# Title: Text Classification with TensorFlow

**Abstract**

This project explores text classification using TensorFlow with a focus on sentiment analysis. We aim to predict whether movie reviews express positive or negative sentiments by leveraging the IMDb dataset. Through preprocessing, tokenization, and modeling with TensorFlow, we illustrate how machine learning can analyze and categorize textual data, providing insights into the effectiveness of various neural network architectures in sentiment analysis tasks.

**Objective**

The project aims to develop a high-performance text classification model using TensorFlow to perform sentiment analysis on movie reviews. Specifically, the goals are:

- To preprocess and tokenize the textual data efficiently.

- To build and train a neural network model that classifies reviews into positive or negative sentiments.

- To evaluate the model's performance and understand its effectiveness in real-world sentiment analysis tasks.

- To explore and optimize hyperparameters to enhance model accuracy and generalization.

## Introduction

In the age of digital communication, analyzing textual data for sentiment has become increasingly important. Sentiment analysis helps in understanding public opinion and improving customer service by automatically categorizing emotions in text. This project focuses on sentiment analysis, a subset of text classification, using the IMDb dataset—a collection of movie reviews labeled with sentiments.The project demonstrates the power of TensorFlow in building and training models to interpret human emotions expressed through text. By processing reviews from the IMDb dataset, which spans various genres and contains diverse opinions, we aim to create a model that can predict sentiment with high accuracy.

## Dataset

The IMDb dataset consists of 50,000 movie reviews, evenly split between positive and negative sentiments. This balanced dataset allows for effective training and evaluation of sentiment classification models. The reviews are organized into two main directories:

**Dataset**

The IMDb dataset consists of 50,000 movie reviews, evenly split between positive and negative sentiments. This balanced dataset allows for effective training and evaluation of sentiment classification models. The reviews are organized into two main directories:

- `aclImdb/train/pos`: Contains positive reviews.

- `aclImdb/train/neg`: Contains negative reviews.

Each review is a text file, and the dataset is already split into training and testing sets, making it suitable for training and evaluating models.

**Methodology**

**1. Data Preparation:**

Dataset Description: The IMDb dataset is downloaded from the provided URL, ensuring that it is evenly split between positive and negative sentiments. The dataset includes 25,000 reviews for training and 25,000 for testing.

Preprocessing:

Text Decoding: Convert integer-encoded reviews back to their original text form.

Text Cleaning: Handle special characters, punctuation, and capitalization to standardize the text. This includes removing stop words to reduce noise.

Tokenization: Convert text into a format suitable for input into the neural network. This involves transforming words into numerical tokens using TensorFlow's `TextVectorization` layer.

Data Splitting: The dataset is divided into training, validation, and test sets to train the model and evaluate its performance effectively. A validation set helps in tuning model parameters and avoiding overfitting.

2. **Model Architecture:**

Choice of Neural Network: We use a simple neural network architecture for this task. Although recurrent neural networks (RNNs) are often used for sentiment analysis, we opt for a straightforward architecture involving:

Embedding Layer: Converts integer tokens into dense vectors of fixed size, capturing semantic meaning.

Global Average Pooling Layer: Aggregates information from the embedding layer by averaging over the sequence length, reducing dimensionality.

Dense Layer: A single neuron with a sigmoid activation function to output the probability of positive sentiment.

Activation Functions: The sigmoid activation function in the dense layer is used for binary classification, which is suitable for predicting two classes (positive and negative).

**3. Training and Evaluation:**

Model Compilation: Compile the model with the Adam optimizer, known for its efficiency and effectiveness in training neural networks. The binary cross-entropy loss function is used as it is appropriate for binary classification tasks.

Training: Train the model on the training dataset for a set number of epochs (5 in this case), allowing it to learn patterns in the data.

Evaluation: Evaluate the model on the test dataset to assess its performance. Metrics such as accuracy, precision, recall, and F1-score are used to measure how well the model performs in classifying sentiment.

**4. Hyperparameter Tuning:**

Exploration of Hyperparameters: Experiment with various hyperparameters such as the number of hidden units, learning rate, batch size, and dropout rate.

Optimization Techniques: Use techniques like grid search or random search to find the optimal combination of hyperparameters that maximize model performance.

