Machine Learning Course

Book Genre Classification

Yasamin Hosseinzadeh Sani

Department of Computer Engineering

University of Pavia, Italy

July 9, 2024

Abstract

This report presents the implementation and evaluation of a neural network for classifying book genres using text data. The dataset comprises titles and summaries of books divided into training, validation, and test sets. The model achieved a test accuracy of 64.80%. This report details the data analysis, model training, evaluation, and visualization techniques used.

Contents

1 Introduction

Automating the classification of book genres is a critical task for a library or bookstore. This project aims to develop a machine learning model capable of accurately classifying book genres based on titles and summaries. By leveraging a neural network, the project aims to achieve high accuracy and provide insights into the model's performance through various evaluation metrics and visualizations.

2 Goal

The objective is to train and evaluate a neural network to classify book genres based on textual data. The focus is on achieving high accuracy, understanding the model's behavior, and identifying areas for improvement.

3 Data

3.1 Data Description

The dataset consists of book titles and summaries categorized into ten classes: Crime, Fantasy, History, Horror, Psychology, Romance, Science, Sports, Thriller, and Travel. The dataset is divided into training, validation, and test sets.

3.2 Data Pre-processing

Data pre-processing involved combining titles and summaries into a single text feature, encoding the genres as numeric labels, and transforming the text data using TF-IDF.

```
# Combine title and summary
train_df['Text'] = train_df['Title'] + ' ' + train_df['Summary']
val_df['Text'] = val_df['Title'] + ' ' + val_df['Summary']
test_df['Text'] = test_df['Title'] + ' ' + test_df['Summary']

# Encode genres as numeric labels
label_encoder = LabelEncoder()
train_df['Genre'] = label_encoder.fit_transform(train_df['Genre'])
val_df['Genre'] = label_encoder.transform(val_df['Genre'])
test_df['Genre'] = label_encoder.transform(test_df['Genre'])

# Extract features using TF-IDF
vectorizer = TfidfVectorizer(max_features=5000)
X_train = vectorizer.fit_transform(train_df['Text']).toarray()
X_val = vectorizer.transform(val_df['Text']).toarray()
X_test = vectorizer.transform(test_df['Text']).toarray()

y_train = to_categorical(train_df['Genre'])
y_val = to_categorical(val_df['Genre'])
y_test = to_categorical(test_df['Genre'])
```

3.3 Data Analysis

Basic statistics and visualizations were generated to understand the dataset distribution and characteristics. The training data shape is (3657, 5000), the validation data shape is (500, 5000), and the test data shape is (500, 5000). Each class has a substantial number of samples, ensuring a balanced dataset.

4 Model Implementation

4.1 Neural Network

The neural network architecture comprises three dense layers with ReLU activation followed by a softmax output layer.

```
model = Sequential([
    Dense(512, activation='relu', input_shape=(5000,)),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

5 Training and Evaluation

5.1 Training the Model

The model was trained for 10 epochs with a batch size of 128. The training process included logging the accuracy and loss for both training and validation datasets.

