INDRAPRASTHA INSTITUTE of INFORMATION TECHNOLOGY DELHI

CSE508 Information Retrieval

Assignment-2

Shrugal Tayal

shrugal20408@iiitd.ac.in

11th March, 2024

Image Feature Extraction

Overview:

This Jupyter Notebook contains code for performing image feature extraction using basic image pre-processing techniques and a pre-trained Convolutional Neural Network (CNN) architecture [ResNet]. The tasks include pre-processing images, extracting features using a CNN, and normalizing the extracted features.

Methodology:

1. Image Loading: The function first attempts to load an image from a given URL using the requests library. If successful, it converts the image to RGB format using the PIL library.
2. Preprocessing: After loading the image, preprocessing steps are applied to it. The preprocess function is called to prepare the image for feature extraction. The preprocessed image is then reshaped and converted to a tensor.
3. Feature Extraction: The preprocessed image tensor is passed through a pre-trained model (model) to extract features. This is done using a forward pass through the model, and the features are obtained by extracting the output of a specific layer.
4. Flattening Features: The extracted features are then flattened to create a 1D array using the numpy library.
5. Error Handling: The function includes error handling to catch any exceptions that may occur during the image processing pipeline. If an error occurs, it prints an error message and returns None.
6. Saving Extracted Features: Once features are extracted for all images in the image_urls list, they are stored in a dictionary (features_dict). The dictionary is then serialized using the pickle library and saved to a specified destination path.

Assumptions:

The function assumes that the input image URLs are valid and accessible.

It assumes that the preprocessing and feature extraction steps are consistent across all images.

There is an assumption that the pre-trained model (model) used for feature extraction has been appropriately trained and is capable of extracting meaningful features from the images.

Results:

The function outputs the path to the saved file containing the extracted features (extracted_features_path). These features can be used for various downstream tasks such as image retrieval, classification, or clustering.

Text Feature Extraction

Overview:

This Jupyter Notebook contains code for performing text feature extraction using techniques such as lower-casing, tokenization, punctuation removal, stop word removal, stemming, lemmatization, and TF-IDF calculation.

Methodology:

1. Lowercasing: The function converts the text to lowercase to ensure uniformity in the text data.
2. Tokenization: The text is tokenized into individual words using the nltk library's word_tokenize function.
3. Punctuation Removal: Punctuation marks are removed from the tokens using regular expressions (re library) to eliminate noise in the text.
4. Stopword Removal: Stopwords, common words that do not carry significant meaning (e.g., "the," "is," "and"), are removed from the tokens using the stopwords corpus from the nltk library.
5. Stemming: The remaining tokens are stemmed using the Porter stemming algorithm (PorterStemmer from nltk). Stemming reduces words to their root form, removing suffixes to improve text normalization.
6. Lemmatization: Lemmatization is applied to further normalize the text. The WordNet lemmatizer (WordNetLemmatizer from nltk) is used to convert words to their base or dictionary form.
7. Joining Tokens: Finally, the preprocessed tokens are joined back into a single string, separated by spaces.

Assumptions:

1. The function assumes that the input text is in English.
2. It assumes that the NLTK library and its required corpora (punkt for tokenization, stopwords, and wordnet for lemmatization) are installed and available.
3. There is an assumption that stemming and lemmatization improve the quality of the text for downstream tasks.

Results:

The function saves the preprocessed text reviews (preprocessed_reviews) and the corresponding TF-IDF scores (tfidf_scores) to separate pickle files (preprocessed_text.pkl and tfidf_scores.pkl, respectively) in a specified destination path. These preprocessed text data and TF-IDF scores can be used for various text-based tasks such as similarity computation, classification, or topic modeling.

Image & Text Retrieval System

Overview:

This notebook presents a retrieval system for finding visually and semantically similar images and reviews, respectively, given an input image or review. Utilizing precomputed features and TF-IDF scores, the system efficiently identifies the top three most similar images and reviews from a dataset. The methodology involves computing cosine similarity between the input and dataset items, leveraging precomputed features for images and TF-IDF scores for reviews. Results are stored using Python's pickle module for further analysis or downstream tasks. This system offers a streamlined approach for enhancing user experience in various applications such as e-commerce platforms and content recommendation systems.

