UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IMGPEDIA: A LARGE-SCALE KNOWLEDGE-BASE TO PERFORM VISUO-SEMANTIC QUERIES OVER WIKIMEDIA COMMONS IMAGES

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TITULO DE INGENIERO CIVIL EN COMPUTACIÓN

SEBASTIÁN CAMILO FERRADA ALIAGA

PROFESOR GUÍA:
BENJAMÍN BUSTOS CÁRDENAS
AIDAN HOGAN

MIEMBROS DE LA COMISIÓN:
JORGE PÉREZ ROJAS
JUAN MANUEL BARRIOS NUÑEZ
RENZO ANGLES ROJAS

Este trabajo ha sido parcialmente financiado por el Núcleo Milenio Centro de Investigación para la Web Semántica

SANTIAGO DE CHILE
2017

Resumen

Motivación. Los estándares de la Web Semántica son útiles para poder organizar la información de la Web de forma que los computadores puedan procesarla y *comprenderla* de mejor manera, pudiendo así los usuarios realizar búsquedas más sofisticadas y contar con un lenguaje más expresivo para realizarlas. Sin embargo, la mayoría de las bases de conocimiento disponibles utilizan solamente la información textual en desmedro del contenido multimedia, el cual ha aumentado enormemente los últimos años y ya es parte fundamental de la Web.

Objetivo. Dado lo anterior, nuestro objetivo en este trabajo es construir una base de conocimientos que nos permita combinar consultas semánticas con consultas sobre el contenido visual de las imágenes de la Web, que llamaremos IMGpedia. Concretamente, se trabajará utilizando las imágenes de Wikimedia Commons.

Contribución. Una vez completado, este trabajo pretende ser un puente entre el mundo del análisis multimedia y el de la Web de Datos. En este sentido, todas las rutinas de descripción de contenido visual serán publicadas como implementaciones de referencia en diferentes lenguajes de programación. Además, la base de conocimientos será una fuente de Datos Enlazados Abiertos de alta calidad, puesto que proveerá enlaces a diferentes fuentes de conocimiento para proveer contexto. Finalmente, estos datos podrán ser consultados a través del SPARQL endpoint público provisto para tal efecto. Esta base de conocimientos es pionera en combinar información del contenido visual de imágenes de la Web con datos semánticos extraídos de DBpedia.

Metodología. Se propone y desarrolla una metodología, dadas las 15 millones de imágenes extraídas de Wikimedia Commons, estas se analicen y procesen para formar una completa base de conocimiento. Primeramente, se calculan sus descriptores visuales; luego se computan sus vecinos más cercanos para establecer enlaces de similitud entre ellas; posteriormente, se propone una estrategia para enlazar las imágenes con recursos de DBpedia si es que las imágenes son utilizadas en el respectivo artículo de Wikipedia; y, finalmente, los datos se publican como un grafo RDF, listos para ser consultados a través de un terminal de consulta SPARQL.

Valor. El valor de este trabajo está en que es el inicio de un proyecto a largo plazo, el cual busca incluir el contenido multimedia dentro de la Web de Datos de una forma automatizada, sin necesidad de etiquetar los medios manualmente, sino que los hechos puedan ser extraídos de fuentes complementarias. De esta forma se hace que el hecho de realizar consultas sobre similitud visual e incluyendo filtros semánticos sea una tarea cada vez más común.

Abstract

Motivation. The Semantic Web standards are used to organize information on the Web in a machine-friendly manner, such that users can ask more complex queries than on the traditional Web. However, most current Semantic Web knowledge-bases use only textual information and disregard multimedia content, which has been growing immensely through the years and has become an elementary component of the Web.

Objective. Given this scenario, the objective of this work is to build a knowledge-base that allows people to combine regular semantic queries with queries involving similarity relationships between images on the Web. This knowledge-base is called IMGpedia, and is based on images from the Wikimedia Commons image dataset.

Contribution. Once finished, this work aims to bridge Multimedia Content Analysis and the Web of Data. To achieve this, all the algorithms implemented to compute the visual descriptors of IMGpedia are being released as reference implementations for anybody to use. Moreover, the knowledge-base is a Linked Open Data source of high quality according to W3C conventions, since it provides links to other data sources such as DBpedia in order to provide context. Finally, this data can be queried through a public SPARQL endpoint provided by us. This is the first knowledge-base that combines visual description of images of the Web with semantic facts extracted from DBpedia.

Methodology. The methodology proposed and developed in this work defines the process of analysis of 15 million images from Wikimedia Commons in order to build the knowledge-base. First, the visual descriptors must be calculated; later, we propose an efficient strategy to compute similarity links between them; afterwards, we define a method for obtaining relations between the images and DBpedia resources of the Wikipedia articles that use them; and finally, the data is published as an RDF graph ready to be queried through the SPARQL endpoint service mounted for IMGpedia.

Value. The IMGpedia knowledge-base is the start of a long term project, which aims to include multimedia content within the Web of Data in an automatized manner, without need for manual labelling and tagging of media. Semantic facts can be extracted from complementary sources of information, thus enabling queries about visual similarity using semantic filters to be executed for the first time on Web.

Para Marisol Aliaga*, por nunca dejar de creer en mi.*

Agradecimientos

Voy a comenzar diciendo que esta página va a quedar corta para agradecer a todas las personas que han contribuido para que este trabajo sea posible. A lo largo de la carrera he contado con un sinnúmero de manos amigas, tanto para cosas académicas y laborales, como para temas personales y sentimentales. Es por eso que estoy infinitamente agradecido de los consejos, el apoyo y por sobretodo del cariño.

A mi familia, por apoyarme e incentivarme en esta aventura de la ciencia, por hacerme notar siempre lo capaz que soy, por escucharme y por quererme tanto como lo hacen. Son fundamentales para mi y sé que nunca voy a poder retrubirles lo mucho que han hecho por mi. Los amo.

A todos los que me han regalado su amistad, pues sin ella no podría haber crecido ni como persona ni como profesional. A todos esos que han atestiguado todo el cambio que he vivido estos 7 años y a los que me han ayudado a seguir en pie cuando la vida se ponía más difícil. En especial quiero nombrar a mis amigos del colegio, mis *manzanas podridas*, esos por los que nadie daba un peso y aquí estamos. También a mis amigos de sección, a todos quienes pasaron por *el almíbar* y me ayudaron a sobrevivir a punta de risas y carretes los primeros años de universidad y que hasta el día de hoy siguen siendo una fuente importante de cariño. Finalmente, quiero agradecer a mis amigos del *culto* y a los *reviejones*, que son una fuente inagotable de ánimo, buena onda, cerveza y conocimiento computín.

Gracias a mis profesores guías, Benjamín y Aidan, que siempre resolvían mis dudas y aportaban con ideas de lo más brillantes. Creo que difícilmente podría haber encontrado mejores ejemplos de investigadores y personas, sus carreras me inspiran mucho. Gracias también a todos los profesores que me han elegido como Profesor Auxiliar o Ayudante, pues me han permitido aprender mucho, tanto de la materia que se ve como de mi vocación por conocer, investigar y enseñar. Gracias también a los funcionarios del DCC, quienes siempre estuvieron dispuestos a ayudarme amablemente, mención especial a Sergio Aguilera, por sacarme de tantos apuros durante la tesis.

Este trabajo no sería posible sin el apoyo económico del Núcleo Milenio Centro de Investigación de la Web Semántica, gracias a quienes me veo inserto en una comunidad científica de lo más agradable y gracias a quienes pude ir a Panamá a presentar mi primer paper. Gracias a Camila Faúndez por el trabajo descrito en la Sección 5.3. Gracias a Felipe Quintanilla por la Figura 2.2, a Patricio Isbej por la Figura 2.6, a Juan Manuel Barrios por la Figura 2.7 y a Benjamín Bustos por la Figura 2.8. Gracias a Sebastián Cisneros por el logo de IMGpedia. Gracias a Daniel Hernandez por ayudarme a levantar el servidor Virtuoso.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1. Motivation

WIKIPEDIA is a global effort to maintain human-readable encyclopedic content. At the time of writing, it contains more than 38 million articles in more than 200 language versions. However, the information of WIKIPEDIA is not machine-readable, making it hard to process automatically. For this reason, the DBPEDIA project [30] automatically extracts and parses the structured information from the articles of WIKIPEDIA and presents it as an RDF data, i.e. data and metadata that is machine-understandable. WIKIDATA [47] is another initiative, organized by the Wikimedia Foundation, that allows people to collaboratively create machine-readable encyclopedic entries.

In terms of multimedia content, WIKIMEDIA COMMONS[1] is a large-scale dataset that contains about 30 million freely usable media files (image, audio and video) that are used within WIKIPEDIA articles. It also contains meta-data about each file, such as its author, licensing and the articles where the file is used. Using this information, DBPEDIA COMMONS [46] automatically extracts the meta-data of the media files of WIKIMEDIA COMMONS pages and presents it as a knowledge-base. However, DBPEDIA COMMONS does not contain nor describe any information about the multimedia content itself, only its meta-data.

Initiatives such as DBPEDIA, WIKIDATA and DBPEDIA COMMONS are all part of what is called the *Web of Data*, whose vision is to enhance the machine readability of the content of the Web. The core premise of the Web of Data is to publish data in a structured format on the Web in such a manner that data provide links to other data, potentially residing at remote locations elsewhere on the Web.

Now the question is, can multimedia be described in a way that is compatible with the Web of Data? And, what will we achieve by doing so? Well, combining the meta-data from DBPEDIA COMMONS with the ontology of DBPEDIA and enriching it with visual descriptors and visual similarity links will allow us to build a knowledge-base, which we call IMGPE-

[1]http://commons.wikimedia.org

dia [11]. Using this, we can combine classic semantic queries with visual similarity queries, which we call *visuo-semantic* queries that can be performed over IMGpedia, e.g., *retrieve images similar to the images of 16th century paintings currently shown in the Louvre* or *given a picture of the Cusco Cathedral, retrieve the top k most visually similar cathedrals in Europe*.

In this work, we want to address the challenges that arise from building such a knowledge-base. Our aim is to include multimedia files as first-class citizens into the Web of Data and demonstrate that both structured data and multimedia should be queried in a holistic manner.

1.2. Objectives

1.2.1. General Objective

The main objective of this thesis is to provide a means to combine semantic queries with visual-similarity queries.

1.2.2. Specific Objectives

The specific objective of this work is to build the knowledge-graph of IMGpedia and to publish it as Linked Open Data, which involves the following sub-objectives:

1. Provide reference implementations for the visual descriptors used in IMGpedia.
2. Generate the IMGpedia dataset from the meta-data and visual descriptors calculated for the Wikimedia Commons images.
3. Find a suitable way to compute static similarity relations for pairs of images.
4. Represent the dataset as RDF and publish it as Linked Open Data
5. Provide a query mechanism where meta-data, image similarity and semantic facts can be combined.

1.3. Methodology

For the generation and publication of the knowledge-base we perform the following steps:

1. We develop reference implementations for four different visual descriptors (see Chapter 5) in three different programming languages (Java, C++ and Python) and demonstrate that these implementations are equivalent.

2. We download the Wikimedia Commons image dataset and store it locally.
3. We calculate the visual descriptors for all the images.
4. We compute the similarity between the images via k nearest neighbors search.
5. We define the relations needed to perform image similarity search in SPARQL and then write the RDF triples with the data.
6. We extract the relations between the images and the articles where they are used.
7. We upload the knowledge-base to the Web as Linked Open Data, including an endpoint to query it.

1.4. Structure of this Work

This work is organized as follows:

1. In Chapter 2, we describe the theoretical framework of this thesis, including different types of visual descriptors, content-based image retrieval, the Semantic Web and how these areas can be united.
2. Next, in Chapter 3, we discuss the goals of this work, the data we use and the technical challenges we face.
3. In Chapter 4, we talk about the visual descriptor extraction process used to build the dataset of IMGpedia and how we address the similarity search problem using approximated techniques.
4. In Chapter 5, we describe the data model we use and how the IMGpedia dataset is published as a knowledge-base and describe the query endpoint is created.
5. In Chapter 6, we show some use-cases and queries that can be asked to IMGpedia.
6. Finally, in Chapter 7, we summarize the conclusions of this work and discuss some of the future research lines for the IMGpedia project.

Chapter 2

Theoretical Framework

This research lies between the Image Processing and Analysis, Information Retrieval, and the Semantic Web areas. We want to process a large amount of images and to construct a similarity graph in order to enrich that graph with semantic data and perform *visuo-semantic* queries.

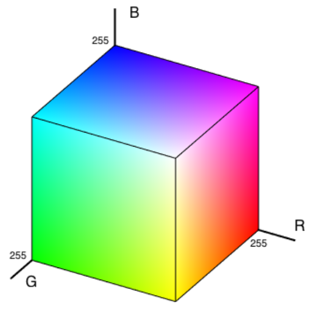
This chapter is organized as follows: first, two sections dedicated to Image Processing fundamentals (Sections 2.1 and 2.2); later, a section regarding Content-Based Similarity Search and Multidimensional Indexing (Section 2.3); and finally, a section describing the Semantic Web principles and foundations, and works on multimedia information within the Semantic Web area. (Section 2.4).

2.1. Image Analysis

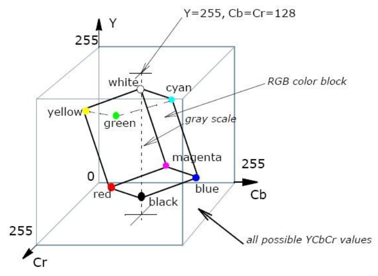
An image is a matrix with a given width and height. Each element of this matrix is called a *picture element* or pixel and describes a punctual portion of the color information of the image [20]. Several color spaces can be used to describe an image, the most common being the RGB space. RGB colors are tuples of three components, each one indicating the intensity value for red, green and blue and every color is the combination of those three components. An illustration of this color space can be found on Figure 2.1a. There are other relevant color spaces: YUV space was used for television broadcasting because it allowed to support both black and white, and color television and now is the standard color encoding for digital television; Y stands for gray intensity (or brightness) and U, V are for chrominances of two colors (red and blue most typically). A slice of the YUV space can be found in Figure 2.1b[1].

Since images are matrices, several operations can be performed on them. Addition, substraction and scalar multiplication, for example, behave the same way they do with typical matrices, but in this case, they represent changes in brightness and contrast of the images. Another interesting operation that can be done on images is convolution: given a kernel ma-

[1]Image taken from Intel tutorial on `https://software.intel.com/en-us/node/503873`



(a) RGB color space.



(b) YUV color space.

Figure 2.1: Color spaces.

trix (the size of the kernel is arbitrary, but 2×2 or 3×3 kernels are often used), each pixel of the convolved image is the result of the summation of all the element-wise multiplications of the neighborhood of the original image with the kernel. In Equation 2.1 we see how each pixel $I(x, y)$ of an image with size $M \times N$ is operated against a kernel k of size $m \times n$, where $a = (m-1)/2$ and $b = (n-1)/2$. Image convolution is useful for sharpening, blurring and edge detection [20].

$$I(x,y) * k = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k(s,t) \cdot I(x+s,\ y+t) \tag{2.1}$$

There are many libraries that help to efficiently process images. OpenCV[2] (stands for Open Source Computer Vision) provides optimized implementations of thousands of algorithms for image and video processing and computer vision tasks. It was specially designed for computational efficiency and has a focus on real-time applications. It was written in optimized C/C++, provides bindings for various programming languages and takes advantage of multithreaded processing.

2.2. Visual Descriptors

As the volume of multimedia content grows through the years, different image content description techniques have been developed in order to efficiently and accurately perform tasks such as searching, identifying and filtering multimedia content. A major effort in this area was the development of the MPEG-7 [12] standard which defines the syntax and semantics for various description tools focused on audiovisual information description at different granularity levels. This standard also classifies the different multimedia descriptors according to the features they are based on, such as *color*, *textures*, *shapes* and others.

In this work we will use four different visual descriptors, three of them are classic standard descriptors and the other one has a more modern basis, not relying on certain features of the image, but rather on its content classification. These descriptors will allow us to perform the

[2] `http://opencv.org/`

visual similarity queries that are to be combined with the semantic queries over the dataset of IMGpedia [17]. All of them will be reviewed next.

2.2.1. Gray Histogram Descriptor

The Gray Histogram Descriptor captures the luminosity information of an image. The image is frequently divided into many blocks because histograms lose the spacial distribution of the information and the blocking helps to preserve a part of it. As a pre-processing step, the image must be converted from RGB color to gray scale using the following equation:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{2.2}$$

where Y is the gray intensity for a pixel and R, B and G are the red, blue and green intensities of the pixel, respectively. The weights for the colors are empirically determined by OpenCV in order to preserve the perception of luminosity of the human eye [45].

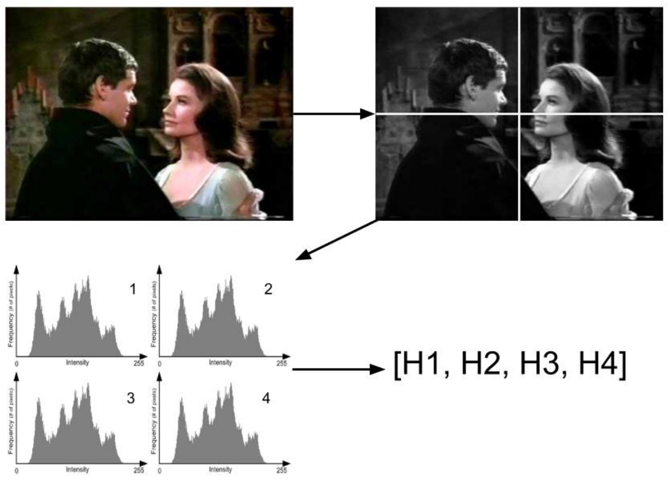


Figure 2.2: GHD calculation process. Picture taken from The Terror motion picture (1963)

Once the image is prepared, a fixed number of blocks is taken and, per each block, the histogram of gray intensities is calculated. Typically, the gray intensities take values between 0 and 255, so the histograms usually have 128, 64 or 32 bins. Finally, all the histograms are concatenated into a single vector. In Figure 2.2 we show the pipeline of the descriptor calculation.

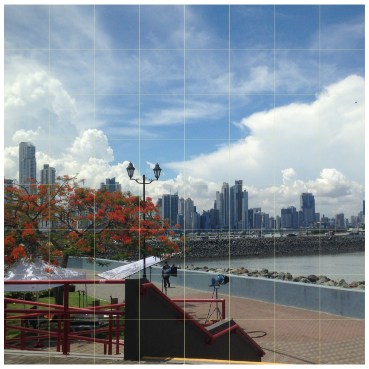
2.2.2. Color Layout Descriptor

The Color Layout Descriptor is based on the color distribution of the image and was proposed as part of the MPEG-7 standard for color description [37]. Its computation requires three steps: the generation of an icon of the image, the computation of the *discrete cosine transformation* and zigzag scanning of the matrices.

