12-2020

PPMExplorer: Using Information Retrieval, Computer Vision and Transfer Learning Methods to Index and Explore Images of Pompeii

Cindy Roullet
University of Arkansas, Fayetteville

PPMExplorer: Using Information Retrieval, Computer Vision and Transfer Learning Methods to Index and Explore Images of Pompeii

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a concentration in Computer Science

by

Cindy Roullet
Institut National des Sciences Appliquées de Rouen
Bachelor of Science in Computer Science, 2011
Institut National des Sciences Appliquées de Rouen
Master of Science in Computer Science, 2015

December 2020
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

John Gauch, Ph.D. Dissertation Director	Wing Ning Li, Ph.D. Committee Member
Khoa Luu, Ph.D. Committee Member	David Fredrick, Ph.D. Committee Member

Abstract

In this dissertation, we present and analyze the technology used in the making of *PPMExplorer*: Search, Find, and Explore Pompeii. *PPMExplorer* is a software tool made with data extracted from the *Pompei: Pitture e Mosaic* (PPM) volumes. PPM is a valuable set of volumes containing 20,000 historical annotated images of the archaeological site of Pompeii, Italy accompanied by extensive captions. We transformed the volumes from paper, to digital, to searchable. *PPMExplorer* enables archaeologist researchers to conduct and check hypotheses on historical findings. We present a theory that such a concept is possible by leveraging computer generated correlations between artifacts using image data, text data, and a combination of both. The acquisition and interconnection of the data are proposed and executed using image processing, natural language processing, data mining, and machine learning methods.

Acknowledgment

First, to my advisor, Dr. John Gauch. Thank you for your time, support, and trust and for being such an outstanding advisor. I am lucky to have had to opportunity to work with you! Thank you for your guidance and wise advice. I was always leaving our meetings with clearer ideas and concrete objectives. Thank you for your kindness and for making sure I was not alone on Thanksgivings; you have made this Ph.D. a blast!

To Dr. David Fredrick, thank you for making this Ph.D. possible and for welcoming me into the Tesseract family. Thank you for taking a chance on me and for serving on my committee. Thank you for your time and mentorship! I admire your vision and I am honored to have had the chance to work with you. I look forward to continuing our collaborations and discussions in the future.

To Dr. Rhodora Vennarucci, thank you for being so helpful and encouraging and for both a fruitful and enjoyable collaboration. I look forward to our future research together!

To Dr. Khoa Luu, thank you for your time serving on my committee, for your advice on the dissertation, and for a pleasant collaboration.

To Dr. Wing Ning Li, thank you for your time in serving on my committee, your advice, and for your thoughtful questions.

Thank you to all the volunteers who took the time to try *PPMExplorer* at its early stage and who answered my questions: Will Loder, Dr. Matthew Notarian, Dr. Rhodora Vennarucci, Dr. David Fredrick, Dr. Jeremy Hartnett, Dr. Rebecca Benefiel, Dr. Bettina Bergmann, Dr. Verity Platt and, Mikaela Hamilton. Your feedback was very valuable and insightful.

Thank you to the library of the University of Arkansas for taking the time to manually scan all the PPM volumes. Thank you so much for your hard work and patience!

To my parents, Philippe Roullet, Marie-José, and Gilles Szilas thank you for your unconditional love and support throughout these years. I am very fortunate to have such wonderful parents who are always supporting and encouraging me to pursue my dreams, even if it means an ocean must be between us... You have given me the means and power to aim high and to reach my goals. I am forever grateful. Thank you for believing in me!

To my sister, thank you for being such an amazing role model... Even though you are the youngest…You have taught me more than how to cook pasta, you have helped me gain confidence in myself more than once, and you were always there to encourage me during the hard times. Thank you for being an amazing sister and always having my back!

To David Philips, thank you for listening to my many practices talks and helping me with my English writing and speaking. You are always so patient and curious; I love discussing with you and brainstorming ideas. You have been there for me during the good and bad times and helped me on so many occasions, words cannot express enough how grateful I am for our friendship.

To Will Loder, thank you for all the good times, for the epic adventures, for being an amazing friend and an outstanding colleague! Thank you for your support and encouragement, especially during stressful times, you always knew what to say to make me feel better. You have helped me to pause, breathe, and enjoy life more. You do make life more fun might I add!

To those that were especially close friends throughout my Ph.D., I am so

thankful for your time, support, and friendship: Kevin Labille, Julio Nolasco, Gustavo Samor, Matt Faries, Brandon Dorsey, Ryan Renfro (In Memoriam), Joel Mandebi, Joe Sirrianni, Mark Mondier, Ervin Moeckel, Sara Hami, Sam Harrison, Charity Smith and Kelsey Myers thank you!

Content

1 Introduction

On a seemingly ordinary day in 79 AD, the Roman city of Pompeii was going about business as usual. Despite a powerful earthquake which shook the region in 62 CE, from which Pompeii was still rebuilding, little did the people know, at nearby Mount Vesuvius, volcanic pressure building beneath the surface was about to unleash catastrophe upon Pompeii and other cities on the plains of Campania. The sleeping giant awoke and blew its top, spewing ash, rocks, and mud into the atmosphere. Possibly 16,000 people tragically lost their lives as they tried to escape with nowhere to go. They were suffocated by the dense clouds of ash and buried under the torrential downfall of volcanic debris. Shortly after, the entire city of Pompeii was entombed in millions of tons of volcanic ash and pumice layers.

1.1 Motivation

While the ancient inhabitants who survived did tunnel into the buried city searching for valuables in the years immediately after the eruption, after this Pompeii was a total loss, it was even erased from maps, and as the years went by, the city was slowly forgotten. It would remain buried and lost to ash and time until 1766. The re-discovery of the ancient city finally brought about some light in shadows of horror and catastrophe suffered by the people of Pompeii. The ruins were unlike any ordinary archaeological find as rapid burial from a volcanic eruption had preserved artifacts by shielding them from air and moisture. In fact, the city had been submerged so quickly and was so well preserved, it seemed almost as if the last moments of Pompeii were frozen in time.

Today, the ruins of Pompeii are a source of many treasures that can provide historians and archaeologists with insight into what everyday life would have been like 2,000 years ago.

Left untouched and unseen for centuries, despite the damage caused by exposure and the crush of tourists, it remains one of the largest and most well-preserved archeological sites known to be in existence. Now the biggest challenge is the preservation of this wealth, 66 hectares of vital information. For the past 200 years, as it has been unearthed, Pompeii has been open to the public. Although approximately one third of the area within the city walls remains unexcavated, being opened to the public has unfortunately degraded the quality of the city's preservation. It also has been subjected to vandalism, wartime bombing, even earthquakes and subsequent eruptions from Vesuvius, which to this day remains an active volcano. The other problem is simply exposure. Now that structures and artifacts have been unearthed, they are no longer protected from the elements. As more time goes by, the degradation becomes worse.

1.2 Driving Problem

However, there is a silver lining. Back when the excavation of the site was at its peak, archeologists and historians recorded what they found as they uncovered the city. Originally, these were drawings, lithographs, and watercolors, but by the late 19th century this record began to include many photographs. These pictures have the benefit of representing the archaeological artifacts in their best-preserved state. *Pompei: Pitture e Mosaici* (PPM) is a collection of books, made in the 1990s, which inventories this set of visual evidence, from drawings to black and white and color photographs from the 1870s accompanied by extensive captions. It also includes hand drawings that illustrate reconstructed art and as well as accurate geolocation annotations almost all date before 1900. The University of Arkansas' library is fortunate enough to have 10 volumes of *Pompei: Pitture e Mosaic,* containing thousands of photographs, illustrations, and other valuable pieces of information. Unfortunately, however, the sheer amount of information makes it extremely challenging for one person to effectively utilize the wealth of data as it is

distributed across printed pages and lacks an index. If relevant connections between artifacts and data contained in the volumes can be made, these connections can reveal pertinent clues to help understand what the ancient culture was like. the collection of *Pompei: Pitture e Mosaic* is a wealth of knowledge but working with it in its printed form would be very cumbersome for an individual or even a small team.

Our work is aimed at examining the possibility and means of creating a system to ease the usability of PPM. We study different methods such as image processing and Optical Character Recognition (OCR) to extract and collect data from the digitized form of PPM. We created an annotated database with the acquired data. We also performed analysis on the data using machine learning methods such as feature extractions with Convolutional Neural Networks (CNN) and Natural Language Processing (NLP), classification with the retraining of a CNN and K-Nearest Neighbors (KNN) using mainly the data from volume 1 and 2. From the analysis results, we developed an interactive software search tool allowing users to explore and make relevant connections between artifacts and data from PPM in a new way. We named this tool *PPMExplorer*. Among studying the various approaches used for the tool creation, this dissertation describes how to multiple technologies can merge to deliver a unique product. We have chosen to create the tool with the game engine Unity. Unity is a very powerful and versatile tool for the development of projects and make it easy to deploy to multiple platforms. Using Unity will give us freedom to build *PPMExplorer* as a standalone application or a WebGL application for the web.

1.4 Organization

The remainder of this dissertation is organized in two main parts, which can be summarized with the diagram in Figure 1. *PPMExplorer* will often be referred as "the tool", "the software research tool", or "interactive software tool".

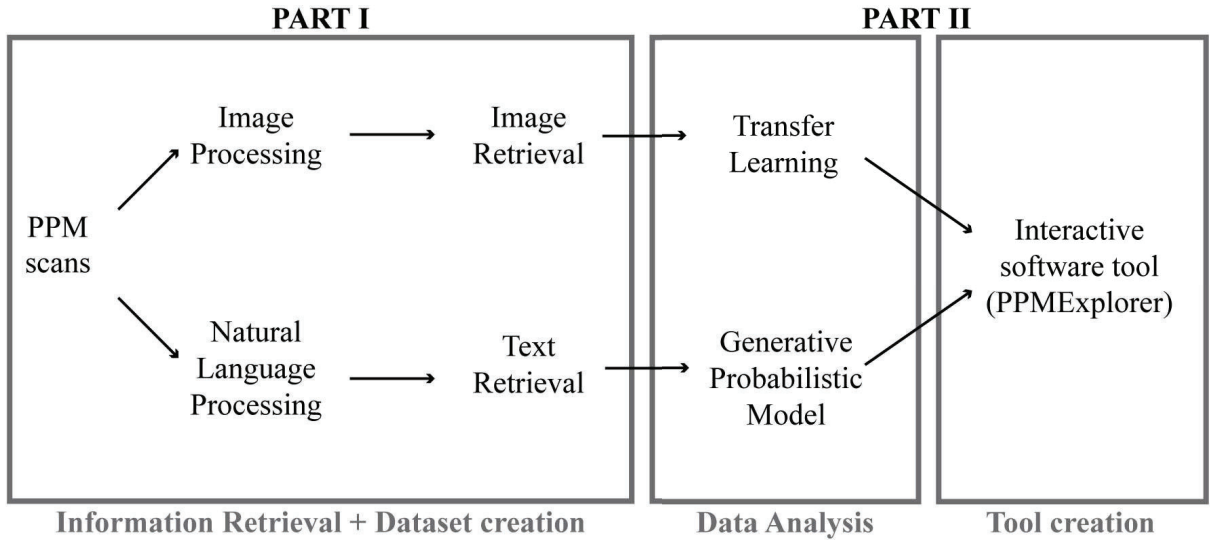

Figure 1: Dissertation Organization. Diagram of the dissertation organization. It is organized in two main parts. Part I contains the elements from the "Information Retrieval + Dataset creation". Part II contains the elements from the two boxes "Data Analysis" and "Tool creation".

1.4.1 Part I: Data Collection

In Chapter 2, we provide necessary background on Image Processing and Computer Vision related to archeology as well as Machine Learning and Natural language processing. Chapter 3 is dedicated to the extraction of the image data from PPM. Chapter 4 describes Image Processing, Optical Character Recognition (OCR) and text processing steps needed to extract the PPM caption text. The database that links image and text together is described in Chapter 5.

1.4.2 Part II: Data analysis

In Chapter 6, we outline the design and implementation of the word search capabilities of *PPMExplorer* as well as a caption similarity calculation. Chapter 7 describe the research conducted to implement the image similarity functionality. Chapter 8 presents how we implemented a similarity search taking information collected from Chapter 6 and 7. Before drawing conclusions, Chapter 9 describes the experiment we conducted with experts in Roman archaeology and art history. It also evaluates the tool and discusses research questions and hypotheses.

PART I: DATA COLLECTION

2 Background

Over the past decade, computer vision and machine learning techniques have proven their usefulness to explore the past. These visualization and computational methods have contributed to the support of archeologists in their exploration in the field, interpretation of discovery and conservation of new findings.

2.1 Computer vision and archeology

Thanks to the increased digitization of historical documents all over the world, we are able to present and explore historical artifacts in innovative ways with new technologies. Algorithms inspired by the structure and function of the brain called artificial neural networks, called Deep Learning algorithms, have brought us many opportunities. Transfer learning technologies, a subset of Deep Learning, have been used in recognizing Batik fabrics from Indonesia [38], generating labels for non-annotated historical images [35], and digitally archiving historical documents [55]. Architecture style recognition and classification of building facades in India [45] has also been possible with transfer learning. Image segmentation has been used to automatically recognize artistic influence of paintings [50]. The RANSAC algorithm has been tested as a support for analysis and interpretation of old paintings [56]. Reorganizing museums by similar objects instead of themes [13] using text similarity has been explored and tested in Europe. Deep learning methods have been utilized to recreate paintings and to restore damaged ones [43]. These are some of the few innovative applications that Deep Learning facilitates. With the help of interactive interfaces, we are able to present data retrieved with this new technology through applications.

Not only do these interactive tools allow information to be shared among more people, but they also present opportunities to improve the technology. Applications using CNNs trained on data can collect user inputs and enhance the tool as the more users interact with them; for instance, when users manually add labels to unlabeled data or upload their own images to databases [45]. Interactive tools can also generate information from one form to another thereby making it accessible to sight or hearing-impaired users (audio file, image, text) [52]. Utilizing mobile apps for quick and easy access to cultural heritage information such as monument recognition [47] or artwork recognition [25] is also a field in development.

In the field, it is very important for archaeological research to accurately record, map, and to monitor each successive layer of the excavation (stratigraphy), and to record each artifact and its location within the site map and the stratigraphic unit. In addition, the expenditure of time is a valuable factor for archaeologists. In modern times, it is not efficient to solely rely upon the traditional time-consuming techniques of mapping excavation sites. Technologies like photogrammetry [1], Geographic Information System (GIS) data collecting [12], on-site laser scanning of structures (remote sensing) [39], [12], [16] assist in the day to day work of archaeologists who must document their excavation and record all they encounter. Geolocalized data is very valuable, but difficult to manage and interpret. The lack of consistency in (spatial) data standards greatly inhibits the ability to develop sustainable solutions for managing, sharing, and analysis of that data [46]. In our project, we have at our disposal, a handmade accurate and annotated map of the Pompeii site. This will enable us to determine the spatial similarity between artifacts in the future, not simply by what was found literally where, but what kinds of artifacts tend to be found in what kinds of spaces.

Surveying, mapping, and other digital documenting techniques are not only applicable for geographical information retrieval, but they can also be utilized for 3D reconstruction which will aid in performing further analysis of artifacts (defragmented or not) and even presenting findings to the public without taking the risk of damaging the original artifacts. It is common for archaeologists to find artifacts that are in a broken state or have missing pieces, therefore automatic reconstruction methods are of great interest in archaeology. A common process to reconstruct the puzzle of an artifact is composed of two main steps: classifying or grouping the fragments and then assembling them. The classification involves feature extraction such as color and texture of the surface [40] or curve patterns [34]. The second stage of matching elements can be done by extracting different features such as 3D information ([36], [40]), aligning them and matching them by calculating similarities between features [40], curve pattern matching algorithms [8], or using Deep Learning methods such as neural networks ([40], [33], [36], [41], [39]).

Collection Management and cataloging are also included in the best practices to adopt and maintain cultural heritage. The classification of artifacts is usually performed by visual inspection. Unfortunately, the metadata often associated with images of the artifacts is sparse or inconsistent. Therefore, there is a research interest in developing tools to assist in labeling tasks [3]. Then there comes the challenge of finding specific artifacts and extracting the wealth of relevant information from thousands of images for research. This is not something that can be reasonably processed by hand. Enhancing metadata on artifacts has been explored using various methods such as extracting image-based features suitable for representing characteristic to classify objects according to the eyes of experts. These features include shape and texture ([30], [4]) for rocks, coins or other biface objects and Scale Invariant Features Descriptors such as

SIFT descriptors [33], which are image descriptors for image-based matching and recognition that are invariant to scale or rotation changes. SIFT have been useful to recognize monuments and determining extra description labels [41] for various excavation site pictures that can be added to the metadata. The more detailed and specific these descriptors are, the more helpful they can be.

Sorting, classifying, and constructing accurate metadata labels are crucial to the analysis and interpretation of findings as a whole. Most recent methods to solve all the above problems use neural networks that keep growing in popularity.

2.2 Convolutional Neural Network

When it comes to classifying images, a lot of research uses Convolutional Neural Networks (CNN). CNN is a neural architecture for Deep Learning which learns directly from images. Just like the other Neural Networks, it consists of several layers that process and transform an input to produce an output. A CNN can be trained to classify images [49] but it can also be trained for object detection [51] [56] and image segmentation [50] [54]. In a typical neural network, each neuron in the input layer is connected to a neuron in a hidden layer. In a CNN, a small region of the input layer neurons connects to a neuron in a hidden layer. These regions are called local receptor fields. Each receptor field is translated across an image to form a feature map of the image in the next hidden layer. This is implemented using a convolution of images, hence the name Convolutional Neural Network (CNN). This process is repeated across all the hidden layers. It can have 10 to 1,000+ hidden layers. Each layer learns to detect different features in the image, the deeper the layer the more complex the feature. For instance, one first layer could detect edges and the last one could detect circles. The last hidden layer is then

flattened, and every neuron is fully connected to the output neurons which produces the final output.

