Sentiment Analysis of IMDb Movie Reviews through the use of Convolutional Neural Networks

Team 7

Kavi Sharma

Gunwoo Lee

April 25th 2024

IS 619

Abstract

This study investigates the application of Convolutional Neural Networks (CNNs) to perform sentiment analysis on the IMDb movie review dataset, containing 25,000 labeled reviews. The objective is to distinguish between positive and negative sentiments expressed in the reviews, leveraging the power of CNNs, which are predominantly celebrated for their performance in image recognition but are also highly effective for sequence data due to their ability to capture local dependencies. Utilizing TensorFlow, we developed a model that processes textual data encoded as word indexes, applying convolution and pooling layers followed by dropout layers to mitigate overfitting. The model was trained on a subset of the dataset, with the validation process revealing a significant generalization capability, achieving a training accuracy of 96.4% and a validation accuracy of 86.61%. The study further explores the challenges of natural language processing, such as managing varied and noisy data, and discusses the potential of extending this model to broader applications beyond movie reviews. The results confirm that CNNs can effectively discern intricate patterns in text data, highlighting their versatility and robustness in handling sentiment analysis tasks.

Introduction:

In the digital age, user-generated content is a common resource for sentiment analysis. This can allow users to gain access to public opinions and preferences. One such source is the Internet Movie Database (IMDb), which hosts a vast collection of movie reviews. Understanding the sentiments expressed in these reviews can provide valuable insights for filmmakers, critics, and audiences alike. Analyzing historical data on movie genres, ratings, box office performance, and other relevant factors can also offer stakeholders, valuable insights and forecasts that assist in making informed decisions as well. This kind of information can also be used globally as well as movies are a pastime that people all over the world enjoy.

The IMDb dataset used in this study comprises 25,000 movie reviews, each labeled with sentiment (positive/negative). These reviews have been preprocessed and encoded as lists of word indexes, with each integer representing a word's frequency in the dataset. This encoding allows for efficient filtering operations, such as focusing on the top 10,000 most common words while excluding the top 20 (Keras, 2024).

By analyzing this dataset, we aim to explore the characteristics of positive and negative movie reviews, identify key features that contribute to each sentiment, and develop a classification model that can accurately predict the sentiment of a given review. Additionally, we seek to investigate the impact of various text analysis techniques, such as tokenization and vectorization, on the performance of the classification model.

Project Schedule:

Phase 1: Project Proposal Submission/Data Collection and Preparation

Duration: 1 week

Activities: Finalize project topic, outline objectives, identify data sources, and submit proposals, gather data from selected sources, clean and preprocess data, handle missing values and outliers, and prepare the dataset for analysis.

Phase 3: Exploratory Data Analysis (EDA)

Duration: 1 week

Activities: Perform descriptive statistics, create visualizations, and identify patterns and correlations in the data.

Phase 4: Model Development and Training

Duration: 2 weeks

Activities: Select and train various predictive models, and tune model parameters.

Phase 5: Model Evaluation and Selection

Duration: 1 week

Activities: Evaluate model performance using appropriate metrics, compare different models, and select the best-performing model.

Phase 6: Report Writing and Revision/Presentation Preparation

Duration: 1.5 weeks

Activities: Document methodology, analyses, model development, results, and insights in a comprehensive report. Revise based on feedback. Prepare a slide deck summarizing the project's purpose, methodology, key findings, and conclusions.

Phase 8: Final Report Submission and Presentation

Duration: On the due date

Activities: Submit the final report and deliver the presentation to showcase the project findings and recommendations.

Responsibilities

Gunwoo

- Load the dataset into the model and clean it for the model.
- Validate and evaluate the model's training

Kavi

- Develop a model and its parameters
- Fine-tune model to reduce val_loss and maximize val_accuracy

Method/Model

<u>Breakdown of a Convolutional Neural Network:</u>

The model architecture used for our analysis is called Convolutional Neural Networks (CNN). To setup our model architecture we utilized TensorFlow. TensorFlow is an open-source

machine learning framework developed by Google. It provides users with libraries, tools, as well as other resources for users to build and deploy machine learning models.