To generate the icon, the image is divided into 64 blocks of the same size. For each block, the average color is computed. Then, an 8×8 icon with those colors is generated. An example of this step is shown in Figure 2.3. Finally, the image has to be converted from RBG color space to YCbCr space: Y stands for the luminance of the image (gray intensity) and Cb, Cr for the blue and red *chrominance* of the image. The conversion is performed using the following equations, where the weights for the different components are determined empirically in order to preserve the human perception of colors, taken from OpenCV documentation [45]:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$
$$Cr = (R - Y) \cdot 0.713 + \delta$$
$$Cb = (B - Y) \cdot 0.564 + \delta$$

$$\delta = \begin{cases} 128 & \text{for 8-bit images} \\ 32762 & \text{for 16-bit images} \\ 0.5 & \text{for floating point images} \end{cases}$$


(a) The image is divided into 8×8 blocks. Photograpy by Sebastián Ferrada

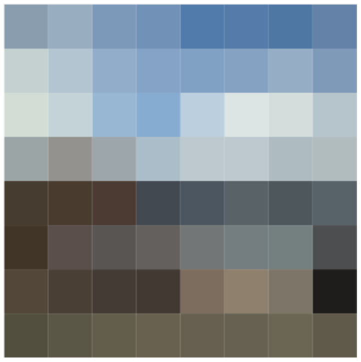

(b) The average color is computed for each block

Figure 2.3: Icon generation example

After the icon is generated, the discrete cosine transformation (DCT) is computed for each color channel (Y, Cb and Cr). DCT expresses the data in terms of a sum of cosine functions oscillating at different frequencies [2]. These frequencies in multimedia files are related to color in images and video and to the key in music. The transformation matrix B for an image I of size $N \times M$ is defined in Equation 2.3, taken from OpenCV documentation [45]. The DCT transform is widely used for lossy compression of media files, such as MP3 for audio files and JPEG for images, because small high-frequency components can be discarded. When the three DCT matrices are computed, we scan them in zigzag order to obtain 3 vectors.

This scanning method is used in order to group the low-frequency coefficients of the matrix. Finally, the scanning vectors are concatenated and presented as the descriptor.

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad 0 \leq p < M,\ 0 \leq q < N \tag{2.3}$$

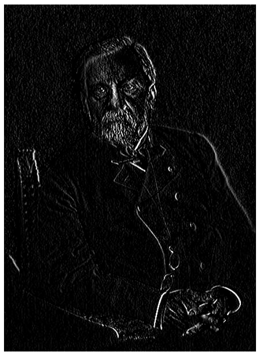
$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & \text{if } p = 0 \\ \sqrt{\frac{2}{M}}, & \text{else} \end{cases} \qquad \alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & \text{if } q = 0 \\ \sqrt{\frac{2}{N}}, & \text{else} \end{cases}$$

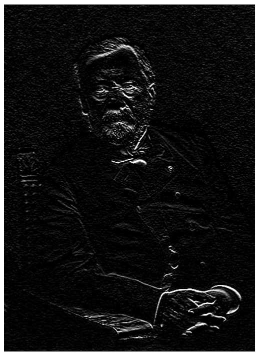
2.2.3. Histogram of Oriented Gradients

The Histogram of Oriented Gradients aims to capture the distribution of the orientations of the edges present in an image by detecting regions where color or luminosity contrast sharply. To compute this descriptor, the image must be converted to gray scale first, using Equation 2.2. Then, the gradient of the image must be calculated, via convolution with derivative kernels. Sobel kernels [42] are the most widely used for this task. After this step we will have two images, one with the x-axis gradient and another with the y-axis (Figures 2.4b and 2.4c respectively), where the full gradient is the sum of both. At this point, a threshold must be applied to the magnitude of the gradient (see Figure. 2.4d). For those pixels that exceed the threshold, the orientation of the gradient is computed according to formula 2.4. Finally, a histogram of all the orientations is computed and used as the descriptor. The histogram contains 18 bins, considering gradient angles in the range $[-180^o, 180^o]$. In Figure 2.4 we can see the sub-products of the process: the angle of the gradient vector, and both horizontal and vertical gradients of the image.

$$\theta = \arctan \frac{G_y}{G_x}, \quad G_y = \frac{\partial I}{\partial y}, \quad G_x = \frac{\partial I}{\partial x} \tag{2.4}$$

(a) Original gray scale image


(b) Magnitude of x-axis gradient


(c) Magnitude of y-axis gradient

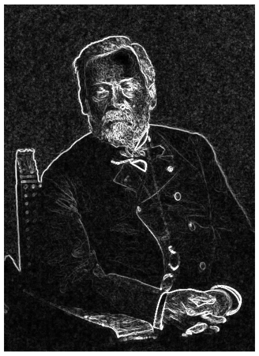

(d) Magnitude of the gradient

Figure 2.4: HOG computation steps. Portrait of Louis Pasteur taken from the Dibner Library of the History of Science and Technology.

2.2.4. DeCAF7

DeCAF7 uses the input for the image classification problem as a feature vector to describe the image, claiming that at that stage, the vector contains relevant semantic information about the image content. This descriptor [16] takes the image as-is and uses a convolutional neural network implemented by Caffe [27] pre-trained with the Imagenet dataset. The implementation [36] resizes the image to 256×256 pixels and takes ten overlapping patches of 244×244 pixels. Then it feeds the network with the patches and extracts the second-last self-convolutional layer of the network (which is a vector with 4096 dimensions). Finally, the descriptor is the average of each of the layers extracted.

2.3. Content-based Image Retrieval

2.3.1. Similarity Search

Similarity search problems rely on the definition of a similarity function δ, which must be good enough to capture the notion of visual similarity between two images. But what is a good similarity function? Generally, distance functions are used as similarity measures, so two points are more similar, the closer they are (the lesser the distance). Moreover, metric distances are the most used, in order to take advantage of their properties for building indexes. A function $f : \mathbb{R}^n \to \mathbb{R}$ is a metric if it is non-negative, reflexive, symmetric and satisfies the triangle inequality (Equation 2.5). Euclidean distance is an example of a metric function.

$$\forall\, x, y, z \in \mathbb{R}^n \quad f(x, y) \leq f(x, z) + f(z, y) \tag{2.5}$$

There are, of course, other distances that can be used, for example the *earth mover's distance* corresponding to the cost of efficiently converting one distribution of features (histogram) to another one [41]. Others include the cosine distance, which is the cosine of the angle between two vectors; the Jaccard distance, which is used to compare sets; and the Levenshtein distance, which is used to measure text similarity; and so on.

As we have seen in the previous section, images can be characterized as a high-dimensional vector that describes different visual features, such as color or texture; as such, an image can be represented as a point in space. Ultimately, the problem of finding similar images is reduced to perform a vector-similarity query over the space defined for the vectors, that is, finding points close enough to each other. There are two main problems to be solved for similarity search in an n-dimensional space $\mathbb{U}$. Let $\mathbb{D} \subseteq \mathbb{U}$ be the set of visual descriptors of the dataset and $\delta : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ a distance function:

- **Range queries.** Given a query vector q and a threshold r, obtain every vector $u \in \mathbb{D}$ so that $\delta(u, q) \leq r$. Also a query rectangle of vertices (v_1, v_2) can be given along with a request for all vectors of the dataset within the rectangle.

- **Nearest Neighbors.** Given a query vector q find the k vectors $u_{\mathrm{i}} \in \mathbb{D}$ so that $\forall v \in \mathbb{D} - \{u_{\mathrm{i}}\}_{\mathrm{i}=1}^{\mathrm{i}=k} \quad \delta(v, q) \geq \delta(u_{\mathrm{i}}, q)$, $\mathrm{i} = 1, \ldots, k$.

2.3.2. Indexing

The initial approach for performing similarity search is a brute force algorithm, i.e., to compare the query vector against all the others in the dataset, obtaining the closest ones. This is, of course, very expensive in time, growing linearly with the size of the dataset. To handle this problem, a lot of indexing techniques in multidimensional and metric spaces have been designed in order to reduce the amount of comparisons that must be made to find similar vectors.

R-tree [21] is a common example of multidimensional index. It groups nearby objects in their Minimum Bounding Rectangles, then those MBRs are grouped in next-level MBRs and so on until all the vectors are contained in a fixed number of MBRs which all form part of the root of the tree. An illustration of an R-tree can be found on Figure 2.5. When searching, we go through the closest MBR in each level discarding all others, therefore decreasing the amount of total comparisons between vectors. The worst case scenario for R-trees is when many or all of the MBRs intersect the range of the query, forcing the algorithm to perform $O(n)$ comparisons.

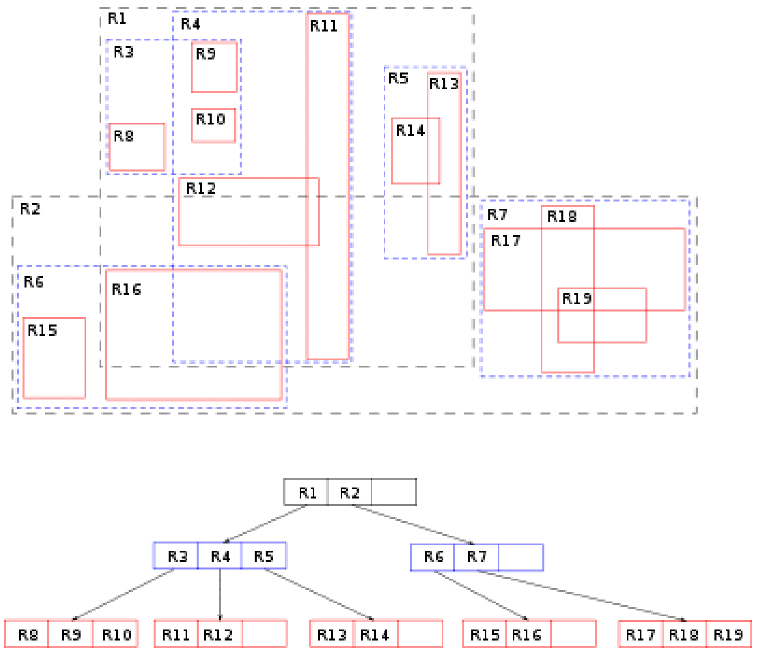
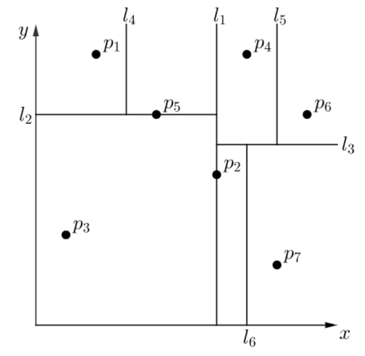


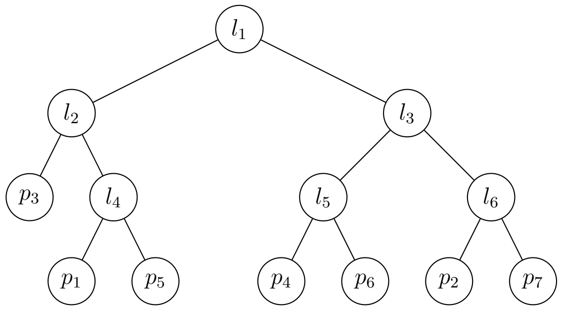
Figure 2.5: R-tree example. In the upper part are the rectangles of the dataset in orange; in dashed lines are the coresponding MBRs. In the lower part is the tree representation of the rectangles. Diagram adapted by Stefan de Konink, from Guttman [21].

K-D trees [7] are binary trees used as multidimensional indexes. Every internal node in a K-D tree sets a hyperplane dividing the space in two parts. Each part of the space is consequently divided until it contains only one vector. An example of how the space is divided can be seen in Figure 2.6. This is useful for solving the nearest neighbors problem using $O(\log n)$ comparisons on average; however this index has a linear worst case.

Nevertheless, these indexing techniques do not scale with high-dimensional spaces due to the *curse of dimensionality* where the vector distribution becomes more and more sparse.



(a) Dataset example and space division for $k-$d tree



(b) Resulting binary tree

Figure 2.6: K-D tree space division and tree representation

There are studies that have shown that these indexing methods fail in high dimensional spaces: Weber et al. [48] conclude that any such indexing or partitioning method has a complexity that tends towards $O(n)$ as dimensionality increases; thus these indexes are not useful to reduce the number of comparisons needed to retrieve the answers of a similarity search in high dimensional spaces.

There are other different approaches that make use the metric properties of the distance function in order to decrease the number of comparisons needed to perform similarity search. As we can see in Chavez et al. [13], pivot-based indexing is widely used for similarity search in metric spaces. A *pivot* is a marked element from the dataset $\mathbb{D} \subseteq \mathbb{U}$ which is used to discard elements in the searching process. The distance of all vectors to the pivot are precomputed, so its cost is amortized in the analysis. Given a range query with a query point q, a similarity threshold r, a pivot $p \in \mathbb{D}$ and a metric distance function δ the following conditions hold:

$$\forall\, u \in \mathbb{D} \quad \delta(p,u) \leq \delta(p,q) + \delta(q,u),\ \delta(p,q) \leq \delta(p,u) + \delta(u,q)$$

Therefore we can state a lower and upper bound for the distance between q and $u \in \mathbb{D}$ in order to discard it or not. Equation 2.6 states the boundaries. In Figure 2.7 we can see these boundaries graphically.

$$|\delta(p,q) - \delta(p,u)| \leq \delta(q,u) \leq \delta(p,q) + \delta(p,u) \tag{2.6}$$

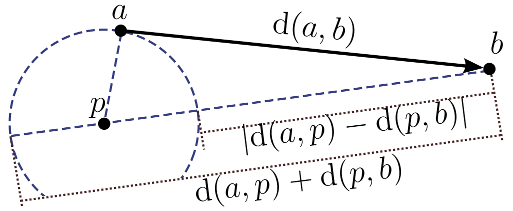

a
$\mathrm{d}(a,b)$
b
p
$|\mathrm{d}(a,p) - \mathrm{d}(p,b)|$
$\mathrm{d}(a,p) + \mathrm{d}(p,b)$

Figure 2.7: Boundaries for distance using pivots

This leads to an exclusion criteria (see Equation 2.7), so every vector u that holds the criteria is discarded and not compared with the query point. In Figure 2.8 we can see that

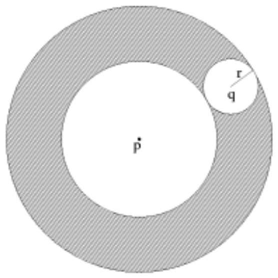


Figure 2.8: Exclusion area in a pivot-based search.

all points in the gray area cannot be excluded from comparison.

$$|\delta(p,q) - \delta(p,u)| > r \tag{2.7}$$

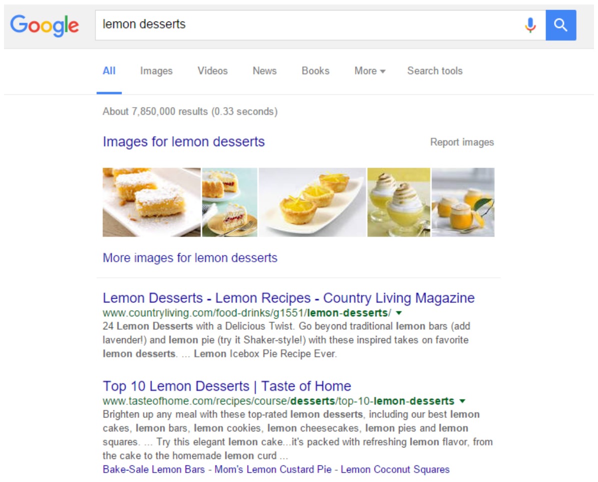
In practice, many pivots are used. First, the distances of the pivots against all vectors in the dataset must be calculated. To solve a range query with a query vector q and a distance threshold r, the distance between q and all pivots is calculated. Then, all vectors that are in the exclusion area are disregarded and all other vectors are compared against q. The choice of the pivots is crucial for the performance of the index: if they are too close, the number of excluded vectors decreases; if they are too far, the exclusion areas are mostly outside of the dataset and become useless. An example of an application for this indexing technique is to perform content-based video copy detection [5], i.e., given a dataset of videos find those on which a short clip appears.

Paredes et al. [38] propose a pivot-based algorithm that solves the k-NN problem (using the string space and edit distance) in $O(n^{1.85..2.10}k^{0.12..0.29})$ time cost and $O(n^{1.26..1.54}k^{0.20..0.48})$ distance computations by making exhaustive use of the properties of metric functions.

2.4. The Semantic Web and the Multimedia Content

2.4.1. The Web of Data

The current Web is strongly document-centric and most of its content makes sense only to humans, not machines. That becomes a problem when attempting to ask more "sophisticated" queries to those documents, requiring some type of reasoning over the semantics of the data instead of matching the query string to the documents. In Figure 2.9 we can see that it is very easy to find relevant web documents for the query "desserts with lemon", but on the other hand "desserts *without* lemon" is poorly addressed.



lemon desserts

About 7,850,000 results (0.33 seconds)

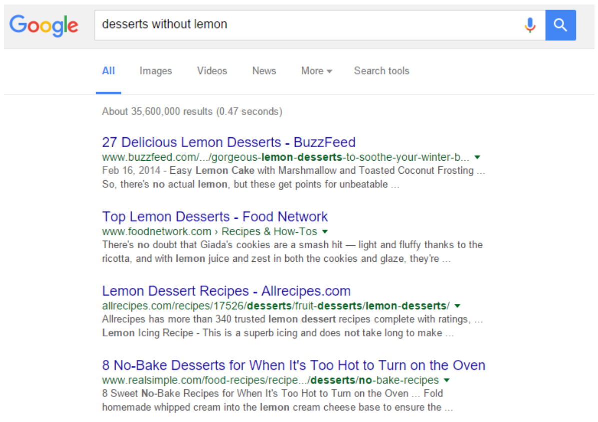
Images for lemon desserts — Report images

More images for lemon desserts

Lemon Desserts - Lemon Recipes - Country Living Magazine
www.countryliving.com/food-drinks/g1551/**lemon-desserts**/
24 **Lemon Desserts** with a Delicious Twist. Go beyond traditional **lemon** bars (add lavender!) and **lemon** pie (try it Shaker-style!) with these inspired takes on favorite **lemon desserts**. ... **Lemon** Icebox Pie Recipe Ever.

Top 10 Lemon Desserts | Taste of Home
www.tasteofhome.com/recipes/course/**desserts**/top-10-**lemon-desserts**
Brighten up any meal with these top-rated **lemon desserts**, including our best **lemon** cakes, **lemon** bars, **lemon** cookies, **lemon** cheesecakes, **lemon** pies and **lemon** squares. ... Try this elegant **lemon** cake...it's packed with refreshing **lemon** flavor, from the cake to the homemade **lemon** curd ...
Bake-Sale Lemon Bars - Mom's Lemon Custard Pie - Lemon Coconut Squares

(a) Search results for "desserts with lemon".



desserts without lemon

About 35,600,000 results (0.47 seconds)

27 Delicious Lemon Desserts - BuzzFeed
www.buzzfeed.com/.../gorgeous-**lemon-desserts**-to-soothe-your-winter-b...
Feb 16, 2014 - Easy **Lemon Cake** with Marshmallow and Toasted Coconut Frosting ... So, there's **no** actual **lemon**, but these get points for unbeatable ...

Top Lemon Desserts - Food Network
www.foodnetwork.com › Recipes & How-Tos
There's **no** doubt that Giada's cookies are a smash hit — light and fluffy thanks to the ricotta, and with **lemon** juice and zest in both the cookies and glaze, they're ...

Lemon Dessert Recipes - Allrecipes.com
allrecipes.com/recipes/17526/**desserts**/fruit-**desserts**/**lemon-desserts**/
Allrecipes has more than 340 trusted **lemon dessert** recipes complete with ratings, ... **Lemon** Icing Recipe - This is a superb icing and does **not** take long to make ...

8 No-Bake Desserts for When It's Too Hot to Turn on the Oven
www.realsimple.com/food-recipes/recipe.../**desserts**/**no**-bake-recipes
8 Sweet **No**-Bake Recipes for When It's Too Hot to Turn on the Oven ... Fold homemade whipped cream into the **lemon** cream cheese base to ensure the ...

(b) Results for "desserts without lemon".

Figure 2.9: Example of bad search results on the Web.

