A web-based user interface for browsing and annotating image collections using a state-of-the-art content-based image search engine

Author: Paula Gómez Duran
Advisors: Xavier Giró-i-Nieto, Eva Mohedano and Kevin McGuinness

Universitat Politècnica de Catalunya (UPC)
Dublin City University (DCU)
2017 - 2018
Abstract

This thesis consists of developing a web user interface for a content-based image retrieval (CBIR) system in order to provide a visualization of the obtained results and eventually to improve them by capturing the user’s feedback.

First of all, it is studied the best way to develop a UI capable of incorporating the machine learning based system that computes a list of similar images given a visual query. The relation between the different programming languages and the amount of data in the different datasets utilized in this thesis are also included in order to achieve a good performance for the system.

The UI consists in a web interface that allows different ways of selecting the query image to search within a particular dataset. Thus, the interface provides an option of using some of the dataset images as queries for the CBIR system, as well as an option to experiment with custom image that is not part of the original dataset, either uploading it from a file or providing its URL to the system.

In addition, when computing the ranking of similar images for a given query, different functionalities or modes are provided to the user: the explorer, the annotation, and the query expansion mode.

The explorer mode allows the user to navigate through a particular dataset by selecting one of the images provided in a ranking and perform a new search.

The query expansion and the annotation modes collect user annotations to improve the ranked list obtained by the CBIR system. In fact, the query expansion mode allows the user to select multiple images as input queries for the system in order to provide a richer representation of the ranking. In the annotation’s mode, instead, the user can annotate the success and failure images of the ranking and submit them to the system, which will finally train an SVM model in order to improve the ranking and update it.

Ten different users were asked to interact during 10 min with the UI and complete a form to evaluate the usability and design of the final of the proposed system. Also, a quantitative and qualitative evaluation was conducted to evaluate the benefit of the annotation and QE modes with respect the original rankings.

Results indicate that the UI achieved is intuitive, robust, consistent and built with a useful purpose. Regarding the utilities, it is concluded that each mode has its own utility and that query expansion mode its better for improving low accuracy rankings, while annotation mode it is also useful when having a high accuracy ranking thus we can annotate just the negative images and use it as a filter.

Finally, the achieved goals are discussed and a study which has also been carried out in this project is used with the purpose of listing all the possible implementations suggested to improve the system.
Resum

Aquesta tesi consisteix a desenvolupar una interfície d’usuari web per a un sistema de recuperació d’imatges basat en contingut (CBIR) per tal de proporcionar una visualització dels resultats obtinguts i, finalment, millorar-los mitjançant la captura dels comentaris de l’usuari.

En primer lloc, s’estudia la millor manera de desenvolupar una interfície web d’usuari capaç d’incorporar el sistema basat en l’aprenentatge automàtic que calcularà una llista d’imatges similars a partir d’una consulta visual. Per aconseguir un bon rendiment del sistema, també s’estudia la relació entre els diferents llenguatges de programació i la quantitat de dades dels diferents datasets utilitzats.

La interfície d’usuari consisteix en una web que ofereix diferents formes d’introduir la imatge que es vol consultar al sistema. Això es pot fer mitjançant l’opció proporcionada pel mateix sistema, que consisteix a usar algunes de les imatges del mateix dataset com consultes per al sistema CBIR, però d’altra banda, també existeix l’opció d’experimentar amb imatges personalitzades que no formen part del dataset original, ja sigui carregant-des d’un arxiu o proporcionant el seu URL al sistema.

A part, en calcular la classificació d’imatges similars per a una consulta determinada, es proporcionen diferents funcionalitats o modes a l’usuari: el mode “explorer”, el mode “annotation” i el mode “query expansion”.

El mode “explorer” permet a l’usuari navegar a través d’un dataset particular en escollir una de les imatges proporcionades al rànquing per així ser capaç de realitzar una nova cerca.

Els modes ”query expansion” i ”annotation” recullen anotacions d’usuari per millorar el rànquing obtingut pel sistema CBIR. De fet, el mode ”query expansion” permet a l’usuari seleccionar múltiples imatges com consultes d’entrada per al sistema per tal de proporcionar una representació més acurada de la classificació. En canvi, el mode ”annotation” permet a l’usuari anotar les imatges d’èxit i fracàs del rànquing per després enviar-les al sistema, que finalment entrenarà un model SVM per millorar el rànquing i actualitzar-lo.

Se’ls va demanar a deu usuaris diferents que interactuessin durant 10 minuts amb la interfície d’usuari i completessin un formulari per avaluar la usabilitat i el disseny final del sistema proposat. A més, es va realitzar una avaluació quantitativa i qualitativa per avaluar el benefici de l’anotació i les maneres QE respecte als rànquings originals.

Els resultats indiquen que la interfície aconseguida és intuïtiva, robusta, consistent i construïda amb un propòsit útil. Pel que fa a les utilitats, es conclou que cada mode té la seva pròpia utilitat i que el mode ”query expansion” és millor en tractar de millorar els rànquings amb baixa precisió, mentre que el mode ”annotation” també és útil en tenir una classificació d’alta precisió. Això és degut al fet que aquest últim mode permet anotar també les imatges negatives arribant a usar el mode com a filtre d’errors.

Finalment, es plantegeu els objectius assolits i es fa ús d’un estudi, també dut a terme en aquest projecte, amb la finalitat de llistar totes les possibles implementacions suggerides per a la millora del projecte.
Resumen

Esta tesis consiste en desarrollar una interfaz de usuario web para un sistema de recuperación de imágenes basado en contenido (CBIR) con el fin de proporcionar una visualización de los resultados obtenidos y, finalmente, mejorarlos mediante la captura de los comentarios del usuario.

En primer lugar, se estudia la mejor manera de desarrollar una interfaz web de usuario capaz de incorporar el sistema basado en el aprendizaje automático que calcula una lista de imágenes similares a partir de una consulta visual. La relación entre los diferentes lenguajes de programación y la cantidad de datos de los diferentes datasets utilizados en esta tesis también se incluyen para lograr un buen rendimiento para el sistema.

La interfaz de usuario consiste en una web que ofrece diferentes formas de seleccionar la imagen de consulta para buscar dentro de un conjunto de datos particular. Por lo tanto, la interfaz proporciona una opción de usar algunas de las imágenes del propio dataset como consultas para el sistema CBIR, así como una opción de experimentar con imágenes personalizadas que no forman parte del dataset original, ya sea cargándolo desde un archivo o proporcionando su URL al sistema.

Además, al calcular la clasificación de imágenes similares para una consulta determinada, se proporcionan diferentes funcionalidades o modos al usuario: el modo "explorer", el modo "annotation" y el modo "query expansion".

El modo explorador le permite al usuario navegar a través de un dataset particular al seleccionar una de las imágenes proporcionadas en un ranking y realizar una nueva búsqueda.