To make a Neural Network (NN), the weights and biases of the hidden layers need to be tuned to produce the best feature maps and therefore make good predictions. To do so, a NN needs to be trained, meaning it needs to learn what are the best weights and biases for each layer. CNN and NN learn the same way. The difference is that in a CNN, each neuron in one hidden layer has the same weight and bias. This is due to the fact that every neuron of one layer is detecting the same feature but at different regions of the image. In a typical neural network, each neuron has different weights and biases. It is possible to create and train a CNN from scratch [53]; it can be highly accurate but also very challenging as one might need more than hundreds of thousands of labeled images as well as significant computational resources. Another way to train a CNN is through transfer learning [29]. It is based on the idea that you can use knowledge of one type of problem to solve a similar problem. Transfer learning is usually done with large architecture CNNs that have be trained on millions of images. ImageNet [7] is one of the common datasets used to train and compare neural networks to each other. There are other such benchmark CNNs, for instance CIFAR-10 [8], CIFAR-100 [37], and SVHN [42]. ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories organized into 1,000 classes. The images were collected from the web and labeled by human using Amazon's Mechanical Turk crowd-sourcing tool [10].

CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class [24]. CIFAR-100 is just like the CIFAR-10, except it has 100 classes containing 600 images each.

There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 “super classes”. Each image comes with a ”fine” label (the class to which it belongs) and a ”coarse” label (the superclass to which it belongs) [23]. SVHN dataset contains different type of images from ImageNet and CIFARs. It contains over 600,000 images of numerical digits obtained from house numbers in Google Street View images. It is composed of 10 classes, one class for each digit. 73,257 digits for training, 26,032 digits for testing, and 531,131 additional, somewhat less difficult samples, to use as extra training data. SVHN is made to identify digits in natural scenes [9].

ImageNet is larger than the previous datasets described above. ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon’s Mechanical Turk crowd-sourcing tool [32].

One of the first winners of the ImageNet Large Scale Visual Recognition Competition (ILSVRC) was Alex Krizhevsky [32], who developed a very deep convolutional neural net. It has a total of 60 million parameters and 500,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and two globally connected layers with a final 1000 SoftMax layer which normalizes the results. His architecture is now called the AlexNet. The ILSVRC competitions use a subset of ImageNet with roughly 1,000 images in each of 1,000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images [32].

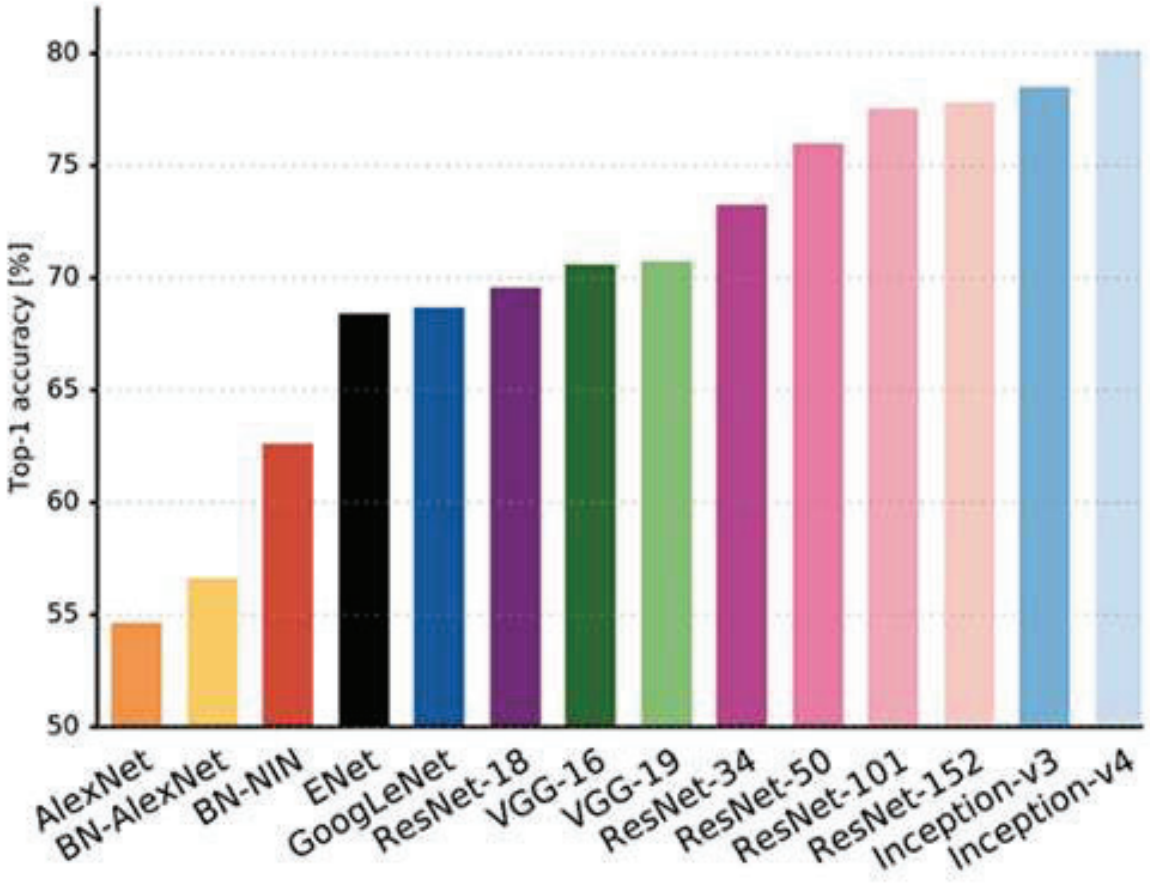

Figure 2: Model Architecture Scoring. Top1 vs. network. Single-crop top-1 validation accuracy for top scoring single-model architectures. We introduce with this chart our choice of color scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink. Picture credit [28].

On the test data of LSVRC-2012, AlexNet achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Multiple other models have been created using CNN as their backbone architecture that have achieved excellent results in ImageNet: ZFNet [21], GoogLeNet [22], VGGNet [19], ResNet [26], DenseNet [31]. GoogLeNet has another name, Inception V1. Since 2013 it has been the father of Inception V2, inception V3 (used in the pilot work, see chapter 7) and now

Inception V4. These multiple neural networks differ in their architecture. In other words, they have different numbers of layers, different number of neurons as well as different activation functions and different connections between neurons. Figure 2 shows one-crop accuracies of the most relevant entries submitted to the ImageNet challenge, from the AlexNet [10], on the far left, to the best performing Inception-v4 [28].

2.3 Natural Language Processing

To perform word similarity calculation, we would need to transform the words into a numerical representation (word embedding) which usually is in the form of a vector. The distance between the two vectors will inform us of how similar the two words are.

For instance, Word2Vec [14] is a two layered neural network trained on a very large corpus of words that can take a word as input and produces a vector as output. Other approaches to convert words into numbers are count-based methods. One popular algorithm for a count-based method is called GloVe: Global Vectors for Word Representation [18]. It creates a matrix that represents words by features. To get this representation, it first constructs a large co-occurrence matrix of words by context. For each word, it counts how frequently that word appears in a certain context (class or category).

Another common method is Latent Dirichlet Allocation (LDA) [44]. LDA states that each document in a corpus is a combination of a fixed number of topics. A topic has a probability of generating various words, where the words are all the observed words in the corpus. These ‘hidden’ topics are then surfaced based on the likelihood of word co-occurrence. This allows for the creation of a lower-dimensional space that best discriminates the samples from different classes [5]. Regarding text classification, after having a numerical representation

of the text, it is possible to use a CNN. It would work the same way as described above but instead of matrices of pixels as an input, it could be a matrix that represents a sentence where each row is a word represented as a vector. This could then detect spatial patterns (groups of words) in the text [27]. Other methods for text classification use typical neural networks [15] or common machine learning algorithms like SVM [5], Naive Bayes ([6], [2]) or KNN ([20],[17]).

2.4 Notations

CNN: Convolutional Neural Network

KNN: K-Nearest Neighbor

LDA: Latent Dirichlet Allocation

NN: Neural Network

OCR: Optical Character Recognition

PPM: *Pompei: Pitture e Mosaici*

SVM: Support Vector Machine

3 Image extraction

This chapter describes the information retrieval process of the images contained in the scanned version of *Pompei: Pitture e Mosaici* (PPM).

The pages of PPM are composed of text and images. By observing the page layout of PPM, we noticed:

- The images are on top of a white page background.
- The images can be black and white or colored.
- The images vary in sizes.
- Images are not framed.
- The number of images on the page can vary from 0 to 8.

Figure 3 describes multiple examples of page layouts. From these observations, we developed an automatic way to detect and crop out the images from the pages using heuristic image processing methods.

3.1 Initial Background/foreground segmentation

One way to separate the foreground (images and text) and background (PPM page background) is to do an image binarization. From the grayscale scan image of a page, we tried using a binarization using a threshold. It puts all the background pixels to 0 and all the foreground pixel to 255.

$$new_{pixel_{value}} = \begin{cases} 0 & if\ pixel_{value} > threshold \\ 255 & otherwise \end{cases} \quad (1)$$

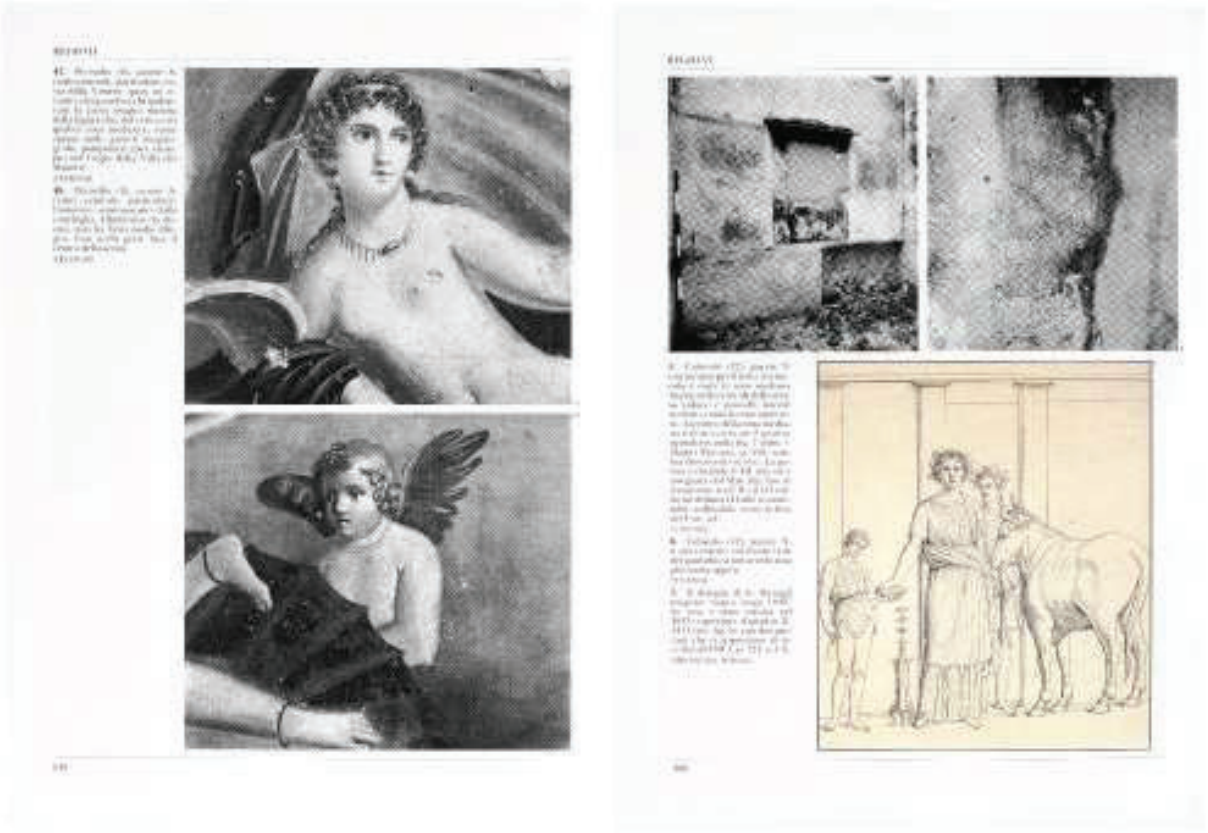

Figure 3: Page Examples. Two examples of scan pages. The one on the left has two black and white images, the one on the right has two black and white images and a color sketch image at the bottom.

Equation (1) describes the binarization function using a threshold. The new pixel value will either be 0 or 255 depending if it is higher or lower than the threshold. Using a threshold of 220 works well on images that have a uniform page light and little scan noise (see Figure 4).

However, on a lot of images this method was not efficient, see Figure 5. Most of the pixels of the bottom image are not being detected as foreground. The problem is that every scanned image has a slightly different global lightning and different local lightning according to how they were placed on the scanner. The ideal global threshold that would extract most of the background is different for each page. Utilizing an adaptive threshold function to create the

Figure 4: Threshold Binarization, Successful Example. Example of a binarization putting all the background pixel to 0 (any pixel with a higher value than 220) and the foreground to white. On the left side is the original page and the binarization result is on the right.

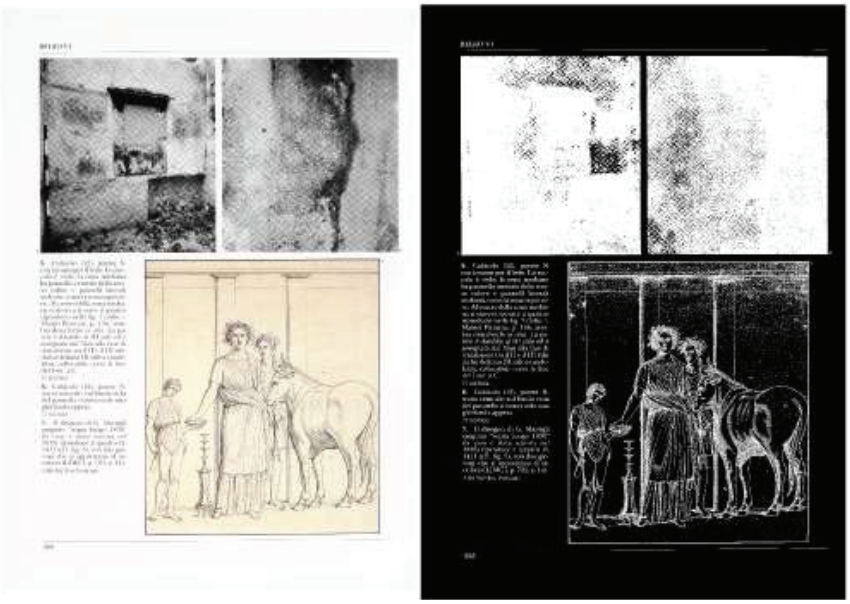

Figure 5: Threshold Binarization - Unsuccessful Example. Example of a binarization putting all the background pixel to 0 (any pixel with a higher value than 220) and the foreground to white. On the left side is the original page and the binarization result is on the right.

binarization improved the results. It calculates a different threshold value for different regions of the page image. The threshold is calculated as a weighted sum of a 3*3 neighborhood of the pixel minus a C value. The weights are a Gaussian window, and the C value is 2 (the most common value used with the adaptive threshold function).

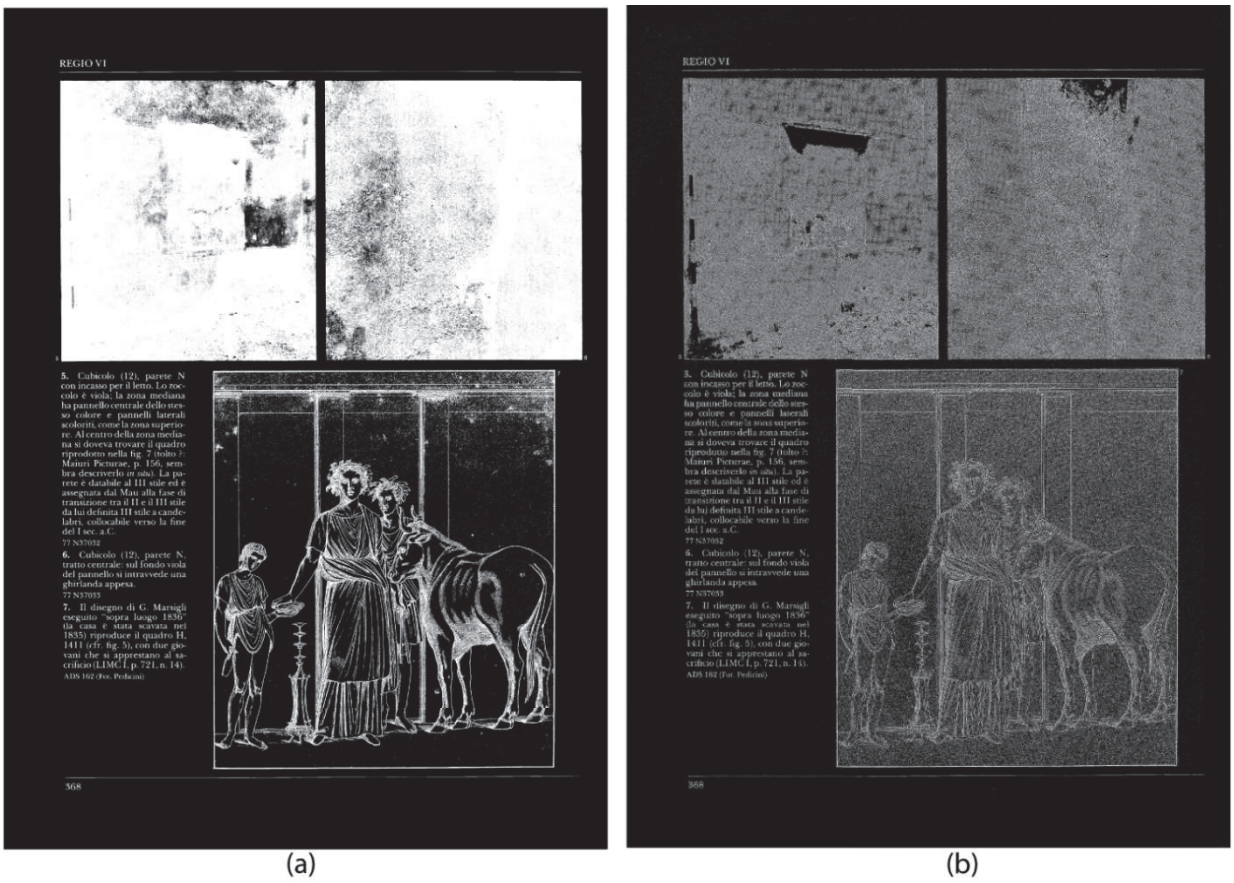

Figure 6: Difference between Threshold and Adaptive Binarization. Two image results of the same page using two different binarization method. (a) is a regular binarization threshold function, (b) is the result of the adaptive threshold function.

