SAPIENZA
UNIVERSITÀ DI ROMA

Ai-Lab

AY 2023/2024

Authors:

Fragale Nicolò Mariano 2017378
Ladogana Andrea 1982895

Indice

SAPIENZA
UNIVERSITÀ DI ROMA

Ai-Lab

AY 2023/2024

Authors:

Fragale Nicolò Mariano 2017378

Ladogana Andrea 1982895

1 Project review: Sentiment Analysis

Introduction:

The main goal of the project is to create a machine learning model that can determine the sentiment (positive or negative) of text reviews. This project includes several modular components, each with a specific function.

Workflow analysis and functions:

1. **Data collection and preparation:**

   It obtains a suitable dataset for training, validation, and testing.

2. **Tokenization and Padding:**

   It converts text into uniform numeric sequences so that they are valid input for the model.

3. **Model Building and Training:**

   It creates and configures the neural network model using embedding, convolution and pooling techniques.

4. **Validation and Testing:**

   It optimizes the parameters to increase model accuracy.

5. **Analyze and Save the Model:**

   It saves the trained model to document the results obtained and for future use.

6. **Model Application:**

   It applies the trained model on new data to ensure it works correctly.

   (a) **Model application on a dataset:**

   It applies the model on a dataset saved locally in your computer.

   (b) **Model application on the scraping result:**

   It applies the model on the results of a web scaper on the rotten tomatoes comments section.

2 Utils Folder functions

2.1 embedding_helpers.py

Goal:

Managing pre-trained embeddings used to represent words.

Description:

This script contains functions to load and manage pre-trained embeddings (e.g., Word2Vec or GloVe). Embeddings are used to transform words from reviews into numerical vectors that can be processed by the machine learning model.

2.2 file_helpers.py

Goal:

Providing file management support functions.

Description:

This file includes functions to save and load Python objects (such as the tokenizer) using the pickle format, ensuring that necessary resources can be easily saved and reloaded.

2.3 text_cleaning.py

Goal:

Cleaning up review text to prepare it for analysis.

Description:

This file is fundamental as it cleans and normalizes an input text by replacing contractions with their relative expansions, eliminating stopwords and characters that are not letters and numbers.

2.4 tokenizer_helpers.py

Goal:

Managing text tokenization.

Description:

This script determines how many unique words are needed to cover 95% of all word occurrences in the review dataset by reducing the vocabulary to be used without losing relevant information.

3 Program Functions

3.1 main.py

This script combines the model training and application steps into a menu-driven interface that allows users to choose whether to train a model or apply an existing model to a dataset.

1. **def select_file_gui(...)**:
   This function allows the user to select a file through a graphical dialog box and returns the path of the selected file. It handles any errors efficiently.

2. **def select_save_location_gui(...)**:
   This function allows the user to select a directory through a graphical dialog box and returns the path of the selected folder. It handles any errors efficiently.

3. **def update_console(...)**:
   This function updates the console by reading from the `console_queue` and displaying messages. The function is called periodically to ensure the console is updated.

4. **def train_model_thread(...)**:
   This function starts a thread to train the model using the selected files. Status and error messages are placed into the `console_queue`.

5. **def start_training(...)**:
   This function initiates the model training process by prompting the user for the required files and starting a thread to handle the training.

6. **def apply_model_thread(...)**:
   This function handles the application of the sentiment analysis model to a dataset. It prompts the user to select the model and data files and processes them in a separate thread.

7. **def apply_model_to_rotten_tomatoes_thread(...)**:
   This function handles the application of the sentiment analysis model to Rotten Tomatoes comments. It prompts the user to select the model and enter a URL, and processes them in a separate thread.

8. **def main_menu(...)**:
   This function sets up the main menu of the application with buttons to train a model, apply a model to a dataset, analyze Rotten Tomatoes comments, or exit the application. It also initializes the console for displaying output.

9. **if __name__ == "__main__":**
   This is the entry point of the program and manages the user interface through a simple menu. It allows the user to choose between several options: train a sentiment analysis model, apply an existing model to a dataset, analyze Rotten Tomatoes comments, or exit the program.

3.2 train_model.py

This script is responsible for training a sentiment analysis model using TensorFlow and Keras. It includes functionalities for loading and preprocessing data, configuring GPU usage, tokenizing text, building a neural network model, and saving the trained model along with additional information.