Los modos de "query expansion" y "annotation" recopilan anotaciones de usuario para mejorar el ranking obtenido por el sistema CBIR. De hecho, el modo "query expansion" permite al usuario seleccionar múltiples imágenes como consultas de entrada para el sistema con el fin de proporcionar una representación más rica de la clasificación. En cambio, en el modo "annotation" el usuario puede anotar las imágenes de éxito y fracaso del ranking y enviarlas al sistema, que finalmente entrenará un modelo SVM para mejorar el ranking y actualizarlo.

Se les pidió a diez usuarios diferentes que interactuasen durante 10 minutos con la interfaz de usuario y completasen un formulario para evaluar la utilidad y el diseño final del sistema propuesto. Además, se realizó una evaluación cuantitativa y cualitativa para evaluar el beneficio de la anotación y los modos QE respecto a los rankings originales.

Los resultados indican que la interfaz lograda es intuitiva, robusta, consistente y construida con un propósito útil. En cuanto a las utilidades, se concluye que cada modo tiene su propia utilidad y que el modo "query expansion" es mejor tratando de mejorar los rankings de baja precisión, mientras que el modo "annotation" también es útil al tener una clasificación de alta precisión, ya que podemos anotar solo las imágenes negativas usando este modo como un filtro.

Finalmente, se discuten los objetivos alcanzados y se hace uso de un estudio también llevado a cabo en este proyecto con la finalidad de enumerar las posibles implementaciones sugeridas para mejorar el sistema.
Acknowledgements

First of all, I would like to thank my tutor Xavier Giró-i-Nieto for making it easy to get in touch with Kevin McGuinness as soon as he was in UPC thus making it possible my collaboration with him in the project.

Secondly, I would like to thank my supervisor on DCU Kevin McGuinness for being available for me whenever I needed to, as well as his patience when explaining me and teaching me all the new content, specifically the one referring to machine learning.

I would also like to thank to Eva Mohedano, for all her advices, her awesome feedback and also her collaboration when integrating my work to her own project. Also for all her explanations and her support when coming up with new ideas.

Thanks to Andrea Calafell for letting me use her template.
## Revision history and approval record

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03/12/2017</td>
<td>Document creation</td>
</tr>
<tr>
<td>1</td>
<td>8/01/2018</td>
<td>Document revision</td>
</tr>
<tr>
<td>2</td>
<td>9/01/2018</td>
<td>Document revision</td>
</tr>
<tr>
<td>3</td>
<td>12/01/2018</td>
<td>Document approbation</td>
</tr>
</tbody>
</table>

### DOCUMENT DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>Name</th>
<th>e-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paula Gómez</td>
<td><a href="mailto:paula.maria.gomez@alu-etsetb.upc.edu">paula.maria.gomez@alu-etsetb.upc.edu</a></td>
</tr>
<tr>
<td>Xavier Giró i Nieto</td>
<td><a href="mailto:xavier.giro@upc.edu">xavier.giro@upc.edu</a></td>
</tr>
<tr>
<td>Kevin McGuinness</td>
<td><a href="mailto:kevin.mcguinness@insight-centre.org">kevin.mcguinness@insight-centre.org</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Written by:</th>
<th>Reviewed and approved by:</th>
<th>Reviewed and approved by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>8/01/2018</td>
<td>Paula Gómez</td>
<td>Xavier Giró i Nieto</td>
</tr>
<tr>
<td></td>
<td>Project Author</td>
<td>Project Supervisor</td>
</tr>
<tr>
<td>Date</td>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>10/01/2018</td>
<td>Kevin McGuinness</td>
<td>Kevin McGuinness</td>
</tr>
<tr>
<td></td>
<td>Project Supervisor</td>
<td>Project Supervisor</td>
</tr>
</tbody>
</table>
# Contents

1 Introduction .................................................. 11

1.1 Statement of purpose ....................................... 11

1.2 Requirements and specifications ............................ 13

1.3 Methods and procedures ................................... 13

1.4 Work Plan .................................................. 14
  1.4.1 Work Packages ........................................ 14
  1.4.2 Gantt Diagram .......................................... 15

1.5 Incidents and Modification .................................. 16

2 State of the art ................................................ 17

2.1 A Front-end in ReactJS ..................................... 17

2.2 A Back-end in NodeJS and Python ......................... 18
  2.2.1 NodeJS ................................................ 18
  2.2.2 Python ............................................... 19

2.3 Deep Learning .............................................. 19
  2.3.1 Single Neural model ................................... 20
  2.3.2 Neural Networks ....................................... 21
  2.3.3 Gradient Descent ..................................... 21
  2.3.4 Convolutional Neural Networks ....................... 22
  2.3.5 Retrieval system ..................................... 23

3 Methodology .................................................... 24

3.1 Baseline .................................................... 24
  3.1.1 Datasets .............................................. 24
  3.1.2 Connection React - node - python .................. 25
  3.1.3 DATABASE and DATASETS ......................... 26
List of Figures

1.1 Subjective ways of annotating images in a dataset .......................... 11
1.2 Gantt Diagram of the Degree Thesis ........................................... 15
2.1 NodeJS handling events. Image source [1] ........................................ 19
2.2 Single neuron .............................................................................. 20
2.3 Popular activation functions ......................................................... 20
2.4 Neural network composition ......................................................... 21
2.5 Visualization of one step of gradient descent optimization ................. 22
2.6 Convolutional Neural Network architecture. Image source[4] ............... 22
2.7 Visualization of the local connection of one neuron within a convolutional layer where the input depth $C = 1$ and the kernel size is $1 \times 3 \times 3$ ........................................ 22
2.8 Spatial max pooling with $2 \times 2$ kernel and stride 2 applied on a single activation map of a convolutional layer ........................................ 23
3.1 Query examples from the three different datasets. Top: Oxford buildings(1st) and Paris buildings(2nd); Bottom: Instre dataset. ..................... 24
3.2 Image to visualize the home page .................................................. 26
3.3 Image to visualize where to put the URL for the query ...................... 27
3.4 Image to visualize where to upload the file for the query .................. 27
3.5 Image to visualize the Oxford dataset selection ................................ 28
3.6 Image to visualize the Paris dataset selection ................................... 28
3.7 Image to visualize the instre dataset selection .................................. 28
3.8 Image to visualize the main Oxford buildings dataset page ............... 29
3.9 Image to visualize the main Paris buildings dataset page .................. 29
3.10 Image to visualize the main Instre dataset page .............................. 30
3.11 explorer mode ........................................................................... 30
3.12 annotation mode ........................................................................ 31
3.13 query expansion

3.14 PARTS url

3.15 Visualization of the Instre dataset structure. The subdivision in different folders (paths) and the number of images in each one can be seen in this figure.

