T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

IMAGE RECOMMENDER

ÇAĞRI ÇAYCI

SUPERVISOR
DR. GÖKHAN KAYA

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

IMAGE RECOMMENDER

ÇAĞRI ÇAYCI

SUPERVISOR
DR. GÖKHAN KAYA

2024
GEBZE

GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 10/10/2023 by the following jury.

JURY

Member
(Supervisor) : DR. GÖKHAN KAYA

Member : PROF. DR. İBRAHİM SOĞUKPINAR

ABSTRACT

In today's rapidly growing digital landscape, the sheer volume of visual content poses a challenge for users seeking to discover and engage with material that aligns with their preferences.

To provide personalized suggestions to user, this image recommendation system uses tag-based image recommendation technique. Tag-based image recommendation technique requires to multi-labeling images with abstract and concrete tags. Labeling images is done by an artificial intelligence system. After the images are labelled, a linked list which consists of images is kept for each tag to increase the efficiency of the system. Each interaction of the user with system makes changes on user's preferences. Recommendation algorithm suggests an image based on these preferences.

The results show that the system can recommend an image with higher success than 50% and makes this without checking all images on dataset.

Keywords: Image Recommendation System, Tag-Based Image Recommendation Technique, Multi-Labeling.

ÖZET

Günümüzün hızla büyüyen dijital ortamında, görsel içeriğin büyük hacmi, tercihlerine uygun materyalleri keşfetme ve bunlarla etkileşime geçme arayışında olan kullanıcılar için zorluk teşkil etmektedir.

Kullanıcıya kişiselleştirilmiş öneriler sunmak için bu görsel öneri sistemi, etiket tabanlı görsel öneri tekniğini kullanır. Etiket tabanlı görsel öneri tekniği, görsellerin soyut ve somut etiketlerle çoklu etiketlenmesini gerektirir. Görüntülerin etiketlenmesi yapay zeka sistemi tarafından yapılmaktadır. Resimler etiketlendikten sonra sistemin verimliliğini artırmak amacıyla her etiket için resimlerden oluşan bağlantılı bir liste tutulur. Kullanıcının sistemle her etkileşimi, kullanıcının tercihleri üzerinde değişikliklere neden olur. Öneri algoritması bu tercihlere göre bir görsel önerir.

Sonuçlar, sistemin veri kümesindeki tüm görselleri kontrol etmeden %50'den daha yüksek beğeni oranına sahip bir görsel önerebildiğini göstermektedir.

Keywords: Görsel Öneri Sistemi, Etiket Tabanlı Görsel Öneri Tekniği, Çoklu Etiketleme

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to Gebze Technical University for providing me with the invaluable opportunity to undertake this project. The academic environment and resources at the university have played a pivotal role in shaping and enhancing my research skills.

I extend my sincere appreciation to my advisor, Dr. Gökhan Kaya, for their unwavering support, guidance, and insightful feedback throughout the entire duration of this project.

ÇAĞRI ÇAYCI

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The increasing amount of content on the web can make it challenging for users to find the content they are interested in. This search process consumes their time and energy. To address this issue, major companies like Netflix, Twitter, and Instagram employ recommendation systems tailored to their platform's requirements. For instance, Netflix utilizes a movie recommendation system, while Instagram employs video and image recommendation systems. These systems enhance user experience by reducing the effort and time users need to spend searching for content they enjoy.

1.1. Project Definition

The project necessitates the development of an image recommender application designed to help users discover content of interest efficiently, minimizing time and energy expenditure. The primary objective of this application is to understand users' preferences and propose new images based on their inclinations. To accomplish this goal, a substantial dataset of images needs to be generated and stored initially. Subsequently, an algorithm must be formulated to provide image suggestions. Finally, a user-friendly web application should be crafted to showcase the functionality of the system.

1.2. Fundamental Assumptions of Project

Recommendation systems are based on predicting what a user might find appealing. However, preferences are inherently subjective and vary from person to person. Consequently, there is no universal truth that guarantees a user will definitely like the content recommended to them. In this project, two primary assumptions are considered for suggesting an image to a user. The first assumption posits that users tend to like images that are similar. The second assumption is that similar images share common tags or labels. To put these assumptions into action, it is imperative to label images and provide image recommendations based on these labels.

2. PROJECT STEPS

This project comprises three main steps. Firstly, a collection of images needs to be assembled. Secondly, these images must undergo the process of labeling. Thirdly, an image recommendation algorithm should be developed to suggest relevant images. Finally, a user-friendly web application must be created, integrating the dataset and the recommendation algorithm for seamless user interaction.

2.1. Collecting Images

Collecting a set of images presents a challenging task where the images need to meet criteria such as uniqueness, uniform distribution, and appropriateness for a work environment. In pursuit of this goal, Random Image Generator APIs like API NINJAS and IMAGECDN were employed. It's worth noting, however, that these APIs do not guarantee the uniqueness or appropriateness of the images obtained. The process of managing the images to ensure their appropriateness and uniqueness is discussed in chapter 3 dedicated to challenges.

2.2. Tagging Images

As mentioned in the abstract, this project employs a tag-based recommendation system. Consequently, the process of tagging images holds significant importance, as the entire recommendation system relies on these tags. Achieving satisfactory results in the recommendation system hinges on ensuring that the tags are not only object-related but also content-related. This approach allows abstract tags to be as influential as concrete tags in enhancing the effectiveness of the recommendation system.