The window (Gaussian combined with the C value) is passed through the entire page, then for each window a threshold is calculated, and it is used to binarize the page. Figure 6 shows the result on page from Figure 5 using the regular threshold function (a) and using the adaptive threshold function (b). It is evident that many more foreground pixels on the bottom

image were detected that were not detected with the regular threshold function. On the other hand, the background accuracy went down as more pixels from background were detected as foreground due to the small window size of the threshold function. The same, but reversed, can be observed in the two top images. They had more accuracy on foreground detection with the regular method. Overall, it still improved the results and brought homogeneity to the detection on the entire page. Ergo, the adaptive threshold function created noise on the background, applying a median blur of size 3*3 removed a lot of it without jeopardizing the foreground pixels. Figure 7 shows a zoomed image of the effect of the median blur.

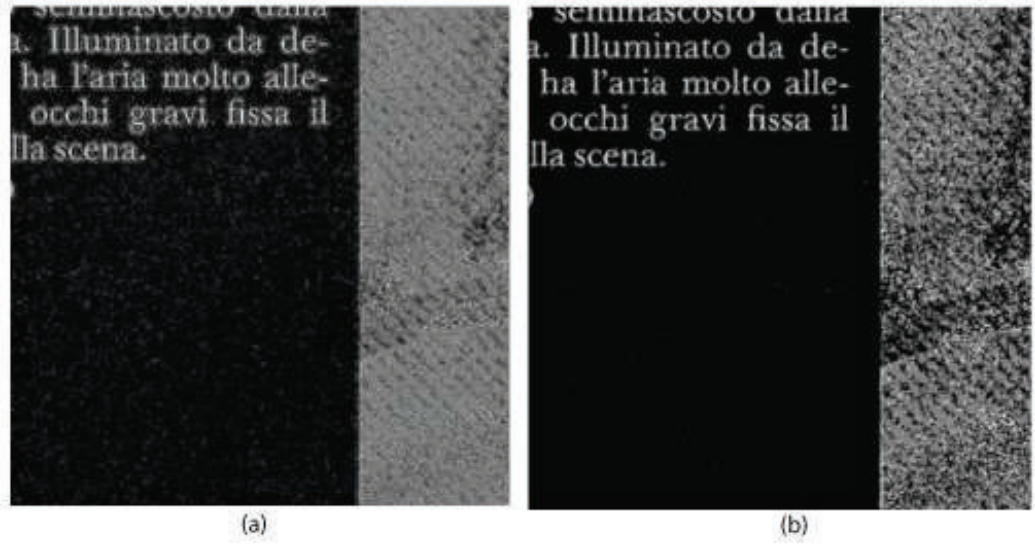

Figure 7: Zoom on Median Blur. Zoom on a page after a binarization. (a) is the image before the median blur and (b) is after the median blur.

3.2 Image Extraction

The foreground and background being identified, the next step was to calculate the coordinates or corners of each image to be able to extract the image region from of the page. Some pixels of the foreground detection correspond to pixels from the images but also the text.

To separate the images pixels from the rest, we used the connected component function. This function connects into groups the pixels connected to each other. We used a connectivity of 8 for the connected function. We also used an average blur of size 5*5 beforehand to increase the connectivity within the images that the median blur might have formed.

Figure 8: Connected Components Two examples of image of page scans after the average blur and connected component function applied. Each color corresponds of a different connected cluster.

Figure 8 shows the results of the connected component function applied to the two pages from Figure 3. It does an excellent job at extracting the different images. Next step was to get rid of the region containing the text and lines. Calculating the size of each region and disposing of

the ones under 16,000 pixels worked well (see Figure 9). This value was determined doing iterations with several values on several types of pages.

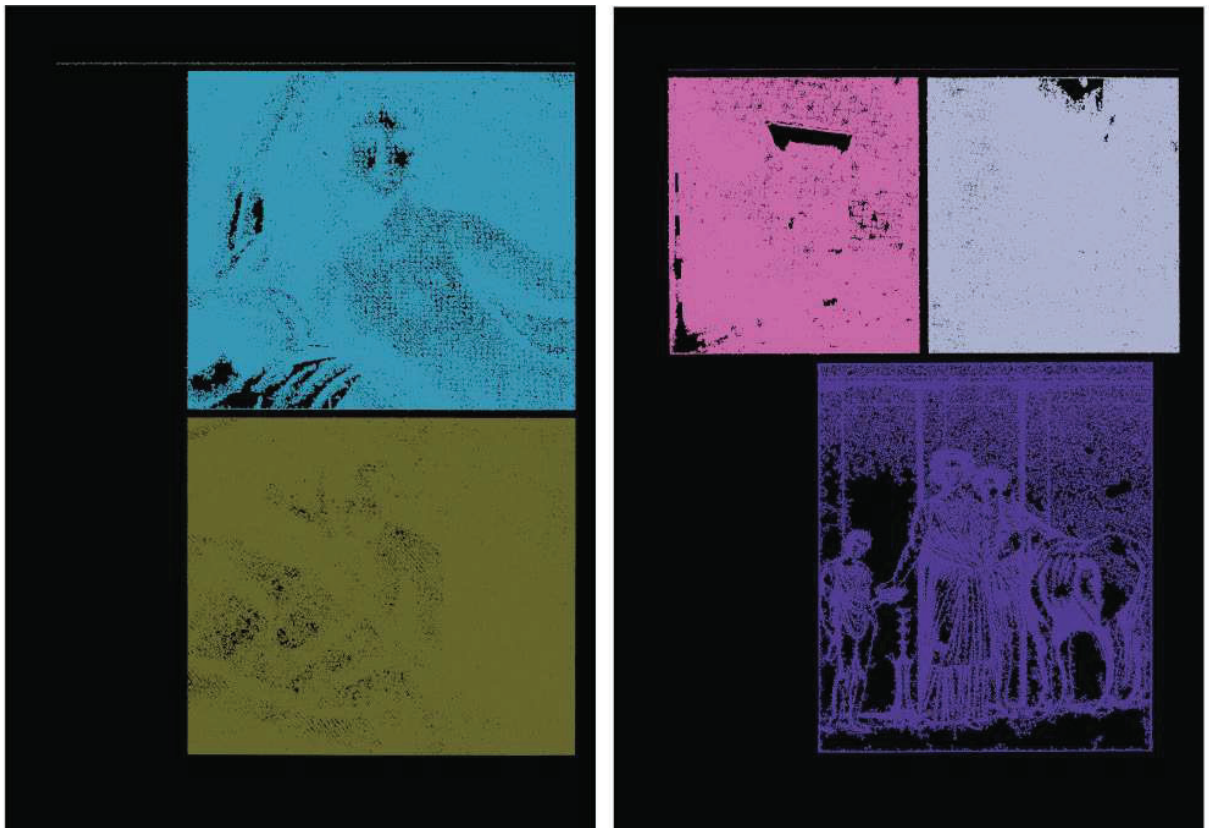

Figure 9:Connected components without small clusters. Two examples of image of page scans after the connected component function applied. Each color corresponds of a different connected cluster, the cluster with less than 16,000 pixels have been deleted.

The area corresponding to each image is extracted by calculating the extremities of the regions:

- Maximum and minimum on the x axis
- Maximum and minimum on the y axis

Once these coordinates were extracted, we eliminated the image regions too big or too small (either height too small or width too small) because they did not correspond to photographs

or drawings in PPM. In the case of region too big being detected, it happened generally when the page had some shade on its borders which created a region taking up the entire page. An example of a case of a region too small can be seen in the Figure 9. The region at the very top contains a decorative line. That region is too small to be an actual image. Figure 10 present the results of the extraction, note that 10 pixels inward are also added to make sure the images are neat and contain no background.

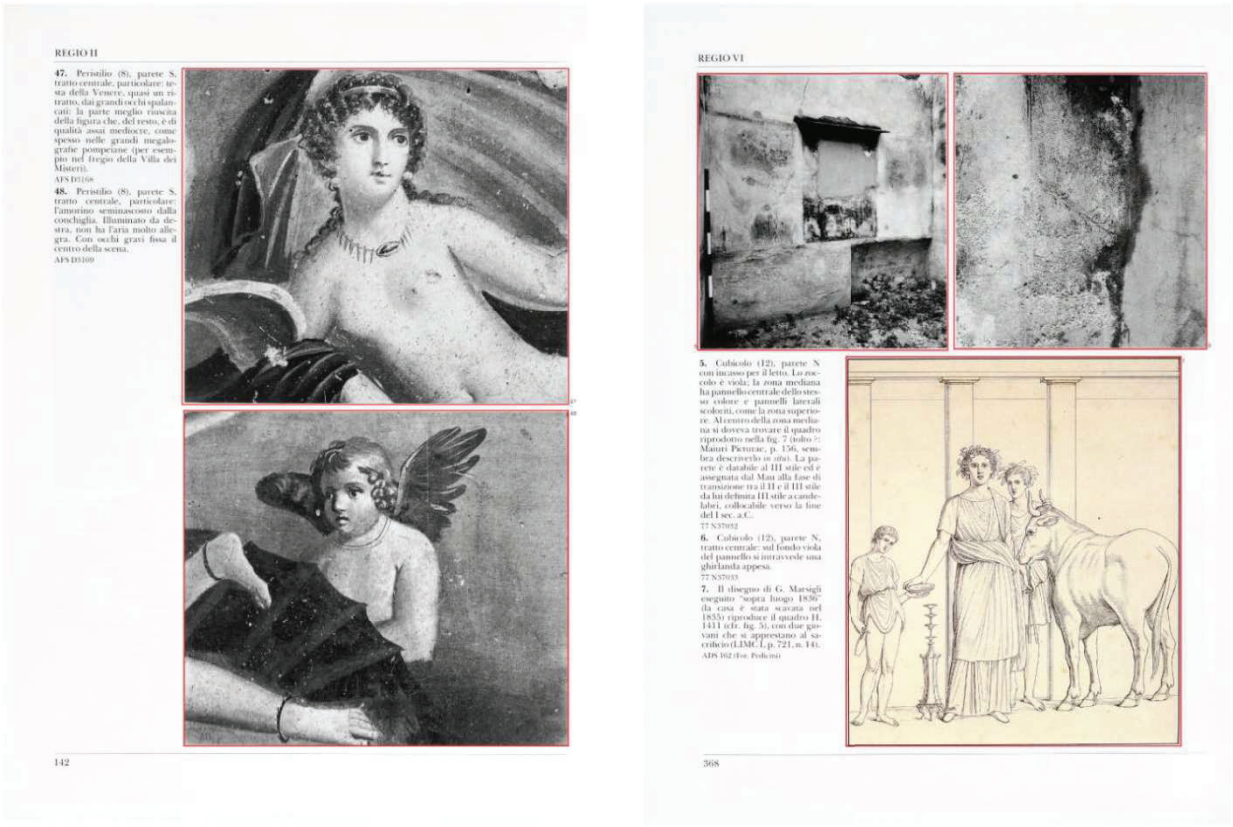

Figure 10: Image Extraction Results. Results of the images' extraction. The red frame boxes correspond to the area collected to save the images.

The detected borders calculated from the coordinates are highlighted in red. Each page with red frames was generated to observed and check the image extraction results quickly. In addition to some regions being too big or too small, some regions were overlapping. One image

can sometime be detected as multiple regions rather than one big one. This happened mostly with maps images (see Figure 11). We fixed most of these problems by combining all overlapping regions.

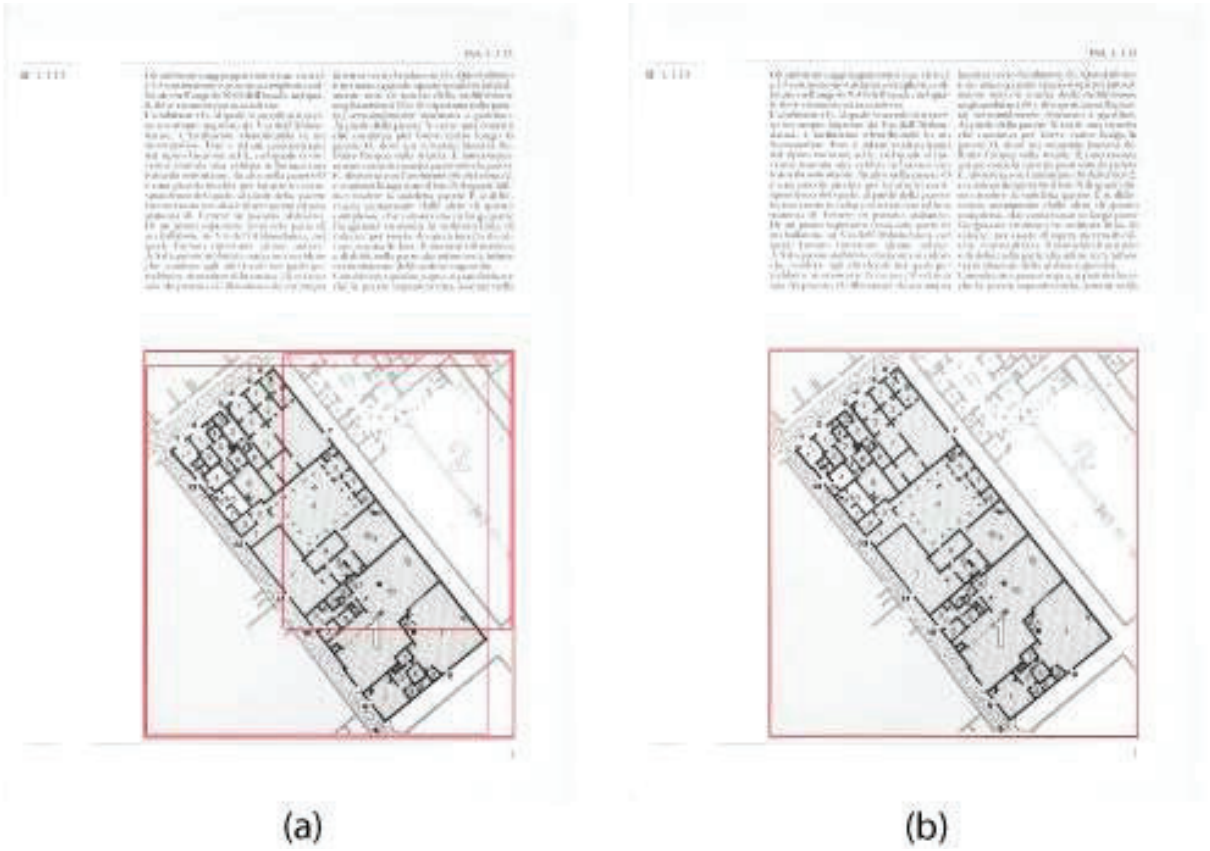

Figure 11: Overlapping and Maps. Page with a map image that creates overlapping problem. (a) has the overlapping problem. (b) has the problem fixed with the coordinates re-calculation.

We identified overlapping regions by looking at the four corners of each region on the image, if at least one of the corner coordinates of one image was inside the square formed by the coordinate of another, we combined the regions by finding the minimum and maximum x and y values from the two regions.

3.3 Evaluation and Observation

Using the methods described above we extracted the images of volume 1-6 and volume 8-10 and the results are displayed in Table 1. Those results have been calculated going through the generated pages with the red frames manually. The total error rate percentage, for these volumes, is 1.58%. The images with errors where manually extracted and saved. The errors were of four different types:

(1) Unavoidable special case

(2) Images cropped too big

(3) Image missed or not detected

(4) Images cropped too small

In the case (1), on some PPM pages the images are placed so that they do not have rectangular shape. It can create overlapping regions which get detected as one big image instead of two. In some cases, the combined region contains unwanted text. Figure 12 is an example of that problem.

These unavoidable errors were counted as cropped too big errors (case 2) when conducting the evaluation of the results described in Table 1.

Cases (2), like cases (1) were not too common. They were occurring mostly if there were scanner noise on the page connecting regions to one another.

Cases (3) usually occur when the image background is very bright, and the foreground region detected (if any) is too small and gets rejected.

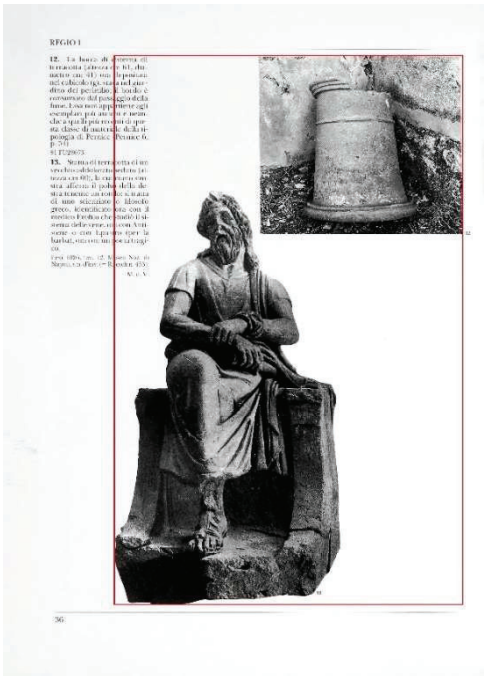

Figure 12: Overlapping Problem Example. Extraction result presenting an unavoidable problem (Case 1). The area extracted contains two images.

Table 1: Image Extraction Result Numbers. Cropping results of Volume1-11 (7 excluded) of PPM. All the volumes have been extracted without the overlay subtraction module except for volume 11.