To address the problem of above and other deeper problems (as seen in the seminal paper of Berners-Lee et al. [9] about their vision for the Semantic Web), the Semantic Web standards are being developed. According to Hogan [24], the Semantic Web can be conceptualized as an extension of the current Web as it enables the creation, sharing and intelligent re-use of machine-readable content on the Web. Moreover in Figure 2.10 we can see the pieces needed to build the ideal Semantic Web and in the following sections we will describe them briefly. The layers shown with dashed lines are not yet reached nor well developed.

Characters are needed to handle textual and numeric information contained in the plain text documents of the Web, so the Semantic Web relies on the Unicode charset. **Identifiers** are needed to refer to and describe *things* of the world as unique instances, whether they are real entities or fictional ones. The natural choice for this are URIs, which are currently being used on the Web to identify resources.

In the following paragraphs we will review the main technologies involved in the fundamental layers of the Semantic Web shown in Figure 2.10. The Resource Description Framework and its languages will allow us to cover the **Syntax** and **Data Model** layers. RDF Schema and OWL will cover the **Schema and Ontologies** layer and, finally, we will review SPARQL to cover the **Querying** layer.

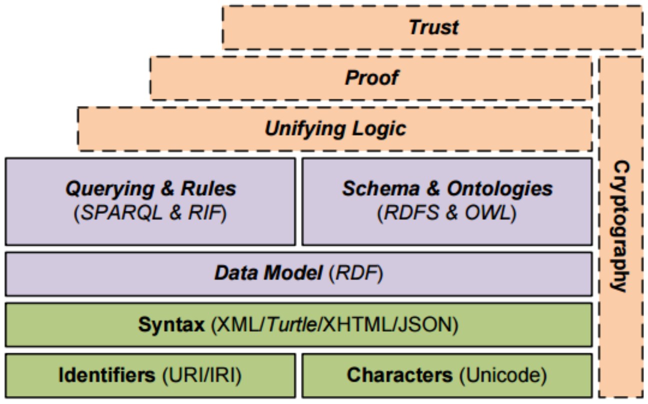

Trust
Proof
Unifying Logic
Querying & Rules (SPARQL & RIF)
Schema & Ontologies (RDFS & OWL)
Data Model (RDF)
Syntax (XML/Turtle/XHTML/JSON)
Identifiers (URI/IRI)
Characters (Unicode)
Cryptography

Figure 2.10: Semantic Web Stack. Source: *Linked Data Management: Principles and Techniques* [24].

Resource Description Framework (RDF)

The RDF standard [32] defines the way data is structured in the Semantic Web for the sake of sharing such information between independent sources. RDF consists basically in Terms, Triples and Graphs. Additionally, we will also need a syntax for our data and in this work we will use Turtle (Terse RDF Triple Language) [6]. Examples of this language will be reviewed throughout the next paragraphs.

Terms refer to atomic resources in the Web which can be either a URI, a literal or a blank node. URIs are global identifiers for resources among the Web, such as `http://dbpedia.org/resource/Chile` to identify the country of Chile. For abreviating the URIs, Turtle language provides prefix definition, so if we state that prefix `db:` relates to `http://dbpedia.org/resource/`, the URI for Chile will be simply `db:Chile`. Literals are primitive values, such as `integer` or `string`; datatypes are often borrowed from XML Schema and used as a postfix: `“9”^^xsd:integer` is the representation of the number 9. Finally, blank nodes are entities that can confirm the existence of a resource without the need of naming it; for example, we know that Chile has a capital city, but if we don't know which city it is, we can use a blank node as a placeholder to denote its existence.

Triples are the main structure for making statements about the data. As one can expect, a triple is just a 3-tuple of RDF terms. The elements of a triple are called the *subject*, the *predicate* and the *object* respectively. The only restrictions to be a valid triple are that only objects can be literals and blank nodes can only be at subject or object positions. As we said before, triples are facts or statements about our information, so as an example (`db:Chile`, `db:population`, 17,000,000) is a valid RDF triple. In turtle, the triple would be written simply as `db:Chile db:population 17000000`. In the example we can see that the subject is the main topic of the statement, the predicate is a property or relation between the subject and the object and the object is the value of such a property. Thus, the given example can be read as “Chile's population is 17 million people”.

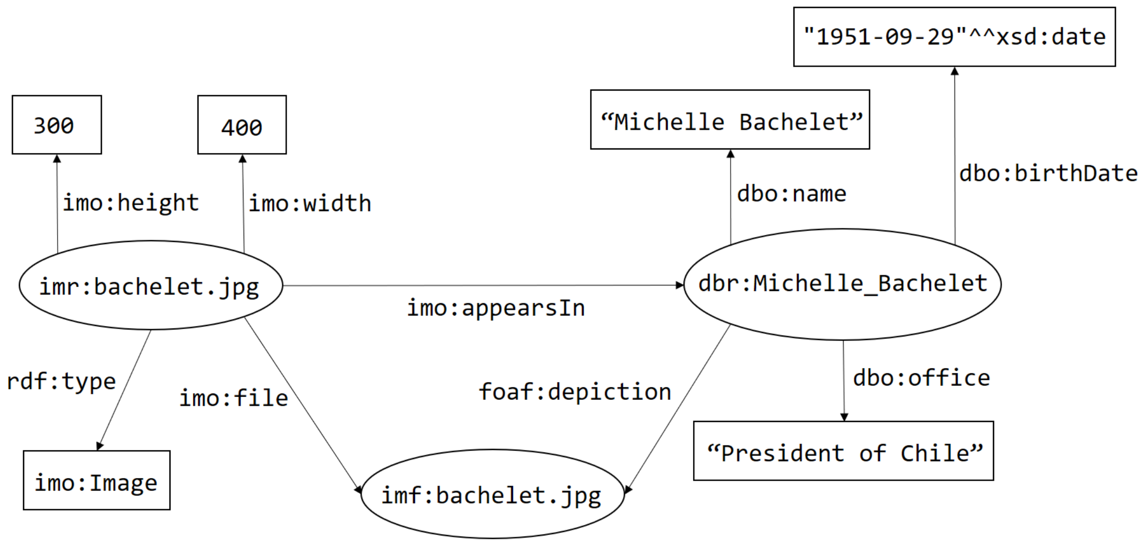


Figure 2.11: Example of an RDF graph. Ellipses show instances of resources identified by IRIs. Classes and primitive datatypes are drawn in boxes. Arrows represent properties pointing from subject to object.

Graphs are sets of triples that may share Terms. Given this graph nature of RDF datasets, URI labeled nodes can be reached from outside in such a way that multiple datasets can be linked together and share information between them. In Figure 2.11 we show a graph example with data about a Chilean president; properties are shown as arrows oriented from subject to object; entities are depicted as ellipses and literal values are shown in boxes; dashed arrows represent relations between classes. In Listing 2.1 there is the graph of Figure 2.11 written in Turtle.

```
#Prefix declarations
@prefix imo: <http://imgpedia.dcc.uchile.cl/ontology#> .
@prefix imr: <http://imgpedia.dcc.uchile.cl/resource/> .
@prefix imf: <http://imgpedia.dcc.uchile.cl/files/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

imr:bachelet.jpg a imo:Image; #; reuses the subject
        imo:appearsIn dbr:Michelle_Bachelet ; #only predicate and object
        imo:height 300 ;
        imo:width 400 ;
        imo:file imf:bachelet.jpg . #. ends triples with the given subject

dbr:Michelle_Bachelet a :Person ;
        dbo:office "President of Chile" ;
        dbo:name "Michelle Bachelet" ;
        dbo:birthDate "1951-09-29"^^xsd:date ;
        foaf:depiction imr:bachelet.jpg .
```

Listing 2.1: RDF graph written in Turtle

RDF Schema and the Web Ontology Language (OWL)

RDF graphs are not enough for describing the complexity of the information. To enrich the expressiveness, define schemas and introduce the ability to reason over the data, RDF Schema [10] and OWL [33] were developed. However, we will rather not emphasize this area, but we will address some of the use cases of these standards. For the rest of the document we will use different prefixes in order to abbreviate the examples. The main prefixes we use are `rdf:` for RDF, `rdfs:` for RDF Schema, `owl:` for OWL and just `:` for a default namespace.

- **Subclasses.** In order to specialize the type of a resource the property `rdfs:subClassOf` can be used. As an example, if a resource is a dog, then the resource is an animal, so the triple (`:Dog`, `rdfs:subClassOf`, `:Animal`) can state that fact.

- **Range and Domains.** To automatically infer the type of the subject and object of a triple, we can define the domain and range of a property using `rdfs:domain` and `rdfs:range` properties. For example, if only persons can have a depiction image, the next triples must be written to state that: `(:depiction`, `rdfs:domain`, `:Person)` and `(:depiction`, `rdfs:range`, `imo:Image)`.

- **Relation properties.** Relational properties such as symmetry or transitivity can be stated using `owl:TransitiveProperty` or `owl:SymmetricProperty`. For example, ancestry is a transitive relation; so if (`:ancestor a owl:TransitiveProperty`) and (`:A`, `:ancestor`, `:B`) and (`:B`, `:ancestor`, `:C`) the following triple is also a fact: (`:A`, `:ancestor`, `:C`).

- **Property chain axioms.** To extract conclusions about the data, chain rules can be stated. Such rules have the form *if A then B*. For example if (`:A :brotherOf :B`) and (`:B :fatherOf :C`) it becomes clear that (`:A :uncleOf :C`), but that triple might not be in the original data. In order to be able to make such entailment, the rule: `:uncleOf owl:propertyChainAxiom (:brotherOf :fatherOf)` has to be included to the graph. Where (:brotherOf :fatherOf) represents a list in RDF.

- **Unions, intersections and more.** Classes can be unions of other classes, intersections, etc. Also there are existential and universal quantifiers and many more schema-defining properties which can be further explored in [10] and [33].

An ontology is a set of rules and concepts that show the properties and relationships of the data, mainly using RDF Schema and the Web Ontology Language. In Figure 2.12 we show an example of an ontology, which is an extract of the ontology of IMGpedia. We see that the RDFS standard is used to define more specific subclasses of the Descriptor class.

SPARQL Protocol and RDF Query Language (SPARQL)

The SPARQL standard is both a specialized RDF query language [40] and a protocol that defines how queries can be invoked and how the results are returned [14]. As explained by Hogan [24], SPARQL queries have a well-defined structure and syntax. A generic query has

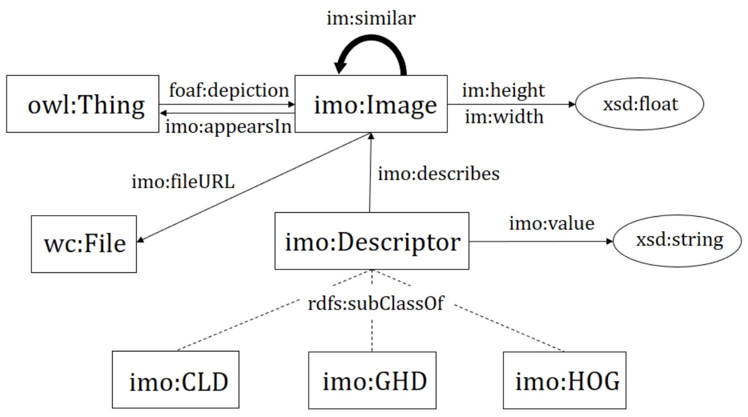


Figure 2.12: Ontology example. Classes are shown in boxes and primitive datatypes in ellipses. Arrows represent relations between instances of the classes, pointing from domain to range. Dashed arrows represent relations between classes themselves.

the following clauses:

- **Prefix Declarations.** As we have seen with Turtle, SPARQL also defines a block for prefix declarations for the sake of abbreviating the following clauses.
- **Dataset Clause.** It is possible to define here which datasets (RDF graphs) are going to be used to resolve the query.
- **Result Clause.** Here the type of query to be executed is defined (more detail in following paragraph) and what results are desired (if any).
- **Query Clause.** This clause contains the patterns to be matched against the graph in order to fetch results. Matching values are stored in free variables denoted with the prefix "`?`" (e. g., `?name`, `?address`).
- **Solution Modifiers.** This block allows pagination, sorting and grouping of results, such as `ORDER BY`, `LIMIT` and `OFFSET` (similar to SQL).

Different kinds of queries can be asked against an RDF graph. The classic query is `SELECT`, which retrieves variables that match the data. `ASK` is a query type that returns true or false if a match against the data exists. Furthermore, `CONSTRUCT` can create an RDF graph and `DESCRIBE` retrieves a description for a matching RDF term or its bindings in the graph.

Consider the RDF data given as example in the previous paragraph. A simple query that can be asked to that graph is shown in Listing 2.2 where the names and depiction images of all presidents of Chile that were born after 1950 are requested with the names sorted alphabetically.

A few years later, SPARQL was extended to support more features. In the SPARQL 1.1 [22] standard, the following features were introduced:

- Property paths: Instead of adding a matching clause for each property, now the complete path can be given in a single line just as `?uncle :brotherOf/:fatherOf ?me` to retrieve the uncle of someone. If the path is made of transitive properties, the operator ∗ can be used, for example, to get all the ancestors of someone we just write `?ancestor :ancestor* ?me`.

- Nested queries: In this extension, a sub-`SELECT` query can be placed within the main query.
- Aggregates: Aggregation functions were added, such as `MIN`, `MAX`, `AVG`, etc.
- Variable binding: For storing values (returned from aggregates for example) into named variables.

```
#Beginning of prefix declarations
PREFIX im: <http://imgpedia.dcc.uchile.cl/resource/>
PREFIX db: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

#Result clause: retrieve president name and depiction image
SELECT ?name ?img

#Query clause: sentences that must match against the data
WHERE{
    ?president db:office "President of Chile" .
    ?president db:name ?name .
    ?president db:birthDate ?bday .
    ?president foaf:depiction ?img_name .
    ?img_name im:file ?img .
    #Filter results, leaving only presidents born after the 1950
    FILTER(?bday > "1950-01-01"^^xsd:date) .
}
#Solution modifiers: sorting names alphabetically order
ORDER BY(?president, ASC)
```

Listing 2.2: SPARQL query example

Linked Open Data

One of the main goals of the Semantic Web is to be able to discover new sources of information and to share and complement structured data on the Web. Linked Data arose as a solution for linking the isolated data sources that were very common in the early Web. In 2006, Berners-Lee [8] published the W3C Design Issues to frame the principles of Linked Data and gave some examples. The four principles stated in the document are:

1. Use `URI`s as names for *things*
2. Use `HTTP URI`s so that people can look up those names

3. When someone looks up a URI, provide useful information, using the standards (`RDF*`, `SPARQL`)

4. Include links to other URIs. so that they can discover more things.

These principles are important because they outline the very fundamental requisites for publishing Linked Data and making it both accessible on the Web and useful for people. Furthermore, to evaluate the quality of an Open Linked Data dataset, a "5 Star Scheme" of evaluation is defined by Berners-Lee [8] and summarized in Table 2.1. Using this rating system, a PDF document under a Creative Commons License and available in the Web is a 1-star source of data; an MS Excel spreadsheet under the same conditions is a 2-star source; a CSV file with the same data is a 3-star data source; an RDF dump with only local URIs for naming things is a 4-star dataset; and, finally, a 4-star dataset that contains links to other sources is a 5-star source of information on the Web.

⋆	Available on the Web, but with an open licence, to be Open Data
⋆⋆	Available as machine-readable structured data
⋆⋆⋆	As (2) plus non-proprietary format
⋆⋆⋆⋆	All the above, plus: Use open standards from W3C to identify things.
⋆⋆⋆⋆⋆	All the above, plus: Link your data to other people's data to provide context

Table 2.1: 5 Star Scheme for Linked Open Data Evaluation. Source [8]

The DBpedia project [30] is a major example of a 5-star dataset in the Linked Open data community, becoming the central node of the Linked Open Data cloud as we can see in Figure 2.13; i.e., several other sources of Linked Data provide links to the DBpedia dataset (and vice versa). DBpedia automatically extracts RDF triples from the semi-structured information contained in Wikipedia articles, such as tables and listings. On the other hand, Wikidata [47] is an initiative of the Wikimedia Foundation to collaboratively build a knowledge-graph about the facts stated in Wikipedia. If we would like to compare both datasets, DBpedia currently contains more information in its graph than Wikidata, but the quality of the facts of the latter is superior than the quality of DBpedia due to the automatic extraction of facts versus human supervised knowledge curation.

2.4.2. Multimedia on the Web of Data

Multimedia is often left behind when a new Linked Data dataset is published since the focus is mostly on textual information. Nevertheless, a few initiatives are trying to include multimedia into the Web of Data. The main example of such initiatives is DBpedia Commons [46], which automatically extracts the meta-data for all multimedia files in the Wikimedia Commons dataset, such as its author, licensing, description and more. However, DBpedia Commons does not describe the content of the media itself nor provide links to the DBpedia [30] entities that use a given multimedia file.

Recently, Wikidata [47] released a prototype for data support for Wikimedia Commons

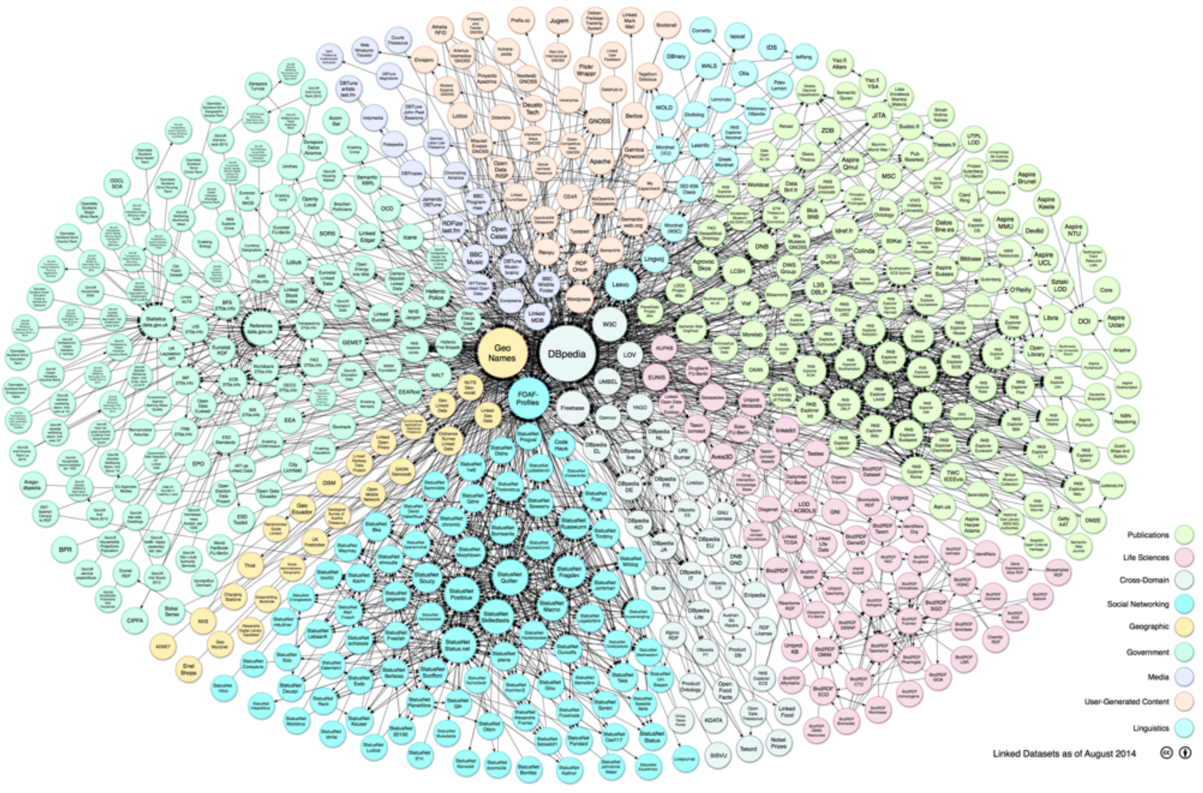


Figure 2.13: Linking Open Data cloud diagram 2014, by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/ .

files[3]. This work is very preliminary, but so far, it allows to upload a file and associate it to the mediainfo page and add statements to existing ones.