2.2.1. Using Google Vision API

The Google Vision API stands out as a robust labeling API, offering a mechanism to assign labels to images based on both object-related and content-related characteristics. The following example illustrates the results obtained from the Google Vision API for a specific image.

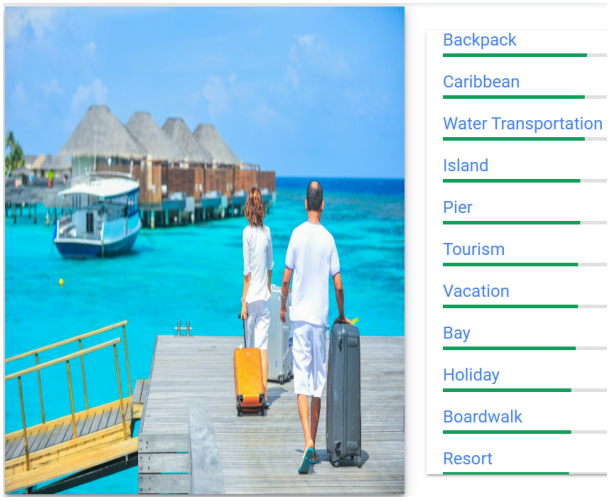


Figure 2.1: An example of labeling with Google Vision API

As the example indicates, the image is labeled with not only object-related labels like ”Backpack” and ”Resort” but also content-related labels such as ”Holiday”, ”Tourism”, and ”Vacation”. This dual approach to labeling enhances the richness and relevance of the tags assigned to the image.

2.2.2. Preparing the Dataset

After selecting the appropriate labeling API, all images in the dataset were labeled with averaging 19.18 labels. To conduct a more detailed analysis, statistics of the labels were calculated. Subsequently, it was determined that there are 4977 unique labels in the dataset. Furthermore, the frequencies of label occurrences were computed and visualized in the following plot.

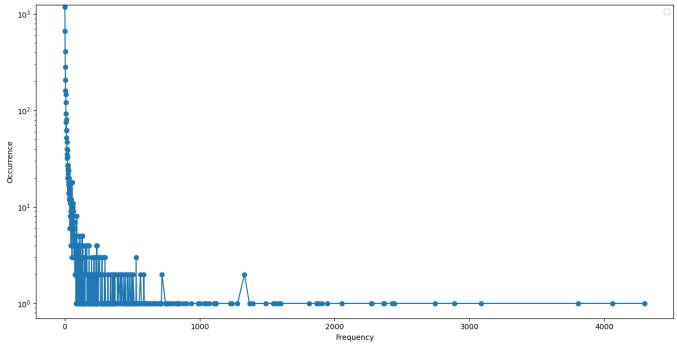


Figure 2.2: Frequencies of each occurrence

Table 2.1: Detailed table of ”Frequencies of each occurrence” graph

Frequency	Number of Labels
1	1187
2	670
3	480
.	.
.	.
4299	1

The graph and table together illustrate that there are 1187 labels that occur only once, 670 labels that occur only twice, and so forth. One challenge highlighted by these statistics is that some labels are very rare. For instance, there is only one image in the dataset that contains the label ”Granola”. Consequently, liking an image with the label ”Granola” would have no significant impact on the recommendation system. This issue is prevalent for labels with relatively small frequencies within the dataset. Another challenge arises from the presence of very frequent labels. Labels that are excessively common pose a problem for the system, as they lack discriminative power when used in the recommendation algorithm. Employing a label that is associated with half of the images in the dataset does not effectively narrow down the recommendation list.

To address the challenges posed by very frequent and very rare labels, the optimal solution considered was their removal. However, outright deletion of these labels might result in some images having a very small number of labels, diminishing their representativeness. After conducting various tests, it was determined that a lower limit of 5 occurrences for very rare labels and an upper limit of 500 occurrences for very frequent labels provided a balanced solution. These limits resulted in a loss of only 200 images, as opposed to other limits.

Table 2.2: Detailed table of image loss with different thresholds (The ”Bottom Limit Frequency” and ”Upper Limit Frequency” signify the frequency thresholds for a label to be retained and not be deleted. The ”Lower Limit Labels” represents the minimum number of labels an image must contain to avoid its deletion.)

Buttom Limit Frequency	Upper Limit Frequency	Lower Limit Labels	Image Loss
10	1000	10	2000
5	1000	10	1500
5	500	10	5000
5	500	4	200

2.3. Structuring the Dataset

The initial version of the dataset structure was designed as follows: the dataset comprises image entries, and each image entry contains the corresponding labels for that image. This structure was chosen to facilitate the application of Naive Bayes Classification. However, it's worth noting that Naive Bayes Classification was ultimately not utilized, as explained in section 4.2.

