Book Genre Prediction using LSTM

Arshan Homi Dastur

November 10, 2024

Abstract

This project leverages the power of LSTM to predict the genre of any book that the user wants , by entering a sentence from that book. My project on book genre prediction using deep learning involves training a model on text data like book descriptions or sample passages to classify books into genres. The project can be hosted as a web page application to allow users to type their sentence and accordingly find the genre they are looking for.

Introduction

Here's an overview of the main steps:

- **Data Collection and Preprocessing**: Text data from books, often descriptions or samples, is collected and cleaned. This involves tokenization, lowercasing, and removing non-essential words (like stop words).
- **Model Design**: A neural network model, often LSTM (Long Short-Term Memory), is built for text analysis. LSTMs handle sequence data well, capturing context across longer text passages by using past inputs to inform current processing.
- **Training**:The model is trained on labelled data with genre tags. It learns patterns in language and phrases that distinguish genres (e.g., horror, romance, sci-fi).
- **Evaluation and Tuning**:Model performance is evaluated using metrics like accuracy and F1-score. Hyperparameters, such as learning rate and dropout, are tuned to improve generalization and reduce overfitting.
- **Deployment**: the model can be deployed to predict genres for new books, aiding in recommendations or cataloguing systems.

Dataset

The data set I AM using is made up of book descriptions and genre classification that I collected from GoodReads. We will simplify our challenge to a Binary Classification problem for this project.Since we don't want to "forget" words too quickly—words early in a phrase might have a big impact on its meaning—we will specifically use LSTM (Long Short-Term Memory) cells.

Data cleaning and Exploration

Data Cleaning and Data Exploration:- This step takes up the bulk of every data scientist's time. We look at every column in the data frame and find out any potential problems we might face. Some common problems include:

- **Missing values**
- **Different languages involved**
- **Non-Ascii characters**
- **Invalid descriptions**
- **Missing spaces in the description.**

The genre will serve as our labels because we are forecasting it, and the qualities will be derived from each book's description. I noticed that several entries had formatting issues; this is where 'langdetect' is useful. A function to eliminate any rows with an invalid description format will be put into place. I noticed that there were many languages involved in my dataset. For simplicity, I only want to get book descriptions that are in English. langdetect allows us to map each description to a ISO 639–1 value to make our lives easier when filtering out English descriptions. I will then retrieve a list of languages and their respective ISO values from Wikipedia.

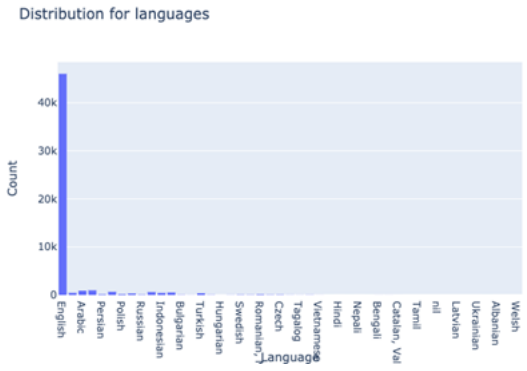

Figure 1: Distribution for languages

Methodology

Data Preprocessing and Exploratory Data Analysis

In every data science project, it is very important to know the distribution of your data and the best way to do so would be to plot graphs. The following plots gave me a good idea about the dataset.If fiction appears as at least one genre in the genre list column, the book is categorized as fiction. It has been noted that all other genres in the same list will be strongly related to fiction if a book includes at least fiction in its genre list. This allows me to convert this into a binary classification problem by comparing the quantity of fiction and nonfiction books in my dataset. We must pay close attention because

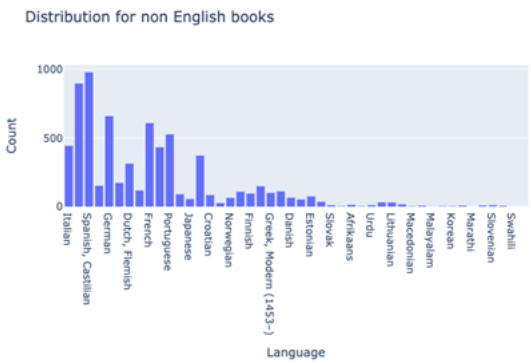

Figure 2: Distribution for non English books

the book description serves as our predictor. Every description must follow the same structure and be the same length. Since fixed-shape tensors and more stable weights can be created, working with fixed input length enhances model training performance. As a result, we shall use padding and clipping. To determine the optimal length, I plot the distribution of description length and observe the most 'common' description length.I decided to observe "population of books" for each level of description length by plotting the Cumulative Distribution Function (CDF). For records where the description is less than 250 words, I pad them with empty values, whereas for records where the description is more than 250 words, I clip them to include just the first 250.

