Non-Textual Data Extraction Assignment

Information Retrieval, Extraction, and Integration

Andrés de Vicente Muñoz; andres.devicente@alumnos.upm.es
Laura Teresa Martínez Marquina; lt.mmarquina@alumnos.upm.es
Paula Manso Zorrilla; paula.manso@alumnos.upm.es

1. Introduction. Application proposal

Nowadays, **personalized user experiences** have become paramount, particularly within the realm of sports entertainment. With the surge in popularity of platforms dedicated to **motorsports**, such as magazines or social media accounts, there exists a prime opportunity to enhance user engagement and satisfaction through tailored content delivery. Our proposed system aims to offer a highly personalized experience based on their preferences.

The application proposal revolves around implementing this system within platforms dedicated to sports entertainment, with a specific focus on motorsports. By **analyzing user interactions with images,** the system can discern their interests and preferences, thereby enabling the **delivery of curated content that aligns with their tastes**.

For instance, if a user demonstrates a keen interest in images featuring cars from the Red Bull Racing team, the system will intelligently identify and prioritize such images for the user's consumption. Moreover, it can expand upon these preferences by recommending related content, such as images of other Red Bull-sponsored events or vehicles from similar racing teams.

Furthermore, the application extends beyond content curation to include targeted advertising opportunities. Through subtle integration within the entertainment platform, users may receive subliminal advertising of products and services related to their preferred motorsports brands, enhancing both user engagement and revenue generation for advertisers.

2. Data Base description

The database has been obtained from the Kaggle platform [1] and is made up of images of Formula 1 cars, divided into folders according to the team. Before starting to work with the database, empty images, images containing two or more cars or containing people, were removed. Finally, it was decided to use only the images of the cars on a white background for the project.

3. Technical proposal

The process is essentially divided into 3 steps:

- Firstly, the system receives a query image from which it extracts two descriptors: the RGB histogram of the image (using the *OpenCV* library in Python and calculating the histogram for 256 levels of each of the RGB channels) and text recognition from the images using the *pytesseract* library.
- In the second step, the same descriptors are calculated for all the images in the dataset, resulting in two values for each image.
- Finally, for each image, a similarity value is obtained with respect to our query image by combining these two values. Thus, the system is capable of ordering each of the images from highest to lowest similarity and returning the 5 most similar images.

3.1. First descriptor: 'Smart' Histogram

We utilize OpenCV, a robust computer vision library, to programmatically analyze images. Using the **cv2.calcHist()** function, we compute histograms to understand the distribution of pixel intensities in RGB images, where each color channel ranges from 0 to 255. By narrowing the bin range to [0,200], we mitigate the influence of white background pixels, ensuring a clearer analysis of other colors present.

Subsequently, we employ the **cv2.compareHist()** function in OpenCV to compute the correlation between two histograms. This provides insight into the similarity of their pixel intensity distributions. Utilizing the **cv2.HISTCMP_CORREL** method within **cv2.compareHist()**, we calculate the correlation coefficient, which ranges from -1 to 1. A coefficient close to 1 signifies high similarity (correlation) between images, while a value near -1 indicates dissimilarity. When comparing histograms, our interest lies in obtaining correlation values closer to 1, indicating high image similarity.

The formula applied to calculate the correlation coefficient is the following:

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H_1})(H_2(I) - \bar{H_2})}{\sqrt{\sum_I (H_1(I) - \bar{H_1})^2 \sum_I (H_2(I) - \bar{H_2})^2}}$$

$$\bar{H_k} = \frac{1}{N} \sum_J H_k(J)$$

where:
and N is a total number of histogram bins.

3.2. Second descriptor: OCR

Optical Character Recognition, or OCR, is a technology that enables the user to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data [2].

In this case the python implementation, Tesseract, has been used. This library incorporates several functions, the most important being **pytesseract.image_to_string()** in which several parameters can be configured according to the text layout. The text of the images in the database shows different sizes, fonts, and addresses, so the parameter "-psm11" has been introduced. While this configuration has proven to be the most effective, overall, Tesseract has given poor results.

The words recognized in the images are stored in a list and then, one by one, a check is made to see if the words (or part of them) match between the input image and the images in the dataset. Then, the number of coincidences is used to calculate a similarity measure (*OCR*).

