CSE508_Winter2024_A2_-2021444-

REPORT

QNS1

Overview

This document provides an in-depth analysis of the Python script aimed at extracting features from images as part of a Multimodal Retrieval System. The system is designed to handle both image and text data, focusing here on the image processing aspect.

Methodologies

Image Feature Extraction:
Pre-trained ResNet Model: The script employs a pre-trained ResNet50 model from PyTorch's torchvision library to extract features from images. This deep learning model is known for its high accuracy in image recognition tasks.

Image Preprocessing: Before feeding images into the ResNet model, they undergo a series of transformations: resizing to 224x224 pixels, converting to tensor format, and normalization using predefined mean and standard deviation values.

Error Handling: The script includes error handling for various exceptions, such as failed HTTP requests, unidentified images, and unexpected errors during the feature extraction process.

Assumptions
The script assumes that the images are accessible via URLs provided in a CSV dataset.
It operates under the premise that preprocessing steps (resizing, tensor conversion, normalization) are essential for preparing images for feature extraction through ResNet50.

Results

The script successfully extracts feature vectors from the images specified in the dataset. These features are then flattened and saved as a list of numpy arrays.
Processed image features are stored in a pickle file, allowing for persistence and future access.

Code Functionalities & Libraries Used

Libraries:
PyTorch and torchvision: For loading the pre-trained ResNet model and performing image transformations.

Pandas: To read the dataset containing image URLs.

Pillow (PIL): For opening and manipulating images fetched from URLs.

Requests: To make HTTP requests for downloading images.

NumPy: For numerical operations and handling the output feature vectors.

NLTK: Although primarily for text processing, it's imported for potential use in handling textual data associated with images.

Ast: For safely evaluating string representations of lists containing URLs.

Key Functions:

extract_image_features(urls): Processes a single image or a list of image URLs to extract features using ResNet50. It handles exceptions gracefully and returns a list of feature vectors.

QNS2

Overview

This document analyzes a Python script focused on text feature extraction for a Multimodal Retrieval System, emphasizing text data preprocessing, normalization, and the computation of TF-IDF scores.

Methodologies

Text Preprocessing and Normalization:

Lowercasing and URL Removal: The script converts all text to lowercase and removes URLs and user mentions to standardize the text data.

Tokenization: A regular expression tokenizer is employed to tokenize the text while omitting punctuation.

Stopwords Removal and Stemming: It removes stopwords (using NLTK's predefined list) and applies stemming to reduce words to their root form, utilizing NLTK's SnowballStemmer.

TF-IDF Computation:
Term Frequency (TF): The script calculates the term frequency for each document, indicating the frequency of each term within individual documents.

Document Frequency (DF) and Inverse Document Frequency (IDF): It computes document frequency for each term across all documents and then calculates IDF scores to measure the importance of terms within the corpus.

TF-IDF Scoring: Combines TF and IDF scores to highlight terms that are important and unique to specific documents.

Assumptions

The script assumes textual data contains descriptive information relevant to the retrieval system, requiring preprocessing to extract meaningful features.
Assumes that less frequent, unique words (higher TF-IDF scores) carry more significance for retrieval tasks.

Results
The preprocessing steps successfully clean and standardize the textual data, making it suitable for further analysis.
The script computes TF-IDF scores for each document, enabling the identification of significant terms for similarity comparisons and retrieval operations.

Code Functionalities & Libraries Used

Libraries:

Pandas: For loading and manipulating the dataset containing review texts.

NLTK: Utilized for text preprocessing tasks including tokenization, stopwords removal, and stemming.

Python Standard Libraries: os for directory operations, re for regular expressions (used in URL removal), and pickle for saving processed data.

Key Functions:

normalize_and_process_text(text): Cleans, tokenizes, and processes individual text entries to prepare them for TF-IDF scoring.

compute_tf_idf(doc_tokens_list): Calculates the TF-IDF scores for a collection of documents, aiding in the enhancement of the retrieval system's capability to perform text-based searches.

QNS 3

Overview

This document reviews a Python script designed for a Multimodal Retrieval System, focusing on retrieving similar images and reviews based on input queries. The system processes image URLs and textual reviews, extracting features and comparing them against a precomputed dataset to identify similarities.

Methodologies

Image Processing:

Feature Extraction with ResNet50: Utilizes the pre-trained ResNet50 model to extract features from images fetched from URLs. The script applies a series of transformations (resizing, center cropping, tensor conversion, and normalization) before feature extraction.

Error Handling: Includes robust error handling for issues encountered while fetching or processing images.

Text Processing:

Basic Text Preprocessing: The text data undergoes lowercasing, punctuation removal, and tokenization. The script allows for optional stopwords removal to refine the text further.

Term Frequency (TF) Calculation: Computes term frequencies for tokenized text, normalizing by the total number of words to prepare for similarity comparisons.

Cosine Similarity Computation:
Implements a versatile cosine_similarity function to calculate the similarity between feature vectors (dense vectors for images and sparse vectors represented as dictionaries for text).