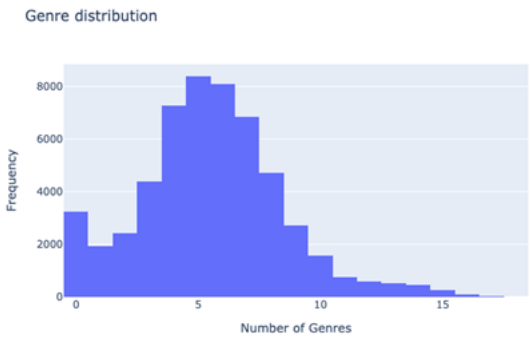

Figure 3: Genre Distribution

Training and testing

For records where the description is less than 250 words, I pad them with empty values, whereas for records where the description is more than 250 words, I clip them to include just the first 250. We should ensure that the training-validation split is stratified when a dataset is unbalanced, meaning that the target variable's distribution (fiction/nonfiction) is not uniform. This guarantees that the target variable's distribution is maintained across the training and validation datasets. Our model will be trained using x_train and y_train, and its validation accuracy will be checked using x_val and y_val. To make sure we are not overfitting, we are aiming for a relatively high training accuracy and a high validation accuracy.

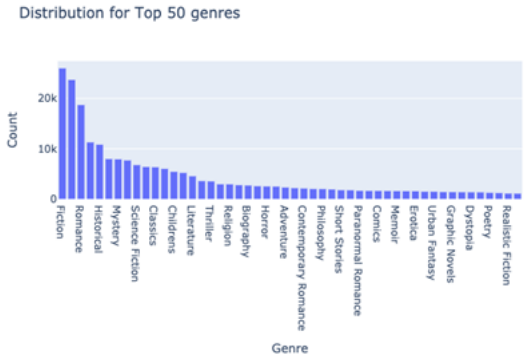

Figure 4: Distribution for top 50 genres

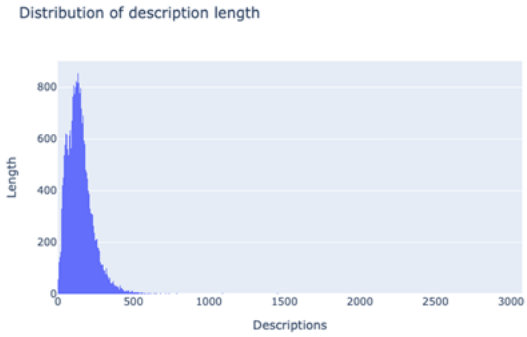

Figure 5:

Model Architecture and Usage

We have implemented an LSTM with one layer. As for the number of hidden nodes, I used the following formula;

$$\text{Accuracy} = \frac{Ns}{alpha * (Ni + No)} \tag{1}$$

Where Ns : number of samples in training data Ni : number of input neurons No : number of output neurons alpha : scaling factor(Indicator of how general you want your model to be, or how much you want to prevent overfitting)

I added a dropout layer which prevents overfitting by ignoring randomly selected neurons during training and reduces sensitivity to specific weights of individual neurons. by ignoring randomly selected neurons during training and reduces sensitivity to specific weights of individual neurons(Dense layer). I have used binary cross-entropy as the loss function and sigmoid as the activation function. Running the model for 5 epochs with a suitable batch size gave a training accuracy of 99.58 %and validation accuracy of 92.47%.

Implementation

The project code was structured using object-oriented programming (OOP) principles, making it modular and maintainable. The code includes functions and classes that handle data loading, model training, prediction generation, and evaluation. Key modules include:

- **urlscraper.py**: Collects the necessary dataset by scraping the designated websites.
- **raw.py**: Performs data cleaning and preprocessing along with LSTM model initialization.

Web page integration

I have implemented the code, keeping in mind industrial practices of writing clean, modular code with necessary comments and indentation to make sure other developers understand the project. In addition to that, I have hosted this project as a web page using Streamlit and ngrok. Streamlit is a powerful tool for building data-driven web applications quickly, especially for machine learning and data science projects. Ngrok is a tool that helps developers expose a local server to the internet through secure tunnels. This is especially useful for sharing a Streamlit or other local application for remote access without complex network configurations. Ngrok provides a temporary, public URL that can be shared, making it easy for collaborators or clients to interact with the application or test features without a full deployment.

Conclusion

Long Short Term Memory proved to be better than other neural networks in this task of predicting genres since it is able to capture the context of the text which has been input.

References

References

[1] TensorFlow, `https://www.tensorflow.org/`

[2] Keras, `https://keras.io/`

[3] Streamlit, `https://streamlit.io/`

[4] ngrok, `https://ngrok.com/`