4.1 Qualitative example of how the AP of an image improves, thus affecting the mAP of the system.

4.2 Quantitative evaluation through a screenshot of the server with AP and mAP computed for the previous queries.

4.3 Example of how the AP of an image does not improve when it is already high in the original rank provided by the BLFC system. Qualitative evaluation.

4.4 Quantitative evaluation through a screenshot of the server with AP and mAP computed for the previous queries. It can be seen how as the AP of the query decreases, it decreases as well the mAP of the system.

4.5 Example of the best 10 queries vs the 10 worst ones.

4.6 Example of ranking updated after some feedback annotations.
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Table showing the results of the UI feedback experiment</td>
<td>33</td>
</tr>
<tr>
<td>5.1</td>
<td>Budget of the project</td>
<td>40</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Statement of purpose

An image retrieval system is a computer system for browsing, searching and retrieving images from a large database of digital images. Early systems [20, 6, 7] relied on image metadata such as a set of keywords or tags, or textual descriptions of the image content. In this way, the user could then search for a particular image by entering a text description or tags, to retrieve all images annotated with the same description. These systems are known as text-based image retrieval systems, and they present two major issues:

The first issue is the one involving manual image annotation. This task is time-consuming, laborious, and expensive due the volume of digital media created nowadays.

The second major and even more important issue is that the text-based image retrieval system relies solely on a textual annotation of an image which can actually be described in multiple ways. This ambiguity on the description is known as the semantic gap problem [12] and involves the subjectivity of human perception, as it can be observed in 1.1, leading in wrong or undesired retrieval results.

Figure 1.1: Subjective ways of annotating images in a dataset
So, it is for that issues that there has been a large amount of research directly analyzing the image content instead of relying in the image annotations.

Content Based Image Retrieval (CBIR)\cite{1} emerged as a research field in the early '90s fixing the task of creating text annotations which described the content of the images to search for the image in the dataset. CBIR aims then at organizing and structuring datasets of images based on their content rather than the associated metadata. The images, are then described by computer vision algorithms which "summarize" the content of an image to a numerical vector called image representation. The fact of not depending of image annotations avoid the problem of having subjective descriptions as in \cite{1} and also having to manually generate them.

This project consists in creating a User Interface for a state of the art content based image retrieval search engine due the need of an UI system for the demos of it\cite{24}. Having a proper UI is extremely useful for an image search engine for two main reasons:

- The first one is the importance of results visualization. Most of the CBIR systems \cite{14}, \cite{16}, \cite{25} base their evaluation in a quantitative assessment (i.e mean Average Precision), so it is possible to perform an objective comparison between them. However, qualitative assessments are even more important. The reason for this is the relevance in terms of showing good and bad examples of the results so that it can be understood where the system fails. This identification allows researchers to improve and to focus on useful and adequate topics.

- The second is the ability to capture the user’s intent. Thus, by means of using a UI, it is possible to collect annotations for a given list of retrieved images, making possible to automatically refine the search results based on the user’s feedback.

The final interface should be capable of visualizing successful or failure cases to assist in improving the search engine technology as much as annotating the images to provide more extensive training data, thus becoming an interface for rapid qualitative evaluation of possible enhancements to the underlying technology. A well designed UI will help in order to understand the functionalities of the system, both in cases where it works well and those in which the system needs to improve.

The project main goals are:

- To understand the image retrieval problem and study the wide range of research on it.
- To design and build a User Interface with all the functionalities needed which interacts with a server which contains the image retrieval system.
- To build a demonstrator for showcasing the latest developments in content based image retrieval.
- Visualization of random experiments by using the query expansion strategy \cite{8}
- Annotation of failure cases in the computed rankings to assist in improving the search engine technology.
- Researching on apply relevance feedback \cite{27} in image retrieval system.
1.2 Requirements and specifications

This project has been carried out at Insight, which is a center of Data Analytics allocated in the Dublin City University (DCU). It has been developed as a tool to use the retrieval image system and it could be used for other students or developers in the future to participate in upcoming challenges by recycling or improving it.

The requirements of this project are the following:

- To explore the similarity among different images by surfing through different datasets, being able also to compare with external images.
- Being able to experiment by studying the relationship among different features of the images using the multi query strategy without annotating the feedback.
- Annotate the failures on the wrong images computed in the rankings thus helping the system to improve.
- Evaluate the results and make a comparative study among all of them.

The specifications are the following

- Use the software of Eva’s Mohedano PHD thesis to compute all the machine learning code (image retrieval engine).
- Developed in Python, ReactJs and NodeJs.

1.3 Methods and procedures

This project arises as a collaboration opportunity with The Insight Center for Data Analytics in DCU. The idea came up from Kevin McGuinness, the supervisor from this center, and the baseline of the project was the participation in Eva’s Mohedano PHD thesis due the need of a UI system as a demo for her state of the art image retrieval engine.

Therefore, the built interface will merge knowledge of machine learning with web programming so given any user’s request in the form of a query image, the system produces a list of images that are relevant to the query, ordered by a similarity score.

Convolutional neural networks (CNNs) based descriptors perform remarkably well in retrieval benchmarks like Oxford and Paris Buildings but, however, state-of-the-art solutions for more challenging datasets such as INSTRE [26] have not yet adopted equal contribution. Nevertheless, our system will try to correct it.

1Code in the repository: https://github.com/imatge-upc/salbow
On one hand, the proposed system provides a way to visualize and to annotate the results on three different image retrieval datasets: Oxford [17], Paris [18] and Instre [26]. While the first two have captured a lot of attention during the past years [14], [25], Instre has not yet being widely analyzed. This project makes it easy to qualitative assess the performance of an state of the art system [16] in the popular Oxford and Paris benchmarks, but also in the new and challenge Instre instance search dataset.

On the other hand, the UI system is built from scratch as well as the server. Then, a python server is also developed in which the image retrieval engine code is incorporated thus all the JavaScript code in ReactJs and in NodeJs is developed in this project while the retrieval engine code is adapted from the Eva Mohedano’s PHD thesis. This code carries out all the computation of the rankings of similarity of the images as well as to retrain the system when the feedback is provided.

1.4 Work Plan

This project has followed the established work plan, with a few exceptions and modifications explained in the section 1.5.

1.4.1 Work Packages

- WP 1: Documentation
- WP 2: Investigation of the involved technologies
- WP 3: Setup Server and Front end
- WP 4: API REST development and integration
- WP 5: App testing (critical review of the project)
- WP 6: Oral communication (writing and presentation of the project)
1.5 Incidents and Modification

During the project we decided to increase the workload and duration of the tasks in the WP 2 at the same time as keeping the timetable. The reason emerged from the opportunity to investigate deep inside in machine learning while following the lecture notes. So while developing the project I was also facing an assignment required for the module in which a ‘Spam classifier’ had to be built and trained.

Regarding the WP5, some modifications in improving some functionalities of the website came up during this package. So, during this period we did not just test the application but amplified with some new functionalities.