TensorFlow can be used for both training a movie sentiment classification model and for analyzing IMDb ratings, genres, and popularity to make investment decisions. In our case, we developed a Convolutional Neural Network (CNN) due to its strengths in capturing local patterns such as spatial features hierarchy through convolution and pooling. CNN models thrive in image recognition where key aspects of an image can be given more importance. Convolution is done through matrix multiplication where a Kernel, typically a 2x2 or a 3x3 matrix, scans across the input matrix one column and then one row at a time (Verma, 2019). Next is the MaxPooling layer where the highest dot product value in a feature is summarized. This allows the model to still capture the important features while also reducing the dimensionality of the matrix (Verma, 2019). The last important layer is the padding layer. Padding is the inserting of zero's on the parameter of the matrix, this allows the model to gain more insight on features inside of the buffer zone.

The use of convolutional and max-pooling layers creates the framework of the Sentiment Analysis model but results in an overly confident model. Through the introduction of Dropout layers, we can randomly select neurons and set their input units to zero during each step. This means that each update to the model during training is performed with a different "thinned" version of the network. By randomly dropping units during training, it ensures that the model does not correlate the noise in the training data as actual patterns. Lastly, flattening occurs, this involves transforming the processed matrix into a single column to be fed into the model for processing.

Depicted below is our rating prediction model. We utilized a batch size of 64 with 15 epochs.

```
batch_size = 64
epochs = 15
```

```
#Assuming the input shape is determined by the number of unique genres
num_genres = train_X.shape[1]

#Define the model architecture
rating_model = Sequential()
rating_model.add(Conv1D(32, 3, padding='same', activation='relu', input_shape=(train_X.shape[1], 1)))
rating_model.add(MaxPooling1D(pool_size=2, padding='same'))
rating_model.add(Dropout(0.25))
rating_model.add(Conv1D(64, 3, padding='same', activation='relu'))
rating_model.add(LeakyReLU(alpha=0.1))
rating_model.add(MaxPooling1D(pool_size=2, padding='same'))
rating_model.add(Dropout(0.25))
rating_model.add(Conv1D(128, 3, padding='same', activation='relu'))
rating_model.add(LeakyReLU(alpha=0.1))
rating_model.add(MaxPooling1D(pool_size=2, padding='same'))
rating_model.add(Dropout(0.4))
rating_model.add(Flatten())
rating_model.add(Dense(128, activation='linear'))
rating_model.add(LeakyReLU(alpha=0.1))
rating_model.add(Dropout(0.3))
#output layer: Single neuron, since we're predicting a single continuous value(movie rating)
rating_model.add(Dense(1, activation='linear'))

#compile the model for a regression task
rating_model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])

#model summary to understand its structure
rating_model.summary()
```

Depicted below is our sentiment analysis model. We utilized a batch size of 128, 15 epochs, and a learning rate of 0.005.

```
[ ]  1 batch_size = 128
     2 epochs = 15

[ ]  1 learning_rate = 0.005
     2 adam_optimizer = Adam(learning_rate=learning_rate)

     1 #Define the model architecture
     2 sentiment_model = Sequential()
     3 sentiment_model.add(Embedding(top_words, 32, input_length=max_words))
     4 sentiment_model.add(Dropout(0.2))
     5 sentiment_model.add(Conv1D(32, 3, padding='same', activation='relu'))
     6 sentiment_model.add(MaxPooling1D(pool_size=2, padding='same'))
     7 sentiment_model.add(Dropout(0.3))
     8 sentiment_model.add(Conv1D(64, 3, padding='same', activation='relu'))
     9 sentiment_model.add(LeakyReLU(alpha=0.1))
    10 sentiment_model.add(MaxPooling1D(pool_size=2, padding='same'))
    11 sentiment_model.add(Dropout(0.4))
    12 sentiment_model.add(Conv1D(128, 3, padding='same', activation='relu'))
    13 sentiment_model.add(LeakyReLU(alpha=0.1))
    14 sentiment_model.add(MaxPooling1D(pool_size=2, padding='same'))
    15 sentiment_model.add(Dropout(0.5))
    16 sentiment_model.add(Flatten())
    17 sentiment_model.add(Dense(128, activation='linear'))
    18 sentiment_model.add(LeakyReLU(alpha=0.1))
    19 sentiment_model.add(Dropout(0.5))
    20 #output layer: Single neuron, since we're predicting a single continuous value
    21 sentiment_model.add(Dense(1, activation='sigmoid'))
    22
    23 #compile the model for a regression task
    24 sentiment_model.compile(optimizer=adam_optimizer, loss='binary_crossentropy', metrics=['accuracy'])
    25
    26 #model summary to understand its structure
    27 sentiment_model.summary()
```