Meta Information		Number of Images				Errors
Volume	Images	Cropped Correctly	Missed	Cropped too small	Cropped too big	
1	1,660	1,653	0	5	2	7
2	1,623	1,620	0	3	0	3
3	1,993	1,982	2	9	0	11
4	2,028	1,995	3	27	3	33
5	1,710	1,687	7	14	2	23
6	1,972	1,910	9	51	2	62
8	2,084	2,019	10	55	0	65
9	1,969	1,927	6	35	1	42
10	820	815	0	5	0	5
11	1,331	1,282	18	28	3	49
TOTAL	17,190	16,890	55	232	13	300

This is an extreme case of case (4). In the case (4), most of the time it is part of the sky missing in an image because the sky is as bright as the page background and the adaptative function has trouble differentiating it from the background, and the page background is not a perfect white. It also happens with sketches for the same reason (see Figure 13).

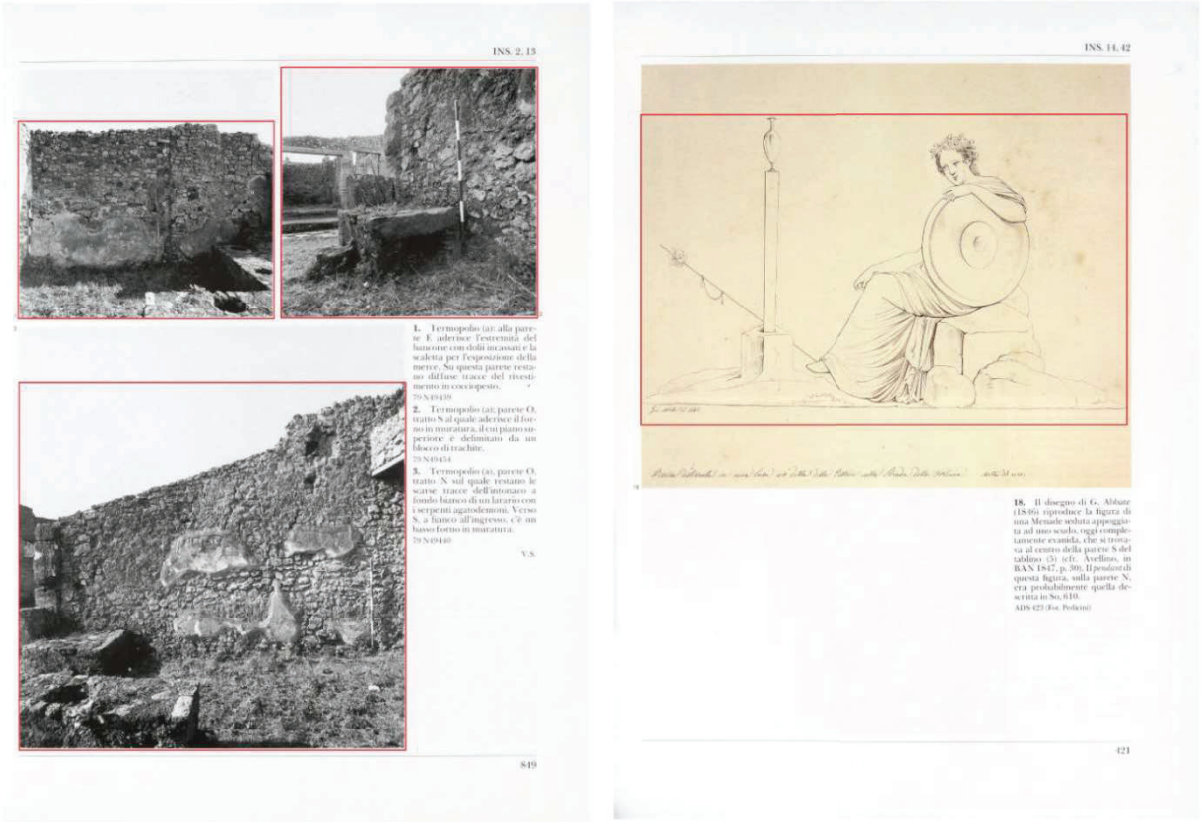

Figure 13: Cropped Too Small Problems. Extraction results presenting problems of extraction too small (Case 4). The area extracted are too small and do not contain the images fully.

The approach described above works very well for black and white images, but in some cases, it does not perform well for images with colored images or pages with yellow backgrounds. This problem occurred so frequently in volume 11 that we developed an alternative pre-processing step to improve segmentation.

Consequently, volume 11 contains a lot more colored images and the error rate for was too high. It was leaving too much labor to be done manually. Table 1 contains the results of the extraction method. Volume 1-10 have been extracted using the method described in the previous section. Volume 11 has a been processed with the addition of the module presented in the following section. Less than 2% of the images had to be extracted manually. For more illustration and details on the error encountered please refer to paper [48].

3.4 Yellow tint overlay subtraction

We worked on improving the automatization of the method by adding a module that calculates mathematically the yellow tint overlay to subtract it from the pages. This cleans every scan leaving the background of each page whiter, while at the same time improving the color on the images themselves. The module also adds colored pixels as foreground pixels to be segmented from the background to further improve the results.

To recreate the noise overlay from the scan we used a bilinear interpolation which is a weighted average of the RGB values from the four corners of the page.

First, we calculated four values for each corner as the average of each L-shaped area that can be seen in Figure 14 below. Let us call these values C1, C2, C3, and C4. We then assumed that the noise overlay that is composed of a combination of those four colors. The overlay pixels are defined as (2):

$$\text{overlayPixel} = C1 * w1 + C2 * w2 + C3 * w3 + C4 * w4 \qquad (2)$$

Where w1, w2, w3, and w4 are weights of values ranging from 0 and 1. They are calculated bilinear interpolation based on the distance from the pixel to each corner (See Figure 15).

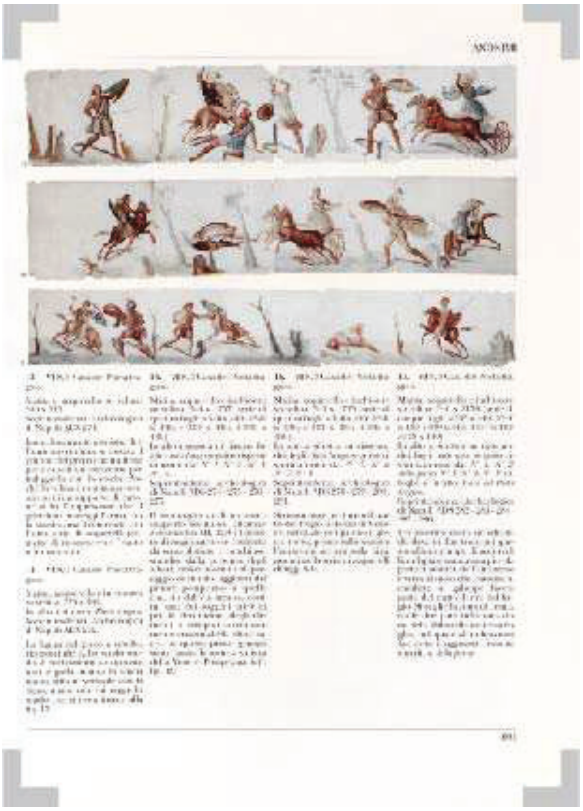

Figure 14: L-shaped Corners. Scanned page with L-shaped corners highlighted, representing the area of the page used to calculate the four colors to interpolate the noise tint overlay, which is usually yellow due to scanning noise.

For instance, the overlay pixel value from the very middle of the image would be:

$$\text{overlayPixel} = C1 * 0.25 + C2 * 0.25 + C3 * 0.25 + C4 * 0.25 \quad (3)$$

Each color's weight is proportional to the opposite area of the corner color. This is illustrated in Figure 15 where the color C1 is proportional to the area grayed out. Meaning w1 would be equal to (1-α)*(1-β). Therefore, equation (2) can be written as:

$$\text{overlayPixel} = C1(1-\alpha)(1-\beta) + C2(\alpha)(1-\beta) + C3(\alpha)(\beta) + C4(1-\alpha)(\beta) \quad (4)$$

Alpha and beta are normalized by the size of the image and represent the percentage of width and height, respectively.

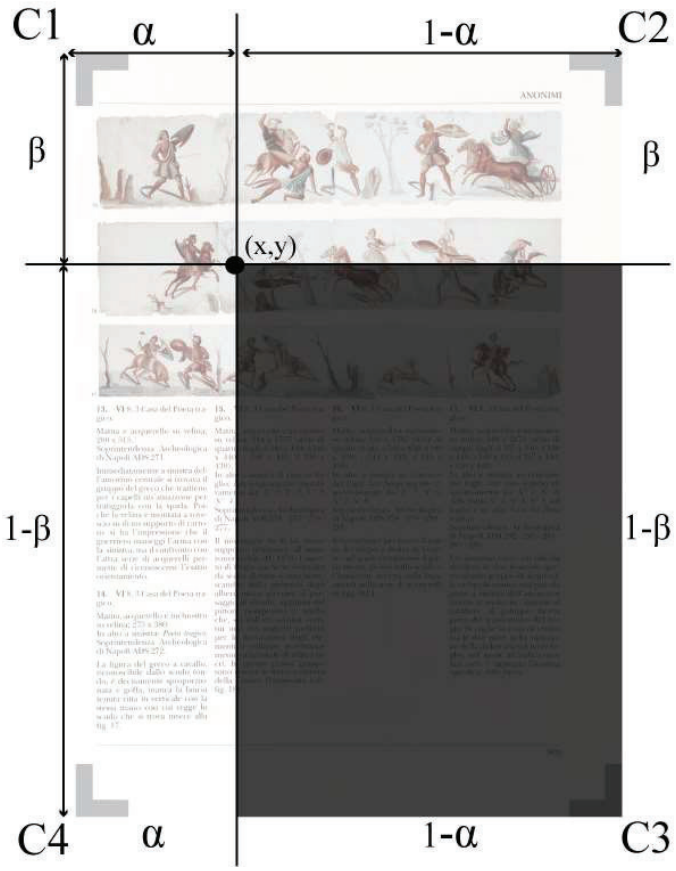

Figure 15: L shaped Corner with Diagram. Page from PPM with the L-shaped corners highlighted as well as an overlay explanatory diagram of the α and β calculation. The overlay pixel (x,y) is at the intersection of the horizontal and vertical line. The weight w1, w2, w3 and w4 are calculated proportionally to the opposite area of the four respective corners C1, C2, C3 and C4. The area to calculate w1 has been grayed out.

Calculating the specific overlay for each page was essential because the type of noise and amount of pixels present differ depending on how the person scanning the page handled the book. The page scans do not all have the same number of pixels, so calculating the overlay to be specific to each page was essential. Figure 16 shows a before and after of the overlay subtraction.

We can observe that the yellow tint has completely disappeared. This significantly improved the detecting the image borders.

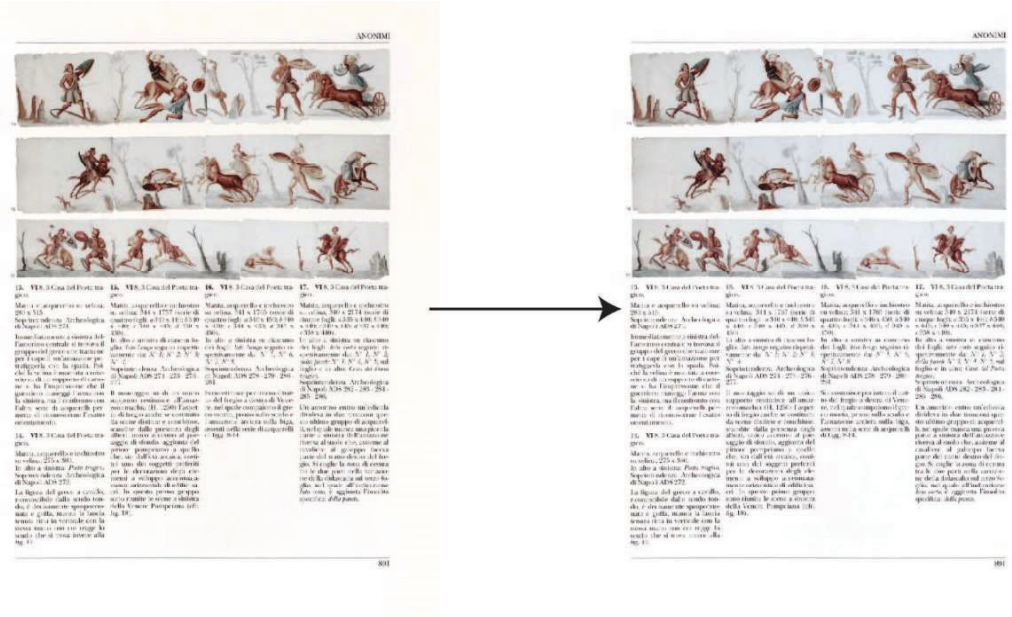

Figure 16: Overlay Subtraction Results. On the left, a page scanned from PPM with yellow tint. On the right, the yellow tint has disappeared after overlay subtraction.

3.4 Evaluation of the module

The overlay subtraction was not applied to volumes 1-10 since the initial results of the other volumes had already been detected and saved, making the reapplication of the algorithm with the new module unnecessary.

Although it would be interesting to compare performances with the heuristic method, we chose not to use a deep learning method to extract the images, as it would have required more time to create a robust training dataset manually that contained the coordinate of the images to extract. Since the margin of error was lower than 2%, we did not explore other methods for extraction.

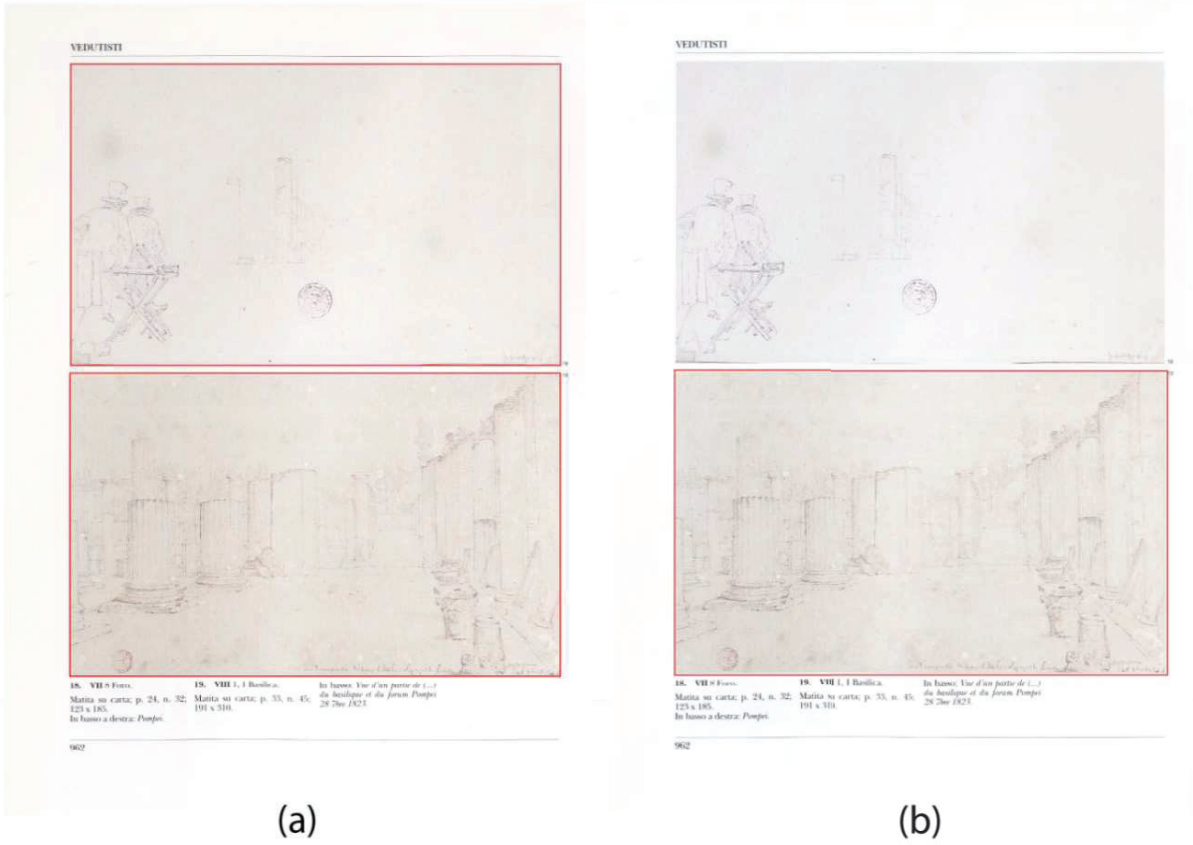

Figure 17: Error with the Overlay Subtraction. The top image of this page has not been detected (but it was detected without the overlay subtraction module). (a) cropped result presented on the raw PPM page. (b) the page with the overlay subtraction applied which brings the background close to the images color.

The result of the extraction of volume 11 with the module implemented are in Table 1. The new module of overlay subtraction and colored pixel addition did improve the results of the color images detection but sometimes introduced new problems that were not there before. For instance, the top image on the page in Figure 17(a) and 17(b), was correctly detected without the module because the image was very white, and the page background had a yellow tint. In this case, the improvement of the process created an error that did not exist before. The page background without its yellow tint noise got too close to the same color as the image. These cases did not counteract the overall positive improvement of the results.

4 Collecting text data

The previous chapter described the process we used to extract and collect all the images of the PPM volumes. This chapter is dedicated on the extraction of the text information associated with the images.

4.1 Page OCR

To extract the captions from the scanned pages, we first experimented on running the OCR algorithm from the library tesseract. By default, the function of the tesseract library applies an Otsu algorithm binarization [11], which was degrading the results. To compensate we had to do some preprocessing on the scanned pages. We had to pre-process the pages by transforming them into grayscale images, then we shifted all the near-white pixels to be absolutely white (255 value), and the rest to be zero, which made sure the letters were black. This corresponds to the inverse binarization of equation (1) with threshold of 200.