These two were basically the main changes in the programmed timetable.
Chapter 2

State of the art

2.1 A Front-end in ReactJS

ReactJS is an open-source JavaScript library which is used for building user interfaces specifically in single page format applications. It was first created by Jordan Walke, who is a software engineer working for Facebook. As it is logical for that reason, ReactJS was first deployed on Facebook (2011) and later on Instagram (2012). Nowadays, a lot of companies such as Netflix, Yahoo, WhatsApp or Atlassian are also implemented by this JavaScript framework.

This new technology allows developers to create large web applications which can change data without reloading the page so that its main features are scalability, speed and simplicity. Furthermore, a one-way data binding is used to simplify things. For example, each time someone types in an input field of the UI, ReactJS doesn’t directly changes the view. Instead, it updates the internal state in the component (model), save these values and only then, update the view with a new render method, so that what it was typed initially is now displayed in the input field.

ReactJs provides utilities which cover the need to develop tools in order to allow users to profit from all the visual information. It allows to locate the needed images with good precision in a reasonable time, being able to achieve many applications and purposes which are necessaries for a relevance feedback in image retrieval.

Dealing with DOM, browsers need to recalculate all the CSS and carry out layout repainting operations on the website, and this takes time. However, ReactJS enable developers to make changes on the website without modifying the DOM, and it is done by the Virtual DOM.

The Virtual DOM is, in fact, the core of what makes ReactJS fast at rendering user interface element and their changes and it can be described as a lightweight and abstract model of the DOM. ReactJS uses the render method to create a node tree from React components and update it in response to changes in data model due from actions. Therefore, each time that the underlying data in the application changes, a new Virtual DOM representation of the UI is created by React.

To sum up, it can be said that when something changes, react’s DOM re-render the UI in a virtual DOM representation, compute the difference between the previous Virtual DOM and the new one, and finally update the DOM with what had actually changed as if a patch had being applied.

---

1. [http://manglar.uninorte.edu.co/handle/10584/7772](http://manglar.uninorte.edu.co/handle/10584/7772)
2. One-way data binding is also known as unidirectional data flow and it provides a predictable application state. The data just flow in one direction breaking the typical structure MVC (model-view-controller) where either the view could change the model state or either way back. In ReactJS, there is only a one-way trip from state to views, there’s no trip back.
3. The Document Object Model is a HTML document created by the browser when a page is loaded. It is a standard for how to get, change, add, or delete HTML elements.
2.2 A Back-end in NodeJS and Python

Web servers are a remote computers or computer programs that delivers web content to the user upon a request through a web browser. Their purpose is to run one or many services in order to solve the needs of the network programs as well as the requests of different clients. Another main purpose of web servers is to manage all the data and be able to test the system without letting all the applications fail in the user’s part. For instance, without having a server, a simple misspelling can cause the website to crash, forcing the system to regain access and undo what it did. However, these are not the only useful features of a web server. In fact, users do not require downloading plugging and thus the website loads faster.

There are two server environments to choose from: local and remote. A local server is hosted locally on an own computer while a remote server is hosted elsewhere. As argued in this article[^4], it is a good practice to first built a local server, for example, because for some debugging and testing operations may be extremely slow or even impossible if attempted on a remote copy of the code. However, there are as well many reasons for building a remote server from scratch, for example because it avoids having to store and maintain two copies of the site files and database. Nevertheless, when there are many tasks that involve the contribution of different work environments[^5] as it is the case of this project, having a local server can help to the parallel work and the correct distribution. More about this topic can be read in the section 3.1.2.

2.2.1 NodeJS

Node.js[^10] is a JavaScript runtime environment built on Chrome’s V8 JavaScript engine. Ryan Dahl, the creator of Node.js, gave to all developers a tool for working in the non-blocking, event-driven I/O paradigm. In fact, after a long term period based on the stateless request-response paradigm, web applications with real-time and two-way connections finally emerged.

Node.js shines in building fast and scalable network applications[^5], as it is capable of handling a huge number of simultaneous connections with high throughput. However, using it for CPU-intensive operations and heavy computation will annul nearly all of its advantages.

Compared to traditional web-serving techniques, NodeJS is completely different. While in the first ones, each connection (request) spawns a new thread taking up system RAM and eventually maxing-out at the amount of RAM available, Node.js operates on a single-thread using non-blocking I/O calls, being able to handle tens of thousands of concurrent connections simultaneously (held in the event loop).

Despite Node.js is extremely powerful in many different aspects, there is one area where it falls for not being able to handle CPU-intensive operations. In this area, other technologies like Python can handle much better the numerical and scientific computations.

[^4]: https://www.ross.ws/content/website-development-local-vs-remote
[^5]: In this code is required the implementation of different part of the code by different people.
2.2.2 Python

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. It is one of the best technologies when dealing with scientific computation, as it consists on a lot of libraries or modules which facilitate it. Besides providing large standard libraries that include areas like string operations, Internet, web service tools and operating system interfaces and protocols, it is characterized by improving programmer’s productivity and integrating the Enterprise Application Integration that makes the development of Web Services easy by invoking COM or COBRA components [22].

Python is actually preferred for most of the software development companies due its versatile features and fewer programming codes. However, python drawbacks appear when accessing to a database, dealing with runtime errors or computing memory intensive tasks. It is also very slow when referring to speed in request or response petitions processing [23].

To summarize, for developing a website where the back-end is as complex as a content based image retrieval system, the fact of combining python and node.js is a good choice.

2.3 Deep Learning

Deep learning is a sub-field of artificial intelligence and family of machine learning methods based on learning data representations. However, it substitutes in fact the old style machine learning tasks, where the features where manually engineered by learning layers of features and computing the representation by using model architectures composed of multiple nonlinear transformations of the input data.

Architectures such as multi-layer perceptron, convolutional deep neural networks and recurrent neural networks have been applied in fields like computer vision, speech recognition or natural language processing due they promising results in all these different tasks [9].

---

6https://medium.com/@ktachyon/javascript-concurrency-model-dc98dabc527
2.3.1 Single Neural model

A single neuron (or perceptron) consists of a number \( I \) of inputs \( x_i \) and one output \( y \). Generally, the inputs correspond to raw features of a single training example and each of them have a weight \( w_i \) \((i = 1, \ldots, I)\) associated. Usually, an additional input \( x_0 \) is permanently set to 1 (associated to a weight \( w_0 \)) and it is called the bias term. The activation of the neuron is computed by adding its weighted inputs \( a = \sum_{i=1}^{I} w_i x_i \). Then, the final output \( y \) is set as a function of the activation.

![Figure 2.2: Single neuron](image)

The activation function \( f \) converts an input signal of a node to an output signal and it is often a non-linear function. In the figure 2.3 below, it can be observed the behavior of three popular activation functions: Sigmoid, Tanh and Rectified Linear Unit (ReLU). A neuron can then be seen as a linear regression or as a linear decision boundary classifier.