Processing a Moving Rating Prediction Model:

To find the correlation between movie ratings and movie genres, the data first needed to be sourced and cleaned. The data was gathered from the IMDb open-source database (https://datasets.imdbws.com/). The data was cleaned to remove the names, production start/end years, and mediums other than movies (video games, plays, TV shows, and shorts) resulting in 150,000 entries. Lastly, the genres were One-Hot Encoded, this creates separate columns for each genre class and the data is represented as either a (1), genre is present, or a (0), genre is not present. This converts textual data into numerical data so that the convolutional model can gather insight. Ratings were converted from type Integer to type Float and then divided by 10 to mirror

the model's input range of 0 to 1. The data was split randomly using the train_test_split function, with the X inputs corresponding with genres and the Y outputs corresponding with an average rating, using the random state of 42.

Defining dataset Classes / One-Hot Encode / Reshaping / Splitting the dataset

```
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
# Assuming 'imdb' is your DataFrame and has been loaded correctly

# Convert genres from string to actual lists
imdb['genres'] = imdb['genres'].apply(eval)

# Initialize MultiLabelBinarizer to transform the genres into a binary format
mlb = MultiLabelBinarizer()
genres_encoded = mlb.fit_transform(imdb['genres'])

# Split the dataset into training and testing sets
# train_X now represents genre encoded data (input features)
# train_Y now represents averageRating (target variable)
train_X, test_X, train_Y, test_Y = train_test_split(genres_encoded, imdb['averageRating'], test_size=0.2, random_state=42)
```

Processing a Sentiment Analysis Model:

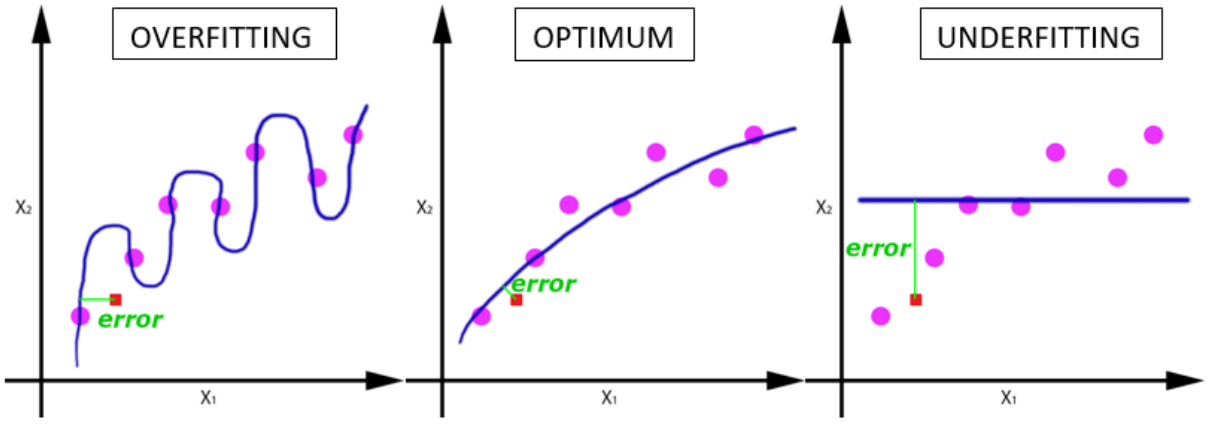
The data for the Sentiment Analysis was sourced from the Keras database (https://ai.stanford.edu/~amaas/data/sentiment/). There we imported the preprocessed review data. The data was segmented with Positive reviews notated with a (1) and Negative reviews notated with a (0). The data was then divided into a training set and a test set each containing 25,000 reviews for a total of 50,000 reviews. The average review length is 234 words with a standard deviation of 173 words. Restrictions were placed such as keeping the top 5,000 words and truncating/padding reviews to 500 words. This was done for model efficiency. Reviews are then encoded as lists of word indexes, with each integer representing a word's frequency in the

```
#load imdb dataset but only keep the top n words, zero the rest
top_words = 5000
(train_X, train_Y), (test_X, test_Y) = imdb.load_data(num_words=top_words)

#Using Keras utility to truncate / pad the data to a length of 500 for each review
max_words = 500
train_X = pad_sequences(train_X, maxlen=max_words)
test_X = pad_sequences(test_X, maxlen=max_words)
```