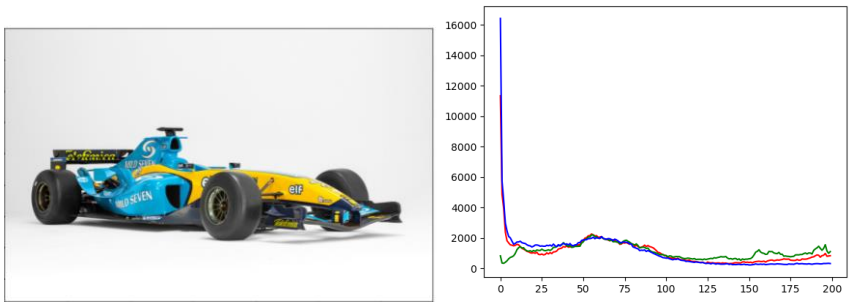
3.3. CBIR implementation

Finally, to implement both predictors as a single similarity value with respect to the query image, it has been decided to weigh each of these values to obtain a value between 0 and 1. Thus, the value obtained with the histogram correlation is weighted by 0.5, so as the text similarity.

$$Similarity\ Value = 0.5x\ histogram + 0.5\ x\ OCR$$

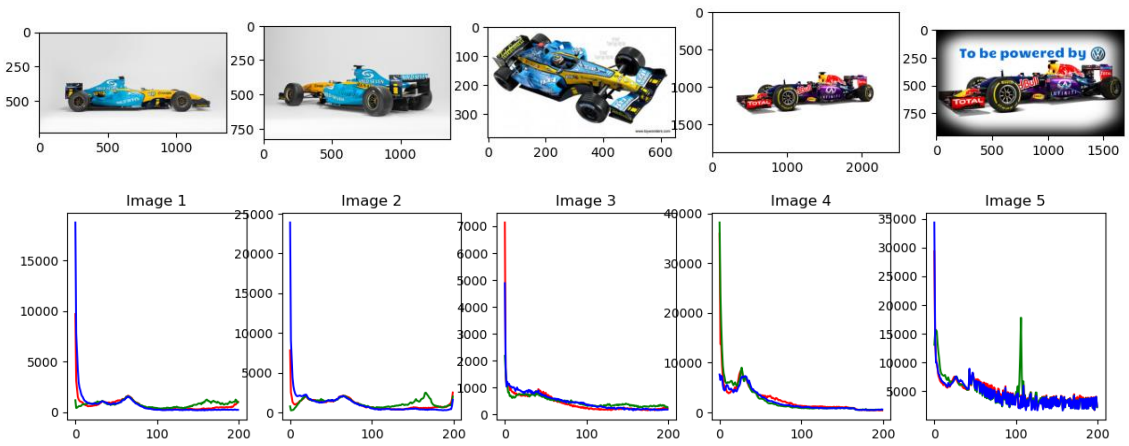
4. Results

Query Image:



```
Words found in query_img: ['mildsevene', 'wie', 'elf', 'stam']
```

Results:



```
Words in image 1: ['mildsev', 'elf,', 'tee', 'mil', 'geve','seeeccnmirm']
---------------------------------
Words in image 2: ['are', 'yraaeeaet', 'sevem', 'wuldseven']
---------------------------------
Words in image 3: ['wee','sex','wnwwtoywonderscom']
---------------------------------
Words in image 4: ['bafanitigd', 'pia', 'otal', 'taa', 'aon', 'wiewo,0y',
                   'cede', 'ges', 'ney', 'pam', 'infinit', 'total']
---------------------------------
Words in image 5: ['tobe', 'powered', 'faite', 'wire', 'hite,gaut',
                   'lwinit', 'total']
---------------------------------
```

5. Conclusion

The outcomes reveal the efficacy of histogram and color comparisons, although text recognition exhibited limitations. Despite the system's simplicity, it exhibits promising potential to fulfill the initial objective outlined in the report.

In conclusion, the CBIR system employed in this work has proven to be effective in its basic functionality. However, to improve accuracy and efficiency the implementation of more advanced algorithms, such as OpenCV EAST [3], is suggested. This algorithm looks for ROIs with a high probability of containing text and uses them as an input for the OCR system. This approach will optimize the process of identifying relevant areas, thus facilitating the work of optical character recognition libraries such as Tesseract.

6. References

[1] Chung, Joon Son & Senior, Andrew & Vinyals, Oriol & Zisserman, Andrew. (2016). Lip Reading Sentences in the Wild.

[2] Sagar Khanna. Formula One Cars (2022). https://www.kaggle.com/datasets/vesuvius13/formula-one-cars

[3]immanuelprathap. OpenCV-Tesseract-EAST-Text-Detector (2021). https://github.com/immanuvelprathap/OpenCV-Tesseract-EAST-Text-Detector

7. Requirements

Open Cv, Pytesseract, numpy, pandas,

8. GitHub repository

https://github.com/LauMarMar17/HMDA_Information_retrieval.git

Note: Refer to “Assignment 2” folder in the repository.