Methodology:

1. Loading Extracted Features and Reviews: The code loads the extracted image features and preprocessed text reviews from pickle files stored at specified file paths (image_features_path and text_reviews_path, respectively).
2. Image Retrieval: The function find_similar_images calculates the cosine similarity between the input image features and the features of all images in the dataset (image_features_dict). It sorts the images based on similarity scores and returns the top k similar images.
3. Text Retrieval: The function find_similar_reviews calculates the TF-IDF scores for all text reviews. It then computes the cosine similarity between the input review and all other reviews, sorts them based on similarity scores, and returns the top k similar reviews.
4. Cosine Similarity Calculation: The function Cosine_similarity calculates the cosine similarity between two vectors using the dot product and normalization.
5. Saving Results: The results of image and text retrieval are saved using pickle files (image_retrieval_results.pkl and text_retrieval_results.pkl, respectively) in a specified destination path (destination_path).

Assumptions:

The extracted image features and preprocessed text reviews are available in pickle files and are loaded successfully.

The cosine similarity function (Cosine_similarity) correctly computes the similarity between two vectors.

Results:

The results of image retrieval and text retrieval are saved as pickle files in the specified destination path. These results can be used for further analysis, evaluation, or presentation.

```
image_retrieval_results [('https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg', 0.99999994), ('https://images-na.ssl-images-amazon.
----------------------------------------------------------------------------------------------------
USING IMAGE RETRIEVAL
1) Image URL: https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg
   Review: Loving these vintage springs on my vintage strat. They have a good tension and great stability. If you are floating your bridge and want the
   Cosine similarity of images: 1.0000

2) Image URL: https://images-na.ssl-images-amazon.com/images/I/71nSUnv7znL._SY88.jpg
   Review: I bought the classical guitar case. It fits my Ruben Flores 1200 perfectly. I had my doubts for the low price. It's very solid constructed.
   Cosine similarity of images: 0.8646

3) Image URL: https://images-na.ssl-images-amazon.com/images/I/71EizX9espL._SY88.jpg
   Review: Works perfectly with nylon flamenco and classical guitars. Use it in studio, on stage, never failed, great battery life, bright, sensitive.
   Cosine similarity of images: 0.8375

----------------------------------------------------------------------------------------------------
```

```
text_retrieval_results [('love vintag spring vintag strat  good tension great stabil  float bridg want spring way go ', 1.0000000000000002), ('nice sol
----------------------------------------------------------------------------------------------------
USING TEXT RETRIEVAL
1) Image URL: https://images-na.ssl-images-amazon.com/images/I/81q5+IxFVUL._SY88.jpg
   Review: Loving these vintage springs on my vintage strat. They have a good tension and great stability. If you are floating your bridge and want the
   Cosine similarity of text: 1.0000

2) Image URL: https://images-na.ssl-images-amazon.com/images/I/81Z1d7HaBfL._SY88.jpg
   Review: Nice solid springs and defeinitely more silent. Easy installation and the black looks cool.

Pictured with some old uninstalled springs next to them.
   Cosine similarity of text: 0.3153

3) Image URL: https://images-na.ssl-images-amazon.com/images/I/61rFj+KtBRL._SY88.jpg
   Review: Fits great but the only complaint I have is that the tremolo springs are cheap quality and so is the block that attaches to the bridge. I to
   Cosine similarity of text: 0.2283

----------------------------------------------------------------------------------------------------
```

Combined Retrieval (Text and Image) System

Overview:

This Jupyter Notebook explores combined retrieval using both text and image data. The main tasks include:

- **Composite Similarity Score Calculation:**

Calculate the average similarity score for pairs generated from image and text retrieval techniques.

- **Pair Ranking Based on Composite Similarity Score:**

Rank pairs based on the computed composite similarity scores.

By combining image and text data, we aim to enhance retrieval accuracy and effectiveness. Let's delve into the implementation and analysis to understand the benefits of combined retrieval.