dataset. We are utilizing the WordPiece tokenizer, which allows us to gather insight from the input sequences. This is done by “decomposes words into subword units and assigns them unique IDs. This process helps capture both the granularity of individual characters and the context of larger words” (WordPiece, 2024) . The X inputs correspond to the individual sentiment and the Y outputs correspond to whether it is positive or negative (Keras, 2024).

Results Analysis

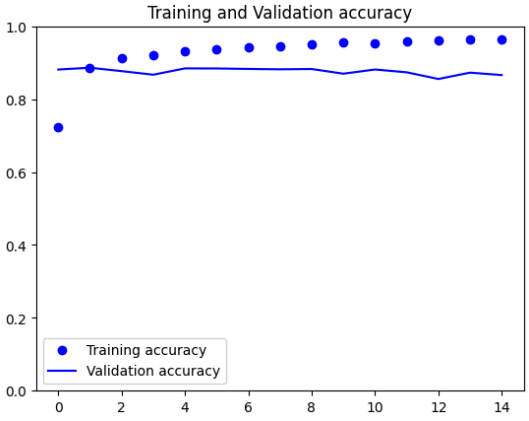
When training a deep learning model there are three possible states our model could finish in. An optimized model would have its training and validation accuracy and loss in line with each other. If the training and validation accuracy produce multiple local maxima/minima then the model is becoming overfit. Overfit is defined when the model is learning “from the noise and inaccurate data entries in our data set” (GfG, 2024). This causes the model to become overly confident resulting in fluctuations in model performance and high variance. Underfitting is when the model is not able to effectively extract information from the data set. This is caused by the model being too “simple or the data being highly regularized” (GfG, 2024), resulting in high bias.

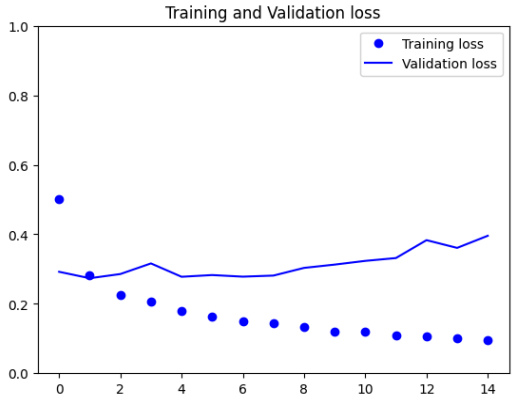


Sentiment Analysis:

The results of the model's training for sentiment show that overall, the training of the model was successful. On the Accuracy graph, we can see a strong correlation between the training accuracy and the validation accuracy as they finalize at 0.9640 and 0.8661 respectively. In the Loss graph, validation loss plateaued between 0.30 and 0.40 while training loss continued to drop.