There are some approaches that try to combine feature description of images with Description Logics to perform Image Retrieval. Di Sciascio et al. [15] intend to retrieve images that match against *high-level* queries such as *all images containing a galloping horse*. In fact, they do not achieve that goal; what they do indeed is to label images with geometric shapes and build composites of them and then match the composite appearance in the images in a dataset, so they can query with sketches or by example. Hu et al. [25] use this approach to automatically classify medical images and identify masses or tumors in X-ray mammographic examinations by comparing them against previously labelled images. The labelling process was performed manually by specialists in the area. Other approaches suggest the use of *fuzzy* description logics, extending the mathematical definitions in order to allow queries with impreciseness, such as visual similarity queries [43, 44].

In general, handling multimedia in the Semantic Web is usually addressed using metadata or tags associated to the media files instead of analyzing the content of the media itself through content descriptors. As Hausenblas et al. propose in [23] it is necessary to tag and annotate multimedia data in a way that Semantic Web principles are taken into account in order to make multimedia semantic retrieval easier. Such annotations can be written manually or automatically depending on the case. A use case they propose is to find

[3]`http://structured-commons.wmflabs.org/`

all interviews of a given artist within a dataset of TV videos, previously tagged by users. Another interesting approach to solve this semantic retrieval of multimedia files is proposed by Passant [39], where they use *semantic distances* between music artists and ratings from users in order to give song recommendations to users. As we stated before, none of these approaches involving multimedia data within Linked Data use analysis of the content of the media, but tags and metadata only. Kurz et al. [29] propose an extension to SPARQL standard, called SPARQL-MM, in order to add said spatio-temporal tags in multimedia files to knowledge-bases, specifying regions of interest and temporal relations. It allows users to ask queries like *retrieve all the frames of video that are between a given time interval* or *a frame that contains certain labels above/within/intersecting specific regions of the image*. However, this latter work again works off meta-data extracted manually from the video, rather than considering the content of it nor proposing ways in which that meta-data can be automatically extracted. Other approaches include efforts to use the MPEG-7 standard as an ontology in order to describe properly the multimedia metadata. Hunter [26] presents an OWL ontology based on the XML Schema definition of MPEG-7 [12] which later became the basis of many other ontologies, since they describe how to extend and specialize their vocabulary. This has lead to collaborative labeling systems for multimedia using the vocabulary defined in MPEG-7 alike ontologies [31, 18, 19]. Along the same lines, Arndt et al. [4] propose the annotation of media fragments to semantically describe the content of an image, and indicate the formal specification of a multimedia ontology based on the MPEG-7 standard for connecting descriptions to parts of a media asset. However, the described works are only focused on the metadata and human added descriptions for multimedia files and disregard the automated feature extraction, which is also described by the MPEG-7 standards. On the other hand, a recent work proposes an ontology to store and describe audio content and audio feature descriptors with the purpose of representing computational workflows of audio features and providing a common structure for feature data to enable the use of Linked Open Data principles and technologies in Music Informatics [3]; it is unclear, however, how this ontology could be extended to images or other multimedia.

2.4.3. Similarity Search with SPARQL

When querying in SPARQL, exact matches must be found in the data, but sometimes we want *imprecise* results, allowing a certain similarity measure to be computed and the most similar matches to be retrieved. Such is the case for content-based image retrieval applications, where it is irrelevant to obtain exact matches due to different compressions, resolutions and image quality, but similar items become a desired answer.

Regarding similarity search using SPARQL: iSPARQL [28] is an extension of SPARQL 1.1 [22] that allows imprecise queries over RDF datasets (the *i* stands for *imprecise*). Of course, they provide similarity measures suited for textual information, such as the Levenshtein string similarity or Jaccard distance. They compare three approaches for collecting similar triples: using virtual triples to add similarity scores between variables, extending SPARQL filter functions to apply similarity thresholds or adding new solution modifiers to the SPARQL grammar. They state that the virtual triple approach is superior to the others and is the one that was implemented. A virtual triple is an RDF triple computed *on demand* and is not part of the actual RDF graph. In Listing 2.3 we can see an example of a query written in iSPARQL (taken from Kiefer et al. [28]) which retrieves similar articles in two different datasets: **opus** and **swrc**. The similarity measure is an aggregation between Levenshtein distance of the titles of the books and Jaccard distance between the titles of the articles. This approach allows to easily aggregate the similarity values since it binds them to variables. Sorting by similarity is also possible because of that. The *cons* of the approach are that it is necessary to extend SPARQL adding the `IMPRECISE` environment for generating the virtual triples and the enormous amount of cross joins that must be performed to compute similarities, but it can be fixed by pre-computing the similarity triples using a `CONSTRUCT` query or by building an index.

There are approaches that use the notion of range queries to retrieve geo-spatial information calculating distance between places. Zhai et al. [49] present a dataset of places with their names, coordinates, spatial relations and others to perform queries such as *all hotels within 10 kilometers from Wuhan University*. Such queries are supported through an extension to SPARQL that allows to filter by the distance between two places. In Listing 2.4 we can see an example query taken from [49].

```
SELECT ?publication1 ?publication2 ?similarity
WHERE{ #Obtaining all pairs book/article title
    ?publication1 rdfs:label ?title1 .
    ?publication1 opus:book_title ?booktitle1 .
    ?publication2 swrc:title ?title2 .
    ?publication2 swrc:booktitle ?booktitle2 .

IMPRECISE { #imprecise block to start generating virtual triples

    #calculating distance and filtering by threshold
    ?sim1 isparql:jac (?title1 ?title2 ) .
    FILTER (?sim1 >= 0.5) .
    ?sim2 isparql:lev (?booktitle1 ?booktitle2 ) .
    FILTER (?sim2 >= 0.5) .

    #aggregating distances and filtering by threshold
    ?similarity isparql:score (?sim1 ?sim2 0.6 0.4) .
    FILTER (?similarity >= 0.5) }
ORDER BY (?similarity, DESC) #sort results by similarity
```

Listing 2.3: iSPARQL query example [28]

```
SELECT ?neighbor ?name ?location
WHERE{
    ?x Geo:name "Wuhan University" .
    ?x Geo:location ?1xLoc.
    ?x Geo:location ?1yLoc.
    ?x Geo:name ?neighbor.
    ?x Geo:location ?2xLoc.
    ?x Geo:location ?2yLoc.

    #Get only those neighbors with "hotel" in its name
    FILTER( REGEX( ?name, "hotel"))
    #Calculate distance and apply the threshold
    FILTER (Geo:distance(?1xLoc, ?1yLoc, ?2xLoc, ?2yLoc) < 10) .
}
```

Listing 2.4: Geo-spatial SPARQL query [49]

Chapter 3

System Overview

IMGpedia is an initiative that aims to bring multimedia content closer to the Web of Data. It has a dataset that allows *visuo-semantic* queries and knowledge discovery by joining visual descriptors and structured data about the images. This work is a preliminary step to start researching ways in which multimedia content on the Web can be queried both visually and semantically. In this chapter we will briefly review the basis, the goals and the challenges of building a knowledge-base such as IMGpedia.

The construction of the IMGpedia knowledge-graph requires the acquisition of all the images from the Wikimedia Commons dataset in order to process and analyse them. In said graph, each image will be represented as a *visual entity* with links to its visual descriptors, its metadata from DBpedia Commons [46] and to other *visual entities* that are visually similar to it. To achieve such goals, a set of steps must be performed.

- To acquire the images of the Wikimedia Commons dataset
- To compute a set of visual descriptors for every image
- To discover similarity links between IMGpedia visual entities
- To find semantic relations between visual entities and DBpedia entities
- To merge all the data sources in a single RDF knowledge-base.

Once all steps are finished, we want to run *visuo-semantic* queries over the resulting knowledge-base. In Section 3.3 we can see how IMGpedia graph is modelled and example queries that can be performed using the data.

Extension	Count
JPG	13,701,448
PNG	1,098,988
SVG	661,552
PDF	171,062
OGG	170,963
GIF	156,242
Other	144,601
Total	16,104,856

Table 3.1: Count of encodings of the images from Wikimedia Commons.

3.1. Images of IMGpedia

In order to start building IMGpedia [11] we must download all images in the Wikimedia Commons dataset. Wikimedia Foundation provides a mirror with the images[1], so they can be downloaded using `rsync` file transfer protocol.

The Wikimedia Commons dataset contains 30 million freely usable media files that can be used within Wikipedia articles. About 15 million of those media files are images, having a total size of 22 TB. The images of the dataset can be easily used in Wikipedia articles and galleries, however, many of the pictures uploaded are there just for a sharing and preserving intention and do not appear in any article or gallerie. The images are originally divided into 16 folders labelled from 0 to f hexadecimal digits. Each folder contains 16 other folders consequently labelled with the same digits. From now on, when we talk about an *image folder* we are referring to one of the top level folders of the mirror's filesystem. In the same manner, an *image subfolder* will be one of the directories within a *folder*. Each *subfolder* contains about 58,000 images.In Table 3.1 we provide a summary of the number of images per encoding. 92 % of the images in the dataset have `.jpg` or `.png` extensions and said images are the ones that we consider in the dataset of IMGpedia for practical reasons; mainly the cost that it would imply to implement non-standard techniques to analyse rare image compressions and encodings such as `.svg` or `.gif`. A small sample of the images downloaded can be seen in Figure 3.1 where the variety of domains and subjects of the images can be appreciated.

Figure 3.1: Wikimedia Commons image sample.

The download process took about 40 days at an average speed of 500 GB per day. To

[1] `rsync://ftpmirror.your.org/wikimedia-images/wikipedia/commons/`

obtain the best download speed possible, several connections via `rsync` were used. Typically, each connection requested a full *folder* to the mirror. The local copy is stored into a Network-attached storage (NAS) system, with 33TB of total disk space.

3.2. Visual Descriptors of IMGpedia

A set of visual descriptors to compute has to be chosen to be part of IMGpedia dataset. We use four different descriptors that capture a variety of aspects about images: color, brightness, border distribution and semantic composition. All of them are represented as very high-dimensional vectors. The detail of how these descriptors are calculated can be found in Chapter 2. These descriptors are:

- Gray Histogram Descriptor (GHD), dividing the image in 4×4 blocks and using 16 bins for the gray intensities. **256 dimensions**.
- Histogram of Oriented Gradients (HOG), dividing the image in 4×4 blocks, using a gradient magnitud threshold of 150 and 18 bins for the gradient orientation. **288 dimensions**.
- DeCAF7 [16], the second-last self-convolutional layer of a pre-trained neural network. **4096 dimensions**.
- Color Layout Descriptor [37] (CLD), the discrete cosine transform of an 8×8 pixel icon of the image. **192 dimensions**.

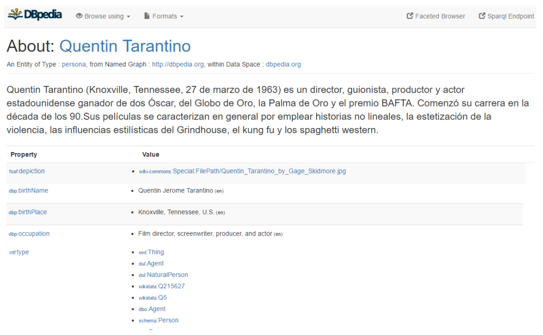
Having a variety of visual descriptors allows us to improve the similarity search, considering both the context of the search and the nature of the source image that is used. To do so, we can choose the descriptor that best fits the problem; i.e., in a plain-coloured image, border-based descriptors will work much better than color-based ones if the goal is to retrieve similar object patterns (such as trees); color-based descriptors will perform better in the search of similar landscapes.

After the local copy was acquired, the descriptors had to be calculated. This required us to estimate the time this stage would take, so we conducted experiments (that will be reviewed in Chapter 4) to have a baseline for the estimation and determine if the project is viable or not. All *classic* descriptors run in a reasonable amount of time: less than 100 ms per image, using multiple execution threads. However, the DeCAF descriptor had a poor time performance, taking more than 3 seconds per image and worse when multiple threads were used. This may have happened because of the unavailability of a machine with GPU: we ran DeCAF only in CPU mode. Despite our efforts, we could not improve the performance of the neural networks underlying the DeCAF7 descriptor sufficiently to be able to calculate the results on the Wikimedia Commons image dataset in reasonable time with available hardware. Hence we decided to abandon this descriptor.

3.3. IMGpedia Knowledge-base

The goal of IMGpedia is to make data available as an RDF-graph with links between similar images, between an image and its descriptors, external links to the related source in DBpedia Commons, to the image URL in Wikimedia Commons and to the DBpedia resources an image depicts.

To facilitate querying IMGpedia, we propose to compute static relationships between images, such as `similar` if the descriptors of two images are close in terms of a distance threshold. The brute-force approach for finding all pairs of images within a given distance takes a quadratic number of comparisons, and would require more than 700TB of disk space to store the matrix distance using simple-precision floating-point representations. Thus, the challenge is to do this similarity search efficiently in both time and space, where we propose to explore: building an index structure to improve the search time over the dataset; using approximate similarity search algorithms; using self-similarity join techniques; and so forth. The detail of this experimental stage of the process and its results can be found in Chapter 4.



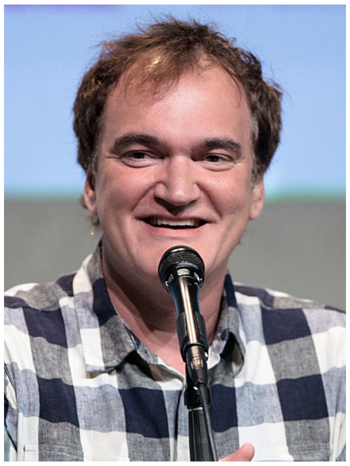
About: Quentin Tarantino

An Entity of Type : persona, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

Quentin Tarantino (Knoxville, Tennessee, 27 de marzo de 1963) es un director, guionista, productor y actor estadounidense ganador de dos Óscar, del Globo de Oro, la Palma de Oro y el premio BAFTA. Comenzó su carrera en la década de los 90.Sus películas se caracterizan en general por emplear historias no lineales, la estetización de la violencia, las influencias estilísticas del Grindhouse, el kung fu y los spaghetti western.

Property	Value
foaf:depiction	• wiki-commons:Special:FilePath/Quentin_Tarantino_by_Gage_Skidmore.jpg
dbp:birthName	• Quentin Jerome Tarantino (en)
dbp:birthPlace	• Knoxville, Tennessee, U.S. (en)
dbp:occupation	• Film director, screenwriter, producer, and actor (en)
rdf:type	• owl:Thing • dul:Agent • dul:NaturalPerson • wikidata:Q215627 • wikidata:Q5 • dbo:Agent • schema:Person

(a) Extract of the DBpedia resource about Quentin Tarantino.



(b) Picture of Quentin Tarantino during an interview. The depiction of `dbr:Quentin_Tarantino`. Photograpy by Gage Skidmore.

Figure 3.2: The data to be merged in the IMGpedia ontology.

As a preliminary step towards creating the IMGpedia knowledge-graph, we also require a novel vocabulary (a new set of RDF terms) and an ontology (a set of inferencing patterns described in RDFS and OWL) to guide the modelling process. A deeper discussion of IMGpedia data modelling and publication can be found in Chapter 5.

One of the core ideas of IMGpedia is to join the data of DBpedia and DBpedia Commons in order to enrich our visual entities with both metadata of the images provided in Wikimedia Commons and semantic tags of the DBpedia resource the image depicts. As an example, in DBpedia, we can see that the resource `dbr:Quentin_Tarantino`[2] is depicted

[2]`http://dbpedia.org/resource/Quentin_Tarantino`

by the image `wiki-commons:Quentin_Tarantino_by_Gage_Skidmore.jpg`[3]. Then, if we enrich our visual entity for said image `imr:Quentin_Tarantino_by_Gage_Skidmore.jpg` with all relevant properties of `dbr:Quentin_Tarantino` such as its `rdf:type`, `dbo:birthName` or `dbo:occupation` we will know that in our image appears a `:Person` named `"Quentin Jerome Tarantino"` who works as `Film director, screenwriter, producer, and actor`. In Figure 3.2 we see an extract of `dbr:Quentin_Tarantino`, and the image that depicts the resource. Given the example, in IMGpedia we would like, for instance, to obtain the depictions of iconic scenes in Tarantino movies and later request similar images in order to obtain the stages or similar scenes in other movies. These questions can be answered storing similarity links between the visual entities or using SPARQL filters over the distances between the descriptors of the resulting images, obtained by joining our own image data with the semantic relations extracted from DBpedia.

[3]`https://upload.wikimedia.org/wikipedia/commons/0/0b/Quentin_Tarantino_by_Gage_Skidmore.jpg`

Chapter 4

Image Processing

In this chapter we will review the Processing and analysis of the images of IMGpedia, starting with the implementation of the visual descriptor algorithms, followed by the descriptor extraction process for the 15 million images. Later, we discuss how the similarity search problem was efficiently solved. And finally, we show and discuss the results of the computation of the similarity graph of the images in the IMGpedia dataset.

4.1. Reference Implementations for Visual Descriptors

Aside from publishing the URLs of images, IMGpedia will contain four different visual descriptors calculated for all the images of the dataset. These descriptors are:

- Gray Histogram Descriptor (GHD)
- Color Layout Descriptor [37] (CLD)
- Histogram of Oriented Gradients (HOG)
- DeCAF7 [16]

However, for practical reasons, DeCAF7 could not be computed for all the images, being thus abandoned. The reasons supporting this decision are given in the following section.

Since our aim is to make multimedia a first-class citizen of the Web of Data, we want to ease the process of feature extraction of images for people outside the multimedia community. For that reason, we release reference implementations of the algorithms that compute the aforementioned visual descriptors in different programming languages. The first three descriptor computers were developed in C++, Java and Python and they all intensively use the OpenCV library to handle image processing tasks, according to the algorithms described in Chapter 2. DeCAF7 was implemented only in Python since the bindings of the neural network framework used are provided in that language. We only use the CPU mode of Caffe framework due to the technical unavailability of an NVIDIA(r) GPU.

The implementations are parametrized according to the values described in Chapter 2. GHD requires the number of rows and columns the image has to be divided into and the number of bins of the gray histogram. CLD and DeCAF7 do not need further parameters. HOG requires the number of rows and columns the image is divided into and the threshold for the magnitude of the gradient.

To make sure the implementations in different programming languages are equivalent we have to prove that they return the same results for every image we run them with. We compared the results of the feature extraction on a dataset containing 2800 images taken from Flickr[1] and we check that the resulting vectors are equivalent component-wise for all implementations [17].

The sources and compilation instructions can be found in the following repository under GNU GPL licensing: `https://github.com/scferrada/imgpedia`

4.2. Descriptor Extraction Process

Once the local copy of Wikimedia Commons dataset and the implementations of the visual descriptor computers are available, we have to start calculating such descriptors over all the images to start building the IMGpedia dataset. The descriptors used and their parameters were mentioned in Chapter 3.

First of all, we run a performance benchmark for the descriptor extraction process to better estimate the time this stage would take. One of our preliminary hypotheses about this benchmark is that there is not a significant difference in execution time between implementations in different programming languages, since they all intensively invoke OpenCV library methods implemented in C/C++ and the intrinsic speed of execution of the language will not be relevant.