We also applied an erosion of size 3*3 to enhance the letters' thicknesses. The erosion removes pixels from background boundaries which add pixels to the letters because they are part of the foreground. Finally, we applied OCR. The results improved significantly, but the OCR tried to make up text from the images. To solve this problem, we use the work describe in chapter 3 to save a version of the pages without their images (see Figure 18). The OCR did a decent job but there are some pages that were completely misinterpreted. When the OCR algorithm failed to detect accurately the columns of text, the result was unreadable. It usually happened when two columns were interpreted as one single paragraph.

Once we had collected all the OCR result of the PPM pages, we had to extract and save in separated files the portions of the text containing every single caption with their corresponding index.

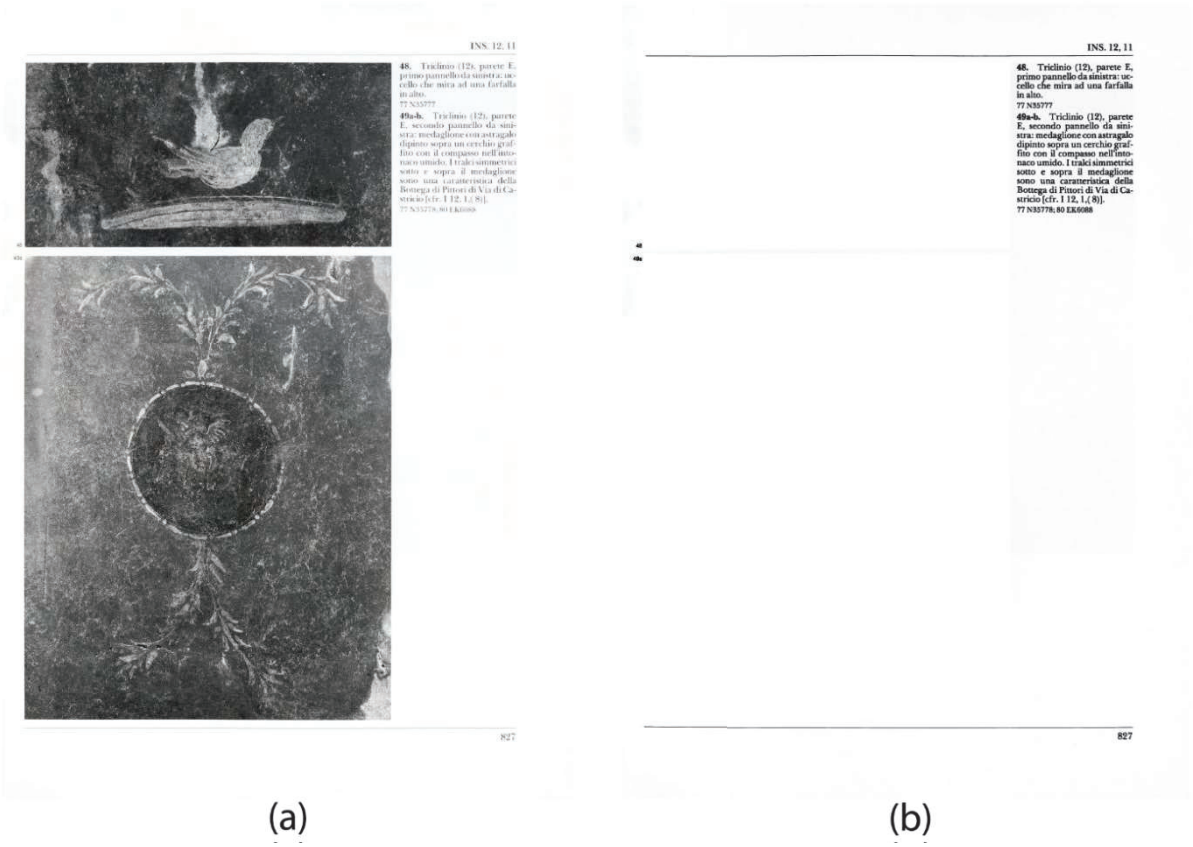

Figure 18: OCR Pre-Processing. Page before and after the preparation for the OCR. (a) Page scan before the text enhancement and image deletion. (b) Page scan without its images and with the text enhancement.

4.2 Collecting caption indexes and cleaning

To extract each caption into a single file, we had to go through every OCR page file and try to detect the begin and end of a caption. Every caption starts with a different possible pattern (beginning patterns):

1. Regular caption: “X.”
2. Caption with a letter index: “Xa.”
3. Same caption for multiple images: “X-Y.”
4. Same caption for multiple images with a letter index: “X-Ya.”
5. Same caption for multiple images with letter indexes: “Xa-Yb.”
6. Same caption for multiple images with letter indexes: “Xa-b.”

The X and Y values are numbers, for instance, a beginning pattern of shape (1) would be “2.”. We created an algorithm that goes through every page text file, detects, and saves the caption into another file. A caption starts to be recorded and written into a separate file when a specific pattern is detected. Which is defined by:

new line *beginning pattern*

Or by:

Start of document *beginning pattern*

We had to add the *new line* as part of the beginning pattern because some captions had numerical information in them. A caption is complete when it is either the end of the page or the beginning of another caption is detected.

4.3 Problems and future work

The algorithm does not handle cases with the beginning patterns of “Xa-Yb” nor “Xa-b”. It also does not handle cases where the caption continues on the following page, therefore some caption files do not contain their caption entirely. Some captions have not been detected altogether due to OCR noise or have a wrong index (see Figure 19). With the latter, we have

been able to automatically fix most of these problems when it is preceded and followed by a logic sequence. For instance, in the case of Figure 19(b), we were able to assign the caption to its correct caption index number because the previous caption index detected was 176 and the following one 178.

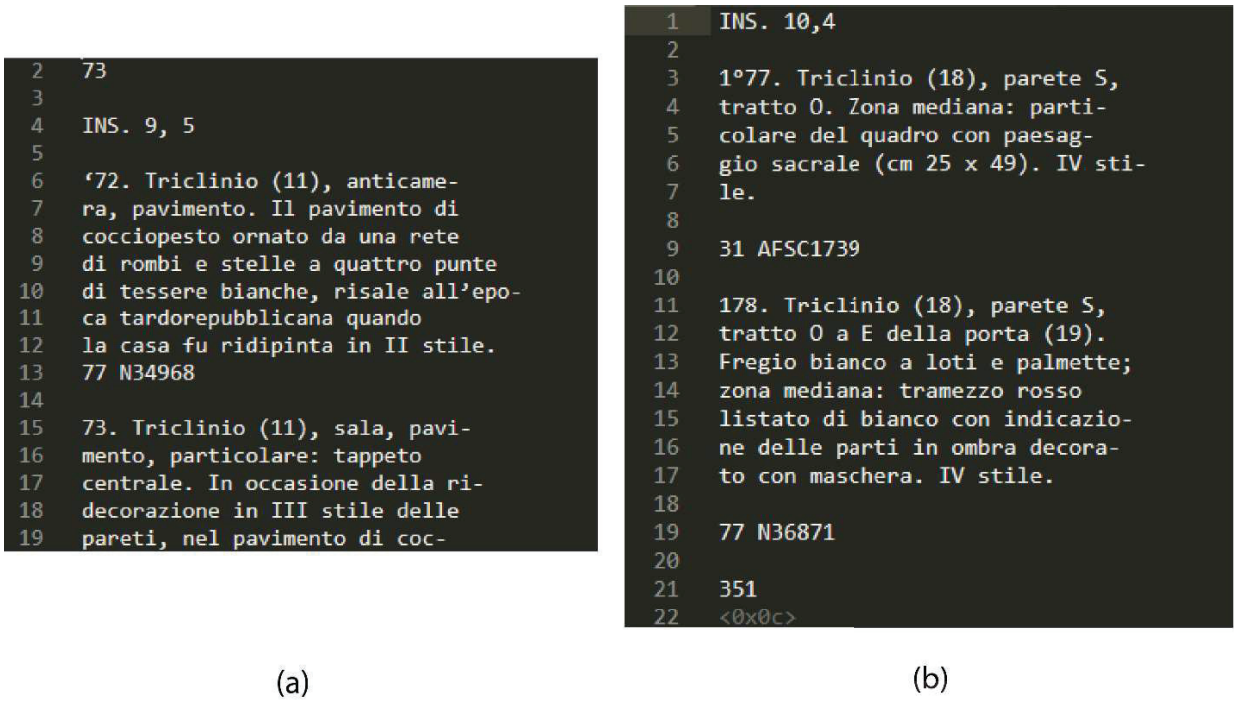

Figure 19: OCR Noise Examples. Two caption results with OCR noise in their index. On the left the caption 72 has ’72. On the right the caption 177 is written 1°77.

The results obtained from the sections above method have not been checked manually. However, the validity of the caption number and association with their image has been checked. This evaluation process in described in chapter 5.

4.4 Collecting the image indexes

The images in PPM are indexed with a small number, located at the image’s top or bottom corner. This number associates the image to a caption indexed with the same number,

sometimes located on a separate page (usually no more than 5 pages apart). Figure 20 illustrates the index's placement of the images and captions.

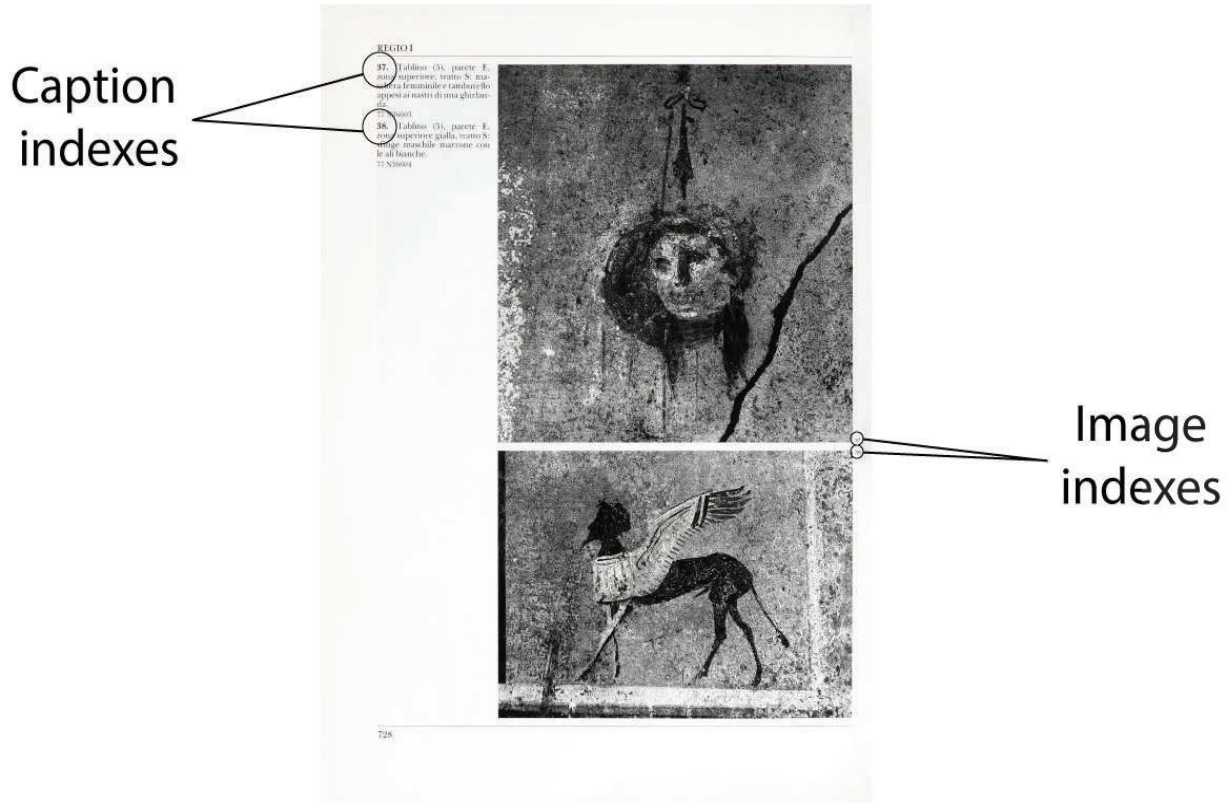

Figure 20: Caption and Image Indexes' Location. Page scan example with the caption indexes circled on the left and the images indexes circled on the right.

We ran experiments trying to detect the image indexes by running OCR on the whole page, but the indexes did not show up in the results. We tried cropping out only the area with the number to detect the number with OCR, which showed better results, but a complex program would have been required to check each corner of every image due to the inconsistent location of the numbers between pages. Moreover, the OCR was performing at less than 100% accuracy.

The PPM volumes are organized by regions in Pompeii (Regio I-IX), and within each volume, the content is structured by city blocks within that region and properties within the

block. An unindexed map introduces the discussion of each property, and every image index restarts at 1 after the map is displayed, indicating that the images of frescoes, mosaics, artifacts, and architecture that follow belong to the property depicted in the map. Figure 21 shows an example of two consecutive pages of PPM, the first one containing a map image.

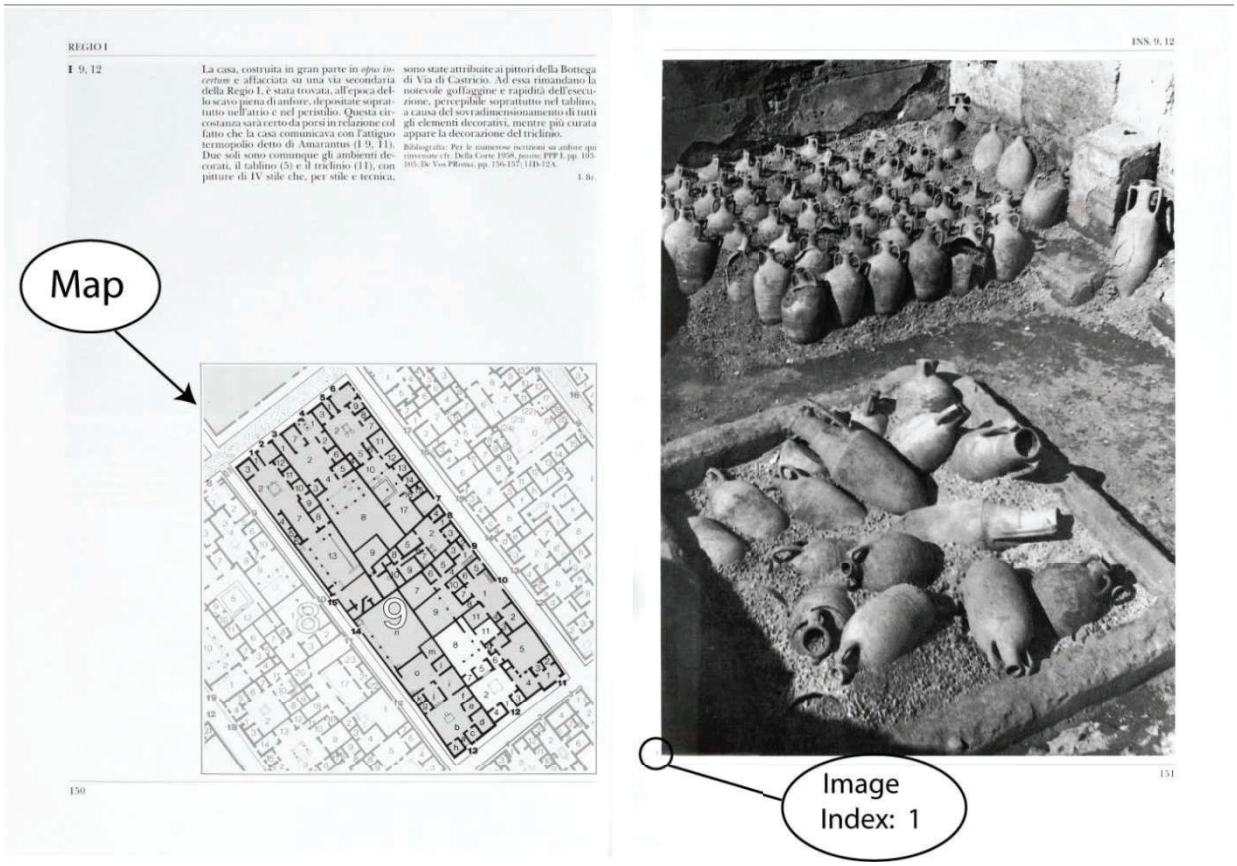

Figure 21: Reset of Image Index Example. Scan of page 150 of Volume 2 on the left next to page 151 of Volume 2 on the right. The page on the left has a map which resets the image index for the image on the following page.

We can observe that the following image has an index of 1. Therefore, we went through each image in order of appearance and indexed them with a counter that restarts when a map is presented. Thanks to some of our previous work, detailed in the chapter 7, we had knowledge of which images were maps. This way, we assigned the image index numbers to their appropriate

captions and saved it in our database. Indeed, previously in our project, we retrained the last layer of a pretrained Inception V3 Convolutional Neural Network [29] to classify images of PPM. The retrained CNN automatically classified the images in volume 2 into 6 categories: ruins, frescoes, schemas, mosaics, ruins with frescoes present, and maps. In classifying maps, our CNN had an error rate of 0.3%, a precision of 99.4%, a recall of 97.8%, and an accuracy of 99.7%.

This method works well, except that the image files have been extracted and saved assuming that the images would always be numbered in order from top to bottom/left to right. This is unfortunately not the case. It is inconsistent throughout the pages.

Another problem was images with indexes of the form “Xa”, with “X” being a number and “a” a letter. Thankfully, that was not the case often but if it encountered an image with a letter index it would disturb all the following images until a map is encountered again. To fix this problem and assign all the correct image indexes, we did two passes on the data using the linking process as a first pass to detect the images with problems (see chapter 5 for details).

5 Data Organization

In this chapter we describe how we linked the images with their captions and how we organized the data in the database.

5.1 Database Organization

Our database is organized with 3 tables:

(5) Page

(6) Image

(7) Caption

Each table has an “ID” field which corresponds to the file name, which is unique.