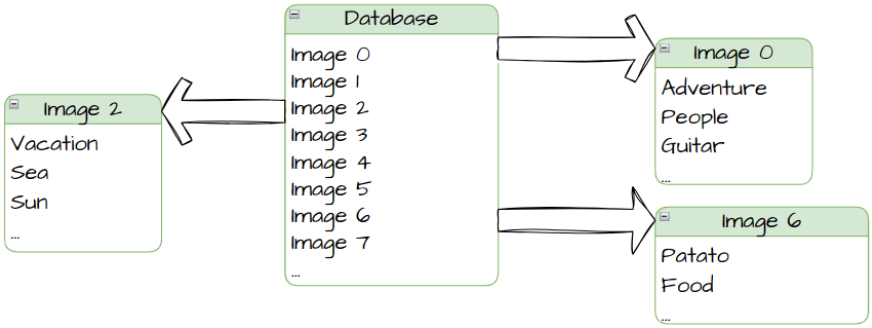


Figure 2.3: Representative image of initial structure of dataset

In addition to the previous data structure, the dataset is enhanced with another structure that enables access to images based on their labels. This supplementary structure involves label entries, where each entry contains the names of images associated with that particular label. Such a structure is crucial for the efficiency of the recommendation algorithm, as it allows for the swift retrieval of images based on their associated labels.

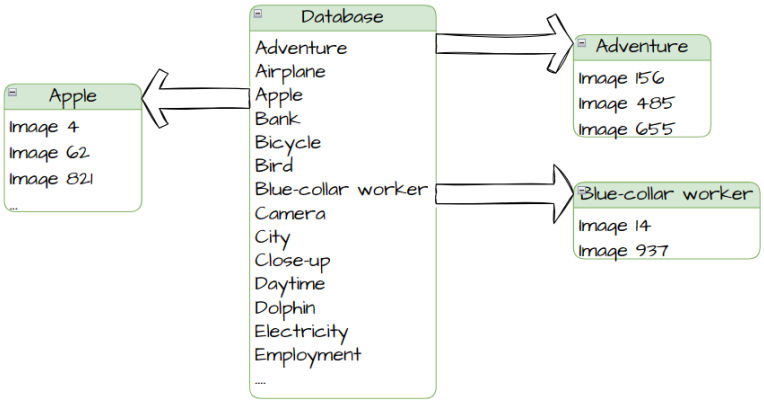


Figure 2.4: Representative image of secondary structure of dataset

2.4. Developing Image Recommendation Algorithm

To provide image suggestions to users, the recommendation system needs to learn users' preferences by enabling them to interact with images in the dataset. The system then predicts an image that the user might like based on their interactions and preferences.

2.4.1. Interacting with Images

Each user is associated with a vector of favorites, where each element in the vector represents a unique label from the dataset. Initially, all elements of the vector are set to 0, and they are updated based on users' interactions. For instance, when a user likes an image, the labels of that image are extracted from the dataset. For each label, the corresponding element in the vector is increased by one. Conversely, when a user dislikes an image, the corresponding element in the vector is decreased by one. This vector serves as a representation of the user's preferences, evolving with their interactions. This process is shown in the following sketch.

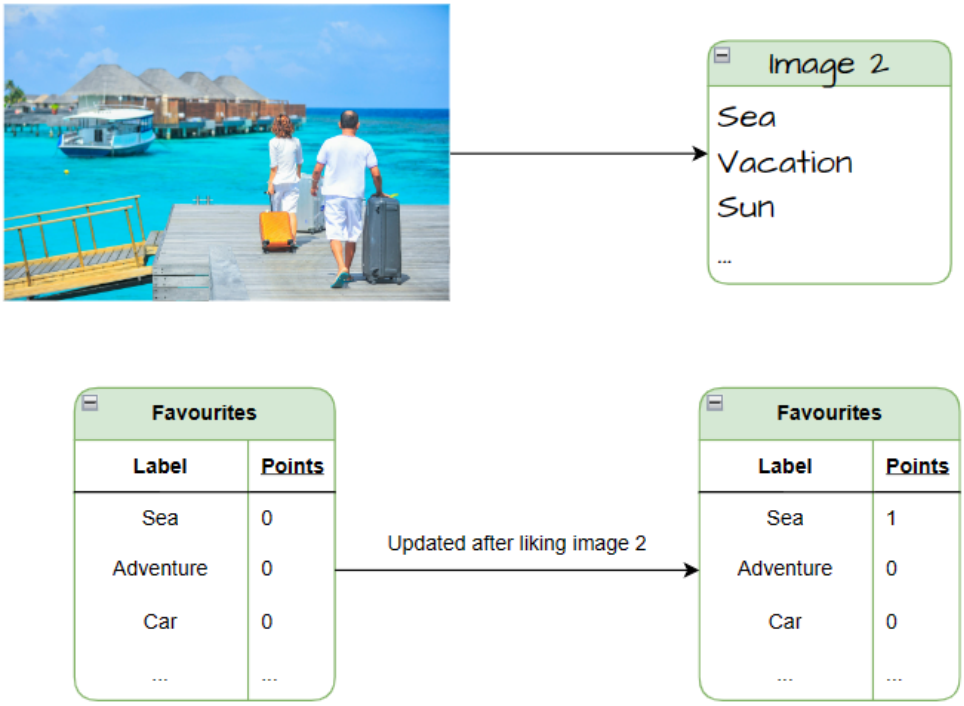


Figure 2.5: Changes of favourites vector after liking an image

2.4.2. Predicting an Image

Predicting an image that a user might like is a complex task. To address this, a point scoring system is employed. Each image is assigned points based on its labels and the weights associated with those labels obtained from the user's favorites vector as in subsection 2.4.1. However, scoring all images in the dataset can be computationally expensive and time-consuming. To mitigate this, only the top 10 favorite labels are considered. Instead of scoring all images, only those containing one of the top 10 favorite labels are evaluated. This significantly reduces the number of images to be checked, as each label is not present in more than 5000 images (according to the Upper Limit Frequency which is determined in subsection 2.2.2). However, in general, a maximum of 2500 images is checked. The following algorithm outlines the precise implementation of the point scoring system.