![Figure 2.3: Popular activation functions](image)

Multi-class classification or regression problems can be addressed by considering simultaneously multiple neurons. In the case of classification, it is common to apply as the activation function the *softmax*, which is defined as:

\[
f(i) = \frac{e^{a_i}}{\sum_j e^{a_j}}
\]  

(2.1)

where \( a_i \) is the weighted sum of the inputs for the node \( i \). This function normalizes the outputs so that they are a categorical probability distribution over the possible outcomes.
2.3.2 Neural Networks

Neural networks are computing systems inspired by the biological neural networks that constitute animal brains. They are composed of one input layer, one output layer and one or more hidden layers, where each is composed of neurons which connect all the inputs between both layers as it is shown in the next figure 2.4 (also called multi-layer perceptron or fully connected networks).

The number of hidden layers and neurons per layer are hyperparameters that need to be carefully selected. Usually, the more layers added the more capacity the network has. However, the amount of parameters to estimate grows very quickly with the number of layers and neurons, and so it can lead to an overfitting model, which will fit the training data extremely well but will have a very poor generalization on unseen data.

2.3.3 Gradient Descent

Gradient descent (GD) is a simple first-order iterative optimization algorithm for finding a (local) minimum of a function. Given a cost function associated with a model $\mathcal{L}(y, \hat{y})$, the algorithm consists in updating the parameters in the direction in which the function changes with the greatest rate.

If $\hat{y}$ is the output of a NN composed of K hidden layers, where $\alpha$ is the hyperparameter learning rate defining the size of the step towards the minimum of the loss, the update to the weights on a neuron in layer $i$ can be described as $W_{(i)} \leftarrow W_{(i)} - \alpha \nabla W_{(i)} \mathcal{L}(y, \hat{y})$

In practice, Gradient descent is very slow, as it requires calculating the loss and the gradients for all the training set to update a single parameter. For that reason, mini-batch Stochastic Gradient Descent (SGD) is used instead, which consists in shuffling the training dataset first and then, taking a batch of N samples to compute a stochastic estimation of the gradient of the cost, updating the $W_i$ parameters.
2.3.4 Convolutional Neural Networks

Deep convolutional neural networks (convnets) have recently become popular in computer vision, since they have dramatically advanced the state-of-the-art in tasks such as image classification [2], retrieval [3] or object detection [11][13].

The main difference between CNNs and the fully connected Neural Networks are the convolutional and pooling layers, which make CNNs particularly suitable for processing spatially structured data such as images.

A convolutional layer outputs a volume of neurons. The depth of the volume is denoted by a set of kernels or filters. Conceptually, the kernels (filters) are slid or ‘moved’ across the spatial dimensions of the input like a sliding window and they change its position by $S$ neurons (stride) each step. Besides, their weights are shared or fixed across the spatial positions of its input, with the assumption that if a particular visual pattern is relevant in one position of the image, it will also be relevant in another one.

Figure 2.5: Visualization of one step of gradient descent optimization.

Figure 2.6: Convolutional Neural Network architecture. Image source[4]

Figure 2.7: Visualization of the local connection of one neuron within a convolutional layer where the input depth $C = 1$ and the kernel size is $1 \times 3 \times 3$. 
The pooling layers are used to reduce the spatial dimension of the convolutional volumes. They operate spatially over a convolution volume. Max pooling is usually applied between consecutive layers to reduce the dimensionality of the representations and achieve some translation invariance.

Figure 2.8: Spatial max pooling with $2 \times 2$ kernel and stride 2 applied on a single activation map of a convolutional layer.

### 2.3.5 Retrieval system

The quality of a retrieval system allows to check whether the system is working properly or not. The estimation of the quality can be done by inspecting the first retrieved elements relevant to a query and this is a good practice just to check the functioning of the system.

However, a quantitative evaluation is necessary to effectively determine the performance of a system. Usually, measures such as precision (seen in equation 2.2), recall (seen in equation 2.3), and mean Average Precision are computed to evaluate the performance of a retrieval system on a particular dataset.

\[
\text{Precision} = \frac{\text{relevant items retrieved}}{\text{retrieved items}} \tag{2.2}
\]

\[
\text{Recall} = \frac{\text{relevant items retrieved}}{\text{relevant items}} \tag{2.3}
\]

Average Precision (AP) is commonly computed to estimate the quality of an ordered list and it is actually the area under the precision-recall curve, which is computed with a finite sum over every position in the ranked sequence of documents: $AP = \sum_{i=1}^{n} p(i) \Delta r(i)$, where $p(i)$ is the precision at $i$-location of the rank, and $\Delta r(i)$ the increment in the recall between positions $i$ and $i-1$.

However, metrics such as memory or search timing are also important factors in real-world problems. A system that achieves $AP = 1$ becomes useless in practice if it requires an infinite amount of time to generate its results. In the same way, a system will not be useful in practice if it requires a large amount of computational resources or memory to store dataset representations.

\footnote{An ideal retrieval system would return all relevant images at the top positions having $AP = 1$. However, precision tends to drop with the number of retrieved elements letting this scenario just as an ideal one.}
Chapter 3

Methodology

3.1 Baseline

The baseline of this project is Eva Mohedano’s PDH thesis. The participation in this project has been a perfect exercise for me to become familiar with the tools and datasets considered in this researching area. It also has provided me the sufficient level of autonomy to proceed with the development of the system as it is reported in the following sections, what represents the core of my work and contributions.

3.1.1 Datasets

Three datasets were used during the development of this project. They are Oxford Buildings\textsuperscript{17}, Paris Buildings\textsuperscript{18} and Instre dataset\textsuperscript{26}. The figure 3.1 includes examples of query images from the three datasets.

Oxford Buildings contains 5,063 still images, including 55 query images of 11 different buildings in Oxford. Paris Buildings contains 6,412 still images including query images of 12 different Paris landmarks. Both datasets have collected the images from Flickr\textsuperscript{1}.

Instre dataset\textsuperscript{26} contains 28,543 images collected from multiple sources, which include various image search engines (e.g. Baidu, Bing, Google, Picsearch, Altavista), social networks (e.g. weibo, facebook) and photo sharing communities (e.g. Flickr, nipic). Their images are distributed into 100 object classes in INSTRE-S1, 100 more into INSTRE-S2 and the last 50 object classes into INTRE-M (as it is shown in the fig 3.15).

Figure 3.1: Query examples from the three different datasets. Top: Oxford buildings(1st) and Paris buildings(2nd); Bottom: Instre dataset.

\textsuperscript{1}Flickr is a photo repository with a public API that allows to query its large database of photos and filter the obtained results by tags, textual data search and geographical location.
3.1.2 Connection React - node - python

Choosing the tools to develop a project is always a difficult task. However, taking into account the different needs of the project when designing its structure can make it easier.