The Page table has the following fields:

(8) ID

(9) Page Number

(10) Volume Number

(11) Location

The Image table has the following fields:

- ID
- PageName (foreign ID)
- ImageIndex
- Category (Maps or no maps)
- VolumeNumber
- PageNumber

- Caption (foreign ID)
- Correct
- Regio (Region of Pompei)
- Location (Insula doorway information)

The caption table has the following fields:

- ID
- PageNumber
- VolumeNumber
- CaptionIndex
- ImageID (foreign ID)
- Category (same as the Image table)
- Regio
- Location

The Location information was retrieved by observing that the information is located on the page with odd page numbers. To save and extract it, we ran a program that detects the “INS.” keyword on the OCR page with an odd page number. Every image on that page and the previous page are from the same location. The region information is located on the page with even page number. To save and extract the region, we ran a program detecting the key word “REGIO” on the page with even page number. The images on that page and the following page are from that region. Note, that each volume of PPM is specific to one to three regions. Volume 2 is specific to Region I only.

5.2 Linking the indexes

The caption associated with each image is either on the same page or in its vicinity. No more than 5 pages before or 5 pages after. From this observation, we made an algorithm that assigns a caption to an image choosing the closest matching caption index counting from the display page.

At this point the linking errors are due to:

(12) Caption not extracted

(13) Caption not being extracted or wrongly extracted due to OCR noise.

(14) Image indexes of the form “Xa”.

(15) Wrong image index assignment due to placement on the page during extraction.

(16) Wrong image index assignment due to a previous image index of the form “Xa”.

After doing this process once, we were able to detect the images with problems. In other words, we knew the images triggering indexing problems such as an index of the form “Xa”. Therefore, we went through the image data a second time with a program that displayed the page scan and the image not having a caption assigned or having a caption with an unexpected number. Once the problematic image is displayed the program enables the user to enter an index manually and it updates the database. This way, we have saved all the images with their correct index number of any form (with or without letters). Figure 22 illustrates the way the images were manually corrected.

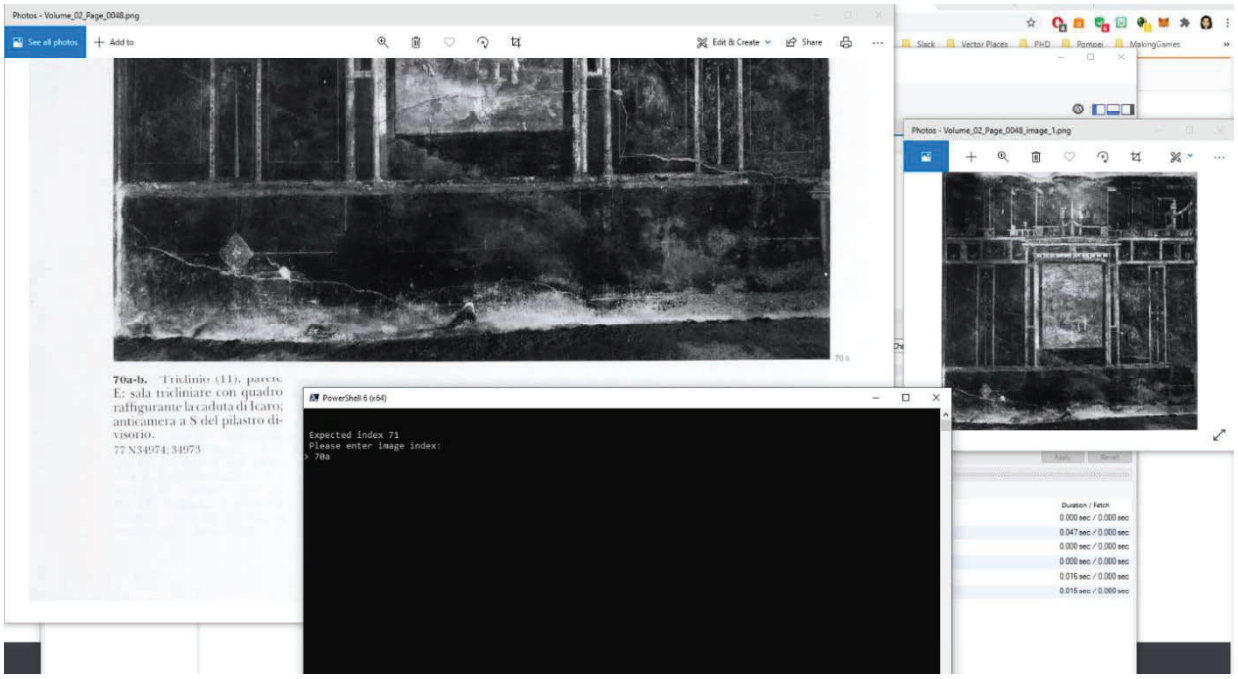

Figure 22: Image Indexes Correction. Illustration of the program to help on the program to help manually assign the image indexes. The program goes through all the images, displays the ones with an indexing problem and their page scan. Then, it asks for the correct index and modifies the database accordingly. On the left side of the figure there is the page scan zoomed in to be able to see the index. On the right side, is displayed the image that the program is asking the index for.

Once the images were all correctly indexed, we ran the linking algorithm once more. The remaining problems are only due to the captions. To check the results, we built a software tool which goes through all the images and displays the caption that has been assigned to them as well as their corresponding scanned page. This tool allowed to manually check the results. The person checking the results has two buttons they can click on: “Correct” and “Incorrect”. Figure 23 illustrates the interface of the checking tool.

This process assigns to the image entries in the database a 0 or 1 value to their “Correct” field. It allows us to store if their index assign is correct as well as if their accurate caption has been found. Table 2 summarizes the results of the indexes matching for volume 2 of PPM.

There are 88.72% of the images of volume 2 that have been correctly matched.

The errors left are the ones with:

(17) The images with indexes containing a letter and the captions (no linking attempt)

(18) Caption containing OCR noise

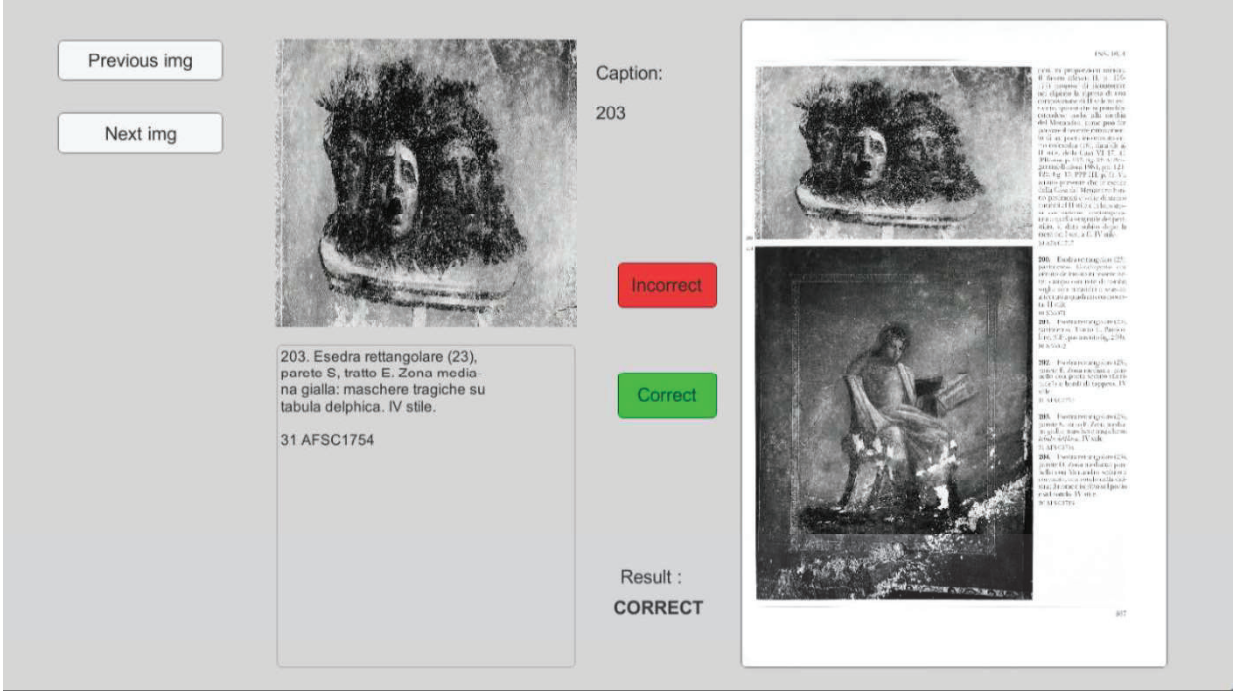

Figure 23: Linking Correction Software Tool Screenshot. Screenshot of the evaluation tool utilized to check if the image indexes match caption indexes.

Table 2: Table of the results from linking the image indexes with the caption indexes from volume 2.

Index matching	
Total number of images	1,623
Map images	67
Missing Caption	78
Incorrect Matching	108
Usable data	1,440

From the work described in chapters 3,4 and 5, we have collected and save data from the PPM volumes, and more specifically:

(19) Entered in the database all the pages entries for volume 2.

(20) Extracted all the images from volumes: 1, 2, 3, 4, 5, 6, 8, 9, 10, 11.

(21) Extracted and correctly linked 1,440 pairs of caption/image from volume 2.

(22) Saved category information on the map images of volume 2.

(23) Prepared the groundwork to extend this work to the other volumes.

The following part of the document is focused on data analysis research and specifically data analysis implementation on the 1,440 pairs correctly collected, as well as the software tool built to navigate through the data.

PART II: DATA ANALYSIS

6 Word Search and Word Similarity

Being able to quickly look up specific artifacts from the PPM volumes is nearly impossible with the print versions of the books. It would require a little bit of luck and a lot of patience to browse through entire volumes to find one specific thing you are looking for and impossible to find the 10 images most similar to this image, scattered across all the volumes. Having the volumes digitized should counter this problem and make the artifact easily searchable. This chapter is dedicated to the method we used to create the part of the tool which can return images with their captions using words. The users can enter keywords they are interested in inside an input field. Clicking on a search button would then return the results of the matching artifacts in the PPM.

6.1 Prepping the corpus

To be searchable, all the words contained in the volumes of PPM need to be pre-processed. It is necessary to have a table containing each caption file name and its associated searchable words. Once captions have been cleared of line break dashes that occur in the text, we collected every word. To do so, we had to eliminate separator characters and other strings such as:

- Period: “.”
- Comma: “,”
- Colon: “:”
- Semicolon: “:”
- Apostrophe: “‘“
- Parentheses: “(” and “)”

- Extra spaces and tabs
- Numerical data
- Words containing less than 2 characters

All words have also been transformed to lowercase. Using python, we created a loadable table containing the caption names and their corresponding searchable word lists that can be used in all the lookup search implementations.

6.2 Term Frequency Search

We proceeded with a first implementation to retrieve a caption and its image using the Term Frequency (TF) (X reference in the background) only. TF measures the frequency of a word in a document. It is defined as (5):

$$TF(t,d) = \frac{\text{number of occurrences of term } t \text{ in document } d}{\text{total number of words in document } d} \quad (5)$$

In our case, the document is one caption, and the terms are the query (reference from notation X) entered by the user in the input field. That query can be one or several words. When the query is composed of several words, the TF of the terms is defined as (6):

$$TF(t,d) = TF(t1,d) + TF(t2,d) \quad \text{where } t \text{ is composed of } t1 \text{ and } t2 \quad (6)$$

When a user looks up terms, we calculate a TF score of each term for every caption and add them together. We then collect and return the caption with the highest score. We made it return 10 results by default, but this a value that the user can modify. Algorithm 1 describes the process, in pseudo code, of returning the highest TF scores for a word which has been implemented in python and run by the research tool. Algorithm 1 can be summarized in 2 parts:

- Calculate the TF score of each query term for each caption
- Return the top TF scores

```
1. Function Calculate_TF_Score(term, numberOfResultToReturn):
2.    initialize All_TF_scores to 0;
3.    foreach caption IN PPM{
4.                  FOREACH word IN caption{
5.                          IF word IS EQUAL TO term{
6.                                  TF_score increase of 1;
7.                          }
8.                  }
9.          TF_score is divided by the number of words in caption;
10.         TF_score is appended to All_TF_scores;
11.  }
12.     return the tops numberOfResultToReturn of highest
   All_TF_scores;
```

Algorithm 1: TF only Term Look Up Algorithm. Pseudo-code for the algorithm for calculating and returning the top N captions with the highest FT scores from the PPM caption database. The pseudo code describes the process for searching one term. For multiple terms, an additional foreach loop is required to go through all the words in the terms searched. This pseudo code has been implemented in Python and is integrated in the search tool.

This method serves its purpose, but it does tend to prioritize the caption with fewer words. This is because the total number of words of a caption is in the denominator of the score calculation, see (5). Therefore, the caption with fewer words will have a higher score even if the term only appears one time.

The idea of the formula (5) is that the term has more value in a caption with fewer words, which is valid. However, PPM has many captions with both very short and long descriptions. So, this method is not ideal if we want to enhance the user experience to show more homogeneous and relevant results. Figure 24 shows the first 6 results from the search of the term “Apollo” (which was recognized as the god of music and prophecy in Greek and Roman religion, often identified with the sun).

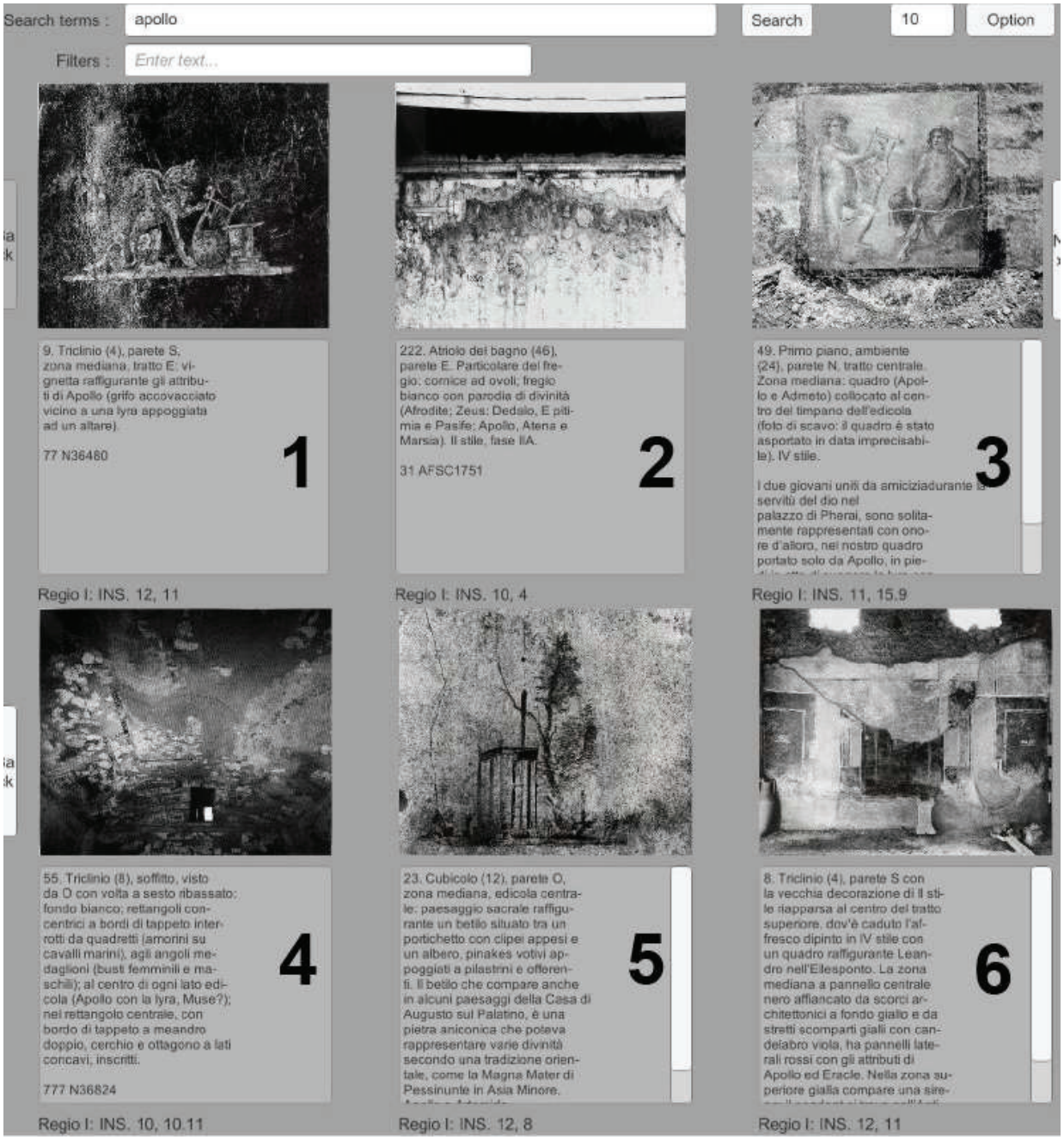

Figure 24: First 6 results of "Apollo" using Term Frequency. First 6 results of the term "Apollo" in the research tool using the TF only method. The captions are returned ranked in TF scores. The big 1-6 digits describe the order of appearance of the captions according to their TF scores. It is observable that the captions are returned in the order of their length. Number 3 might appear longer than number 4 to the human eye, but this is only because the caption contains a line return due to OCR noise. One might also notice the word "apollo" was searched and the results contain "Apollo". The search "apollo" or "Apollo" will return the same results because of the pre-processing describe in section 6.1.

Figure 24 illustrates that the results are returned in order of the caption length. One approach to counteract this is the use of an improved method, Term Frequency and Inverse Dense Frequency (TF-IDF).