Algorithm 1 Recommendation Algorithm

Require: favourites[] subsection 2.4.1
Require: images[] (Figure 2.3) ▷ images keeps the labels for images
Require: labels[] (Figure 2.4) ▷ labels keeps the images for labels
  $favourites \leftarrow sort(favourites)$
  $possibleImages \leftarrow []$
  for $i \leftarrow 0$ to 10 **do** ▷ Getting images from top 10 labels
    for $j \leftarrow 0$ to $length(labels[favourites[i])$ **do**
      $imagegetslabels[favourites[i]][j]]$
      if $image$ not in $possibleImages$ **then**
        $possibleImages.append(image)$
      end if
    end for
  end for
  $max \leftarrow 0$
  $maxIndex \leftarrow -1$
  for $i \leftarrow 0$ to $length(possibleImages)$ **do**
    $currentPoint \leftarrow 0$
    for $j \leftarrow 0$ to $length(images[possibleImages[i]])$ **do**
      $currentPoint \leftarrow currentPoint * favourites[images[possibleImages[i][j]]]$
    end for
    if $currentPoint >= max$ **then**
      $max \leftarrow currentPoint$
      $maxIndex \leftarrow i$
    end if
  end for
  return $maxIndex$

2.5. Developing Web Application

To demonstrate that the system works, it needed to be integrated with a platform. In this project, integrating the system into a web application was deemed more appropriate. This choice was motivated by the need to test the system with objective real users, and a web application is generally more accessible than other platforms.

The web application required an authentication mechanism and a user-friendly interface for interacting with images. Authentication was necessary so that each user could have an account to track their statistics and store their favorite labels.To implement the authentication mechanism, login and sign-up pages were designed.

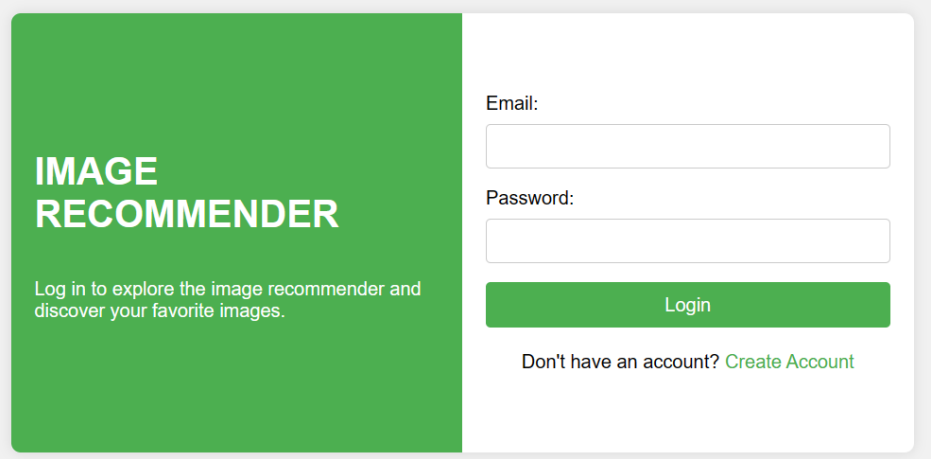


Figure 2.6: Login Page

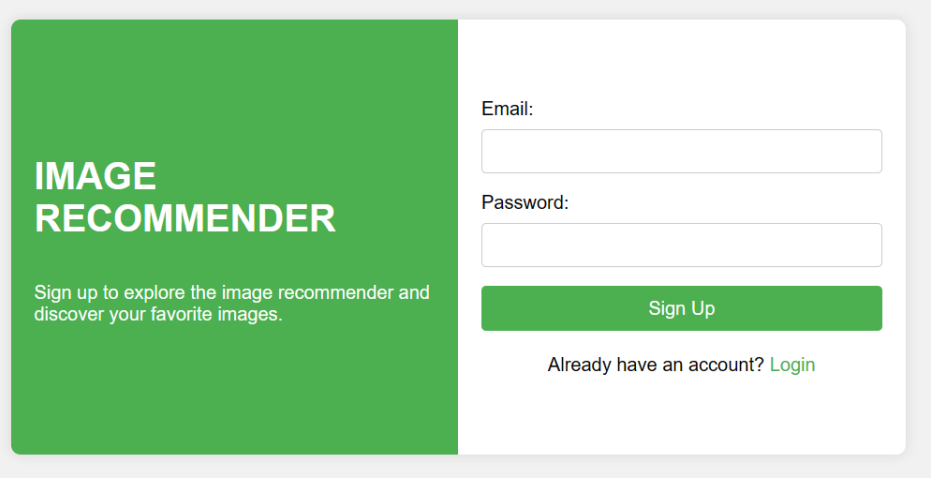


Figure 2.7: Sign up Page

The web application prompts the user to select 5 positive and 5 negative images sequentially. The primary reason for this requirement is to establish a foundation for understanding users' preferences. Upon completion of this process, the web application is now ready to suggest new images to the user based on their interactions and established preferences.