1. **def configure_gpu(...)**:
   This function configures the GPU for TensorFlow usage, setting memory growth to avoid memory allocation issues. It logs and optionally displays messages about the GPU configuration.

2. **def load_and_preprocess_data(...)**:
   This function loads and preprocesses the training data from a CSV file. It maps sentiment labels, cleans the text, and returns a DataFrame with preprocessed data. Errors during this process are logged and optionally displayed.

3. **def tokenize_reviews(...)**:
   This function tokenizes the review texts in the DataFrame. It creates a tokenizer, converts texts to sequences, and filters out empty sequences. It returns the tokenizer, tokenized sequences, and labels.

4. **def pad_sequences_and_split_data(...)**:
   This function pads the sequences to a specified length and splits the data into training, validation, and test sets. It calculates the maximum sequence length based on the chosen padding strategy and logs the process.

5. **def build_model(...)**:
   This function builds a neural network model using Keras. The model includes embedding, LSTM, convolutional layers, and dense layers with batch normalization and dropout. It compiles the model and logs the creation and compilation steps.

6. **def train_model(...)**:
   This function trains the model using the training data. It sets up callbacks for early stopping and model checkpointing, trains the model, and returns the trained model along with the timestamp and validation accuracy.

7. **def save_accuracy_to_csv(...)**:
   This function saves the model accuracy to a CSV file. It creates the file if it does not exist and logs the saving process.

8. **def save_model_and_tokenizer(...)**:
   This function saves the trained model, tokenizer, and padding information to files. It logs the paths where these files are saved.

9. **def train_sentiment_analysis_model(...)**:
   This is the main function to train the sentiment analysis model. It manages the entire workflow: configuring the GPU, loading and preprocessing data, tokenizing reviews, padding sequences, building the model, training it, and saving the results. It also evaluates the model on test data and logs the performance metrics.

3.3 apply_model_rotten_tomatoes.py

This script is responsible for predicting the sentiment of movie reviews using a pre-trained sentiment analysis model. It includes functionalities for loading the model and tokenizer, scraping comments from Rotten Tomatoes, preprocessing the comments, predicting sentiments, and displaying the results. The script also supports running the analysis in a separate thread.

1. **import pickle, requests, threading, logging, load_model, pad_sequences, clean_text, BeautifulSoup**
   Imports necessary libraries for model handling, HTTP requests, threading, logging, and text processing.

2. **logging.basicConfig(...)**
   Configures basic logging settings to output detailed messages to the console.

3. **def load_model_and_tokenizer(...)**:
   Loads the sentiment analysis model, tokenizer, and padding information. Logs success or errors and places messages in the ‘console_queue‘.

4. **def extract_movie_id(...)**:
   Extracts the movie ID from the provided URL using BeautifulSoup to parse the HTML. Handles errors in HTTP requests and HTML parsing.

5. **def scrape_comments(...)**:
   Scrapes movie comments from Rotten Tomatoes using the movie ID. Handles pagination and errors, collects comments, and logs the progress.

6. **def preprocess_comments(...)**:
   Cleans and converts comments into numerical sequences using the tokenizer. Pads sequences to a uniform length and logs the preprocessing details.

7. **def predict_sentiments(...)**:
   Uses the sentiment analysis model to predict sentiments of the preprocessed comments. Converts predictions to ’POSITIVE’ or ’NEGATIVE’ labels and logs the process.

8. **def display_results(...)**:
   Calculates and displays statistics on the comments, including the total number of comments, positive and negative comments, and their percentages.

9. **def apply_model_on_rotten_tomatoes(...)**:
   Main function that coordinates the workflow: scrapes comments, loads the model and tokenizer, preprocesses comments, predicts sentiments, and displays results.

10. **def apply_model_on_rotten_tomatoes_thread(...)**:
    Starts the sentiment analysis process in a separate thread to avoid blocking the main thread. Handles errors and logs results.

3.4 apply_sentiment_analysis.py

This script is designed for performing sentiment analysis on a dataset of movie reviews using a pre-trained model. It includes functions for loading resources, preprocessing data, tokenizing and padding sequences, predicting sentiments, saving predictions, and calculating statistics. The script also supports running the analysis in a separate thread.