When the design of this project was achieved, the conceptions about the use of a technology in the wide range of options were reduced to fit into the project. The fact that the front-end part had to be started from scratch, made the choice of any simpler tool easier for not having any restrictions. However, since the machine learning code had already been programmed in python, this affected the back-end structure by adding a restriction in the tool to be chosen, reducing the options to one in which the incorporation of python code it is possible.

Regarding the front-end part, ReactJS was the more accurate option to develop the UI of this project, as this system can be considered as a webtop which requires fast speed on images visualization amongst other features.

In addition, a back-end structure was required to connect the system and there were two main functionalities that had to be taken into account when choosing the programming tools:

- The server should allow exchanging data with the client, thus providing fast speed and consistent data structures.
- The server must be able to support all the mathematical operations to calculate the rankings and to store the user feedback to improve the system.

NodeJS provides this two features, in addition of being very suitable for interacting with ReactJS. However, it is not the most adequate tool when it is necessary to perform mathematical operations. For this end, Python is in fact a better option with the motivation of having already a large amount of the machine learning code implemented in Python for this project.

An initial plan to interconnect the system was carried out with the purpose of arranging the compromise between the two mention issues. The plan consisted in using Javascript (NodeJS) as the main language of the server. Thus, it was possible to provide a simple form of connection server-client, and then open a shell between nodejs and python when required to compute the different rankins or to store the feedback collected by the system. In this way, all the code would be executed in the nodejs server, incorporating also the machine learning code within it by calling a python process, which would run in the background to perform the heavy computational work and returning the results.

Unfortunately, after implementing this structure it came up that this method was slowing down a lot the system due the need of calling the python script every time that it was necessary to get a ranking. The only possible solution was to implement an internal function to make some validations first, in order to avoid unnecessary python computations. This function would first need to check if the file asked is already stored in the server. If it is, it would need to check whether it has might changed and if there is a need to replace it. If it was not stored in the server, then the function would need to call the python process anyway so it ended up being also a very bad option.

---

2 A web desktop or webtop is a desktop environment embedded in a web browser or similar client application
3 Other features which ReactJS provide are explained in section 2.1
Once this failed structuring of the system was discussed and studied, the idea of building two servers arose (one for the nodejs and the other for python). This method would allow bidirectional interaction between both servers quickly and concisely. After some research, the solution seemed to be ZERORPC\(^4\) which is a lightweight, reliable library that does not accentuate the language for distributed communication between the processes of the server side.

3.1.3 DATABASE and DATASETS

When designing the system, the question of whether to build a database or not also arose. However, this problem was not difficult to solve, since it did not make sense to build a database to store images or to store only the ranking files, which also do not last long without having to overwrite them.

The best solution for data management in this project seemed to be to store the images into the nodejs server and compute the rankings into txt files, which would be stored into the python server. In this way, when a ranking is asked by a query made from the UI, the nodejs server just needs to send the “query id” to the python server and it can compute the ranking and reply with a list of images id’s that the nodejs server will just take from itself.

3.1.4 WEB DESIGN

The website design consists of a homepage plus a full page for each dataset.

3.1.4.1 Home page

In the home page, as it can be observed in the figure 3.2 below, all the different options are provided. A more detailed explanation with pictures of this each option is given in the following points.

![Figure 3.2: Image to visualize the home page.](http://www.zerorpc.io/)
• **Image from url**

The first consists in introducing an URL into the search engine. In this option, the query will be done with the image depicted by the URL into the dataset, which must also be selected. An error handling is computed so that if a dataset is not chosen as in the right part of the figure 3.2, the query will not be computed.

![Image from url](https://example.com/image.png)

Figure 3.3: Image to visualize where to put the URL for the query.

• **Image from file**

The second option consists in dragging and dropping or upload the files. It is also required to select the dataset where the query wants to be searched into.

![Image from file](https://example.com/file.png)

Figure 3.4: Image to visualize where to upload the file for the query.
• **Image from dataset**

The last option is to select first the dataset where we want to search into, and then clicking on the image that we want to use as the query.

![OXFORD DATASET](image)

Figure 3.5: Image to visualize the Oxford dataset selection.

![PARIS DATASET](image)

Figure 3.6: Image to visualize the Paris dataset selection.

![INSTRE DATASET](image)

Figure 3.7: Image to visualize the instre dataset selection.
3.1.4.2 Main pages of each dataset

Even in the home page, as it has already been shown, the three datasets can be visualized, a main page for each one has been created. In the main page a 'qimlist' is rendered in order to provide some query examples for the user. An example of each page can be observed in the following points.

- Oxford

![Image to visualize the main Oxford buildings dataset page.](image)

Figure 3.8: Image to visualize the main Oxford buildings dataset page.

- Paris

![Image to visualize the main Paris buildings dataset page.](image)

Figure 3.9: Image to visualize the main Paris buildings dataset page.

Qimlist is a list with the main queries of each dataset. There is one list for each dataset.


- Instre

![Figure 3.10: Image to visualize the main Instre dataset page.](image)

### 3.1.4.3 DIFFERENT MODES

When a query is computed and the respective ranking is displayed, three different modes can be selected in the system. They are explained in detail below.

- **explorer mode**

  The explorer mode is thought to compute queries infinitely. When the first query is computed and the ranking of the similar images is received, whichever other image of the ones which appears below can be selected now to compute a new query.

![Figure 3.11: explorer mode](image)
• annotation mode

The annotation mode is created to give feedback to the system. As we can see in the figure 3.12 below, a red caution sign is displayed for a remind of the actual functionality. The fact that makes this mode so ‘dangerous’ is that the current annotations will affect to the system due a real feedback is collected by the annotations of the user by the UI. When the feedback is given and submitted, it is used to improve the system as it is explained in the section 4.2.

![Figure 3.12: annotation mode](image)

• mode Query expansion

The query expansion mode allows to the user to select multiple images for the query in order to provide a richer query representation for the image search engine. By combining the representations of the selected images (by averaging its representations), it is possible to provide a more informative query image representation that would potentially improve the quality of the returned rank list of images. However, this mode can also be used to experiment with queries by using extremely different images when creating the new query representation. This mode allow us then to observe how the output ranking varies. In the section 4.2 some experiments are done due to see when a query improves or deteriorate depending on the input queries selected.

![Figure 3.13: query expansion](image)
3.1.4.4 Data structure processing

The datasets used in this project have a hierarchical structure of files. All of the files have a name but, in some datasets, their identification also depends on the folder where they are allocated. For that reason, when treating the data into the code some issues came up in data treatment process.

The main problem was not having the possibility of generalize all the code, as it had to be different when using a name as an id than when the whole path was used. On one hand, if a name was used as an id for the query, there was no issue with the system. However, on the other hand, if a path was used as the id in the query, the browser recognize it as a part of the path instead of as a part of the query causing the system to crash. The parts of an URL can be seen in the figure 3.14 attached below.