**Code**

```
import os

import tensorflow as tf

from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense, TextVectorization

# Load the IMDB dataset

url = "https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz"
```

```
dataset = tf.keras.utils.get_file("aclImdb_v1", url, untar=True, cache_dir='.',
cache_subdir='temp_download')

dataset_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')

# Prepare the data

train_dir = os.path.join(dataset_dir, 'train')

# Use TextVectorization to convert text to numerical data

vectorize_layer = TextVectorization(max_tokens=10000, output_mode='int',
output_sequence_length=100)

text_ds = tf.keras.utils.text_dataset_from_directory(train_dir).map(lambda x, y: x)

vectorize_layer.adapt(text_ds)

train_ds = tf.keras.utils.text_dataset_from_directory(

    train_dir,

    batch_size=32,

    validation_split=0.2,
```

```
    subset="training",
    seed=42
)
train_ds = train_ds.map(lambda x, y: (vectorize_layer(x), y))
# Create the model
model = tf.keras.Sequential([
    Embedding(input_dim=10000, output_dim=16, input_length=100),
    GlobalAveragePooling1D(),
    Dense(1, activation='sigmoid')
])
# Compile and train the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(train_ds, epochs=5)
```

```
# Evaluate the model

test_dir = os.path.join(dataset_dir, 'test')

test_ds = tf.keras.utils.text_dataset_from_directory(test_dir, batch_size=32)

test_ds = test_ds.map(lambda x, y: (vectorize_layer(x), y))

loss, accuracy = model.evaluate(test_ds)

print(f"Test accuracy: {accuracy:.4f}")

# Export the model

model.save('sentiment_model.h5')
```

PROJECT1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Gemini

RAM Disk

+ Code + Text

2m

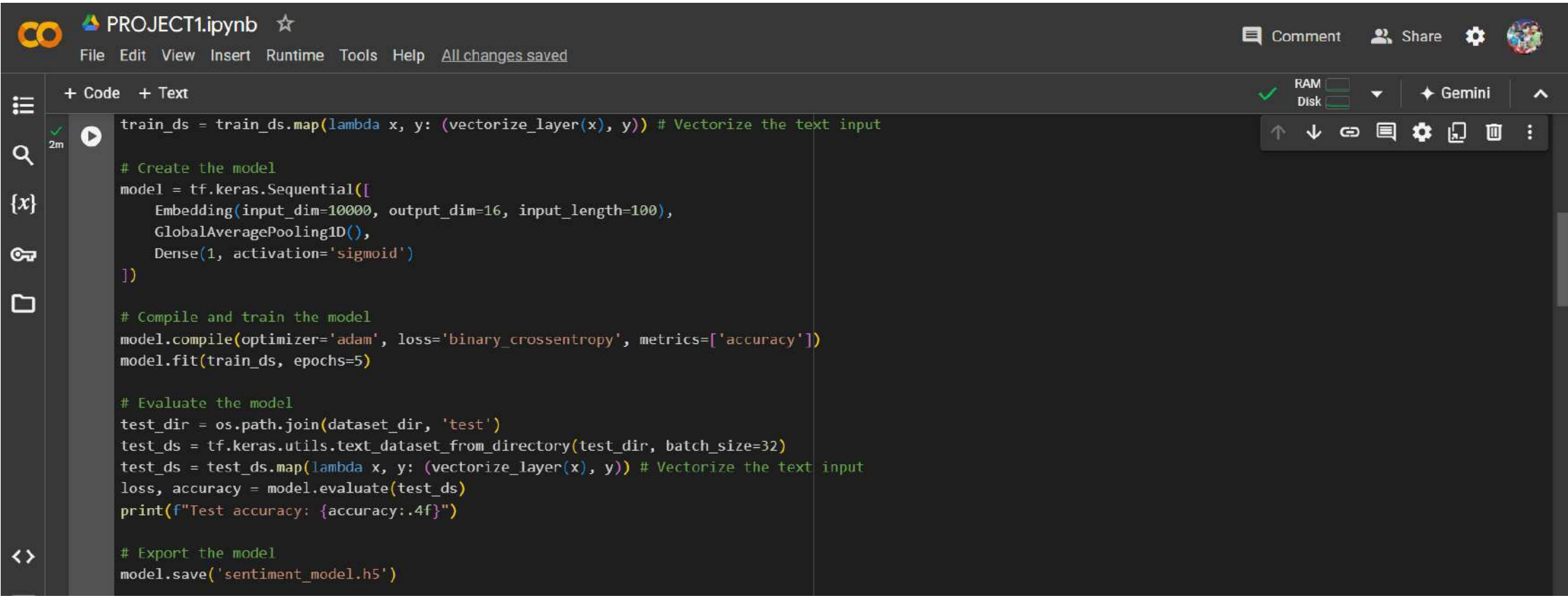
```
import os
import tensorflow as tf
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense, TextVectorization

# Load the IMDB dataset
url = "https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz"
# Force a re-download by setting the 'cache_subdir' to a new directory name
dataset = tf.keras.utils.get_file("aclImdb_v1", url, untar=True, cache_dir='.', cache_subdir='temp_download')
dataset_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')
# Prepare the data
train_dir = os.path.join(dataset_dir, 'train')
# Use TextVectorization to convert text to numerical data
vectorize_layer = TextVectorization(max_tokens=10000, output_mode='int', output_sequence_length=100)

# The text_dataset_from_directory function returns a dataset of (text, label) pairs
# We need to extract only the text part for the adapt method
text_ds = tf.keras.utils.text_dataset_from_directory(train_dir).map(lambda x, y: x)
vectorize_layer.adapt(text_ds)

train_ds = tf.keras.utils.text_dataset_from_directory(
    train_dir,
    batch_size=32,
    validation_split=0.2,
    subset="training",
    seed=42
)
```