This pattern suggests the model seems to be performing well, showing good accuracy and declining loss on both training and validation datasets. There is no large gap between the training and validation lines, which can often indicate overfitting. Instead, the model's performance on unseen validation data is close to its performance on the training data, which generally implies that the model has learned to generalize well.





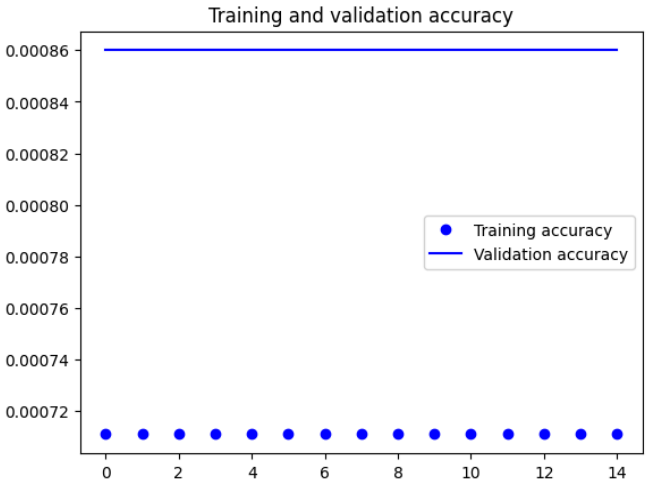

```python
# Assuming test_X is your features and test_Y is your labels from the test set
# Let's randomly select an index
random_index = np.random.randint(0, len(test_X))

# Selecting a random sample from the test set
sample_features = np.expand_dims(test_X[random_index], axis=0)  # Model expects a batch dimension
sample_label = test_Y[random_index]

# Predicting the sentiment
prediction = sentiment_model.predict(sample_features)

# Interpreting the result
predicted_sentiment = 'Positive' if prediction >= 0.5 else 'Negative'
actual_sentiment = 'Positive' if sample_label == 1 else 'Negative'

print(f'Predicted Sentiment: {predicted_sentiment}')
print(f'Actual Sentiment: {actual_sentiment}')
```

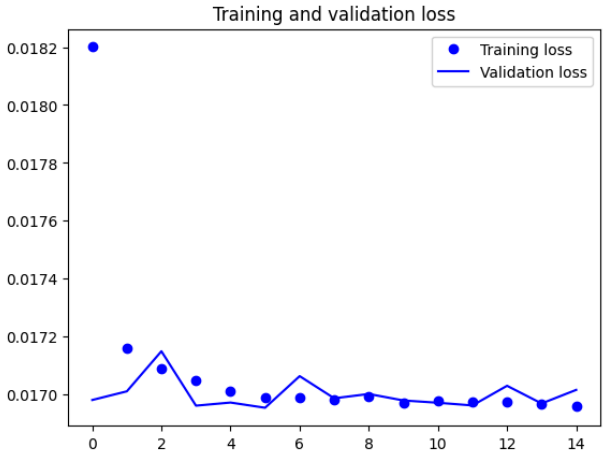
```
1/1 [==============================] - 3s 3s/step
Predicted Sentiment: Positive
Actual Sentiment: Positive
```

The above block of code validates the accuracy of the model by selecting a random sentiment from the test set. If the model prediction was equal to or greater than 0.5, there is a high correlation to a positive review. If the model prediction was less than 0.5, it was determined to be a negative review.

Rating Prediction:

As depicted on the Accuracy graph, accuracy on both the training and validation sets was extremely low, finishing at 0.00086 or 0.086%. This is due to insufficient input variables when training this model. This indicates that there is not sufficient correlation between how successful a movie performs and the genres that make up that movie. We believe with additional input data in the form of directors, main actors, movie budgets, and gross profits, we can produce a more robust model that can accurately predict how a movie will perform at the box office.





Model inferencing was not conducted on the rating prediction model as both training and validation accuracy were too low to gather meaningful insight.