![Figure 3.14: PARTS url](image)

Amongst the different datasets used in this project, there is just difference in structure between Instre dataset and the others two. In fact, Oxford and Paris building datasets have the same structure and they are also about the same size\(^6\). However, Instre dataset has its data structured in folders and it is of much bigger size\(^6\). Figure 3.15: Visualization of the Instre dataset structure. The subdivision in different folders (paths) and the number of images in each one can be seen in this figure.

![Figure 3.15: Visualization of the Instre dataset structure](image)

The difference in their structure and in their size have implied some issues when developing the system, not just because of the browser’s misunderstood with the id’s but also because of the speed when loading the main pages.

When a ranking is computed for an Oxford query, this is returning a json or txt with about 5000 images (6000 when the query is computed on the Paris dataset). Loading less than 10k images is fast in React Js. However, when computing a query in the Instre dataset, the images to load are more than 28k, which require a lot of time to load in memory.

Thus, when dealing with different datasets we have mainly these two issues: dealing with structure and dealing with size. In this project we propose some alternatives to deal the issues, described in section 4.3.

---

\(^6\) Oxford building dataset has 5062 images and Paris dataset has 6412 images. They are both images collected from Flickr by searching for particular filter depending if it is the Oxford or the Paris dataset.
Chapter 4

Results

This chapter presents the results obtained with the techniques presented in Chapter 3 to improve the baseline system described in Section 3.1.

4.1 Getting feedback of the UI

In this section, an experiment for getting the feedback of the UI from the users has been carried out. The experiment consisted in one person using the UI for 10 minutes and answer a short questionnaire. Ten volunteers participated in the experiment and answered the questionnaire regarding the usability of the proposed UI. Table 4.1 shows the obtained probabilities for each of the possible answers (results can also be seen in\[1\]). They have been graded following a probability distribution among the different options: Strongly Disagree (SD), Disagree (D), Neutral (N), Agree (A), and Strongly Agree (SA). They will all be graded between 0 and 1.

<table>
<thead>
<tr>
<th>Questions</th>
<th>SD</th>
<th>D</th>
<th>N</th>
<th>A</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 I think this UI is intuitive.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>2 In terms of understanding each function, I think that the existing explanations are good enough.</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>3 I have understood the purpose of the UI.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>4 I understand the different modes of operation.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>5 Regarding the ‘annotation mode’, I think that the UI is useful to improve the accuracy of a trained model.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>6 Regarding the ‘query expansion’ mode, I think that it is useful to experiment with similar and non-similar queries without affecting the accuracy of the system.</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>7 Regarding the ‘Explorer’ mode, I think that it is useful to have ‘clickable’ images in order to navigate through the system.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>0.4</td>
</tr>
<tr>
<td>8 Regarding the UI infrastructure, I think that it is robust and consistent.</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>9 I think that the UI is fully featured regarding its functionalities.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 4.1: Table showing the results of the UI feedback experiment.

\[1\]https://www.surveymonkey.com/results/SM-YYH37N5N8/
The question number 10, was to mention two liked features about the website and 2 things that the user might change. The most common items mentioned by users regarding the liked features were:

- Clean Design.
- The click through query feature (Explorer mode)
- The annotation tool.
- The option of uploading external images as queries.
- The fact of using query expansion mode to experiment without impacting the system.
- The option of searching ranks for my own images.
- The online training.

Regarding the most common things that the users would add were:

- Add Info icons with explanations.
- Include text-based queries.
- Include a "crop" mode on the query images to specify the region of interest.
- Include “on the fly” learning by using external images for the query expansion and/or the relevance feedback.
- Include a mechanism to measure the time expand per query image.
- Include the ability to search within all photos in the three datasets.
- In “query expansion mode”, make the submit button more visible/central.
- Include the option of visualizing the picture uploaded when using drag and drop tool.

The statistics conclude that the UI is intuitive, robust, consistent and built with a useful purpose. The aim of its different modes of operation is fully understood by the users although an extended explanation inquiring into the difference between the annotation and query expansion modes would further improve the understanding of these. Even that, the users agreed on the fact that each mode has its own utility thus it is very useful either giving straight feedback to the system as well as being able to experiment with hypothetical situations without affecting it.

The most liked feature is the fact that an explorer mode with “clickable” images have been developed and it has also contemplated as a great idea the fact of implementing hovering help icons in each functionality instead of giving a huge explanation in the home page. All the comments collected in the question 10 has been really useful to define possible future work and improvements. Some of the ideas, were already contemplated before the questionnaire as described in the section 4.3.
4.2 Quantitative and qualitative evaluations

4.2.1 Query expansion mode

This section shows how the query expansion mode affects the Average Precision (AP) and the mean average precision (mAP) of the system. Three experiments are done:

The first experiment consists of selecting a query where the automatic BLCF system performs poorly. Thus, a query image with 0.299 AP was selected. The first 6 relevant images of the original ranking (left side of Figure 4.2) were selected for Query Expansion (QE). QE is a procedure that takes the image representation associated to the manually selected images, averages them, $l_2$-normalize them, and issues a new query that is compared to the visual representations of the whole dataset, as in [16]. Figure 4.2 shows how the ranked list of the top 5 retrieved images is changed after QE. This experiment improves the query AP to nearly 0.6 points.

![Figure 4.1: Qualitative example of how the AP of an image improves, thus affecting the mAP of the system.](image1)

![Figure 4.2: Quantitative evaluation through a screenshot of the server with AP and mAP computed for the previous queries.](image2)

The second experiment consists of selecting an image that generates a ranked list with already a high AP score (the BLCF system generates automatically a "good" ranked list for the selected query). Then it is proceed as well as in the previous section, by selecting the first 6 relevant images in the original rank to perform query expansion. In the Figures 4.3 4.4 it can be seen that the AP slightly decreases. This is due the fact that when an image has a good accuracy, using...
the query expansion mode could just be adding noise instead of helping the system to improve its ranking.

Figure 4.3: Example of how the AP of an image does not improve when it is already high in the original rank provided by the BLCF system. Qualitative evaluation.

Figure 4.4: Quantitative evaluation through a screenshot of the server with AP and mAP computed for the previous queries. It can be seen how as the AP of the query decreases, it does it as well the mAP of the system.

The third and last experiment in this section consists of evaluating the results qualitatively when doing a multi-query with the best 10 images and as well as with the 10 worst ones. Thus, the first 10 images of the ranking were selected to compute the top-10 multi-queries example, and the last 10 images of the last page of the ranking were selected to compute the bottom 10 multi-queries example. The results can be observed in the figure 4.5 below.