6.3 TF-IDF

Term Frequency and Inverse Dense Frequency (TF-IDF) allows us to measure how relevant a word (or term) is to a document in a collection of documents. TF-IDF makes it possible to analyze a word with respect to the entire corpus of PPM and not only to one caption. It combines the TF score as explored in section 6.2 above and the Inverse Dense Frequency (IDF) (7). The IDF serves as an indicator of how frequently a term appears. The less frequent a word the higher its IDF because the word becomes “rarer”.

$$IDF(w) = \begin{cases} 0 & \text{if } w \text{ is not in any document} \\ \ln\left(\dfrac{\text{total number of documents}}{\text{number of documents containing } w}\right) & \text{otherwise} \end{cases} \quad (7)$$

The Inverse Dense Frequency is equal to zero if the word does not appear in the corpus. Otherwise, it is the logarithm of the total number of documents over the number of documents containing the word. The logarithm (base e) is used so that the value does not explode when the number of documents expands considerably.

The final relevance score of a word is then calculated as TF (5) times IDF (7) for each document (8). When searching for a word using the TF-IDF measure in PPM, each caption will have a TF-IDF score calculated according to that word, the highest score can then be selected.

$$Relevance(w, d) = TF(w, d) * IDF(w) \quad (8)$$

In our specific case, the equation is used with the word w being a word from the search query of the search tool and the document d is a caption from PPM.

When searching for multiple words at once, the relevance is calculated as a TF-IDF vector. For instance, with a search query of 2 words, the relevance vector or TF-IDF vector for each caption will be of size 2 as well. Each element of the relevance vector is calculated as equation (8) describes. To select and rank the results of the caption search, we use a distance calculation between the query vector and the relevance vectors (see section 6.4 under for details). The query vector is calculated the same way that the TF-IDF is calculated considering the search query as a document. For example, a search query of "Apollo uccello" would have a vector of $[½ * IDF(\text{“}Apollo\text{”})\ ½ * IDF(\text{"}uccello\text{"})]$.

The IDF scores of each word of PPM have been pre-calculated beforehand to be used in the query vector and TF-IDF vectors calculation. They have been saved in a matrix that we call the IDF matrix.

The overall process of searching using TF-IDF is described in the pseudo code of Algorithm 2 and Algorithm 3 which can be summarized in 4 parts:

- Calculating the IDF matrix for each word of PPM (Algorithm 3 part 1)
- Calculating the TF-IDF of the search query (Algorithm 2)
- Calculating the TF-IDF of each caption of the terms in the query (Algorithm 3 part 2)
- Returning the top results. (Algorithm 2)
- A multiple search request using the TF method calculates a score for each caption, which is then ranked from high to low. A multiple search request using the TF-IDF method calculates a vector for each caption, which then needs to be compared to the query vector to be ranked as more or less relevant. This is what brings us to the next part of this chapter which presents the different method we use to compare the query vector and the caption vector.

```
1. function LookupQuery_using_TF_IDF(query, numberOfTopResults){
2.     query = CleanQuerry(query)
3.     query_vect = Initialize query vector with values of ones;
4.     index=0;
5.     for w in querry:
6.         idf = GetIDFScore_from_IDFMAtrix(w);
7.         query_vect[index] = (float(query_vect[index])/len(query));
8.         query_vect[index] *=float(idf);
9.         IDF_vect[index]=idf;
10.          index++;
11.     // Need to calculate a vector of the size of the query for
12.     // each document
13.     captions_TF_IDF = CalculateTF_IDF_vectors(query, IDF_vect);
14.     distanceMatrix = CalculateDistanceMatrix(captions_TF_IDF,
   query_vect);
15.     return the numberOfTopResults top results from the distance
   matrix;
16. }
```

Algorithm 2: TF-IDF Search Process. Function taking a query and the number of results to return as input. It returns the "numberOfTopResults" top results. It cleans the query as explained in section X, calculates the TF-IDF vector of the query, calls the function to calculates the TF-IDF vectors of all the captions, calls the function to calculate the distance matrix between the query vector and all the caption vectors (see details in section 6.4). Then returns the top result recorded in the later. This pseudo code has been implemented in Python and is integrated in the search tool.

6.4 Vector Comparison

When comparing vectors, we can either calculate how similar they are to one another or how different they are. This following part discusses different ways to compare vectors and what has been implemented in the PPM research tool. When entering a query in the research tool, one has the possibility to search using the TF measure or the TF-IDF measure. When using the TF, the results are returned using a max function. When using the TF-IDF, we need to compare the query vector and the caption vectors to one another.

```
1. Function CalculateIDFMatrix(){
2.     Initialize empty IDFMatrix;
3.    //Counting the number of documents containing each word
4.    //which creates word -> number of documents, to be appended in
5.    //the IDFMatrix
6.    foreach(caption c in PPM){
7.       Initialize empty words checked list;
8.       foreach(word w in c){
9.          if(w has already been checked in the caption)
10.              skip;
11.           else{
12.              Increase document count of w in the IDFMatrix;
13.              Put w in the words checked list;
14.           }
15.        }
16.    }
17.    //Calculating the IDF values in the IDFMatrix
18.    foreach(element in IDFMatrix){
19.       element = ln(total number of document in PPM/element)    ;
20.    }
21.    return IDFMatrix;
22. }



23.  Function CalculateTF_IDF_vectors(query, IDF_vect){
24.     //IDF_vector contains the IDF values of the words in the query
25.     Initialize tf_idf_vectors-result;
26.     foreach caption c in PPM{
27.        index=0;
28.        Initialize tf_idf_score to a list of size of len(query)to
   zero  values;
29.         foreach word w in query{
30.            foreach w2 in the words in_caption c{
31.               if(w == w2){
32.                  tf_idf_score[index]=tf_idf_score[index]+1;
33.               }
34.             }
35.
   tf_idf_score[index]=(tf_idf_score[index]/len(words_in_caption))*IDF_
   vect[index])
36.            index ++
37.          }
38.          append tf_idf_score to tf_idf_vectors-result;
39.     }
40.     return tf_idf_vectors-result;
41. }
```

Algorithm 3: Calculating the IDF Matrix and the TF-IDF vectors for each caption. Two functions used in the TF-IDF search process (called by Algorithm 2). The first one (part 1) calculates the IDF matrix of each word contained in PPM. The second (part2) calculates the TF-IDF vectors of all the captions.

6.4.1 Euclidean distance

One way to evaluate the closeness between two vectors is to calculate the distance between them. One way to do so is to use the Euclidean distance, see equation (9).

$$d(\vec{u}, \vec{v}) = \|u - v\| = \sqrt{\sum (u_i - v_i)^2} \tag{9}$$

In the research tool, when using the Euclidean distance to compare the query and the captions, it collects the Euclidean distances from the query TF-IDF vector to each caption vector.

Then it returns the results with the lowest value. The bigger the value the more distant the two vectors are. The Euclidean distance gets more complex to calculate as the vectors grow and does not take into consideration the direction of the vectors.

6.4.2 Cosine similarity

Another way to compare two vectors, is to calculate how similar they are to one another. Cosine similarity measures the similarity between two vectors of an inner product space which is equal to the cosine of the angle between them see equation (10).

$$similarity(\vec{u}, \vec{v}) = \cos\theta = \frac{\vec{u} \bullet \vec{v}}{\|u\|\|v\|} \tag{10}$$

The cosine of the angle is the inner product of the two vectors normalized to both have length 1. This is because the inner product is defined as equation (11).

$$\vec{u} \bullet \vec{v} = \sum u_i v_i = \|u\|\|v\| \cos\theta \tag{11}$$

The cosine similarity can also be viewed geometrically as shown in Figure 25. The smaller the angle between the vectors the more similar they are. Therefore, the similarity value

will be between zero and one. The higher the cosine (closest to 1) the more similar the two vectors are, see Figure 26 for a trigonometry reminder.

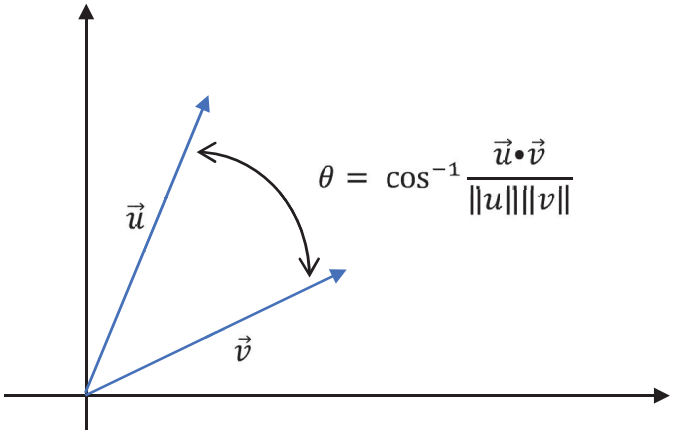

Figure 25: Angle θ between vector $\vec{u}$ and $\vec{v}$. Illustration of the angle θ between two vectors. The bigger the angle, the bigger the difference between the vectors.

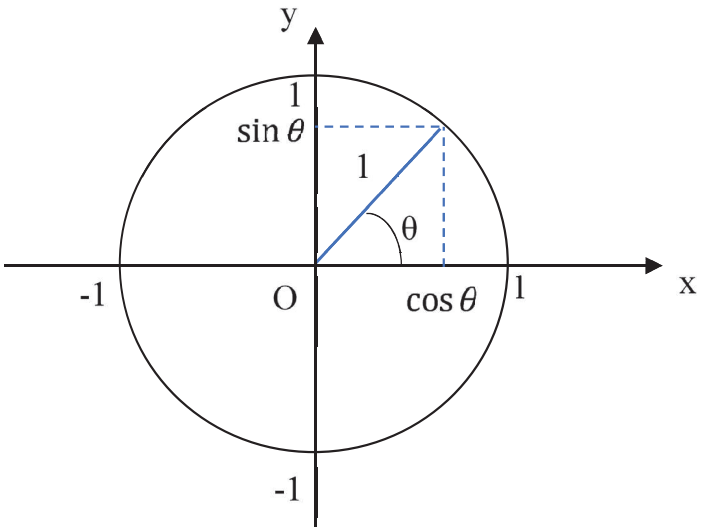

Figure 26: Trigonometry Circle. Schema of a trigonometry reminder which illustrates why the similarity value is between 0 and 1. It also exposes why a value of 0 means the vectors are very different and a value of 1 means they are similar.

One advantage of cosine similarity is its low complexity, especially for sparse vectors. However, with a query composed of 1 word, the Euclidean distance is more helpful because the cosine similarity returns the results containing the word in the caption in order of their appearance in PPM (because of the normalization).

Using the Euclidean distance or cosine similarity for a query with several words will not necessarily prioritize the captions containing both words. It will depend on their TF and TF-IDF score which depends on their frequency in PPM. To get a result with a particular word in the captions, using a filter is needed.

6.5 Filters

To help researchers to get relevant artifacts related to specific information, the tool needs to return results containing that specific data. Using the search query with multiple words might not return first the captions containing all the words from the query. It might instead return some with the first word appearing multiple times (with greater term frequency score). Therefore, we added a word list filter input field to the research tool. When using the filter field, the results from the query appear only if they contain at least one of the words in the filter list.

The filter field is particularly useful for instance to retrieve artifact data combined with spatial data. For example, the query “apollo” with the filter “mediana”, will return Apollo artifacts appearing around the middle zone of a wall. It would be relevant to implement more types of filters like:

- A result must contain all the words from the filter field
- A result must not contain the words front the filter field
- Filter a specific area of Pompeii like an insula or region

All the components and options available when looking for a list of words in PPM have been described. The following part is an overview and a discussion of all the functionalities in the research tool.

6.6 Overview

This segment presents an overview of the tool and presents the different search options. The research tool has an input query field, a filter field, and an option window so the user can pick which search option to use. Figure 27 explains visually where to find all these components and how to do a word search with the research tool.

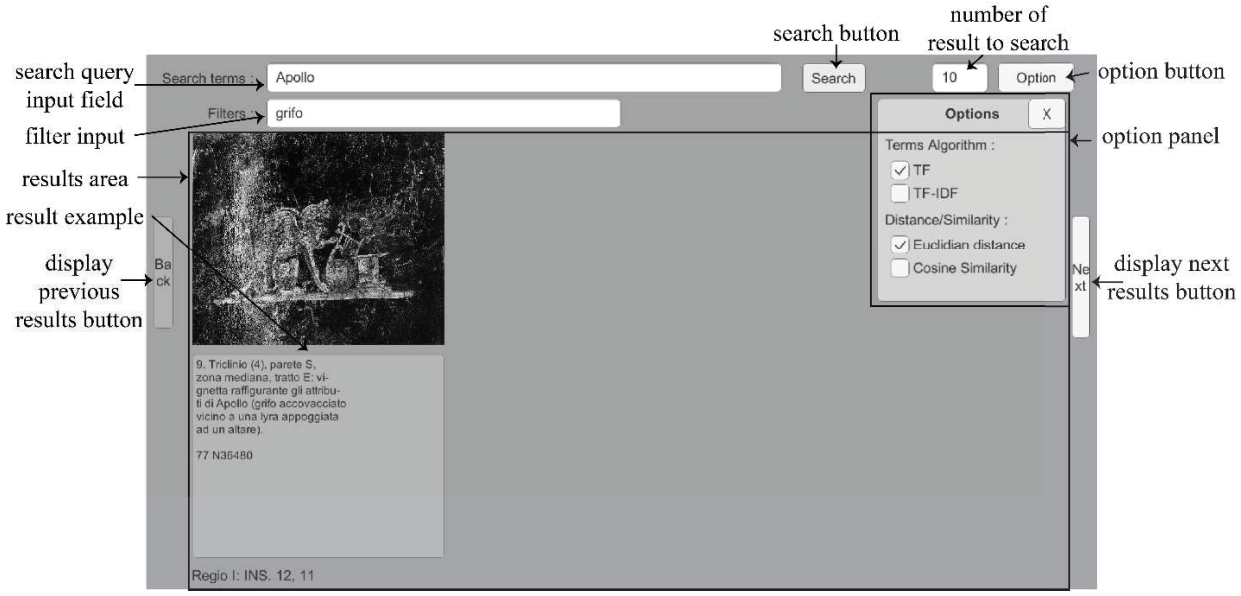

Figure 27: Overview of the Research Tool: Word Look Up Area. *O*verview of the search tool, with an example of a "Apollo" search query and the filter "grifo" with top 10 results to return. All the components descriptions are displayed and assigned with arrows. The results area and option panel are boxed. The option panel contains the search options examined chapter 6. The user can pick one term frequency option and one comparison option by clicking on toggles. The search displayed in the figure presents one result after clicking on the search button.

6.7 Word Similarity

We have examined how we used the OCR results from chapter 4 to explore the data in PPM. This section is dedicated to another way to search and use the text data to find connection between artifacts of PPM.

6.7.1 LDA

To compare and analyze captions together we needed to extract feature vector from them. First, we got rid of line breaks and dashes that were in the caption text file from the OCR process. Indeed, the captions in the books are laid out in columns which forced the writers to cut a lot of the words with dashes to create line breaks. To create feature vectors for each caption we used the Latent Dirichlet Allocation (LDA) method. LDA states that each document in a corpus is a combination of fixed numbers of topics. A topic has a probability of generating various words, where the words are all the observed words in the corpus. By calculating the likelihood of word co-occurrence, we can uncover these "hidden topics" in the text. In our case, the LDA model would generate a predefined number of topics from PPM, represented by a distribution of words. Then each caption would be represented by a distribution of these topics, aka its feature vector. To generate the LDA model or the caption feature vectors, there are 5 preparatory steps:

- Tokenization: Splits all the corpus into sentences and the sentences into words. It also makes all the words lowercase and removes punctuation.
- Words that have fewer than 3 characters are removed.
- Numerical values are removed.
- All common Italian stop-words are removed (most common words in the language).
- Stemming: words are reduced to their root form.

All those steps have been performed using the NLTK (Natural Language Tool Kit) library specific to the Italian language. After these steps, the LDA model is ready to be calculated by defining the bag words:

- Mapping each existing word of the corpus with an ID number and a number of occurrences (creation of a dictionary).
- Mapping each caption with the number of words it contains and their number of occurrences in that same caption (bag of words).

From these two elements and a designated number of topics, we can generate and save an LDA model. Selecting the number of topics to use is tricky and requires the performance of various iterations. We collected three different models to observe:

- 10 topics (illustrated in Figure 28)
- 50 topics
- 100 topics

In the stemming step, we wanted to make sure the cardinal coordinates information (North, South, West, East) would be taken into account in the feature vector. Indeed, the captions often include detailed spatial information to specify where the image is located on a wall in a room in a house. For instance, a caption might indicate that an image was located in the middle horizontal zone of the left vertical track of the northern wall of certain room which in a specific house. This is critical to attempt to understand what kind of decoration is found in what kind of space. For example, the northern wall would be described as “parete N” in the caption; however, by default, the stemming process of the LDA cuts the word’s ending and any word shorter than three letters. Therefore, “parete N” would become “par”, and we would lose the cardinal

information. To avoid this, we pre-processed the text and moved the cardinal letter to the beginning of the word so the stemming would not get rid of it.