We performed two experiments, both of them computing separately the five descriptors of 57,377 images within a *subfolder* of the dataset, using the Python implementation of the descriptor computers. One of the experiments calculated the descriptors sequentially in a single thread of execution and the other simultaneously, using multiple threads. The calculation process has three steps: reading the image from disk, calculating the descriptor vector and saving the vector to disk. We measure the time of the full process. Experiments were performed on a machine with Debian 4.1.1, a 2.2 GHz (24 core) Intel(r) Xeon processor, and 120 GB of RAM. For the parallel computation experiment, 28 threads were used [17].

The results of the experiments are shown in Table 4.1. There we can see that using multiple threads of execution improves the performance in one order of magnitude in the calculation of classic descriptors (GHD, CLD, and HOG). However, DeCAF7 does not benefit from using multiple threads because the Caffe [27] implementation of neural networks already uses multithreading to process a single image, so it becomes a slight overhead in the computation to have several neural networks in parallel [17]. Therefore, the estimated time for running

[1]`http://www.flickr.com`

Descriptor	**Average time per image** [ms]	
	Single Thread	*Multithread*
GHD	119.4	10.4
CLD	436.7	30.4
HOG	477.7	25.7
DeCAF7	3634.8	3809.7

Table 4.1: Descriptor calculation benchmark.

DeCAF7 over the whole dataset is about 1.8 years whereas the computation of classic descriptors would take only a few days: 45 hours for the gray histogram (the fastest) and 124 hours for the Color Layout Descriptor, the most expensive.

A way to improve DeCAF7 execution time is via GPU optimizations of Caffe. Preliminary experiments were made, since we have no direct access to a GPU, and we found that a 4x improvement is achieved. From previous estimation we can conclude that CPU computation of DeCAF7 is non-viable for this work and the GPU computation of it will be left as future work.

We repeated the same experiment using C++ implementations to see if there was a significant improvement in execution time because of the intrinsic time overhead interpreted languages have. Compared with the Python experiments, C++ took only 3 seconds less over all the images, using same settings as before. This is because the Python implementations invoke C/C++ primitives of OpenCV and Numpy, so the overhead of the interpreted programming languages does not really affects the overall performance of the feature extraction process. For this reason, all further algorithms and routines will be developed in Python since it simplifies the programming process.

4.3. Similarity Search

Defining similarity relations between visual entities is crucial to accomplish the main objective of this work. For a first version of IMGpedia, we will include the 10 nearest neighbors for each image of the dataset which are to be linked using the `similar` relation. The decision of only considering the 10-NN is based on keeping an upper bound on the number of triples of the resulting IMGpedia dataset around 5 billion triples. Ideally, we are able to store all pair-wise distances but, as was stated before, it is too costly in terms of time and memory to materialize all distances. Thus only considering the 10-NN of each image produces three reasonably large similarity graphs (one for each visual descriptor) with about 500 million of triples in each.

To obtain said similarity relations, we need to choose between many methods to solve three 10-NN problems (one for each descriptor) in three datasets of ∼15 million high-dimensional vectors in a reasonable time. Using a brute force algorithm is non-viable, but we could use it over a smaller dataset to build a ground truth in order to compare with other methods.

Approximated KNN methods arise as a reasonable solution to solve this problem, but their accuracy has to be tested first. However, the exact computation of the nearest neighbors through pivot indexing (using Paredes et al. algorithm [38]) and a further analysis of the distance distribution will be addressed as future work.

We test FLANN (Fast Library for Approximated Nearest Neighbors) [34] because it claims to be scalable and simple to use: both handy qualities to solve a big data problem like ours. FLANN automatically chooses the best approximated strategy to solve the KNN problem depending on the dataset and the precision of the approximation can be manually set. These strategies include randomized k-d forests, priority search k-means trees and hierarchical clustering trees [35]. Following, a brief characterization of each strategy, for further details refer to Muja et al. [35]:

- A k-d forest is a set of randomized k-d trees stored on a priority queue, which are queried in parallel. These k-d trees are randomized because the split hyper-plane of the space is chosen at random on the first N dimensions.

- A k-means tree is a data structure that has on each level k clusters of the dataset, so each cluster is subsequently clusterized k times until the raw data is on the leaves of the tree.

- Hierarchical clustering trees use binary features of the data, random hierarchical clusterization of the data based on those features and using multiple types of trees.

To gain insight into the performance of FLANN we conducted a preliminary experiment. First, we build a ground truth with the 10-NN of $20,000$ GHD descriptors over a dataset with the descriptors of a full *folder* of images ($\sim 800,000$ vectors) using the brute force algorithm, i.e., to compare each of the $20,000$ query images against each of the images of the dataset, computing Euclidean distance and keeping the 10 minimum answers. The construction of the ground truth took 3.5 days using 16 threads of execution, one for each *subfolder* of the dataset. After that, we give FLANN the same input and distance function and we let it choose the best approximated search strategy for the dataset and we manually set the precision up to $90\,\%$. FLANN took only 13 minutes to deliver the results using a *single* execution thread. After parsing and comparing these results against our ground truth we found out that FLANN *real* precision was $79\,\%$ for the stated problem. We decided that the gain in execution time compensates the loss in precision, so we choose FLANN to compute the approximated nearest neighbors between all images.

In order to use FLANN over the whole dataset, descriptors have to be preprocessed. This is because FLANN requires the vectors to be in a single row-major matrix and give it as input. Since each vector will be on a particular row of the matrix, we need to keep track of the name and location of the corresponding image so we can parse the search results afterwards. So first of all, we need to build a dictionary to enable the translation from row index in the matrix to the original image path. To secure the repeatability of the results, whenever we obtain the files in a given folder, we sort the filenames alphabetically. Both the dictionary and the input matrix can be computed on a single execution.

The output of a KNN search are two matrices: one with the indexes of the nearest vec-

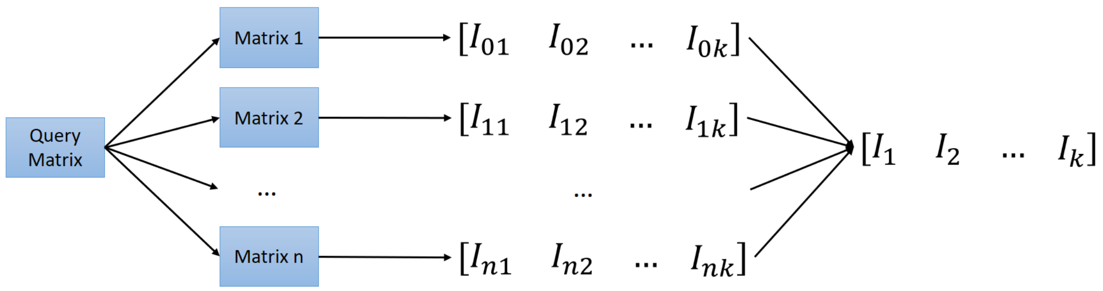

Query Matrix
Matrix 1
Matrix 2
...
Matrix n
$[I_{01} \quad I_{02} \quad ... \quad I_{0k}]$
$[I_{11} \quad I_{12} \quad ... \quad I_{1k}]$
...
$[I_{n1} \quad I_{n2} \quad ... \quad I_{nk}]$
$[I_1 \quad I_2 \quad ... \quad I_k]$

Figure 4.1: Pipeline of computation of the nearest neighbors using FLANN

tors for each query vector in a row-major order; and the other with the computed distance between the query vector and its neighbors. Since these matrices do not fit with the data model we want to build, we have to parse them into a more machine-readable way: that is, triples (`source_image`, `target_image`, `distance`), where `source_image` is the query vector, `target_image` is one of the nearest neighbors, and `distance` is the computed distance between them.

The last thing to overcome before we can run the similarity search is to choose a fitting distance function general enough to solve the problem in any dataset. For this work we choose Manhattan distance function, given the results shown in Aggarwal et al. [1] regarding the bad performance of high-order Minkowski distances (see Equation 4.1) in similarity search applications in high-dimensional spaces.

$$L_p(x,y) = \left(\sum_{\mathrm{i}=1}^{n} |x_{\mathrm{i}} - y_{\mathrm{i}}|^p \right)^{1/p} \tag{4.1}$$

Finally, to obtain the 10-NN of every image in IMGpedia dataset, we use the following pipeline for each of the three descriptors:

- We parse the single vector files into the FLANN input matrix described above. We build one matrix for all descriptors within a *folder* of images. As a result of this stage, we have 16 row-major ordered descriptor matrices.
- We build a FLANN index for each matrix using L_1 (Manhattan) distance. FLANN decides in each case which approximated strategy fits better for the subset. FLANN precision is set to 90 %.
- For each *folder*, we launch a thread that searches for the approximated 10-NN of all images of the *folder* over every FLANN index. When this stage finishes we have 160 candidates to be the 10-NN (10 per index).
- We obtain the 10 minimum distances from the 160 candidates for each image and store the results as triples (`source_image`, `target_image`, `distance`).

In Figure 4.1 we graphically show the pipeline of the nearest neighbors calculation, descri-

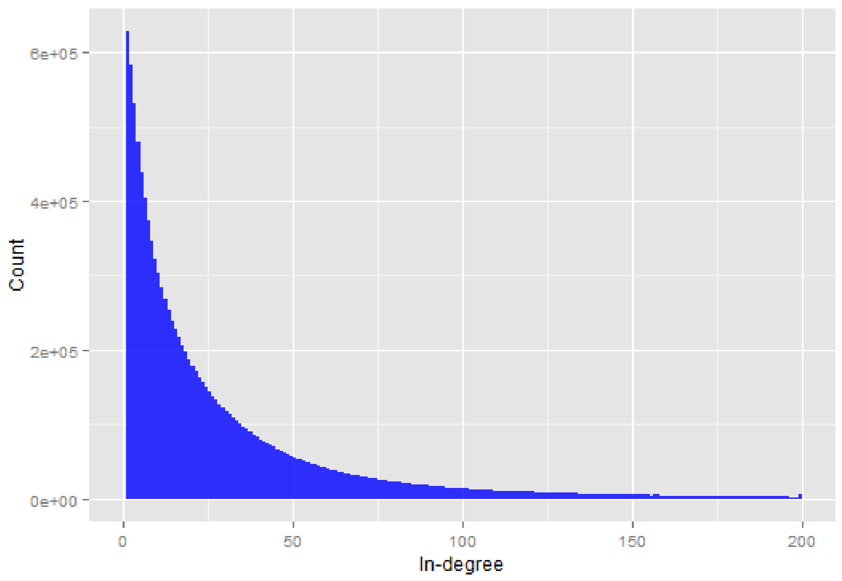


Figure 4.2: Distribution of the in-degree of the images on the similarity graph of IMGpedia.

bed above. The calculation of the approximated nearest neighbors took on average 3.9 hours per descriptor. The parsing of the results took 4 more hours per descriptor, which is a good performance, considering that the most optimistic estimations for the brute force algorithm were of the order of years using a multithreaded brute-force implementation.

4.4. Similarity Search Results

The work described in the previous section produced the core data for the IMGpedia dataset, which is the similarity links between images, depending on a specific visual descriptor. In this section we will review some of the results retrieved in the similarity search process. The similarity links define a graph with 14,765,300 vertices (from which 11,917,390 appear as a nearest neighbour of another image) and 473,680,270 directed edges. In Figure 4.2 we show the in-degree distribution of the graph, which shows that there are many images that are similar to a few others and few images that are similar to many other images. Given the scale of the graph, further analysis and characterization of the graph is left for future work.

In Figure 4.3 we see the similarity search results for an image of *Hopsten Marktplatz*[2] in Germany. There is the source image and its 10-NN using the Histogram of Oriented Gradients descriptor. We can remark that the visual similarity is evident, all the neighbors present a centered tree as the main object in the picture. This is mostly because of the descriptor used, since HOG captures the border distribution of the image in small portions of it, and a tree shape has a very particular edge distribution that matches among all the neighbors.

In the following, we will review the results of the similarity search using the same image

[2]Original image can be found on `https://commons.wikimedia.org/wiki/File:Hopsten_Marktplatz_3.jpg`

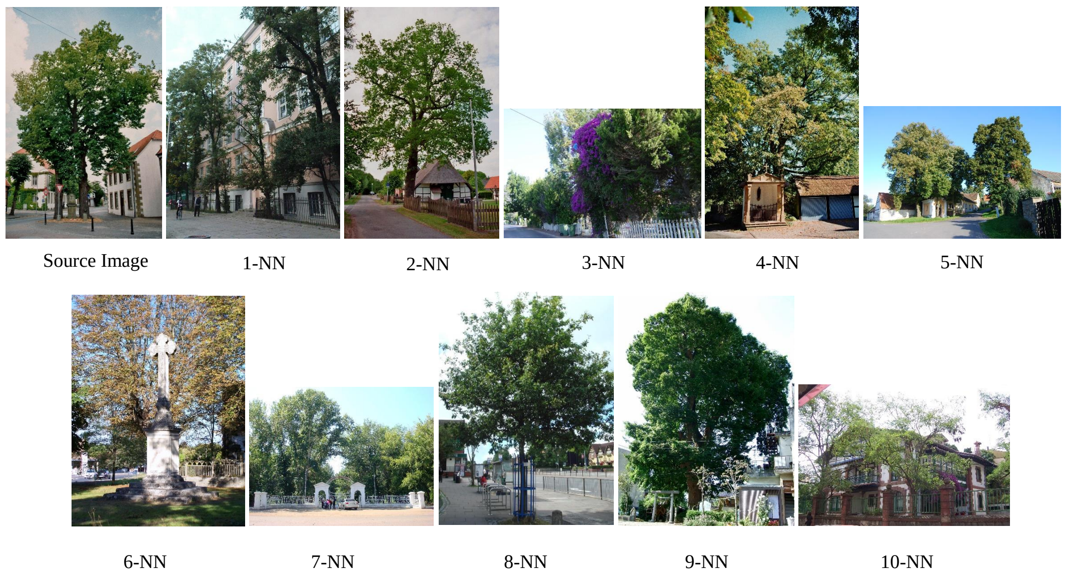


Figure 4.3: 10-NN of Hopsten Marktplatz picture using HOG descriptor.

but different descriptors. In Figure 4.4, said results are shown. We can see that this time, results are not just trees in the foreground: we also see a shore and a basketball ring with trees in the foreground. What changes between the Histogram of Oriented Gradients and the Gray Histogram? They are meant to extract different visual attributes from the image. GHD takes into account the brightness distribution of the picture which, in the case of trees, is a pretty regular pattern of light and dark pixels. Such regularity can be seen in the 6-NN picture which is a painting from Friedrich Ernst Wolfrom. This painting could not be retrieved as a nearest neighbor using HOG, because edges in paintings are softer and blurrier than in real-world pictures. But since GHD takes into account color intensity only, paintings can be linked to similar real-word images.

In Figure 4.5 we have the first 5 of the 10-NN of Hopsten Marktplatz using the Color Layout descriptor. It only takes to see those results to realize they are not good in means of visual similarity. Why are the nearest neighbors so visually different? CLD computes the average color on each of the 8×8 blocks the image is divided. And, as can be expected, the average colours of the source image are all similar to white with a green stain in the center. Surprisingly, the icons of the nearest neighbors present the same shapes as the source image. In Figure 4.6 we can see visualizations of the icons (zoomed in) on which CLD is computed for Hopsten Marktplatz image and the first two nearest neighbors and tell the similarity between them. A visualization is used, since the icon is transformed to YC_bC_r colorspace, so it helps to understand better the similarity between the intensities of each color channel. These results are not an argument against the usefulness of Color Layout Descriptor in image similarity search, but it is a useful example to state that not all descriptors work well on the same kind of image. There is a particular visual descriptor that behaves well on a particular image and in a particular task and there are others that do not.

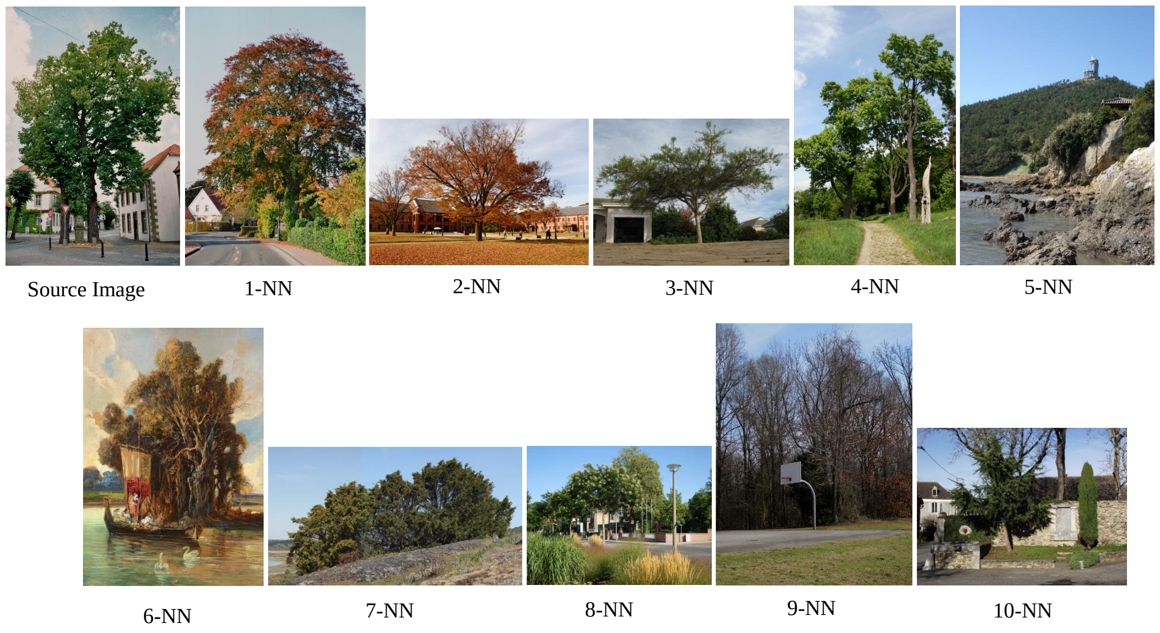


Figure 4.4: 10-NN of Hopsten Marktplatz picture using GHD.

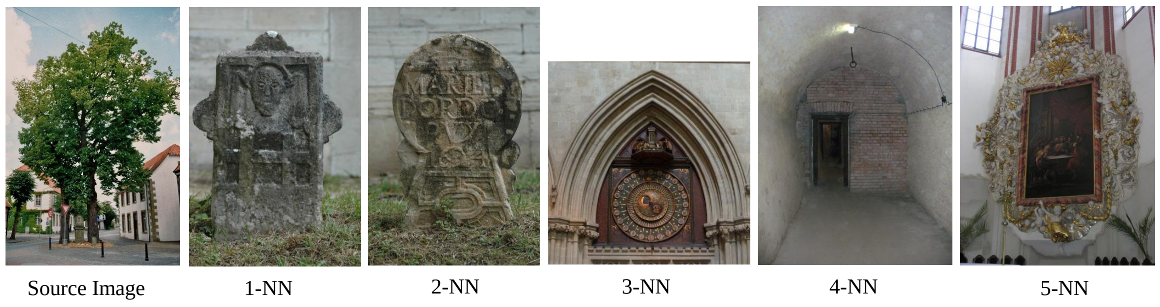
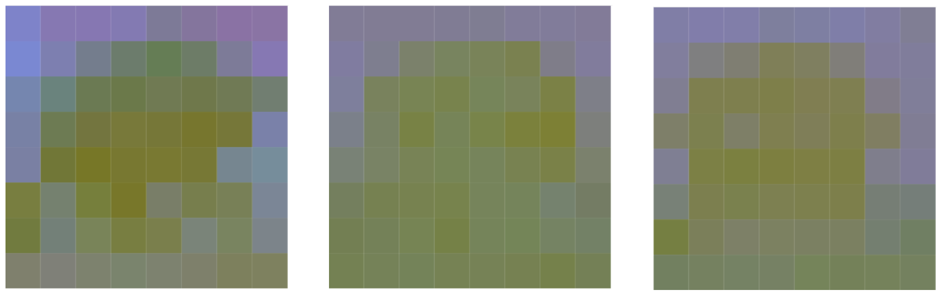


Figure 4.5: 5-NN of Hopsten Marktplatz picture using CLD.