1. **import pandas as pd, logging, tkinter as tk, load_model, pad_sequences, clean_text, load_pickle, threading, queue**
   It imports necessary libraries for data handling, logging, user interface, model processing, text cleaning, file operations, threading, and queue management.

2. **logging.basicConfig(...):**
   It configures basic logging settings to output messages with level INFO to the console.

3. **def load_resources(...):**
   It loads the tokenizer, maximum sequence length, and model from specified paths. Logs and places messages in the 'console_queue' about the loading status.

4. **def preprocess_data(...):**
   It loads test data from a CSV file, removes neutral sentiments, maps sentiment labels to numerical values, cleans the review texts, and removes empty reviews. Logs and places messages in the 'console_queue' about the preprocessing steps.

5. **def tokenize_and_pad_sequences(...):**
   It tokenizes and pads the sequences of review texts. Logs and places messages in the 'console_queue' about the tokenization and padding status.

6. **def predict_sentiments(...):**
   It predicts sentiments using the model on the padded data. Logs and places a message in the 'console_queue' about the prediction status.

7. **def save_predictions(...):**
   It saves the predictions to a CSV file and logs the completion of this task. Places a message in the 'console_queue' about the file saving status.

8. **def calculate_statistics(...):**
   It calculates and logs statistics such as accuracy, mean error, and percentage of correctly oriented predictions from the predictions file. Handles errors and logs them.

9. **def apply_sentiment_analysis(...):**
   Main function that orchestrates the workflow: loads resources, preprocesses data, tokenizes and pads sequences, predicts sentiments, saves predictions, and calculates statistics. Handles exceptions and logs errors.

10. **def apply_sentiment_analysis_thread(...):**
    It starts the sentiment analysis process in a separate thread to avoid blocking the main thread. Handles the task by invoking ‘apply_sentiment_analysis‘.