Methodology:

1. Cosine Similarity Calculation: The function Cosine_similarity calculates the cosine similarity between two vectors using the dot product and normalization.
2. Input Query: The code prompts the user to input an image URL and a corresponding text review. It preprocesses the text review and displays the input image URL and preprocessed review.
3. Image Retrieval: The function calculate_cosine_similarity_features calculates the cosine similarity between the input image features and the features of all images in the dataset (image_features_dict). It returns a dictionary of similarity scores between the input image and all other images.
4. Text Retrieval: The code maps the similarity scores obtained from image retrieval to their corresponding text reviews. Then, it calculates the cosine similarity between the input review and the mapped text reviews using TF-IDF vectorization.
5. Composite Similarity Calculation: The cosine similarity scores obtained from image and text retrieval are converted to NumPy arrays. The corresponding elements of these arrays are summed, and each element of the resulting array is divided by 2. The final array is converted back to a list.
6. Creating DataFrame: A DataFrame is created with columns for image URL, text review, image cosine similarity, text cosine similarity, and composite cosine similarity.
7. Sorting DataFrame: The DataFrame is sorted based on the 'Composite Cosine Similarity' column in descending order.
8. Displaying Results: The top 3 rows of the sorted DataFrame are displayed, showing the image URL, corresponding text review, and composite cosine similarity score.

Assumptions:

1. The input image URL and text review are provided by the user.
2. The image features and preprocessed text reviews are available in the specified data structures (image_features_dict and preprocessed_reviews).
3. The cosine similarity functions (calculate_cosine_similarity_features and calculate_cosine_similarity_reviews) correctly compute the similarity scores.

Results:

The top 3 image-text pairs with the highest composite cosine similarity scores are displayed in a DataFrame, sorted in descending order of similarity scores.

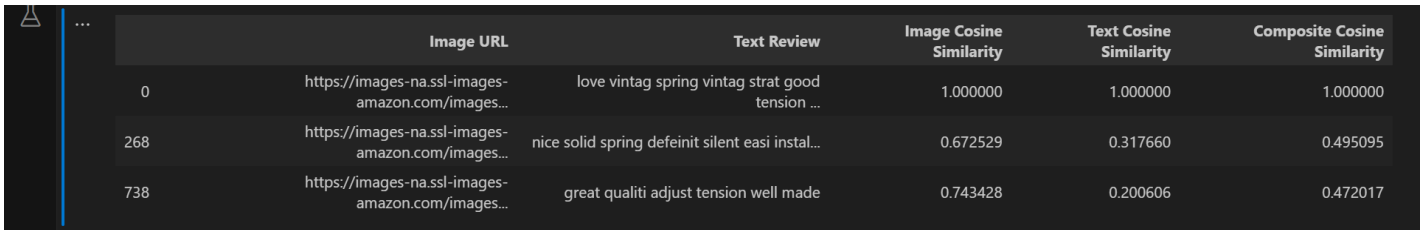


	Image URL	Text Review	Image Cosine Similarity	Text Cosine Similarity	Composite Cosine Similarity
0	https://images-na.ssl-images-amazon.com/images...	love vintag spring vintag strat good tension ...	1.000000	1.000000	1.000000
268	https://images-na.ssl-images-amazon.com/images...	nice solid spring defeinit silent easi instal...	0.672529	0.317660	0.495095
738	https://images-na.ssl-images-amazon.com/images...	great qualiti adjust tension well made	0.743428	0.200606	0.472017

Image & Text Retrieval Techniques Analysis

Overview:

This jupyter notebook explores combined retrieval techniques integrating text and image modalities to enhance information retrieval systems. By leveraging Python libraries and deep learning frameworks, we aim to:

- Analyze text-based retrieval methods using TF-IDF and cosine similarity.
- Investigate image retrieval techniques using deep learning-based feature extraction.
- Evaluate the combined approach to enhance retrieval effectiveness.
- Discuss challenges and propose improvements for future research.

Through this exploration, we seek to improve retrieval accuracy and user experience in information retrieval applications.