On the other hand, in Figure 4.7 we can see a much better example (in terms of visual similarity) of the nearest neighbors search results using CLD taking a picture of a building in Dreifertstraße street[3], Germany. This improvement is only because of the color distribution of the source image. The average color icon is mostly brown on the center and below and sky blue on the top, just as with all the nearest neighbors. However we can see that the images are not really close semantically, since the nearest neighbors are mostly landscapes of fields and not buildings like the source image. This leads to the thought that CLD may work better if it is combined with other descriptors such as HOG.

[3]Original image can be found in `https://commons.wikimedia.org/wiki/File:91_Dreifertstrasse_1.JPG`



(a) Icon for source image (b) Icon for 1-NN (c) Icon for 2-NN

Figure 4.6: Average color icon for Hopsten Marktplatz and its first two nearest neigbors

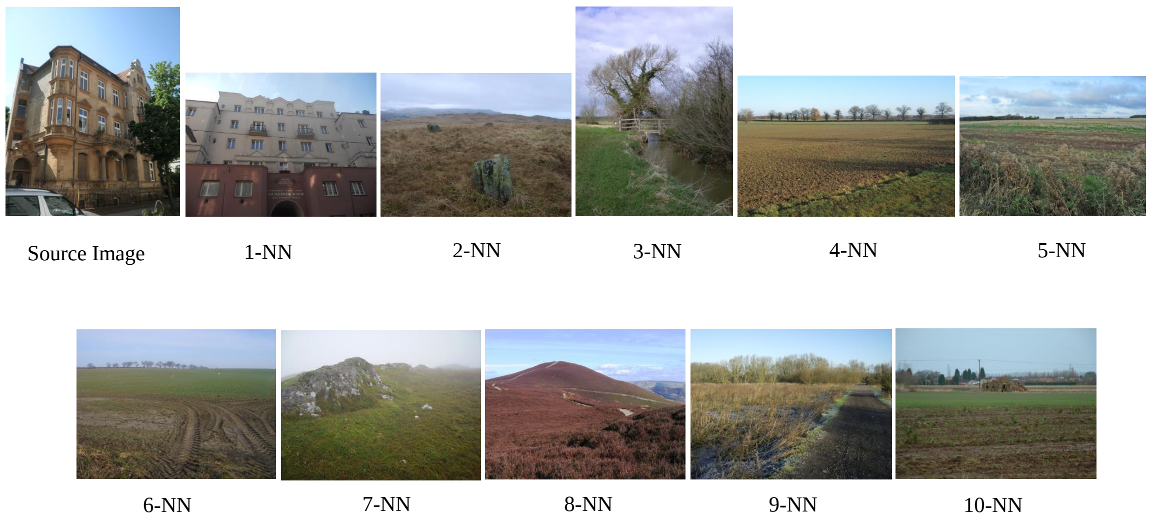


Figure 4.7: 10-NN of a picture of a building in Dreifertstraße street using CLD

Chapter 5

Data Modelling and Publication

The ultimate goal of IMGpedia is to provide a way to perform queries over an image dataset that involve both visual and semantic information. To do so, we rely on Semantic Web Technologies (details in Chapter 2). In this Chapter, we review the design and the process of building the knowledge-base of IMGpedia and how to perform queries over it. We start discussing the design of the IMGpedia ontology, which defines the vocabulary to be used within the knowledge-base data. Later we show the structure of the RDF data that will be inserted into the graph. We finish by talking about the public SPARQL endpoint where the data was loaded.

5.1. IMGpedia Ontology

Ontologies are required to establish a schema for the data to be included in a knowledge-base. Here, we define the classes of IMGpedia entities and the available relationships between their instances. The classes included in the IMGpedia ontology are:

- **Image:** it represents an image from the dataset as a visual entity of the graph. It holds the size of the image and links to the resource of DBpedia Commons that describes the same image.

- **Descriptor:** it holds the vector of a visual descriptor of a certain image. A Descriptor can be of the following subclasses of Descriptor, where each one represents one of the visual descriptors computed in this work: CLD, for Color Layout Descriptor; HOG, for Histogram of Oriented Gradients; and GHD, for Gray Histogram Descriptor.

- **Image Relation:** this is a structure that will contain the similarity relations between Images, including the computed distance and the visual descriptor involved in the calculation.

Object properties are also defined in order to build semantic bridges between IMGpedia instances, resources of other graphs and primitive values. These properties are:

- **Similar:** this is the main relation of IMGpedia. It links two images and states that the object of the relation is within the 10 nearest neighbors of the subject using any of the visual descriptors.

- **Height:** represents the height of an Image. Its range is a number.

- **Width:** represents the width of an Image. Its range is a number.

- **Describes:** this relation links every Descriptor with the Image it was extracted from.

- **Value:** represents the value of a Descriptor as a string of the values of each component of the vector separated with commas.

- **Source Image:** is a relation between an Image Relation and an Image. Since Image Relations are placeholders for the similarity search results, they have to contain both images, the distance and the descriptor that was used. Source Image is the Image used as the query point in the nearest neighbors search.

- **Target Image:** in the context of the similarity search, a Target Image is one of the computed nearest neighbors for the Source Image. This object property links an Image Relation with such an Image.

- **Distance:** is the Manhattan distance computed between Source Image and Target image in the context of an Image Relation and, therefore, using the specified visual descriptor. The range of this property is float numbers.

- **Uses Descriptor Type:** refers to the descriptor that was used in the similarity search. The range of this property is a Class, but it is expected to be used with one of the IMGpedia sublasses of Descriptor.

- **Appears in:** this property links a visual entity of IMGpedia with the resources of DBpedia that represent the Wikipedia articles in which the image of the visual entity appears in.

- **File URL:** with this property we link the visual entity to their URL in the Wikimedia Commons dataset.

The complete and detailed vocabulary established by the ontology can be appreciated in Appendix A, written in Turtle RDF syntax (details about the syntax can be seen in Chapter 2). There we define the classes and object properties to be used in IMGpedia, along with the range and domain restrictions and description labels in both English and Spanish. An online version of the ontology can be found in `http://imgpedia.dcc.uchile.cl/ontology`. In Figure 5.1 we show graphically how classes interact with each other and with primitive datatypes through the described object properties. Visual entities can also be related with resources of other knowledge-bases via `imo:appearsIn` relation. There, we also show that any resource of type `owl:Thing` can have as depiction an IMGpedia visual entity. In Figure 5.2 we show the structure of the similarity links stored in IMGpedia. There, the properties `sourceImage`, `targetImage` and `distance` can be better understood. If two images are a pair (`sourceImage`, `targetImage`) in an Image Relation, they are also related with the `similar` property. In both figures, classes have been represented as rectangles, datatypes as ellipses and properties with arrows pointing from subject to object. Dashed lines represent subclasses. In Figure 5.2 we show that every image contained in IMGpedia has an equivalent resource in DBpedia Commons, by using the OWL [33] property `owl:sameAs`.

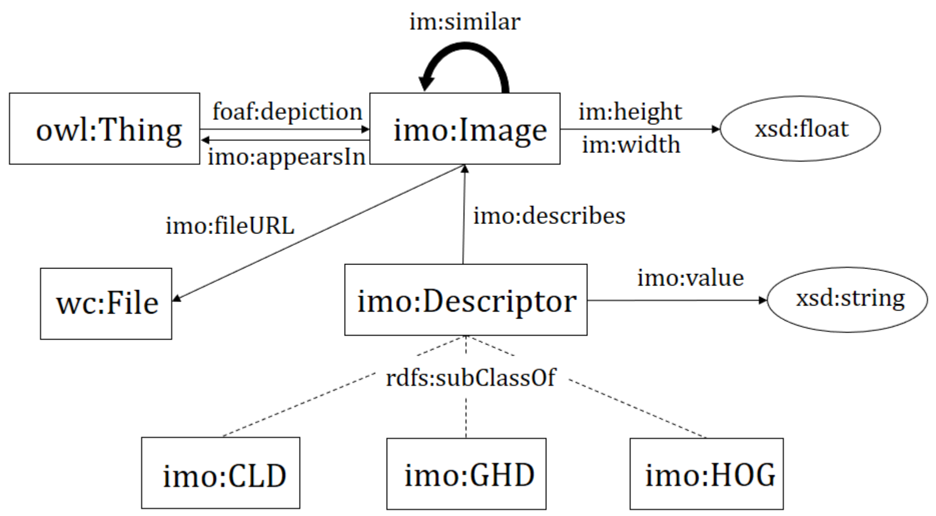


Figure 5.1: IMGpedia ontology core.

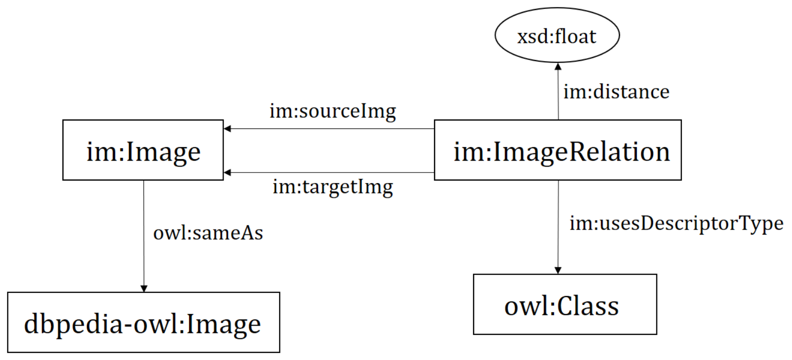


Figure 5.2: IMGpedia ontology for similarity links.

5.2. Knowledge-base Structure

After the vocabulary for the knowledge-base of IMGpedia is agreed, we start structuring the raw data obtained in the image processing stage. At this point we have:

- The images of the Wikimedia Commons dataset
- The three visual descriptors (GHD, HOG and CLD) for each image with each vector in a different file.
- The results of the similarity search, for each descriptor, in CSV files.

We need to transform this raw data into an RDF graph, according to the ontology that was defined. To do so, we implemented multiple parsers that take the raw data and write the equivalent RDF information using Turtle [6] syntax for RDF. The first thing is to set up the visual entities of IMGpedia, i.e., the resources that would describe an actual image of the dataset, containing its width and height, a reference to the Wikimedia Commons media, and a reference to the DBpedia Commons URI of the resource that describes the same image. In Listing 5.1 we can see an example of the RDF for a visual entity written in Turtle.

```
@prefix imo: <http://imgpedia.dcc.uchile.cl/ontology#> .
@prefix imr: <http://imgpedia.dcc.uchile.cl/resource/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dbcr: <http://commons.dbpedia.org/resource/File:> .
@prefix wcf: <commons.wikimedia.org/wiki/File:>

imr:Quentin_Tarantino_by_Gage_Skidmore.jpg a imo:Image ;
    owl:sameAs dbcr:Quentin_Tarantino_by_Gage_Skidmore.jpg ;
    imo:width 400 ;
    imo:height 300 ;
    imo:fileURL wcf:Quentin_Tarantino_by_Gage_Skidmore.jpg .
```

Listing 5.1: RDF example of a visual entity

Later, we need to parse the descriptors for each image, stating the type of descriptor that it is, the image it describes and the value of the vector. We decided to model the vectors as strings, with the values of each component separated with commas. The name of the resources corresponds to the name of the image it describes, following with a dot and the name of the descriptor used; for example, `dog.jpg.HOG` will be the name of the resource about the HOG descriptor of an image named `dog.jpg`. In Listing 5.2 we show an example of an RDF file describing a descriptor, written in Turtle.

```
@prefix imo: <http://imgpedia.dcc.uchile.cl/ontology#> .
@prefix imr: <http://imgpedia.dcc.uchile.cl/resource/> .

imr:Quentin_Tarantino_by_Gage_Skidmore.jpg.GHD a imo:GHD ;
    imo:describes imr:Quentin_Tarantino_by_Gage_Skidmore.jpg ;
    imo:value "[ 0.34418711, 0.10582313, 0.05867421, ...]"
```

Listing 5.2: RDF example of a visual descriptor

Finally, it is necessary to express the 150 million Image Relations of IMGpedia, i.e., the results of the similarity search. To do so, we name the resource using the `md5` hash over the name of the source image, the target image, the distance and the name of the descriptor that was used and, to avoid the unlikely scenario in which two similarity relations collide in the hash, we concatenated the hash with the name of the visual descriptor involved in the relation. In Listing 5.3 we see an example of an image relation modelled in RDF according to IMGpedia ontology. If two images are related by an Image Relation, one as source image and the other as target, we will state that they are `similar` and thus we add the corresponding RDF triple to the graph.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix imo: <http://imgpedia.dcc.uchile.cl/ontology#> .
@prefix imr: <http://imgpedia.dcc.uchile.cl/resource/> .

imr:176147ac95660a4a49b535a06b3f3b30a78bd5d58c57d5260572cdce11f98ad4.HOG
    a imo:ImageRelation ;
    imo:sourceImage imr:Hopsten_Marktplatz_3.jpg ;
    imo:targetImage imr:Boze_Cialo-glowny.JPG ;
    imo:distance 1.219660e+01 ;
    imo:usesDescriptor imo:HOG .

imr:Hopsten_Marktplatz_3.jpg imo:similar imr:Boze_Cialo-glowny.JPG .
```

Listing 5.3: RDF example of an Image Relation

The parsing process takes a few days to be completed. We put several visual entities and their descriptors in a single file; many Image Relations were placed in the same file. The IMGpedia dataset thus consists of 8046 Turtle files containing all the triples following the structure shown in the previous examples, and has a total size of 488 Gigabytes. However, links to DBpedia resources are still missing from the graph; generating said triples will be addressed in the following section.

5.3. Links to DBpedia Resources

To effectively ask queries to IMGpedia we need more information than just the visual descriptors and the similarity links between images. It is necessary to know to which Wikipedia articles the image belongs to, so we can link our visual entity to the corresponding DBpedia or Wikidata resource, using the property `appearsIn` defined in the IMGpedia ontology in Section 5.1.

To obtain said links, we downloaded a dump of Wikipedia[1] in English: an SQL relational database containing all the information of the English version of Wikipedia as it was as of December 1st, 2015. According to the dump documentation[2], there are two relevant tables for our purpose: `page` and `imagelinks`. `page` contains metadata from a Wikipedia article, which are identified by their name; however, they also use a numeric unique identifier. `imagelinks` contains the identifier number of a page and the name of an image that is used within that `page`.

What is left to be done is to join the two tables and do the projection of the columns of interest: the name of the `page` and the name of the image of the `imagelink`. Taking into account the size of both tables, we use a distributed map-reduce join with Hadoop, which

[1] `https://dumps.wikimedia.org/enwiki/20151201`

[2] `https://phabricator.wikimedia.org/source/mediawiki/browse/master/maintenance/tables.sql`

splits the tables into buckets according to the hashing of the columns involved in the join and process each bucket in parallel on a separate node of execution. Finally, the resulting tuples are merged and thus we obtain a list of pairs (*article*, *image*).

At this point we need to decide whether to use DBpedia or Wikidata resources and link them to IMGpedia. As a first iteration of the project, we choose DBpedia because it is easier to find the corresponding resource about a given Wikipedia article. Performing the same task with Wikidata is more obscure, since the resources and properties are labeled with a unique code (for example, the resource referring to Quentin Tarantino is named `Q3772`), instead of the label of the article. Thus, linking to Wikidata requires another parsing and query step that is left as future work.

As a final step we filter from this list only the relevant images for this work, i.e., the images with `png` or `jpg` extension and then we parse the pairs to RDF triples using the `imo:appearsIn` object property. In Listing 5.4, we show a couple of RDF triples in Turtle that are present in our knowledge-base.

```
@prefix imo: <http://imgpedia.dcc.uchile.cl/ontology#>
@prefix im: <http://imgpedia.dcc.uchile.cl/resource/>
@prefix dbr: <http://dbpedia.org/resource/>

im:Chamomile_original_size.jpg imo:appearsIn dbr:Nephelium_hypoleucum .
im:Rose_Amber_Flush_20070601.jpg imo:appearsIn dbr:Nephelium_hypoleucum .
im:HondaS2000-004.png imo:appearsIn dbr:Alfa_Romeo_Scighera .
im:Rose_Amber_Flush_20070601.jpg imo:appearsIn dbr:Acer_shirasawanum .
```

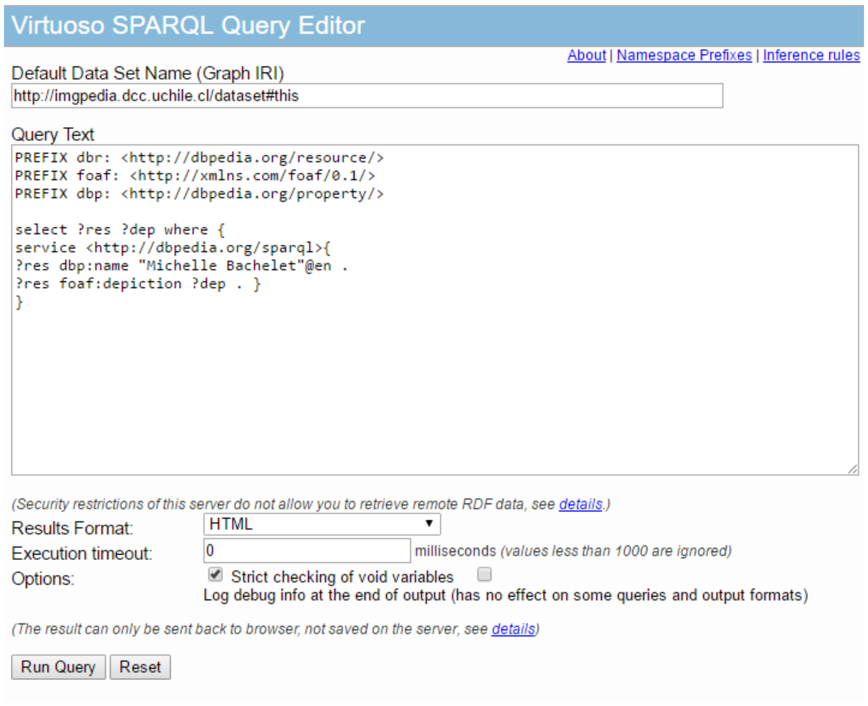
Listing 5.4: RDF example of links to DBpedia resources

5.4. SPARQL Endpoint

The data described in the previous section has to be uploaded to a public SPARQL endpoint in order to be queried by users and to become Open Linked Data. There are many SPARQL server managers that allow to load graphs and provide a user interface to ask queries. Virtuoso Open Source[3] was chosen because it can handle very massive data and provides a Bulk Loader that is useful for loading several RDF files in a fast and concurrent manner. We run the Bulk Loader in multiple threads, according to the specification in Virtuoso documentation[4]. The process of loading all the 8045 files took 79 hours using 10 execution threads and allowing Virtuoso to use 64 GB of RAM.

The server runs at `http://imgpedia.dcc.uchile.cl` and the endpoint can be accessed through the '`/sparql/`'path. The IMGpedia data is stored in the default graph; the IRI of the graph is `http://imgpedia.dcc.uchile.cl/dataset#this`. The user interface is provided by Virtuoso; an image showing the details can be found in Figure 5.3. There, we see a

[3]`https://github.com/openlink/virtuoso-opensource`
[4]`https://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtBulkRDFLoader`


Virtuoso SPARQL Query Editor
About | Namespace Prefixes | Inference rules

Default Data Set Name (Graph IRI)
http://imgpedia.dcc.uchile.cl/dataset#this

Query Text
```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbp: <http://dbpedia.org/property/>

select ?res ?dep where {
service <http://dbpedia.org/sparql>{
?res dbp:name "Michelle Bachelet"@en .
?res foaf:depiction ?dep . }
}
```

(Security restrictions of this server do not allow you to retrieve remote RDF data, see details.)

Results Format: HTML
Execution timeout: 0 milliseconds (values less than 1000 are ignored)
Options: ✓ Strict checking of void variables ☐ Log debug info at the end of output (has no effect on some queries and output formats)

(The result can only be sent back to browser, not saved on the server, see details)

Figure 5.3: Virtuoso endpoint query interface

text area for editing the SPARQL query, an input for the IRI of the graph, and some configuration settings that allow to change the answer format (HTML, XML, JSON) and set a query execution timeout.

Chapter 6

Results

IMGpedia is a large-scale knowledge-base containing visual similarity information. In this Chapter we review statistics about the dataset and we show some of the use-cases of the dataset through query examples.

IMGpedia is built over the 14,765,300 images available in the Wikimedia Commons dataset. In Table 6.1 we show relevant numbers about the data contained in the graph. First we show the number of instances of each class defined in IMGpedia ontology (Details in Chapter 5). Later, we show the total amount of instances, the total number of RDF triples and the number of links to DBpedia through the `imo:appearsIn` relation.

Class	Number of elements
imo:Image	14,765,300
imo:Descriptor	44,295,900
imo:ImageRelation	442,959,000
Instances	502,020,200
Triples	4,111,399,891
Links to DBpedia	12,683,423

Table 6.1: Number of resources of IMGpedia

The main results of this work are the various types of queries can be asked to IMGpedia. A basic query might be to request a specific visual descriptor for an image or obtaining all images that are similar to another one. In Listing 6.1 we can find a query example that requests the similar images of the image of Hopsten Marktplatz: this will retrieve the nearest neighbors found using any of the visual descriptors. In Table 6.2 we show the results retrieved by IMGpedia endpoint, limited to the first ten. In Figure 6.1 we show the corresponding images, retrieved with Listing 6.1.