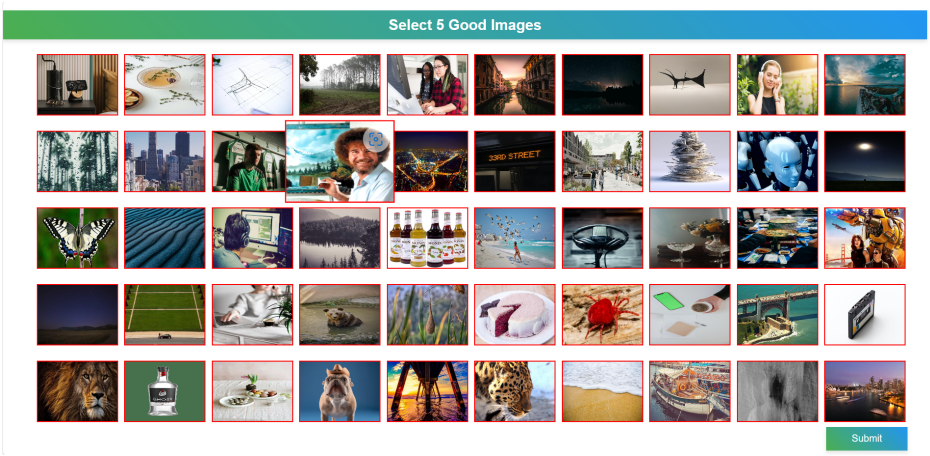


Figure 2.8: Selecting Images

The home page offers various functionalities, including liking and disliking an image, displaying the labels of the image, and showing users' statistics. When the user interacts with an image, the system takes into account both previous and current interactions to recommend a new image immediately. Additionally, it updates the labels associated with the current image and relevant statistical data.

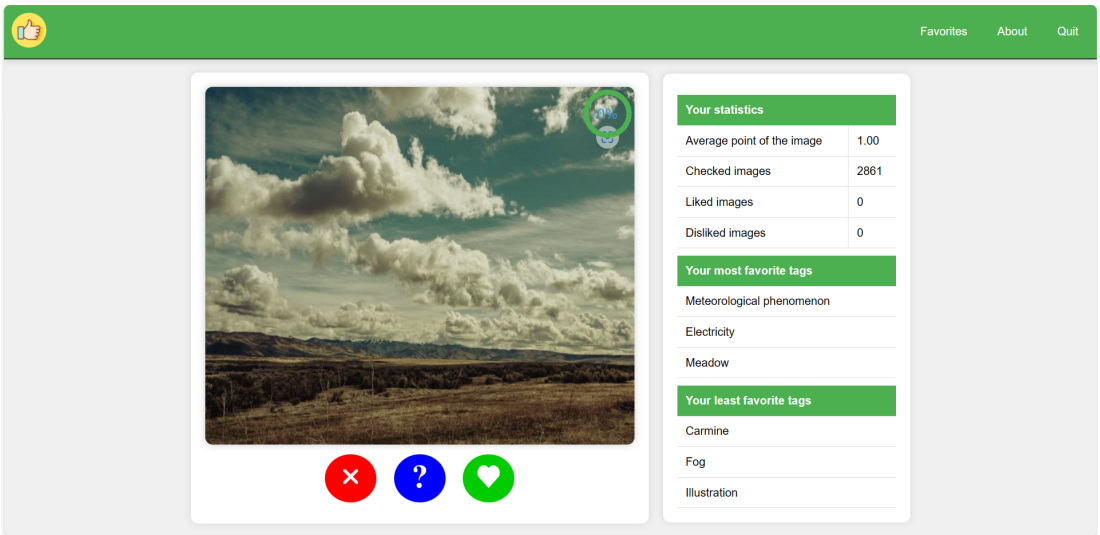


Figure 2.9: Home Page

The web application also includes an ”about” page that introduces the project, and a ”favorites” page that displays the top 20 favorite labels as a histogram.