Figure 4.5: Example of the best 10 queries vs the 10 worst ones.
4.2.2 Annotation mode - Training an SVM

In this section, an experiment to improve the ranking of a query is done. Thus, the user should give feedback to the system regarding the ranking of the query chosen. This is done by selecting the similar images displayed on the ranking as a successful images and, the ones which should not appear in the ranking as a failure ones. Once the feedback is received by the system, the annotations are used to train an SVM model with a linear kernel.

To train the LinearSVM model it is needed to have a matrix which stores all the visual descriptors related to all the positive and negative annotated images. With this matrix the model can be trained, and the visual representations of the whole dataset can be classified as relevant or nor relevant for a particular query. The confidence score of the LinearSVM model is used to re-rank the dataset images and generate a new ranked list. All the negative annotated images are filtered out so they are not displayed in the new ranking. Finally, this ranking is sent to the UI and it is updated in order to display it for the users.

Figure 4.6 shows an example of how the first images of the ranking change when sending feedback annotations. More dynamic examples can be seen in the demo of the project.²

![Figure 4.6: Example of ranking updated after some feedback annotations.](image)

4.2.3 Annotations vs query expansion

Additional experiments have been carried out to study the difference between the annotation mode and the query expansion mode. The purpose was to discover which of the two modes improved more a ranking for a specific query.

After some tests and experiments, it has been concluded that query expansion mode allows to improve the accuracy when the ranking is, in average, well represented. However, the annotation mode works much better in other situations such as having a completely poor ranking or having

²https://youtu.be/wNfn4ydCLng
specific failure images to discard in an extremely accurate ranking.

This is due to the fact that the annotation system is capable of removing these images selected as failure ones. This leads to an advantage of improving the system by annotating just the negative images without needing to annotate also the positives ones. Thus, in a scenario where the ranking computed is extremely accurate but with some specific failures, the best option is this mode for its ability to directly remove those cases. Besides, in a very poor ranking it is also more useful to annotate the positive and the negative feedback than just giving the positive one, as it is just done in the query expansion mode.

4.3 Goals achieved and future work

The list of goals defined for this project were:

1. Understanding how image retrieval is done for best integration of the project.
2. Design and build a User Interface with all the functionalities needed and merge it with a suitable server which interact with machine learning side.
3. Develop an API REST for the machine learning side so that it can be accessible by the server.
4. Build a demonstrator for showcasing the latest developments in content based information retrieval.
5. Build a platform for annotation of images to provide more extensive training data.
6. Build an interface for rapid qualitative evaluation of possible enhancements to the underlying technology.
7. Visualization of success and failure cases to assist in improving the search engine technology.

The results achieved in chapter 4 reflect much of the initial objectives set, that can be summarized into two main goal topics. On the one hand, objectives 2, 3 and 5, involve to develop a UI which is intuitive, robust and consistent. On the other hand, objectives 1, 4, 6 and 7 require to have a high level understanding of CBIR techniques, that in the case of this project required to get familiar with some deep learning techniques applied to vision problems, a basic machine learning tools such as LinearSVM, as well as knowledge of how to assess the qualitative performance of a retrieval system.

Regarding the first main goal topic and based on the questionnaire results obtained regarding the UI design, we can conclude that a robust and interactive UI has been developed. The main functionalities have been achieved and the system interacts properly between the two servers and also with the UI. The system is also capable of annotating the successful and failed images of a ranking and therefore training an SVM model which is capable of predicting the annotations of all the other images.

Regarding the second goal, it was required in order to develop the UI the study and understanding of the CBIR systems. Thus, a deep research of BLCF [15] was carried out so that new
concepts such as deep learning or Bag of Words could be learned. With those new concepts, we were finally able to understand how an image retrieval system works and, therefore, improve the performance of the original system.

Many ideas aroused while the project was being carried out. However, it was no possible to carry them all out. As a future work we plan to:

- Incorporate the possibility of selecting only part of the image when making the query (bounding box).
- Extend query expansion and annotation modes to images that are not part of the dataset, either images entered through the url or images uploaded from file.
- Unify the structure of the data so that all datasets can be treated by re-using the same code. An implementation for this improvement can be carried out in the case of Instre dataset. It consists of, instead of modifying the UI code to deal with the dataset structure, be able to create a file that maps the path of each image with an id so that it can end up treating as if it had the same structure as Oxford and Paris datasets.

Possible publications and open-source contributions have been discussed in order of improving the image retrieval engine system and also the UI. A demo of this project can be seen in the attached link.

[https://youtu.be/wNfn4ydCLng](https://youtu.be/wNfn4ydCLng)
Chapter 5

Budget

This project has been developed using the resources provided by Insight, a center of Data Analytics allocated in the Dublin City University (DCU), so there are not maintenance costs.

Thus, the main costs of this projects comes from the salary of the researches and the time spent in it. I consider the position of one of the PhD student supervising me with a wage/hour of a junior engineer, plus the position of two professors who were advising me with a wage/hour of a senior engineer. I will consider that the total duration of the project was of 22 weeks, as depicted in the Gantt diagram in Figure 1.2.

<table>
<thead>
<tr>
<th></th>
<th>Amount</th>
<th>Wage/hour</th>
<th>Dedication</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Junior engineer</td>
<td>1</td>
<td>8,00 €/h</td>
<td>4 h/week</td>
<td>704 €</td>
</tr>
<tr>
<td>Senior engineer</td>
<td>2</td>
<td>20,00 €/h</td>
<td>4 h/week</td>
<td>3,520 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>4,224 €</strong></td>
</tr>
</tbody>
</table>

Table 5.1: Budget of the project
Chapter 6

Conclusions

The main objective of this project was to develop a web user interface for a content-based image retrieval system developed in [16].

On one hand, the UI developed in this project allows to visualize the results generated by a state-of-the-art CBIR pipeline, providing a mechanism to interact with the system. It is also a tool to collect user’s annotations on the different visual results, facilitating the creation of machine learning models to improve the performance of the automatic CBIR engine.

On the other hand, the UI also allows to navigate through three different image retrieval benchmarks by means of the “explorer mode”, which allows queries images from inside and outside the dataset (uploading the file or by directly inserting an image URL).

Additionally, two different modes were designed to improve the quality of the ranked lists obtained by the CBIR system: Query expansion and annotation mode, that were shown to improve performance specially in the cases where the CBIR system achieved low AP.

Regarding the results obtained, it is made clear that the main purpose has been accomplished if we look at the user’s feedback questionnaire regarding consistence of the UI as well as if we do the same regarding the purpose achieved with all the different modes implementations.

Reaching this point though has not been a straight line. Many inconveniences arised when developing the UI, specifically when looking for a way to establish the connection between NodeJs and Python languages. However, it all finally worked out and the connection established now is not just robust but faithful.

As a future work, we plan to make the code open-source thus other software developers can contribute in the project. Additionally, the improvements of the section [4,3] will also been taken into account in order to improve the project.
Bibliography