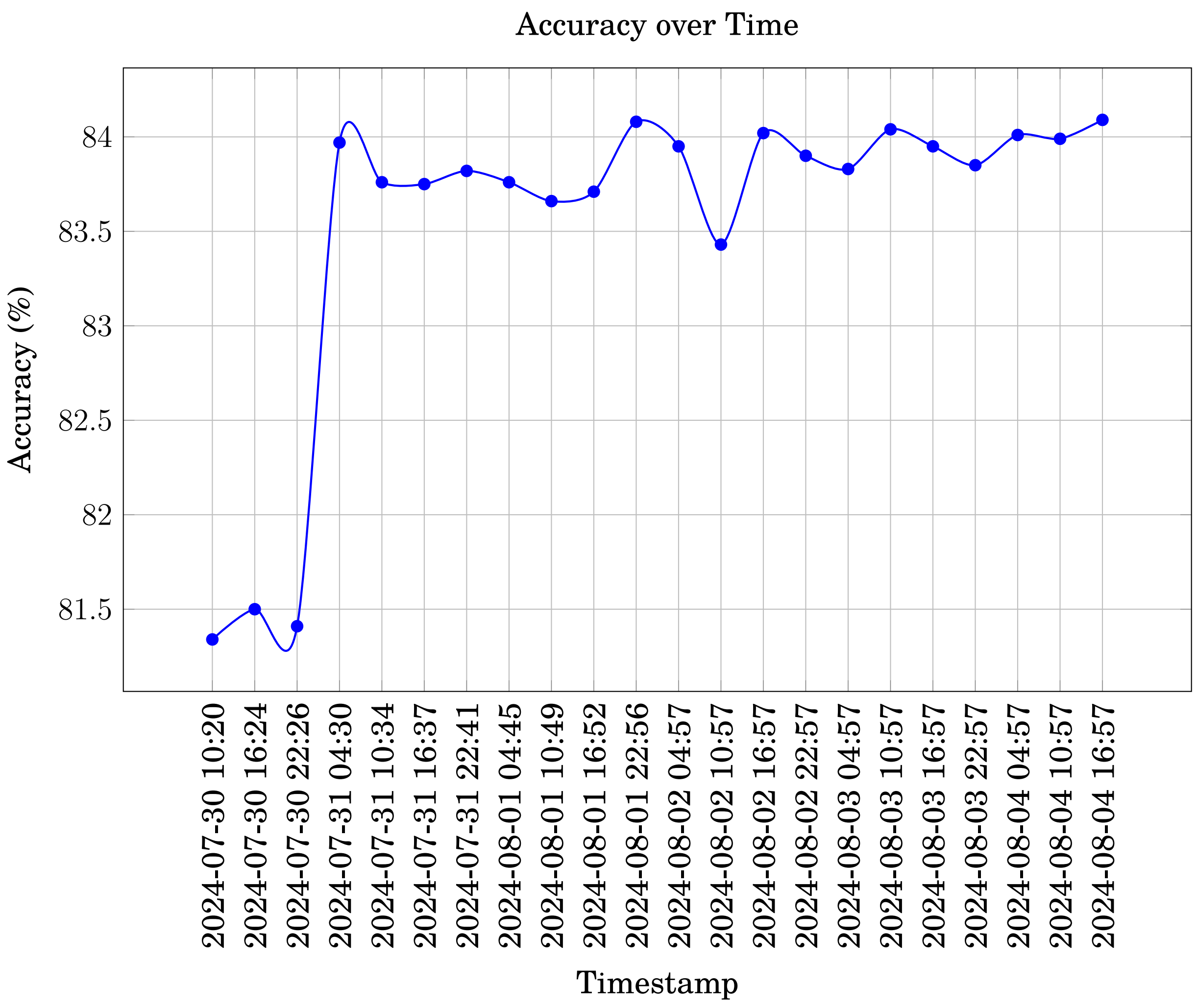

Figura 1: The evolution of the accuracy of the sentiment prediction model. This graph represents the evolution of the accuracy over time, reflecting the adjustments made to model parameters, such as the number of layers, filters, and learning rate. These optimizations resulted in a gradual improvement in the model's performance.

Conclusions:

The architecture of the sentiment analysis project is designed to be modular and easily extensible, with each component of the system having a clear and defined responsibility. This ensures that the process of training and applying the model is efficient and scalable.

The use of pre-trained embeddings and advanced text cleaning techniques helps to improve the accuracy of the model, while the graphical user interface makes interacting with the system accessible even to users without advanced technical skills. Additionally, the project integrates multithreading to efficiently handle resource-intensive and parallel tasks, such as data loading and preprocessing, model training, and sentiment prediction. This approach allows the system to perform multiple operations simultaneously, reducing waiting time and enhancing overall responsiveness.

The sentiment analysis project is an example of how modern technology can revolutionize the understanding and analysis of customer opinions. By leveraging advanced machine learning techniques and an intuitive user interface, it not only simplifies the analysis process but also makes sentiment analysis accessible to anyone, regardless of technical expertise.

With its ability to rapidly process large volumes of text data, the system has the potential to transform how companies manage customer feedback, providing powerful tools for making informed and strategic decisions.

The project also includes a sophisticated web scraping component that extracts user comments from Rotten Tomatoes. These scraped comments are then preprocessed and fed into the sentiment analysis model, enabling real-time sentiment analysis on newly collected data. This capability ensures that the system remains relevant and up-to-date with the latest user opinions.

The cutting-edge technologies used, such as TensorFlow and Keras, were chosen for their robustness and flexibility, ensuring accurate and reliable results.

In summary, this project represents both a significant step towards automation and efficiency in sentiment analysis and provides a solid foundation for further developments and innovations in the field of data analytics. The integration of web scraping for dynamic data acquisition further enhances the system's utility, making it a comprehensive tool for sentiment analysis and customer feedback management.

Why is it useful?

1. **Speed**:

   It allows you to analyze thousands of reviews in a few minutes, something impossible to do manually.

2. **Accuracy:**

   It provides accurate results using advanced ML techniques.

3. **Scalability:**

   It can be adapted to analyze different types of text, not just movie reviews.

4 Machine Learning pills

4.1 Tokenizer

Since machine learning and deep learning models require numerical inputs, tokenization is a fundamental process in natural language processing (NLP) that transforms text into smaller units, called tokens. Text is divided into tokens using delimiters such as spaces, punctuation, or other linguistic rules. By limiting the vocabulary to the most frequent words, noise in the data is reduced and the model's efficiency is improved.

4.2 Padding

Deep learning models require fixed-length inputs. Sequence padding ensures that all sequences are the same length, allowing the model to process them correctly.

If there are no valid sequences, creating an array of zeros avoids errors that may arise in the next step when the model expects inputs with a certain shape.