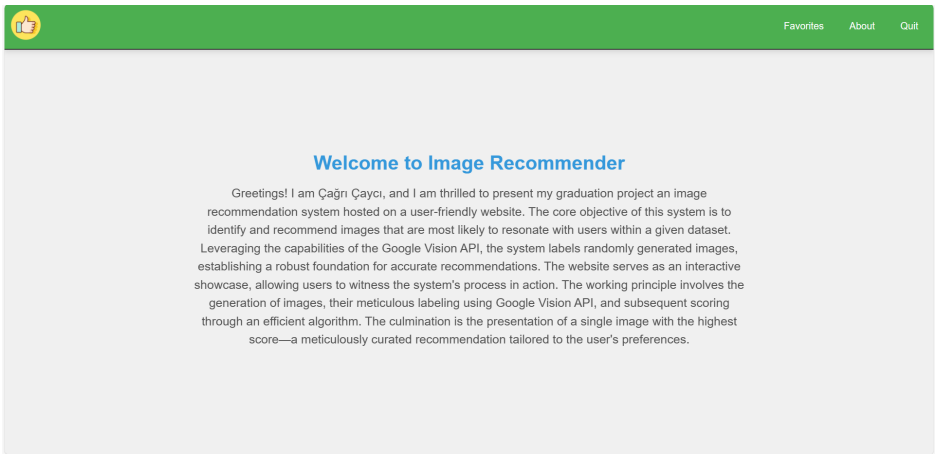


Figure 2.10: About Page

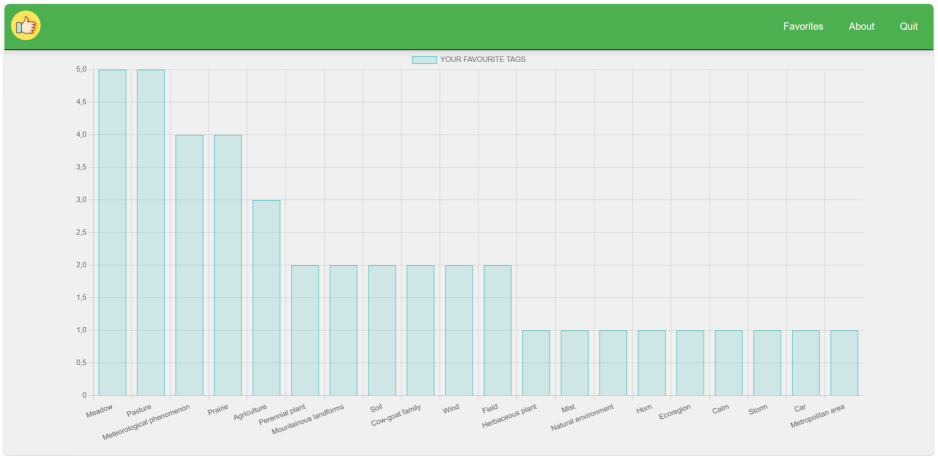


Figure 2.11: Favourites Page

2.5.1. Services

The web application relies on Firebase to manage all backend operations, encompassing crucial tasks like authentication and storage. This ensures seamless and secure user interactions. Additionally, Github serves as the platform for hosting the web application, providing a reliable environment for its accessibility and performance. Together, Firebase and Github play integral roles in the functionality and availability of the web application, working hand in hand to deliver a smooth user experience.

3. CHALLENGES

During the development of the project, several challenges were encountered and addressed through practical solutions.

3.1. Collecting Unique Images

Following the integration of the web application with both the dataset and the recommendation algorithm, it became apparent that the dataset contained duplicate images, with several copies of the same pictures. To address this issue, an algorithm was developed to identify and remove duplicate images from the dataset. Upon applying this algorithm, the dataset was successfully refined from 35,000 images to 9,000 unique images.

The identification of unique images was accomplished through this algorithm, which involves comparing the size and hash of each image. The comparison of sizes serves as a preliminary step, and hashing is only utilized when two images share equal sizes. The ”hashlib” library was employed for the hashing process, providing a robust mechanism for generating unique identifiers for images. This method ensures a comprehensive and reliable approach to identifying unique images within the dataset.

Algorithm 2 Finding Unique Images

Require: images[]
  $imageSizes \leftarrow []$
  $uniques \leftarrow []$
  for $i \leftarrow 0$ to $length(images)$ **do**
    $imagesSizes.append(getSize(images[i]))$
  end for
  for $i \leftarrow 0$ to $length(images) - 1$ **do**
    $unique \leftarrow True$
    for $j \leftarrow i + 1$ to $length(images)$ **do**
      if $imageSizes[i] == imageSizes[j]$ **then**
        if $hash(images[i]) == hash(images[j])$ **then**
          $unique \leftarrow False$
        end if
      end if
    end for
    if $unique == True$ **then**
      $uniques.append(images[i])$
    end if
  end for

3.2. Increasing Number of Images

The reduction in dataset size due to the removal of identical images necessitated an increase in the number of images. However, the API initially used for generating random images had depleted its supply of unique images. To overcome this limitation, the Google Search Images API was utilized to obtain additional unique images. In each iteration, a label was randomly selected, and this label was then used to search for an image on Google. This approach ensured the addition of uniformly distributed types of images to the dataset. Ultimately, this process successfully augmented the number of unique images from 9,000 to 13,000.

```
def getRandomImageFromGoogle(developer_key, custom_search_cx, labels):

    index = random.randint(1, len(labels))

    for i in range(0, 3000):
        try:
            gis = GoogleImagesSearch(developer_key=developer_key, custom_search_cx=custom_search_cx)
            search_params = {
                'q': labels[index],
                'num': 10,
                'safe': 'high',
                'imgType': 'photo',
                'fileType': 'jpg',
                'imgSize': "large"
            }

            gis.search(search_params=search_params, custom_image_name=str(counter))

            if(len(gis.results()) > 0):
                image = gis.results()[random.randint(0, len(gis.results()) - 1)]

                image.download(source_folder)

                resize_image(source_folder+ str(counter) + ".jpg", source_folder + str(counter) + ".jpg", (640,480))

                counter += 1

        except Exception as e:
            print(f"Error in iteration {i}: {e}")
```

Figure 3.1: The implementation of creating images using Google Images Search

3.3. Collecting Proper Images

Given that the images used in this project were sourced from the internet, there was a possibility that some of them might not be suitable for all audiences. To mitigate the risk of exposure to inappropriate content, a thorough manual inspection of all images in the dataset was conducted. As a result of this process, 300 images deemed inappropriate were identified and subsequently removed from the dataset.

4. FAILED APPROACHES

During the project's development, various approaches were experimented with. However, not all of them proved successful due to several reasons, leading to their modification or replacement.

4.1. Hierarchical Clustering

During the dataset preprocessing, it was identified that there are 4977 unique labels for all images, with some being deemed unimportant or very rare. For instance, the label "poached eggs" occurs only 8 times in the dataset. To address this, the idea of merging less important labels with more general ones using Hierarchical Clustering became tempting. To implement this, the "spaCy" library was employed. The library offers functionality to convert words into vectors, simplifying the measurement of distance between two labels.