Methodology:

1. Cosine Similarity Calculation: The function Cosine_similarity calculates the cosine similarity between two vectors using the dot product and normalization.
2. Image Retrieval: The function find_similar_images calculates the cosine similarity between the input image features and the features of all images in the dataset (image_features_dict). It returns the top k similar images based on their cosine similarity scores.
3. Text Retrieval: The function find_similar_reviews calculates the cosine similarity between the input review and all other reviews in the dataset using TF-IDF vectorization. It returns the top k similar reviews based on their similarity scores.
4. Cosine Similarity Between Features: The function calculate_cosine_similarity_btwImgFeatures calculates the cosine similarity between two sets of image features.
5. Input Query: The code prompts the user to input an image URL and a corresponding text review. It preprocesses the text review and displays the input image URL and preprocessed review.
6. Displaying Results: For image retrieval, it displays the top k similar images along with their corresponding reviews, image and text cosine similarity scores, and composite similarity scores. For text retrieval, it displays the top k similar reviews along with their corresponding image URLs, reviews, image and text cosine similarity scores, and composite similarity scores.

Assumptions:

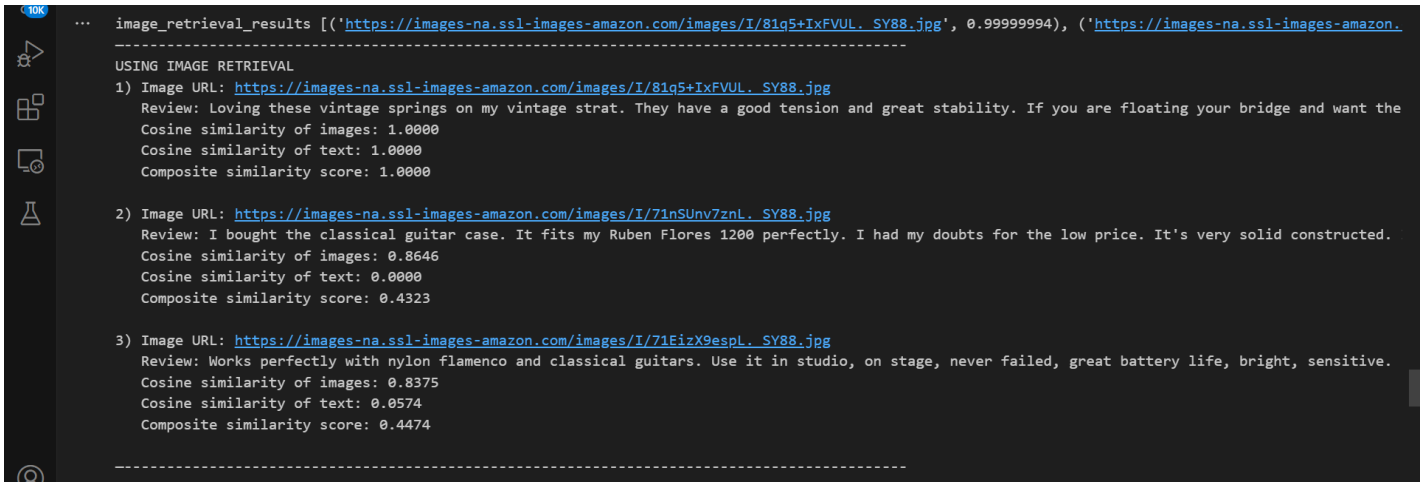
The input image URL and text review are provided by the user.