Discussion

There are many implementations for a sentiment analysis model in the movie industry, but in a broader sense, any industry that relies on customer sentiment. Movie studio executives, marketing agents, directors, and consumers themselves, can gauge how sentiment shifts throughout the production process. As information is released to the public, companies can deduct if the current sentiment is inline with company goals. Movie studios are already implementing deep learning models that are more accurate, efficient, and cost effective compared to traditional sentiment analysis (Sheils, 2023). These deep learning models can provide advertisement companies with enriched data that can better reach target audiences.

Any industry that utilizes customer feedback can make use of our company's product, this extends to e-commerce, journalism, advertisers, and local businesses. The scalability of our product makes it an easy adoption from small business to large corporations. Individuals can train this model with their specific needs by introducing reviews left by their customers.

We plan to further train the sentiment model by further tokenizing the input sequences so that adjectives and adverbs provide more weight. This will allow the model to become more applicable to any company's needs. Another future addition will be the use of attention mechanisms. Typically in neural networks, an encoder-decoder pair is only able to gather insight from the last state of the input sequence. This causes problems when the user wants to gather insight from long chains of text. "Instead of paying attention to the last state of the encoder as is usually done with RNNs, in each step of the decoder we look at all the states of the encoder, being able to access information about all the elements of the input sequence. This is what attention does, it extracts information from the whole sequence, a weighted sum of all the past encoder states" (Vaswani, 2023). This application will allow our model to not only place

importance on the ends of a sequence but the sequence as a whole. This will allow companies to extract information from blogs and news articles, where entries can easily become several hundred words.

Conclusion

The exploration of sentiment analysis using a Convolutional Neural Network (CNN) on the IMDb dataset has demonstrated that deep learning can effectively discern and categorize the sentiment of movie reviews. Our model, which processed 25,000 reviews, was able to achieve high accuracy in distinguishing between positive and negative sentiments, indicative of its ability to capture the nuances in the data provided.

The successful training of the model was characterized by closely aligned training and validation accuracy, suggesting a good balance between bias and variance, and the ability of the model to generalize well to unseen data. While the validation loss showed a slight plateau, it did not significantly diverge from the training loss, mitigating concerns over potential overfitting.

Despite the complexity inherent in natural language processing, the CNN's utilization of convolution, max-pooling, and dropout layers effectively managed to reduce dimensionality and prevent overfitting, thereby enhancing the model's capability to learn significant patterns. The presence of dropout layers likely contributed to the network's resilience against overfitting, allowing the model to maintain robustness even when faced with varied or noisy data.

As we reflect on our findings, we recognize the vast potential of applying deep learning to sentiment analysis, not only in the realm of film reviews but across diverse sectors where understanding consumer sentiment is key. The model presents opportunities for businesses and creative professionals to harness the wealth of data available online, translating it into actionable insights that can inform decisions and strategies.

Works Cited

GfG. “ML: Underfitting and Overfitting.” *GeeksforGeeks*, GeeksforGeeks, 11 Mar. 2024, www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/.

Sheils, Matt. “AI-Based Audience Sentiment Analysis for Film Marketing: The Future of Targeted Advertising.” *LinkedIn*, 31 Aug. 2023, www.linkedin.com/pulse/ai-based-audience-sentiment-analysis-film-marketing-future-sheils/.

Team, Keras. “Keras Documentation: IMDB Movie Review Sentiment Classification Dataset.” *Keras*, keras.io/api/datasets/imdb/. Accessed 14 Apr. 2024.

Vaswani, Ashish, et al. “Attention Is All You Need.” *arXiv.Org*, 2 Aug. 2023, arxiv.org/abs/1706.03762.

Verma, Shiva. “Understanding 1D and 3D Convolution Neural Network: Keras.” *Medium*, Towards Data Science, 17 July 2023, towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610.

“WordPiece.” *What Is WordPiece?*, h2o.ai/wiki/wordpiece/#:~:text=During%20tokenization%2C%20WordPiece%20decomposes%20words,the%20context%20of%20larger%20words. Accessed 25 Apr. 2024.