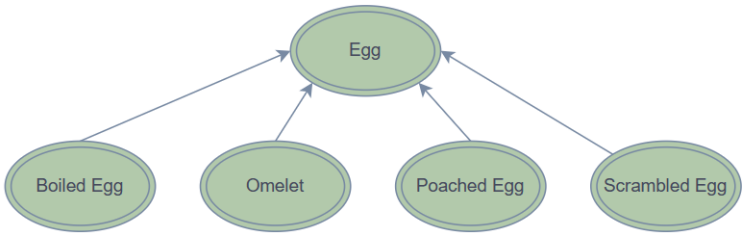


Figure 4.1: Hierarchical Clustering for Egg.

Hierarchical clustering is accomplished through an iterative process where, in each iteration, all labels in the dataset are compared with each other using cosine similarity. At the conclusion of each iteration, the two labels with the highest similarity are merged into one. This process continues until there are k clusters remaining, where k is determined based on the characteristics of the dataset. The representative label for clusters is determined as the label that is both present in the cluster and occurs most frequently within that cluster.

Algorithm 3 Hierarchical Clustering

Require: labels[]
  $N \leftarrow$ labels length ▷ N represents the length of labels and change dynamically.
  $max \leftarrow 0$
  $maxIndices \leftarrow -1, -1$
  while $N > 1$ **do**
    for $i \leftarrow 0$ to $N - 1$ **do**
      for $j \leftarrow i + 1$ to N **do**
        $currentSimilarity \leftarrow sim(labels[i], labels[j])$
        if $currentSimilarity$ bigger than max **then**
          $max \leftarrow currentSimilarity$
          $maxIndices \leftarrow i, j$
        end if
      end for
      Merge $labels[i]$ with $labels[j]$ and add labels array
      Remove $labels[i]$ and $labels[j]$ from labels array
    end for
  end while

4.2. Naive Bayes Classification

The initial strategy when designing recommendation algorithm involved the application of Naive Bayes Classification to categorize images as either positive (liked by the user) or negative (disliked by the user). Naive Bayes was selected for two primary reasons. Firstly, as a probability-based classification algorithm, it facilitates the calculation of the likelihood that a user will like a particular image. This not only yields a label (positive or negative) but also provides a probability associated with the classification. Secondly, the simplicity and ease of implementation of Naive Bayes made it an obvious choice for this phase of the project.

However, it became evident that calculating the probability for each image was computationally expensive and time-consuming. This prolonged processing time could negatively impact the efficiency of the user-centric system. As a result, alternative approaches were explored to address these computational challenges.

Algorithm 4 Finding $P(feature \mid status)$

Require: $trainX[], trainY[], feature$
  $totalYes, totalNo, occurenceYes, occurenceNo \leftarrow 0, 0, 0, 0$
  for $i \leftarrow 0$ to $length(trainX)$ **do**
    if $trainX[i] == 1$ **then**
      $totalYes \leftarrow totalYes + 1$
      for $j \leftarrow 0$ to $length(trainX[i])$ **do**
        if $trainX[i][j] == feature$ **then**
          $occurenceYes \leftarrow occurenceYes + 1$
        end if
      end for
    end if
    if $trainX[i] == 0$ **then**
      $total_no \leftarrow total_no + 1$
      for $j \leftarrow 0$ to $length(trainX[i])$ **do**
        if $trainX[i][j] == feature$ **then**
          $occurenceNo \leftarrow occurenceNo + 1$
        end if
      end for
    end if
  end for
  return $occurenceNo/totalNo$, $occurenceYes/totalYes$

Algorithm 5 Naive Bayes Classification

Require: $trainX[], trainY[], test_x[]$
  $probabilities \leftarrow \{\}$
  $results \leftarrow []$
  for $i \leftarrow 0$ to $length(test_x)$ **do**
    $probabilityNo \leftarrow 1$
    $probabilityYes \leftarrow 1$
    for $j \leftarrow 0$ to $length(test_x[i])$ **do**
      $feature \leftarrow test_x[i][j]$
      if $featurenotinprobabilities$ **then**
        $probabilities[feature] \leftarrow [probability(trainX, trainY, feature)]$
      end if
      $probabilityNo \leftarrow probabilityNo * probabilities[feature][0]$
      $probabilityYes \leftarrow probabilityYes * probabilities[feature][1]$
    end for
    if $probabilityYes >= probabilityNo$ **then**
      $results.append(1)$
    else
      $results.append(0)$
    end if
  end for
  return $results$

5. EVALUATION

The success of the project is contingent upon meeting specific criteria. For this project, the success criteria include achieving a recommendation algorithm with high accuracy, surpassing 50%. Additionally, the process of finding a suitable image to suggest should take less than 10 seconds. These benchmarks serve as crucial indicators for the effectiveness and efficiency of the recommendation system.