```
PREFIX im: <http://imgpedia.dcc.uchile.cl/resource/>
PREFIX imo: <http://imgpedia.dcc.uchile.cl/ontology#>

SELECT DISTINCT ?Target
WHERE {
    im:Hopsten_Marktplatz_3.jpg imo:similar ?Target .
}
```

Listing 6.1: SPARQL query example over IMGPEDIA dataset

Target
im:Hopsten_Gustav_Lampe_Strasse_05.jpg
im:Liriodendron_tulipifera_01.JPG
im:Boze_Cialo-glowny.JPG
im:Koos_de_la_Rey.jpg
im:Nuernberg_Felsengaenge_010.JPG
im:Boxtown_Park_Memphis_TN_2012-12-30_015.jpg
im:Nový_Dvůr_(Písek)_(5.).jpg
im:Friedrich_Ernst_Wolfrom_Die_Überfahrt.jpg
im:TU_Dresden_76.jpg
im:2000_Selly_Oak.JPG

Table 6.2: Results of Listing 6.1 query

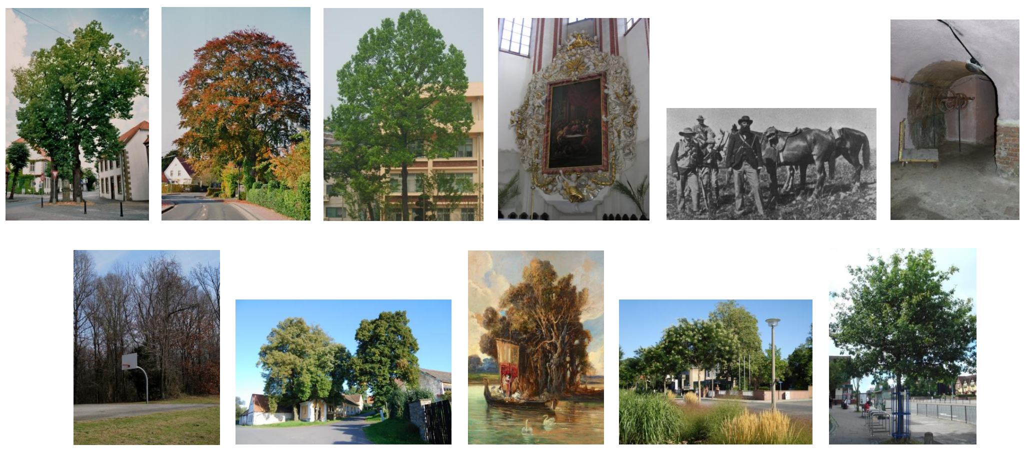


Figure 6.1: Results of Listing 6.1 query. The first image is the query image, the others are ordered according to Table 6.2, from left to right.

We could also request only the nearest neighbours that were found using a particular visual descriptor and sort them by the distance between the images. In Listing 6.2 we show such a query, requesting the nearest neighbours of the image of Hopsten Marktplatz, using HOG descriptor. In Table 6.3 are the results of the query. In Figure 6.2 we show the results graphically.

```
PREFIX im: <http://imgpedia.dcc.uchile.cl/resource/>
PREFIX imo: <http://imgpedia.dcc.uchile.cl/ontology#>

SELECT DISTINCT ?Target ?Distance
WHERE {
?rel imo:sourceImage im:Hopsten_Marktplatz_3.jpg ;
    imo:usesDescriptorType imo:HOG ;
    imo:targetImage ?Target ;
    imo:distance ?Distance .
}
ORDER BY ?Distance
```

Listing 6.2: SPARQL query example over IMGpedia dataset

Target	Distance
im:Mokotów_Gimnazjum_im.J.Piłsudskiego_i_dom_mieszkalny-_10.jpg	11.0907
im:Krippken_Mettingen_3.jpg	11.2779
im:PikiWiki_Israel_11675_Landscape_in_Moshav_Tsofit.JPG	11.6791
im:Mettingen_Prozessionshaeuschen_Nordhausen_3.jpg	11.9080
im:Nový_Dvůr_(Písek)_(5.).jpg	11.9844
im:Rauzan_Croix_de_chemin.jpg	11.9857
im:Recreation_in_Ishimbay.jpg	12.0976
im:2000_Selly_Oak.JPG	12.1016
im:Mishima-no-Ookeyaki024.jpg	12.1759
im:Cantabria_Santoña_casa_familia_Albo_lou.JPG	12.1966

Table 6.3: Results of Listing 6.2 query

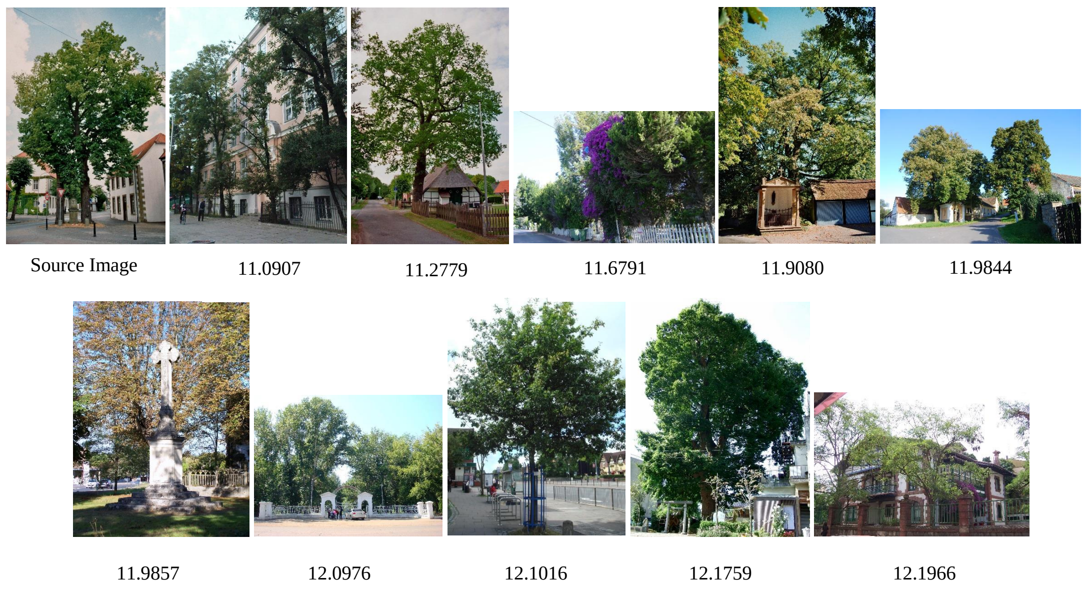


Figure 6.2: Results of Listing 6.2 query

In IMGpedia we can also try federated queries, in order to obtain the depiction of other articles of Wikipedia, using DBpedia or Wikidata, and merge them with the visual information. To do so, we simply use the relation `imo:appearsIn` that links our images with the DBpedia resource that uses it. For example, we can get the labels of the resources connected with images that are similar to the depiction of the Sugarloaf Mountain on Wikipedia. In Listing 6.3 we show the federated query that does so, requesting the labels to DBpedia within the `SERVICE` clause. The `OPTIONAL` clause is used, since not all images from the dataset are depictions of an article in Wikipedia so they will not appear in DBpedia extraction, even if the image is present elsewhere in the article. In Table 6.4 we can see the output of the query: labels with a language tag.

```
PREFIX imo: <http://imgpedia.dcc.uchile.cl/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?Img ?Label WHERE{
<http://imgpedia.dcc.uchile.cl/resource/PaodeAcucar.JPG> imo:similar ?Img .
?Img imo:appearsIn ?Res .
SERVICE <http://dbpedia.org/sparql>{
    ?Res rdfs:label ?Label .
}
FILTER(LANG(?Label)='en')
}
```

Listing 6.3: Federated SPARQL query example over IMGpedia dataset

Label
“Laxminarayan Temple”@en
“ArcelorMittal Orbit”@en
“Asagiri-class destroyer”@en
“Gifu Castle”@en
“Gran Sabana”@en

Table 6.4: Results of Listing 6.3 query

The results shown in Table 6.4 are poor, considering that most images just belong to Wikimedia Commons galleries and are not present in any Wikipedia article. Besides, there are images that are used in other language versions of Wikipedia, so the `imo:appearsIn` relation does not capture them. Also, there is the chance that the process of obtaining said relation is still not complete due to the image dataset being older than the dump. This process must continue being tested and improved.

As anticipated in Chapter 1, IMGpedia can also be used as a semantic image retrieval system, since we provide links to the Wikimedia Commons original media. We could retrieve the URLs of the images that appear in the paintings of the Louvre that were painted during the 16th century. In Listing 6.4 we show the SPARQL query that retrieves them, in Table 6.6 we show the results and in Figure 6.3 we show some of the pictures from the result set. Additional facts can be retrieved from DBpedia about the resources if needed; for example, we can retrieve the artist of the paintings using the `dbp:artist` object relation.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX yago: <http://dbpedia.org/class/yago/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>
PREFIX imo: <http://imgpedia.dcc.uchile.cl/ontology#>

SELECT ?url ?label WHERE{
    SERVICE <http://dbpedia.org/sparql>{
        ?res a yago:Wikicat16th-centuryPaintings ;
             dcterms:subject dbc:Paintings_of_the_Louvre ;
             rdfs:label ?label ;
        FILTER(LANG(?label)='en')
    }
    ?img imo:appearsIn ?res ;
         imo:fileURL ?url
}
ORDER BY(?label)
```

Listing 6.4: SPARQL query that uses IMGpedia as a semantic image retrieval system

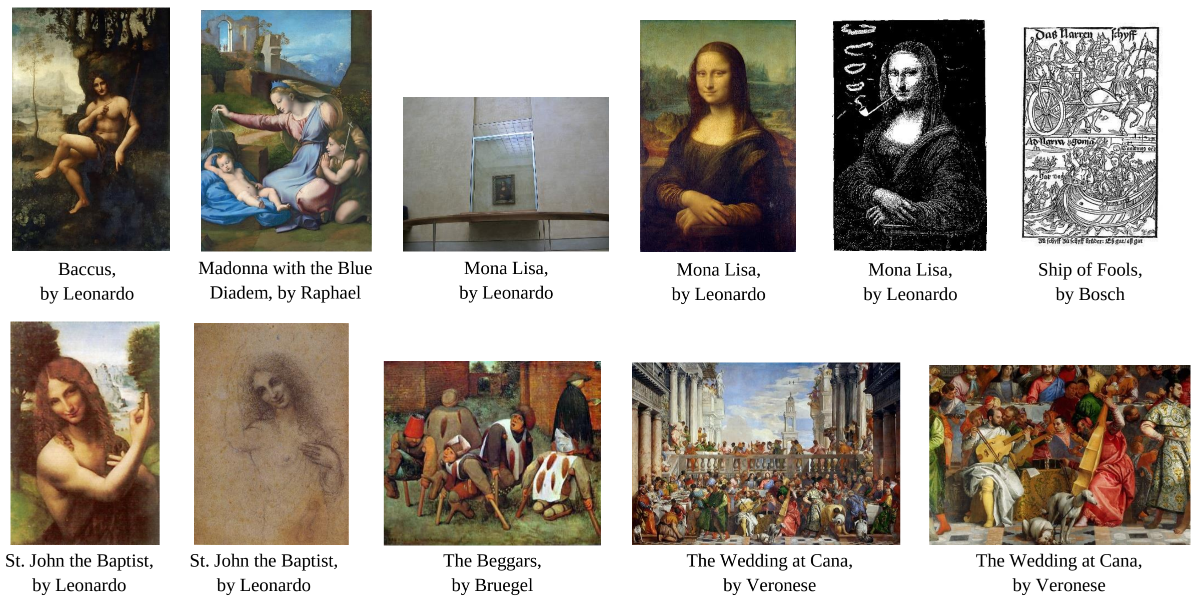


Figure 6.3: Some results of Listing 6.4 query. Images are labelled with the name of the Wikipedia article in which they appear and the name of the painter.

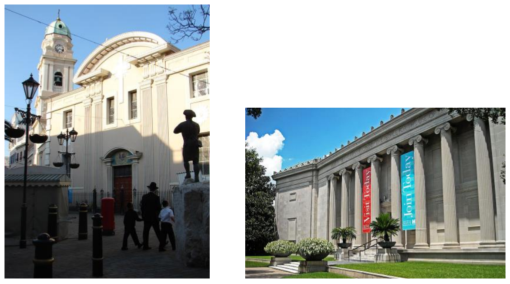
Finally, we can apply semantic filters to the results of a query and obtain similar images from different domains. For instance, in Listing 6.5, we show a SPARQL query that takes the images from articles categorized as "Roman Catholic cathedrals in Europe" and looks for the similar images that are categorized as "Museum". The results of the query are listed in Table 6.5 and in Figure 6.4 we show the retrieved images.

```
PREFIX im: <http://imgpedia.dcc.uchile.cl/resource/>
PREFIX imo: <http://imgpedia.dcc.uchile.cl/ontology#>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX yago: <http://dbpedia.org/class/yago/>
PREFIX dbc: <http://dbpedia.org/resource/Category:>

SELECT DISTINCT ?urls ?urlt WHERE{

    SERVICE <http://dbpedia.org/sparql>{
        ?sres dcterms:subject dbc:Roman_Catholic_cathedrals_in_Europe .
    }

    ?source imo:appearsIn ?sres ;
            imo:similar ?target ;
            imo:fileURL ?urls .
    ?target imo:appearsIn ?tres ;
            imo:fileURL ?urlt .

    SERVICE <http://dbpedia.org/sparql>{
        ?tres dcterms:subject ?sub
        FILTER(CONTAINS(STR(?sub), "Museum"))
    }
}
```

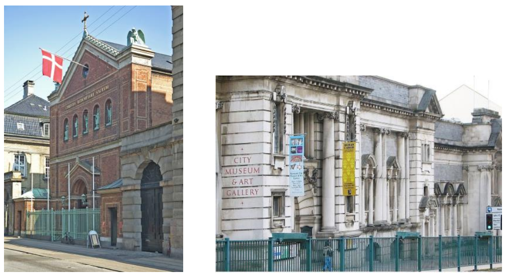
Listing 6.5: SPARQL query example over IMGpedia dataset using semantic filters

European Cathedral	Museum
wcf:Cathedral_of_St_Mary_ the_Crowned.JPG	wcf:MFA_houston.jpg
wcf:SanktAnsgar01.jpg	wcf:Plymouth_Museum_and_Art_ Gallery.jpg
wcf:Giovanni_Laterano_fc07.jpg	wcf:Helsingin_Luonnontieteellinen_ museo.JPG
wcf:Cathedral_of_St._Mary_ the_Crowned_courtyard_entrance.jpg	wcf:Dumbarton_House.jpg

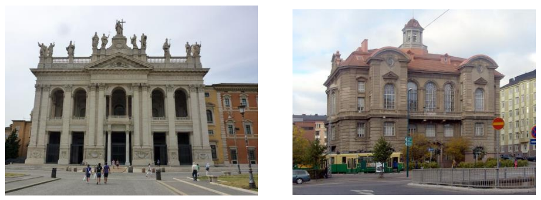
Table 6.5: Results of Listing 6.5 query



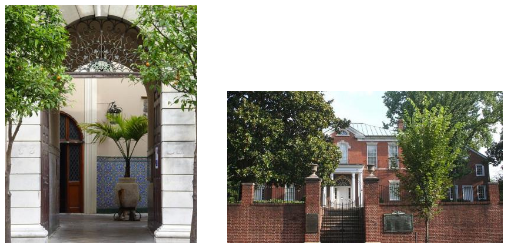
Cathedral of St. Mary and Museum of Fine Arts



St. Ansgar's Cathedral and Plymouth Museum



Basilica of St. John Lateran and Natural History Museum of Helsinki



Cathedral of St. Mary (Entrance) and Dumbarton House Museum

Figure 6.4: Results of Listing 6.5 query

There is room for improvement for the results of the last query (Listing 6.5), since we use all the images from WIKIMEDIA COMMONS instead of only using those that have links to DBPEDIA. Because of this, semantic facts cannot be retrieved for all the images present in similarity relationships; hence the results of the queries are limited to find such images that can be related to a resource on the Web of Data. However, these are promising results taking into account the objectives of this work, which are to perform *visuo-semantic* queries for the first time on the Web.