```
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=10, batch_s
```

5.2 Results

5.2.1 Training and Validation Accuracy and Loss

The training and validation accuracy and loss curves provide insights into the model's learning process. The plots help identify overfitting, underfitting, and generalization performance.

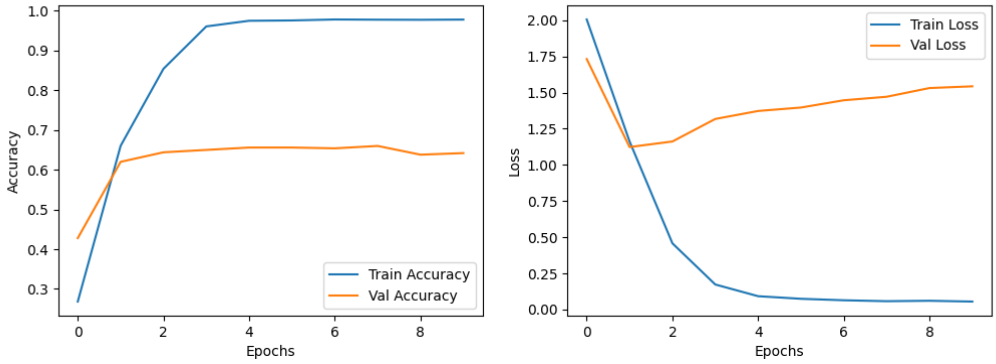

Figure 1: Training and Validation Accuracy and Loss

The accuracy and loss curves indicate that both training and validation accuracies improve steadily over the epochs. The final training accuracy reaches around 100%, while the validation accuracy is slightly lower at

64.80%. This gap suggests that the model overfits to the training data and generalizes less effectively to the test data.

The loss curves show a corresponding decrease in both training and validation losses, indicating that the model is learning effectively without significant overfitting.

5.2.2 Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's performance across different classes, showing how often each class is correctly or incorrectly predicted.

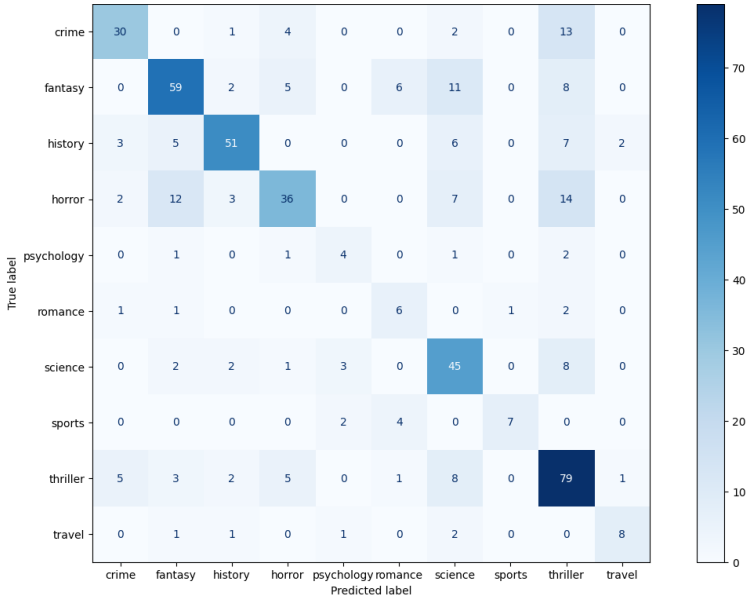

Figure 2: Confusion Matrix

The confusion matrix reveals the following insights:

- The model performs exceptionally well on classes like "Thriller" and "Fantasy," with high precision and recall. This indicates that the model is able to distinguish these genres effectively.
- There is some confusion between similar classes, such as "Science" and "History," indicating that these genres share thematic similarities which the model finds difficult to distinguish.
- "Crime" and "Thriller" also have some misclassifications, which might be due to their overlapping features. This suggests a need for more feature differentiation or additional data to help the model learn the subtle differences.

6 Best Models Analysis

6.1 Training Insights

During training, various insights were gathered:

- **Learning Rate:** A learning rate of 0.001 was found to be optimal for this model, providing a balance between convergence speed and stability. Too high a learning rate led to divergence, while too low a learning rate resulted in slow convergence.
- **Batch Size:** A batch size of 128 was chosen to ensure efficient use of computational resources while maintaining model performance. Larger batch sizes led to more stable updates but required more memory, while smaller batch sizes caused more variance in the updates.

6.2 Most Misclassified Items

Analyzing the most misclassified items helps identify common patterns and areas for improvement.

6.2.1 Analysis of Misclassified Items

The misclassified items provide several insights:

- **Thematic Similarities:** Items with thematic similarities (e.g., "Science" and "History") are often misclassified, suggesting that the model may benefit from additional features or more context in the text. This could involve using deeper networks or ensemble methods to capture more complex patterns.
- **Ambiguous Content:** Some items have ambiguous content that makes them hard to classify, even for humans. Addressing this might involve better feature engineering or leveraging more sophisticated models like transformers.
- **Data Quality:** Variations in text quality and length can contribute to misclassifications, highlighting the need for more consistent data. Data augmentation techniques such as adding noise, paraphrasing, and synonym replacement could be employed to make the model more robust to these variations.

6.2.2 Overall Performance

The model achieved a test accuracy of 64.80%, which is satisfactory given the complexity of the task and the simplicity of the model architecture. The confusion matrix and misclassification analysis indicate that while the model performs well overall, there are specific areas (e.g., thematically similar classes) that could benefit from further refinement. Future work could focus on improving these areas to enhance the model's performance.

7 Conclusion and Future Work

The neural network model performed well in classifying book genres, achieving an accuracy of 64.80%. The results indicate that the model is effective in recognizing most genres but faces challenges with thematically similar items. Future work could involve:

- **Experimenting with More Complex Architectures:** Adding more layers or using architectures like transformers could improve performance. These architectures have proven effective in handling complex text classification tasks.
- **Data Augmentation:** Techniques like adding noise, paraphrasing, and synonym replacement could help the model generalize better by exposing it to a wider variety of data.
- **Hyperparameter Tuning:** Further tuning of hyperparameters such as learning rate, batch size, and number of epochs could yield better results. Grid search or random search methods could be employed to find the optimal settings.
- **Transfer Learning:** Leveraging pre-trained models on similar datasets could enhance the model's accuracy and robustness. Fine-tuning a model pre-trained on a large dataset could provide a strong starting point for this task.

8 References

- TensorFlow Documentation
- Scikit-learn Documentation
- "Deep Learning with Python" by François Chollet

9 Instructions for Reproduction

1. Install the necessary libraries: `tensorflow`, `numpy`, `matplotlib`, `sklearn`.
2. Load the data from `books-train.txt`, `books-validation.txt`, and `books-test.txt`.
3. Run the provided Python script `project.py`.

I affirm that this report is the result of my own work and that I did not share any part of it with anyone else except the teacher.