5.1. Efficiency

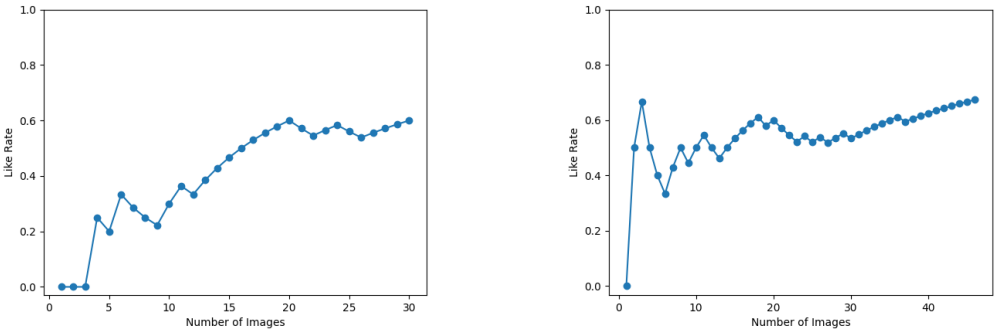
Efficiency is a crucial aspect of the project, given its user-centric nature. The primary goal of a recommendation system is to save users' time in finding content they are interested in. Therefore, the process of finding an image to predict must be quicker than the time it would take for a user to find it manually. This emphasis on efficiency is fundamental to enhancing the overall user experience and fulfilling the core objectives of the recommendation system.

As stated in subsection 2.4.2, the recommendation algorithm utilizes a point scoring system to score images based on the labels they contain and the weights of those labels (where the weight represents the number of images containing the label and liked by the user). If the number of images in the dataset is denoted as N, and the number of labels for an image is denoted as M, the time complexity of this process would be O(NM). However, as determined in subsection 2.2.2, a label contains an image list with at most 500 images. Additionally, as established in subsection 2.4.2, not all labels are used in the recommendation algorithm; only the top 10 favorites of the user are considered. Consequently, the algorithm operates with a constant number of labels, each containing 500 images at most. Therefore, the time complexity of the algorithm decreases to O(5000M), which is equivalent to O(M). Although 5000 images might be considered high and potentially threatening for a dataset consisting of 13,000 images, the number of images checked is independent of the dataset volume. No matter the size of the dataset, the number of images checked is at most 5000. Additionally, experiments show that, in each iteration, at most 2000 images are checked and the process described takes at most 2 seconds.

On the other hand, the space complexity of this process is O(NM) because all images and their corresponding labels must be stored for the recommendation algorithm.

5.2. Accuracy

Another crucial aspect of the project is delivering more accurate results to users. Users do not want to waste their time on images they do not like, so higher accuracy contributes to an enhanced user experience. To measure the accuracy of the recommendation algorithm, the system needed to be tested by objective real users. After 50 different trials of the system with 20 different users, the following statistics were calculated.



(a) label 1 (b) label 2

Figure 5.1: Like rate graphs of random two users

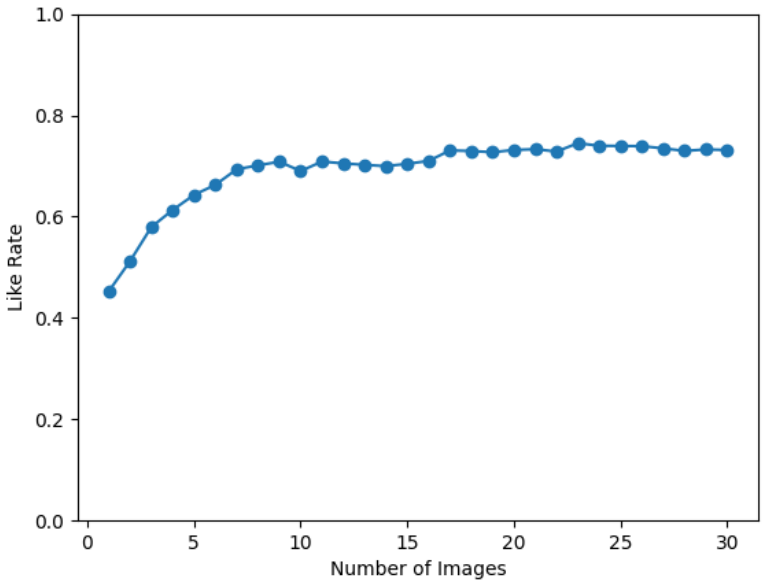


Figure 5.2: Average like rate of all users in system

While the statistics for each user may vary, the overall data suggests that the system better understands the user as the number of interactions with images increases. After a user interacts with 20 images, the accuracy of the system converges to 0.8. This indicates that the recommendation algorithm becomes more effective and accurate as users provide more input and the system learns their preferences over time. However, the decreasing like rate after 20 images may caused by over-fitting.

5.3. Conclusion

After evaluating the system performance through tests with objective real users, it was observed that the system meets the necessary requirements. It can suggest images with high accuracy, averaging 0.6919 in 30 images, and accomplishes this in less than 10 seconds, specifically 2 seconds. This demonstrates the effectiveness and efficiency of the recommendation system in providing relevant image suggestions to users.

SOURCES

- Das, D., Sahoo, L., & Datta, S. (2017). A survey on recommendation systems. International Journal of Computer Applications, 160(7). Foundation of Computer Science.
- Webb, G. I., Keogh, E., & Miikkulainen, R. (2010). Naïve Bayes. Encyclopedia of Machine Learning, 15(1), 713-714.
- Understanding Hierarchical Clustering.
- Creating Graphs.
- Google Vision API Documentation.
- Firebase Documentation.
- Google Images Search API.
- Random Image API of API NINJAS.
- Random Image API of IMAGECDN.