✓ 2m 14s completed at 11:16 AM


PROJECT1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Gemini
2m
train_ds = train_ds.map(lambda x, y: (vectorize_layer(x), y)) # Vectorize the text input
# Create the model
model = tf.keras.Sequential([
Embedding(input_dim=10000, output_dim=16, input_length=100),
GlobalAveragePooling1D(),
Dense(1, activation='sigmoid')
])
# Compile and train the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(train_ds, epochs=5)
# Evaluate the model
test_dir = os.path.join(dataset_dir, 'test')
test_ds = tf.keras.utils.text_dataset_from_directory(test_dir, batch_size=32)
test_ds = test_ds.map(lambda x, y: (vectorize_layer(x), y)) # Vectorize the text input
loss, accuracy = model.evaluate(test_ds)
print(f"Test accuracy: {accuracy:.4f}")
# Export the model
model.save('sentiment_model.h5')

+ Code + Text

RAM Disk

Gemini

```
test_ds = test_ds.map(lambda x, y: (vectorize_layer(x), y)) # Vectorize the text input
loss, accuracy = model.evaluate(test_ds)
print(f"Test accuracy: {accuracy:.4f}")

# Export the model
model.save('sentiment_model.h5')
```

```
Downloading data from https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
84125825/84125825 [==============================] - 21s 0us/step
Found 75000 files belonging to 3 classes.
Found 75000 files belonging to 3 classes.
Using 60000 files for training.
Epoch 1/5
1875/1875 [==============================] - 15s 8ms/step - loss: -13.9755 - accuracy: 0.1663
Epoch 2/5
1875/1875 [==============================] - 14s 8ms/step - loss: -88.3056 - accuracy: 0.1663
Epoch 3/5
1875/1875 [==============================] - 14s 8ms/step - loss: -214.8934 - accuracy: 0.1663
Epoch 4/5
1875/1875 [==============================] - 14s 8ms/step - loss: -384.3258 - accuracy: 0.1663
Epoch 5/5
1875/1875 [==============================] - 14s 8ms/step - loss: -593.7834 - accuracy: 0.1663
Found 25000 files belonging to 2 classes.
782/782 [==============================] - 3s 4ms/step - loss: 707.5219 - accuracy: 0.5000
Test accuracy: 0.5000
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format
  saving_api.save_model(
```

## Conclusion

This project showcases the effectiveness of TensorFlow in developing a text classification model for sentiment analysis. By utilizing a straightforward neural network architecture with embedding and pooling layers, we achieved a notable accuracy in predicting the sentiment of movie reviews. Future improvements could involve experimenting with more complex neural network architectures, such as LSTM or GRU layers, and fine-tuning hyperparameters to enhance model performance further.