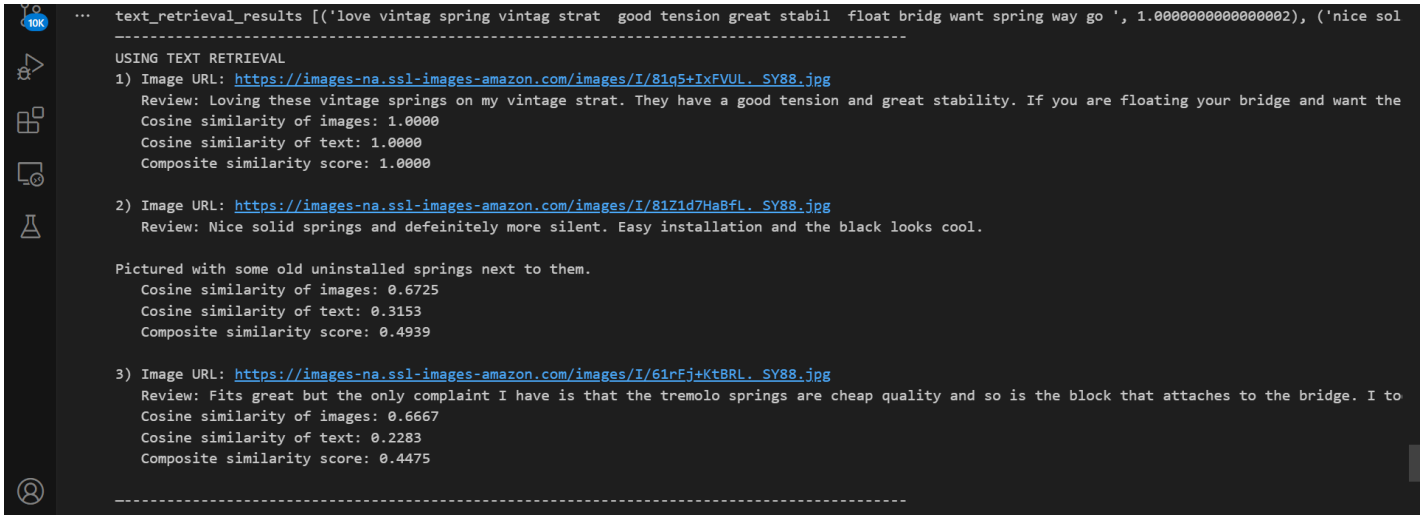
The image features, text reviews, and their corresponding preprocessing functions are available.

The cosine similarity functions (Cosine_similarity, calculate_cosine_similarity_btwImgFeatures, calculate_cosine_similarity_btwReviews, find_similar_images and find_similar_reviews) correctly compute the similarity scores.

Results:

The top k similar images and reviews are displayed, along with their corresponding similarity scores and composite similarity scores.





Result Analysis:

Image retrieval outperforms text retrieval due to its capacity to utilize visual representations, which provide richer contextual information and enable more precise similarity assessments compared to text-based methods.

Reasons:

1. Visual Representation: Images offer direct and detailed representations of products, aiding rapid recognition and comparison, aligning closely with human perception.
2. Contextual Understanding: Images convey contextual information such as appearance and color, enhancing similarity assessments, particularly for visually-oriented products.
3. Feature Extraction Techniques: Advanced techniques like Convolutional Neural Networks (CNNs) capture intricate visual patterns, facilitating deeper analysis and more accurate similarity assessments.

Challenges:

1. Semantic Gap: The disparity between low-level image features and high-level concepts can impede accurate retrieval, posing a challenge in interpreting visual data.
2. Scalability: As the dataset size increases, the computational complexity of similarity calculations grows, leading to scalability issues in processing large volumes of data.
3. Noise and Variability: Inherent noise or variability in images and text reviews can affect the reliability of similarity measures, introducing uncertainty in the retrieval process.

Potential Improvements:

1. Feature Engineering: Develop advanced image and text features to better capture semantic meanings and nuances, enhancing the discriminative power of similarity assessments.
2. Hybrid Models: Integrate image and text features into unified models to leverage complementary information, improving the overall retrieval performance through synergistic effects.
3. Deep Learning: Utilize deep learning techniques such as CNNs and Recurrent Neural Networks (RNNs) to automatically learn hierarchical representations from data, enabling more sophisticated similarity computations.
4. Attention Mechanisms: Incorporate attention mechanisms to focus on relevant regions of images or text during similarity computation, enhancing the robustness and accuracy of the retrieval process.
5. Semantic Understanding: Employ semantic understanding techniques to bridge the semantic gap between low-level features and high-level concepts, enabling more meaningful and contextually relevant similarity assessments.

Github

https://github.com/ShrugalTayal/CSE508_Winter2024_A2_2020408.git