Commons URL	Wikipedia label
wcf:Dagger-14-plain.png	"Bacchus (Leonardo)"@en
wcf:Bacchus_(painting).jpg	"Bacchus (Leonardo)"@en
wcf:La_Vierge_au_voile,_by_Raffaello_Sanzio,_from_C2RMF_retouched.jpg	"Madonna with the Blue Diadem"@en
wcf:WP0.5_Icon.png	"Mona Lisa"@en
wcf:MonaLisaShield.jpg	"Mona Lisa"@en
wcf:Sapeck-La_Joconde_fumant_la_pipe.jpg	"Mona Lisa"@en
wcf:Dagger-14-plain.png	"Mona Lisa"@en
wcf:Mona_Lisa_(copy,_Thalwil,_Switzerland).JPG	"Mona Lisa"@en
wcf:Raffael_046.jpg	"Mona Lisa"@en
wcf:Isleworthml.JPG	"Mona Lisa"@en
wcf:Gioconda_(copia_del_Museo_del_Prado_restaurada).jpg	"Mona Lisa"@en
wcf:Mona_Lisa_stolen-1911.jpg	"Mona Lisa"@en
wcf:Mona_Lisa_margin_scribble.jpg	"Mona Lisa"@en
wcf:Mona_Lisa,_by_Leonardo_da_Vinci,_from_C2RMF_retouched.jpg	"Mona Lisa"@en
wcf:Mona_Lisa_(copy,_Hermitage).jpg	"Mona Lisa"@en
wcf:Narrenschiff.jpg	"Ship of Fools (painting)"@en
wcf:Dagger-14-plain.png	"St. John the Baptist (Leonardo)"@en
wcf:Leonardo_da_Vinci_-_Angelo_Incarnato.jpg	"St. John the Baptist (Leonardo)"@en
dbr:John_the_Baptist_-_Salai.jpg	"St. John the Baptist (Leonardo)"@en
wcf:Petro_Bruegel_Pictori.png	"The Beggars"@en
wcf:Pieter_Bruegel_the_Elder_-_The_Cripples.JPG	"The Beggars"@en
wcf:Veronese,_The_Marriage_at_Cana_(1563).jpg	"The Wedding at Cana"@en
wcf:Paolo_Veronese_-_The_Marriage_at_Cana_(detail)_-_WGA24859.jpg	"The Wedding at Cana"@en
wcf:Paolo_Veronese,_avtoportret.jpg	"The Wedding at Cana"@en
wcf:Paolo_Veronese_008.jpg	"The Wedding at Cana"@en
wcf:Suleiman_in_Veronese_The_Wedding_at_Cana_1563.jpg.jpg	"The Wedding at Cana"@en

Table 6.6: Results of Listing 6.4 query

Chapter 7

Conclusions

In this work, we have presented how we built the IMGpedia knowledge-base. This knowledge-base includes visual and similarity information about 14,765,300 images from the Wikimedia Commons dataset. IMGpedia allows to ask SPARQL queries combining the image information and semantic facts stored in other knowledge-bases such as DBpedia or Wikidata. In this chapter we discuss the main conclusions about the objectives of this work, the construction and relevance of the knowledge-base, as well as the results, and main contributions of this work. We end by addressing the future of IMGpedia, what can be improved, what has to be kept and what can be incorporated in to the knowledge-base.

In Chapter 1 we stated several objectives for this work. In this section, we will first address them in order to check if they were completed or not and to what extent.

Provide Reference Implementations for the Visual Descriptors The visual descriptors implemented are built using the OpenCV library (using any of the implementations in different programming languages). The numbers and some insight of the efficiency can be found in Section 4.2. These visual descriptors are freely available on the Web[1] as an effort to bring the Image Analysis process closer to the Semantic Web community and, in general, to bring Multimedia closer to the Web of Data.

Generate the IMGpedia Dataset The dataset of IMGpedia is composed of four main data sources: the images from Wikimedia Commons, the visual descriptors of the images, the links between the images and the Wikipedia articles in which they are used, and the similarity links between the images. All these sources of information are stored in the IMGpedia server and can be reused, updated and expanded. The update process requires the download of new images, the computation of the visual descriptors for those images and perform the similarity search for all the images of the dataset; the later join of Wikipedia dump tables can be done using just the new images and, finally, all the resulting new triples can be updated and/or inserted in the graph.

[1] `http://github.com/scferrada/imgpedia`

Find a Suitable way to Compute Static Similarity Relations In Chapter 4 we show different ways in which the problem of efficiently computing the similarity links for 15 million images can be solved. We found that FLANN is an efficient tool that computes the approximated nearest neighbors of large-scale datasets with a satisfying precision. However, further analysis can be made and exact methods will be tested in the future.

Represent the Dataset as RDF and Publish it as Linked Open Data We developed specific parsers to transform the raw data of the different sources (descriptors, similarity links, links to DBpedia) into the RDF standard [32], using Turtle syntax [6]. IMGpedia is part of Linked Open Data, since it is freely available on the Web and its resources are reachable through HTTP protocol using URIs.

Provide a Query Mechanism IMGpedia provides a public SPARQL endpoint[2] to ask SPARQL queries against our knowledge-base, combining both visual and semantic information. Some results of this endpoint are reviewed in Chapter 6.

7.1. Relevance and Contribution

According to the 5-star model for Linked Open Data by Berners-Lee [8], IMGpedia is a 5-star data source, since it is an RDF graph that uses IRIs to identify its resources and provides links to other data sources in order to provide context. IMGpedia contains links to Wikimedia Commons, joining the visual entities with the real-world image it represents; and to DBpedia, stating that a visual entity is used in a specific resource through the `imo:appearsIn` object property. There are 12,683,423 triples interlinking the images of IMGpedia with the resources of DBpedia that use them. However, the quality of said links still need to be improved, since currently they are being automatically obtained and must be refined and cleaned. Furthermore, we want to add links to Wikidata because their data is often richer than that stored in DBpedia, so it will allow us to ask queries with more expressive semantic filters. Additionally we will prefer to use only images that are used in at least one article, so the semantic links make sense.

Many kinds of queries can be asked to IMGpedia through its public SPARQL endpoint. These queries may refer to the retrieval of feature vectors of certain images, to find the nearest neighbors of a given image, and most importantly to combine visual queries with the semantic content of the resources that use the images. For the latter task, federated queries using `appearsIn` property can be helpful to add semantic filters over the visual results or to obtain related information for the images, enriching the results. In Chapter 6 we show some use-cases and results about querying IMGpedia.

The methodology defined in this work for image analysis can be applied in any other image dataset. The only requirement is that the dataset is large enough so similarity links make sense. However, linking to external resources has to be handled in a specific way depending

[2]`http://imgpedia.dcc.uchile.cl/sparql`

on the nature and provenance of the images. As defined in Chapter 1, the methodology is comprised of the following steps:

1. Locally store the images to be used.

2. Compute the visual descriptors for each image

3. For each image, compute the 10 nearest neighbors using a fixed amount of buckets and running a FLANN instance on each bucket. Finally, merge the answers keeping the 10 minimum distances.

4. Parse the visual entities, the descriptors and the similarity relations to RDF according to the IMGpedia ontology.

5. Select the source of semantic facts and provide links to it.

6. Provide a public SPARQL endpoint so users can ask queries and browse the data.

7.2. Future Work

The knowledge-base of IMGpedia, as it is presented in this work, is merely the start of a long term project. It is the foundation of many research tasks that can be performed, regarding the similarity search of multimedia content on the Web of Data. In this section, we will address some of the ideas, open research questions and potential work to improve and update IMGpedia in the future.

First of all, we believe that computing the DeCAF7 [16] descriptor can be a useful addition to IMGpedia. To do so, we must run our feature extractor over a GPU and measure the time difference against our CPU benchmark. Furthermore, it can be interesting to store the labels obtained with the neural network, and see how the classification of the content of the image can enrich the queries or extend the expressiveness of the questions that can be asked to IMGpedia. As was proposed by Ferrada et al. [17], we can label images with categories, types, entities, etc. Thus IMGpedia could serve as a useful resource for the multimedia community; e.g., since we are using Caffe framework for neural networks to compute DeCAF7, we could train a network using the DBpedia categories extracted for each image as labels, allowing us to automatically classify new images in the dataset or benchmark performance against other classifiers. We could also label images with the specific entities they contain using DBpedia/Wikidata based on the article(s) in which it appears: we may need to tread careful since the images related to DBpedia resources through `imo:appearsIn` property might not contain the subject of the resource but something related to it; e.g., in Barack Obama's Wikipedia article appears a map of the electoral results in which Obama himself does not appear. However, we could consider some heuristics to address this such as the location of the image in the article and some further content analysis.

One of the main contributions of this work is the image similarity graph, which is large-scale and sparse. However, further analysis and characterization can be performed over the graph: spectral analysis can be used to detect connected components and communities within the graph; centrality measures can be computed; we could research if there is any correlation between vertex similarity and image similarity; we could try to find shortest path between any two images, to see how one image can be converted into another through the similar images that connect them.

Restricting the similarity search only to pre-computed nearest neighbors over the whole dataset seems fine for a first iteration. However, we want to allow more flexible similarity queries in IMGpedia. To achieve that goal, we propose to extend SPARQL standards [22] so it can allow vectorial operations, such as runtime distance computations. These might be based on the works of iSPARQL [28] and SPARQL-MM [29] regarding similarity search, vector handling and multimedia queries in SPARQL. We also want to research if there are other possibilities for the nearest neighbors search beyond FLANN. We thus may compare the results obtained with FLANN against other libraries or other manual configurations of FLANN. Furthermore we want to compute exact nearest neighbors efficiently using pivots and to perform an analysis of the distance distribution, so an image can have a custom number of nearest neighbors instead of a fixed number. Along these lines we also want to add query-by-example features to IMGpedia, so people can upload an image and ask visuo-semantic queries about it.

Linking to DBpedia did not provide good examples of *visuo-semantic* queries. This may be due to errors in the extraction of the (`image`, `article`) pairs or to the poor structure of the facts in DBpedia. Wikidata facts are much more clean and rich, since they are curated by users, hence it can allow us to support more expressive, relevant queries. In this matter, we want to research ways in which we can find the relations between a Wikipedia article and the corresponding Wikidata resource.

Finally, to get more people using and browsing the data of IMGpedia we will develop a web application that allow them to review the images and its metadata and descriptors, to ask queries with syntax highlighting, and to obtain the results in a human-friendly manner rather than the plain HTML tables retrieved by Virtuoso.

Bibliography

[1] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434. Springer, 2001.

[2] Nasir Ahmed, T Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[3] Alo Allik, György Fazekas, and Mark Sandler. Ontological representation of audio features. In *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II*, pages 3–11, Cham, 2016. Springer International Publishing.

[4] Richard Arndt, Raphaël Troncy, Steffen Staab, Lynda Hardman, and Miroslav Vacura. Comm: Designing a well-founded multimedia ontology for the web. In *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings*, pages 30–43, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[5] Juan Manuel Barrios Núñez. Content-based video copy detection. *(Doctoral Thesis, University of Chile, Santiago, Chile). Retrieved from `http://repositorio.uchile.cl/handle/2250/115521`*, 2013.

[6] David Beckett, Tim Berners-Lee, and Eric Prud'hommeaux. Turtle-terse RDF triple language. *W3C Team Submission*, 14:7, 2008.

[7] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[8] Tim Berners-Lee. Linked Data. W3C Design Issues, July 2006, 2010.

[9] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[10] Dan Brickley, RV Guha, and Andrew Layman. Resource Description Framework (RDF) Schemas. W3C Recommendation, April 1998.

[11] Benjamin Bustos and Aidan Hogan. IMGpedia: A Proposal to Enrich DBpedia with Image Meta-Data. In *Proc. 9 Alberto Mendelzon International Workshop on Foundations*

of Data Management (AMW), pages 35–39, 2015.

[12] Shih-Fu Chang, Thomas Sikora, and Atul Purl. Overview of the mpeg-7 standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):688–695, Jun 2001.

[13] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM computing surveys (CSUR)*, 33(3):273–321, 2001.

[14] Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. SPARQL Protocol for RDF (W3C Recommendation 15 January 2008). *World Wide Web Consortium*, 2008.

[15] Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. Structured knowledge representation for image retrieval. *Journal of Artificial Intelligence Research*, 2002.

[16] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pages 647–655, 2014.

[17] Sebastián Ferrada, Benjamin Bustos, and Aidan Hogan. IMGpedia: Enriching the Web of Data with Image Content Analysis. In *Proc. 10 Alberto Mendelzon International Workshop on Foundations of Data Management (AMW)*, 2016.

[18] Jose Emilio Labra Gayo, Patricia Ordóñez de Pablos, and Juan Manuel Cueva Lovelle. Wesonet: Applying semantic web technologies and collaborative tagging to multimedia web information systems. *Computers in Human Behavior*, 26(2):205 – 209, 2010.

[19] Eirini Giannakidou, Ioannis Kompatsiaris, and Athena Vakali. Semsoc: Semantic, social and content-based clustering in multimedia collaborative tagging systems. In *IEEE International Conference on Semantic Computing*, pages 128–135. IEEE, 2008.

[20] Rafael C Gonzalez and Richard E Woods. *Digital image processing.* Prentice Hall, 3 edition, 2008.

[21] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, pages 47–57, New York, NY, USA, 1984. ACM.

[22] Steven Harris and Andy Seaborne. SPARQL 1.1 query language. W3C Recommendation, W3C, March 2013.

[23] Michael Hausenblas, Raphaël Troncy, Tobias Bürger, and Yves Raimond. Interlinking multimedia: How to apply linked data principles to multimedia fragments. In *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.*, 2009.

[24] Aidan Hogan. Linked data and the semantic web standards. *Linked Data and the Semantic Web Standards*, 2013.

[25] Bo Hu, Srinandan Dasmahapatra, Paul Lewis, and Nigel Shadbolt. Ontology-based medical image annotation with description logics. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pages 77–82. IEEE, 2003.

[26] Jane Hunter. Adding multimedia to the semantic web: Building an mpeg-7 ontology. In *Proceedings of the First International Conference on Semantic Web Working*, SWWS'01, pages 261–283, Aachen, Germany, Germany, 2001. CEUR-WS.org.

[27] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM.

[28] Christoph Kiefer, Abraham Bernstein, and Markus Stocker. The fundamentals of isparql: A virtual triple approach for similarity-based semantic web tasks. In *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings*, pages 295–309. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[29] Thomas Kurz, Kai Schlegel, and Harald Kosch. Enabling access to linked media with sparql-mm. In *Proceedings of the 24th International Conference on World Wide Web*, pages 721–726. ACM, 2015.

[30] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2014.

[31] Ching-Yung Lin, Belle L Tseng, and John R Smith. Video collaborative annotation forum: Establishing ground-truth labels on large multimedia datasets. In *Proceedings of the TRECVID 2003 Workshop*, 2003.

[32] Frank Manola, Eric Miller, and B McBride. RDF Primer. W3C Recommendation, February 2004, 2004.

[33] Deborah L McGuinness and Frank Van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation, February 2004, 2004.

[34] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[35] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.

[36] David Novak, Michal Batko, and Pavel Zezula. Large-scale image retrieval using neural net descriptors. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 1039–1040, New

York, NY, USA, 2015. ACM.

[37] Jens-Rainer Ohm, Leszek Cieplinski, Heon Jun Kim, Santhana Krishnamachari, B Manjunath, Dean S Messing, and Akio Yamada. The mpeg-7 color descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 2001.

[38] Rodrigo Paredes, Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. *Practical Construction of k-Nearest Neighbor Graphs in Metric Spaces*, pages 85–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[39] Alexandre Passant. DBREC - Music Recommendations Using DBpedia. In *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*, pages 209–224, 2010.

[40] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Recommendation, January 2008, 2008.

[41] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[42] Irwin Sobel and Gary Feldman. A 3×3 isotropic gradient operator for image processing. *Stanford Artificial Project*, 1968.

[43] Giorgos Stoilos, Giorgos Stamou, Vassilis Tzouvaras, Jeff Z Pan, and Ian Horrocks. A fuzzy description logic for multimedia knowledge representation. In *Proc. of the International Workshop on Multimedia and the Semantic Web*, pages 12–19, 2005.

[44] Umberto Straccia. Reasoning within fuzzy description logics. *J. Artif. Intell. Res.(JAIR)*, 14:137–166, 2001.

[45] OpenCV team. OpenCV 2.4 Documentation.

[46] Gaurav Vaidya, Dimitris Kontokostas, Magnus Knuth, Jens Lehmann, and Sebastian Hellmann. Dbpedia commons: Structured multimedia metadata from the wikimedia commons. In *The Semantic Web - ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, pages 281–289, Cham, 2015. Springer International Publishing.

[47] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September 2014.

[48] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

[49] Xiaofang Zhai, Lei Huang, and Zhifeng Xiao. Geo-spatial query based on extended SPARQL. In *2010 18th International Conference on Geoinformatics*, pages 1–4. IEEE, 2010.

Appendix A

IMGpedia ontology in Turtle

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix imo: <http://imgpedia.dcc.uchile.cl/ontology#> .

<> a owl:Ontology ;
    rdfs:label "The IMGpedia Ontology"@en ,
        "La Ontología de IMGpedia" .

imo:Image a owl:Class ;
        rdfs:label "Image"@en ,
            "Imagen"@es .

imo:Descriptor a owl:Class ;
        rdfs:label "Descriptor"@en ,
            "Descriptor"@es .

imo:CLD a owl:Class ;
        rdfs:subClassOf imo:Descriptor ;
        rdfs:label "Color Layout Descriptor"@en ,
            "Descriptor de la Distribución del Color"@es .

imo:GHD a owl:Class ;
        rdfs:subClassOf imo:Descriptor ;
        rdfs:label "Gray Histogram Descriptor"@en ,
            "Histograma de Grises"@es .

imo:HOG a owl:Class ;
        rdfs:subClassOf imo:Descriptor ;
        rdfs:label "Histogram of Oriented Gradient"@en ,
            "Histograma de Orientación del Gradiente"@es .
```

```
imo:ImageRelation a owl:Class ;
        rdfs:label "Image Relation"@en ,
            "Relación entre Imágenes"@es .

imo:similar a owl:ObjectProperty ;
        rdfs:domain imo:Image ;
        rdfs:range imo:Image ;
        rdfs:label "A similar image"@en ,
            "Una imagen similar"@es .

imo:nearCopy a owl:ObjectProperty ;
        rdfs:domain imo:Image ;
        rdfs:range imo:Image ;
        rdfs:label "Near copy of an image"@en ,
            "Copia cercana de una imagen"@es .

imo:height a owl:ObjectProperty ;
        rdfs:domain imo:Image ;
        rdfs:range xsd:float ;
        rdfs:label "Image height"@en ,
            "Altura de la imagen"@es .

imo:width a owl:ObjectProperty ;
        rdfs:domain imo:Image ;
        rdfs:range xsd:float ;
        rdfs:label "Image width"@en ,
            "Ancho de la imagen"@es .

imo:appearsIn a owl:ObjectProperty ;
        rdfs:domain imo:Image ;
        rdfs:range owl:Thing ;
        rdfs:label "Tha image appears in the resource"@en ,
            "La imagen aparece en el recurso"@es .

imo:fileURL a owl:ObjectProperty ;
        rdfs:domain imo:Image ;
        rdfs:label "The URL of the image in Wikimedia Commons"@en ,
            "La URL de la imagen en Wikimedia Commons"@es .

imo:describes a owl:ObjectProperty ;
        rdfs:domain imo:Descriptor ;
        rdfs:range imo:Image ;
        rdfs:label "Describes an image"@en ,
            "Descriptor de una imagen"@es .

imo:value a owl:ObjectProperty ;
```

```
        rdfs:domain imo:Descriptor ;
        rdfs:range xsd:string ;
        rdfs:label "Descriptor value"@en ,
            "Valor del descriptor"@es .

imo:sourceImg a owl:ObjectProperty ;
        rdfs:domain imo:ImageRelation ;
        rdfs:range imo:Image ;
        rdfs:label "The source of an image relation"@en ,
            "El sujeto de la relación entre imágenes"@es .

imo:targetImg a owl:ObjectProperty ;
        rdfs:domain imo:ImageRelation ;
        rdfs:range imo:Image ;
        rdfs:label "The target of an image relation"@en ,
            "El objeto de la relación entre imágenes"@es .

imo:distance a owl:ObjectProperty ;
        rdfs:domain imo:ImageRelation ;
        rdfs:range xsd:float ;
        rdfs:label "The distance between the images in the relation"@en ,
            "La distancia entre las imágenes de la relación"@es .

imo:usesDescriptor a owl:ObjectProperty ;
        rdfs:domain imo:ImageRelation ;
        rdfs:range im:Descriptor ;
        rdfs:label "The descriptor used in the relation"@en ,
            "El descriptor usado en la relación"@es .
```

Listing A.1: IMGPEDIA ontology in Turtle language for RDF