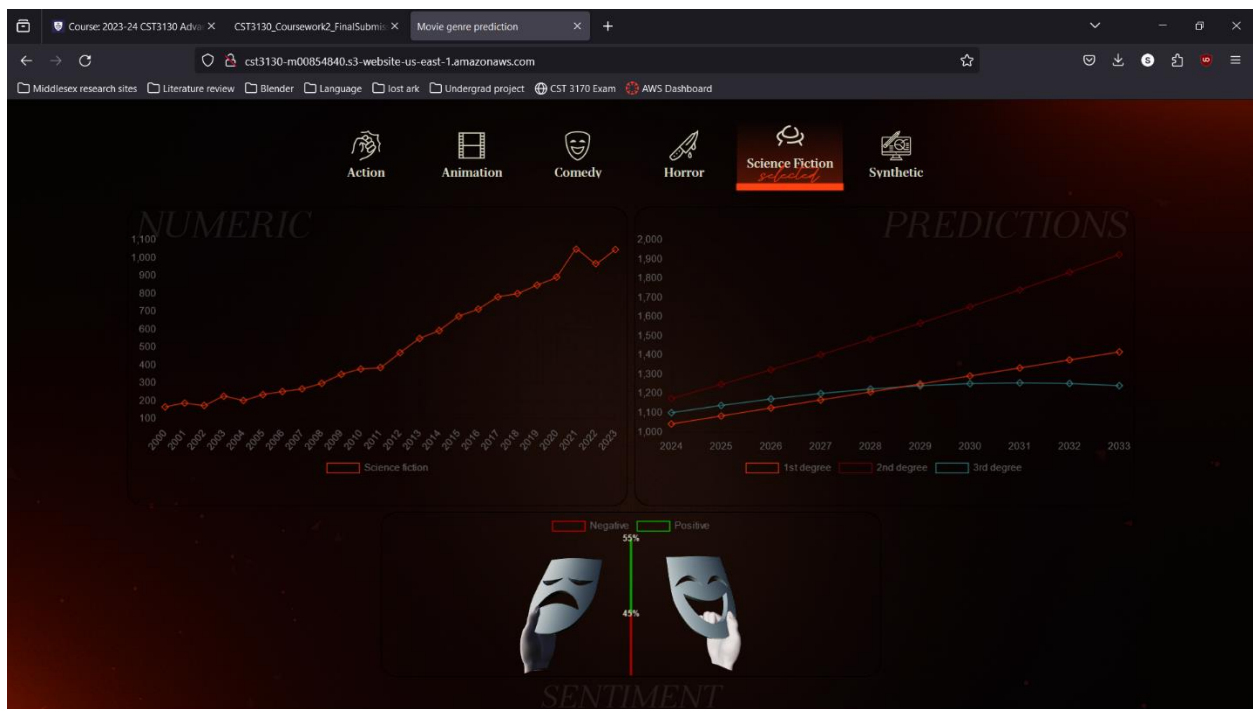
Figure 4 Horror section

Figure 5 Science fiction section

Figure 6 Synthetic section

4 ARCHITECTURE

This section presents and explains the architectural diagram and its corresponding parts, which outline the framework of the website from text/sentiment uploading to AWS services.

The program outside of the client from the developer side initiates with the Webservice scraper, which accesses the TMDB API to retrieve data on movies released in specific genres during designated years. This data is then sent to the sentiment scraper, which forwards it to a text-processing API. This API evaluates and returns the proportions of negative, neutral, and positive sentiments in the text and sends it back to the scraper. Concurrently, as each genre from each year is finished processing, the information is uploaded to DynamoDB.

The middle section of the diagram details client interactions. Clients send action requests to the WebSocket via specific routes. The "no_of_movies" route retrieves numeric and sentiment data by filtering through DynamoDB using the "genreName" specified by the client. The "numPrediction" route involves predictive analytics, where a polynomial regression model predicts future values of numeric data at the 1st, 2nd, and 3rd degrees. This model is chosen due to the limited number of numeric data points available—approximately 87 instances generated from SageMaker, as opposed to the required 300. After processing, the data is returned to the client and displayed using React Chart.js.

Finally the left section of the diagram illustrates the setup of a SageMaker endpoint for synthetic data. Initially, training files are uploaded to an S3 bucket. These files are then utilized by a SageMaker training job to develop a model and establish an endpoint. This endpoint is subsequently accessed via Postman to generate predictive data for the synthetic dataset, which is then integrated into the webpage.
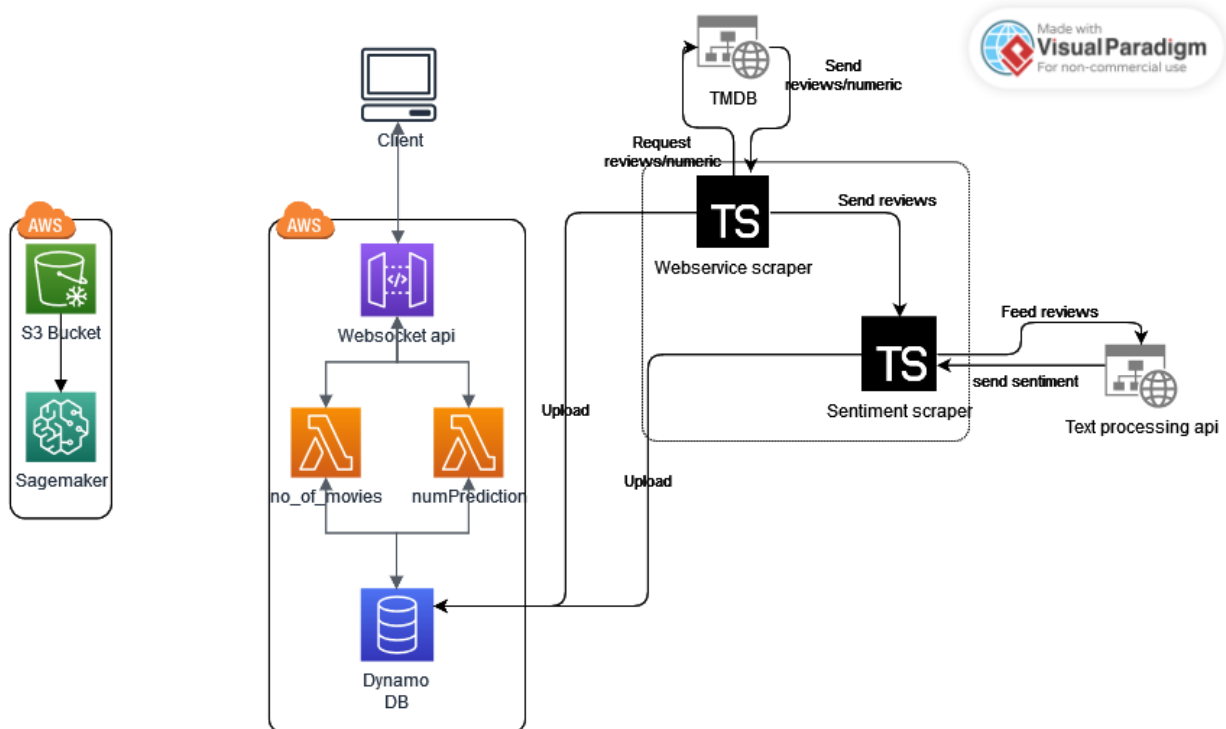

Figure 7 Architecture diagram