Assumptions

Assumes images are accessible via URLs and can be processed into feature vectors compatible with the system's retrieval mechanism.

Text data requires basic preprocessing to enhance retrieval accuracy, assuming that normalized term frequencies provide a sufficient basis for comparison.

Results

Successfully processes input images and reviews, extracting and comparing features to identify the most similar items in the dataset.
Outputs similarity indices and scores for both images and reviews, demonstrating the system's multimodal retrieval capabilities.

Code Functionalities & Libraries Used

Libraries:

PyTorch and torchvision: For image feature extraction using ResNet50 and image preprocessing.

Pandas: To load the dataset containing URLs and review texts.

Pillow (PIL): For opening and processing images from URLs.

Requests: To fetch images from the internet.

NLTK: Used for optional text processing steps, like stopwords removal.

SciPy: Imported but not explicitly used in the provided code snippet; could be intended for additional mathematical operations.

Key Functions:

preprocess_image(image_url): Fetches and preprocesses an image from a given URL, returning extracted feature vectors.

preprocess_text(text): Performs basic text preprocessing and computes term frequencies for the input review text.

cosine_similarity(v1, v2): Calculates cosine similarity between two vectors or sets of vectors, handling both dense and sparse representations.

find_most_similar_images and find_most_similar_reviews: Retrieve top-N similar items based on cosine similarity scores, demonstrating the script's ability to perform targeted retrieval in a multimodal context.

QNS 4

Overview

This document evaluates a Python script designed for ranking and presenting the results of a multimodal retrieval system. The system aims to identify and rank items based on similarities across both image and text modalities, utilizing precomputed similarity scores.

Methodologies

Composite Score Calculation:

Combining Similarities: The script calculates overall scores by averaging the image and text similarity scores for each item. This method allows for a holistic evaluation of items based on their relevance across both modalities.

Result Sorting and Presentation:

Ranking: Items are ranked based on their composite scores, with the highest scores indicating the most relevant items.

Display: The script outputs a list of top matches, providing the indices and composite scores to highlight the most relevant images and reviews.

Assumptions

Assumes that both image and text similarities have been precomputed and stored, facilitating a straightforward averaging process.
Assumes the existence of a dataset containing URLs and review texts corresponding to the indices used in the similarity scores.

Results

Composite Scoring: The script successfully computes overall scores for each item, integrating image and text similarities.

Ranked Output: Outputs a ranked list of items, offering a clear view of the top matches based on composite similarity scores.

Code Functionalities & Libraries Used

Libraries:

Pandas: Used for loading and handling the dataset, facilitating access to URLs and review texts based on indices.

Pickle: For loading precomputed similarity scores and saving the ranked results for future use.

Numpy & Scipy: Although scipy.spatial.distance.cdist is imported, it's not explicitly used in the provided code snippet. The script mainly relies on NumPy for numerical operations.

Key Functions:

compute_overall_scores(img_similarities, txt_similarities): Computes the average of image and text similarities for each item.

fetch_items(data, img_idxs, review_idxs): Retrieves the image URLs and review texts for the top-ranked items, ensuring they are within the data boundaries.

QNS 5

Overview

This document provides a detailed analysis of the Python script A2_ans5.py, which plays a crucial role in the presentation and analysis phase of a Multimodal Retrieval System. The script processes and displays ranked retrieval results, integrating similarities from both image and text data.

Methodologies

Aggregate Score Calculation:

Combining Similarities: For each pair of image and text data, the script calculates aggregate scores by averaging their similarity scores. This method balances the contribution of both modalities towards the final relevance assessment.

Result Presentation:

Ranking and Display: The script ranks items based on their aggregate scores, ensuring the most relevant items are highlighted. It then displays the image URL, review text, and individual similarity scores for each ranked item.

Assumptions

The system assumes the retrieval scores for images and reviews are precomputed and accessible in a structured format, enabling straightforward aggregation.
It presumes that the dataset contains relevant URLs and review texts indexed similarly to the retrieval scores, allowing direct mapping.

Results

Aggregate Scoring: Successfully computes an aggregate score for each item, encapsulating both visual and textual relevance.

Ranked Display: Outputs a list of top matches, specifying URLs, review texts, and similarity scores, effectively summarizing the retrieval results.

Code Functionalities & Libraries Used

Libraries:

Pandas: Facilitates loading and manipulation of the dataset, enabling efficient retrieval of URLs and review texts based on indices.

Pickle: Used for deserializing precomputed retrieval scores, ensuring the persistence of retrieval data across sessions.

Numpy: Employs mathematical operations to compute average scores and aggregate statistics, enhancing data analysis.

Key Functions:

compute_aggregate_scores(img_scores, txt_scores): Merges image and text similarity scores into a unified metric for ranking.

fetch_content_by_index(content_frame, selected_indices): Extracts URLs and review texts for the ranked items, ensuring relevance to the user's query.

showcase_results(content_frame, agg_scores, modified_reviews=None): Presents the retrieval results in a structured and informative manner, with an optional parameter for handling modified or preprocessed reviews.