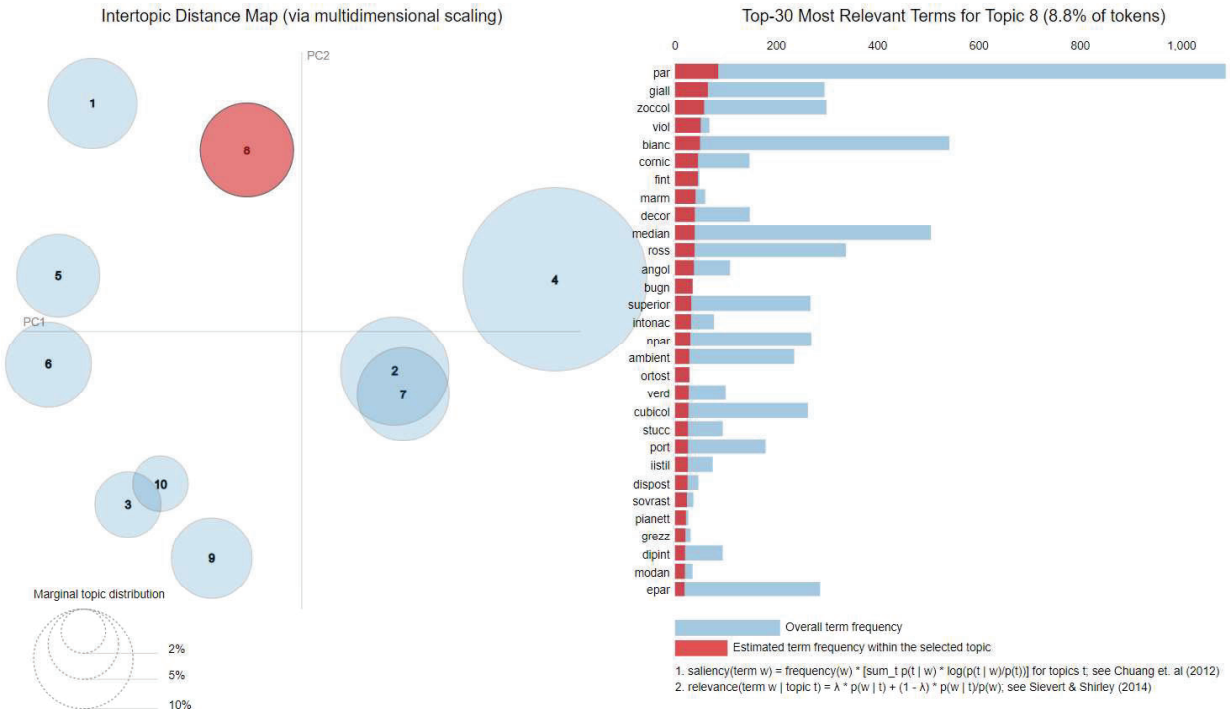

Figure 28: LDA model 10 Topics Visual. Visual on the LDA model with 10 topic word distributions. The topic number 8 is highlighted in red on the left. The word distribution of that topic is displayed on the right.

For instance, “parete N” would be transformed into “Nparete” which would be stemmed to be “npar”. The captions also sometimes contain a fresco’s style designation, classifying it as one of the four main styles of Roman wall painting, appearing in the text as “stile I”, “stile II”, etc. To conserve the style information in the feature, we changed the wording to “Istile,” which gets transformed into “istil”. After the LDA creates a topic distribution, we can observe in our results that the algorithm did consider topics containing cardinal information and style information.

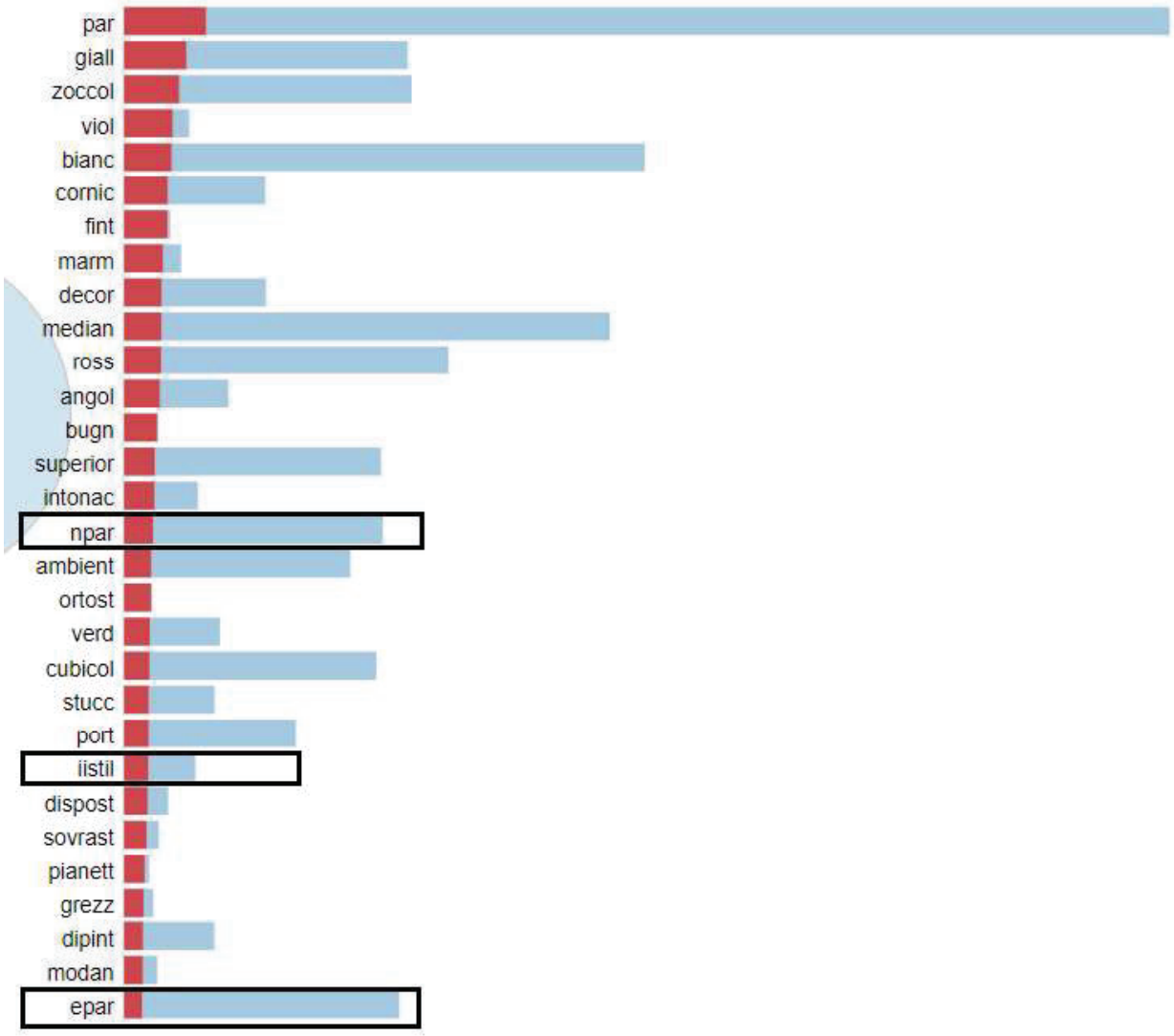

Figure 29: Zoom on Words. Zoom on the word distribution of topic number 8. “npar”, “iistil”, and “epar” are emphasized. The blue parts of the bars represent the overall term frequency over the whole data. The red parts of the bars represent the estimated term frequency within the topic.

Figure 28 is the illustration of the topic distribution created with the LDA model with 10 different topics. In Figure 29, we zoomed in on the word distribution of topic number 8 so that we can observe that one of the predominant words is “npar”, “iistil” and “epar”. This means that topic 8 contains cardinal information and style information. It also implies that we could detect captions describing images similar to one another containing location information in the houses

of Pompeii in their feature vectors. We could potentially answer questions such as such as whether bird representations are often located on the southern part of walls. We used the feature vectors of the captions from the LDA model to calculate a distance matrix of each vector to one another using the cosine similarity. This is the matrix that is used in the research tool to return the captions that are found to be similar to one another.

6.7.2 Caption similarity search

Within the software research tool, the user has access to the similarity calculation area by clicking on either the image or caption on one of the image/caption pairs returned.

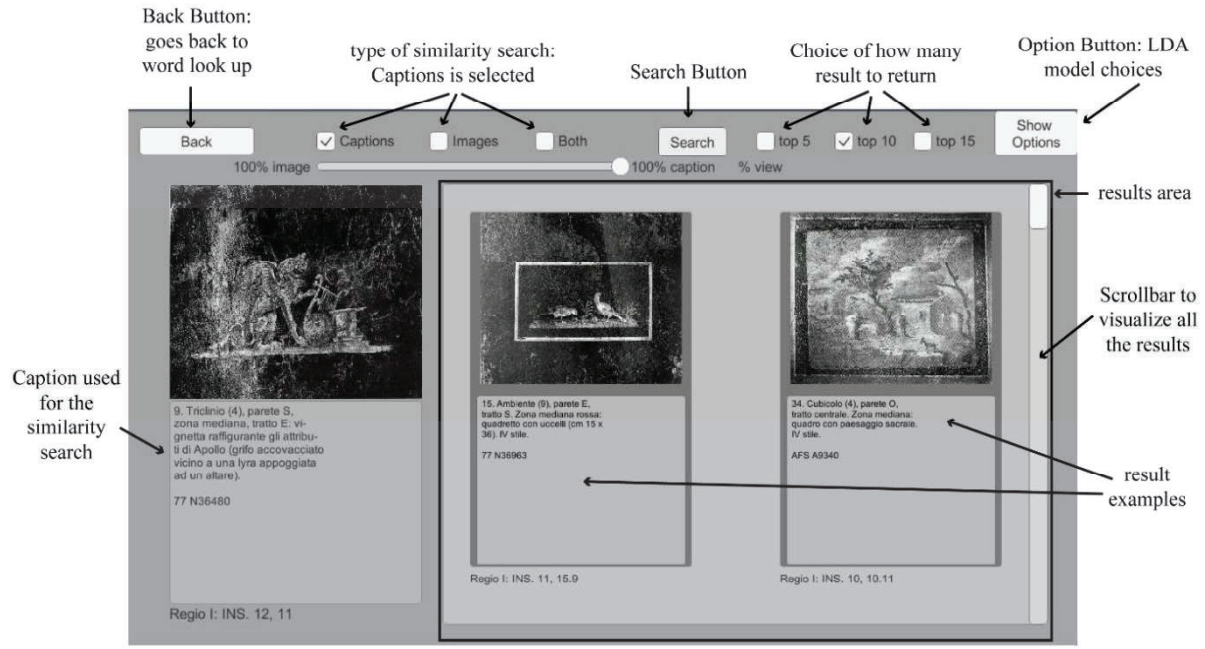

Figure 30: Caption Similarity Search Overview. Overview of the similarity search area of the tool. The legend explains the different functionality of the tool. The image/caption displayed on the left is used for the caption similarity displayed in the result area. The results showed on this figure are from an LDA model of 50 topics and 500 passes.

Once they have clicked on that image or caption, they will be presented on a screen where they can perform similarity search on the image they have clicked on. Figure 30 is the

tool's view of the similarity search area. The parts associated with the caption similarity calculation have been emphasized.

We have calculated the three LDA model: 10 topics, 50 topics, and 100 topics. We put in the software research tool the option to pick one of the models to use for the similarity search. We have precalculated in advance the distance matrices for each model. Therefore, when a user clicks on the button search, it takes the distance matrix of the selected LDA model, looks for the row containing the feature vector from the caption on the left and returns, ranked in order, the most similar ones with in the whole corpus (volume 2 data only so far).

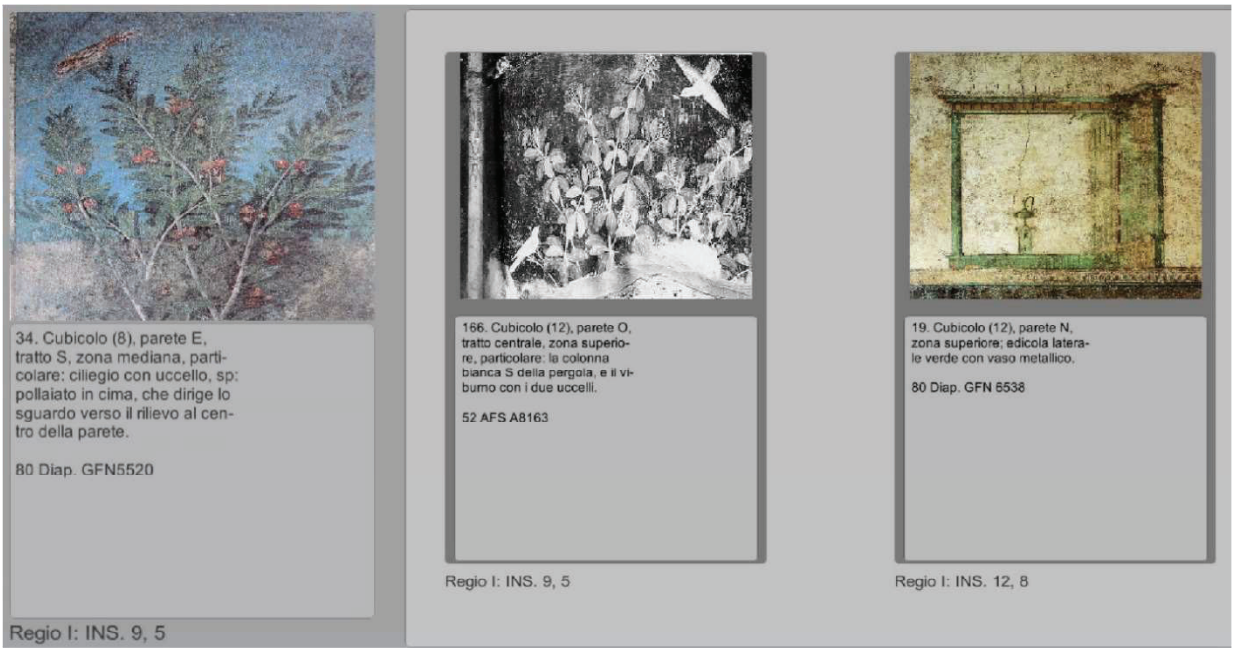

Figure 31: Caption Similarity Example. Example of a caption similarity search result with the search tool. The image/caption pair on the far left is the one used for the similarity search. the two image/caption pairs on the right are the top 2 results from a caption similarity search using an LDA model with 100 topics.

Figure 31 illustrates the results of a caption similarity search using an LDA model with 100 topics (only the top 2 images are visible). The results are returned fast because the distance

matrices have been pre-calculated. There is no significant increase in speed of response going from 10-50-100 topics either. What takes the longest for the tool is displaying the images within Unity.

7 Image Similarity

We have presented how one user of the tool can look up image/caption pairs from PPM using either terms or using an entire caption. In this chapter we study how to use the image information from the pair to retrieve artifact connections.

7.1 Previous Analysis

We had explored the feasibility of finding a way to classify the images, and possibly create a program to automatically annotate the image of PPM. It would have been a program that classifies each image into a category such as: Maps, Mosaics, Ruins, Schemas, Murals, Ruins with mural. These categories were too broad, and a lot of images of PPM are composed with elements from these different categories. Although the classification was presenting decent results for these categories, they are too arbitrary to be accurate enough and useful. However, this was handy to categorize and identify very quickly the images that were maps. Thanks to these pre-categorizations, we have put in the database if the image is a map or not. This is what we used to index all the images since they reset the number to 1 (see chapter 5).

To improve our image classification results, we retrained the last layer of a pre-trained Inception V3 [29] CNN architecture. We got the image training set by going over manually images from the volume 1 of PPM, and manually assigning them to a category. Once we had at least a hundred images in each, we used them to retrain and adjust the weights of the CNN. Once retrained, we tested it on all the images of volume 2. In information retrieval, the quality of search results is typically described in terms of how the retrieval results match the true results. To do this, we calculate the following: true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

7.1.1 Accuracy results

Accuracy: The accuracy describes how often the classifier is correct.

With the formula: Acc = (TP+TN)/total.

The accuracy of each class is:

1. Maps: 0.997
2. Mosaics: 0.979
3. Ruins: 0.916
4. Schemas: 0.986
5. Murals: 0.944
6. Ruins with Murals: 0.939

7.1.2 Error rate results

Misclassification Rate or Error Rate: The error rate describes how often the classifier is wrong: ER = (FP+FN)/total.

1. Maps: 0.003
2. Mosaics: 0.021
3. Ruins: 0.084
4. Schemas: 0.014
5. Murals: 0.056
6. Ruins with Murals: 0.061

7.1.3 Recall results

What percentage of the correct results does the model predict? That is the question the recall represents. It can be defined as: Recall = TP/(TP + FN).

Sensitivity or Recall results:

1. Maps: 0.978
2. Mosaics: 0.558
3. Ruins: 0.913
4. Schemas: 0.739
5. Murals: 0.877
6. Ruins with Murals: 0.858

7.1.4 Precision Results

What percentage of the predicted matches were correct? This is what the precision answers. Precision can be defined as: Precision = TP /(TP + FP).

1. Maps: 0.994
2. Mosaics: 0.674
3. Ruins: 0.810
4. Schemas: 0.739
5. Murals: 0.964
6. Ruins with Murals: 0.841

7.2 A New Road

The goal of our work is to develop a functionality that could help archeologists and historians make connections between images that they might not have been exposed to by just looking at the paper version of PPM. This functionality is calculating a degree of similarity between the image of PPM. One user could find quickly, for instance, an artifact's image with the same color range or same motifs as one they are interested in.

7.2.1 Transfer Learning

To implement this similarity search with images, we had to extract features from each image. Using transfer learning and comparing CNNs' feature extraction will answer the research questions of which CNN architecture would work best for calculating similarity between archaeological images.

Transfer learning is an important tool in machine learning to solve the basic problem of insufficient training data. It tries to transfer the knowledge from the source domain to the target domain by relaxing the assumption that the training data and the test data must be independent and identically distributed. We wanted to collect the feature from a pre-trained CNN to transfer into another model.

We first used the Inception V3 pre-trained on 1,000 classes on the ImageNet dataset which has over a million images, meaning that all the weights of the CNN were already adjusted to detect over 1,000 types of various images. We ran it through our PPM images from Volume 2 and it provided us with a 1,000 long feature vector for each image.

Once we had this data, we implemented a K-Nearest Neighbors (KNN) algorithm to classify the images into 10 classes. First, we picked a feature vector far away from all the other

ones as the first seed, a second vector was selected the furthest away from the first one, a third the furthest away from the first and second vectors, and so on until we had 10 seeds. We then added all the feature vectors into clusters according to how close they were to a seed and recalculated the seeds according to the vectors in the clusters, repeating that step until the seeds did not change anymore.

Figure 32 and Figure 33 show a sample of each class that the KNN created with the features from Inception V3. The following list is a description of each category as one could observe:

- 0: 183 images, mostly images of ruins, black and white.
- 1: 52 images, frescoes containing square structures.
- 2: 69 images, long walls with rectangular shapes (either vertical or horizontal).
- 3: 386 images, maps and schemas, and complex shapes.
- 4: 47 images, close-ups black and white, flying people or animals.
- 5: 62 images, closeups with round shapes.
- 6: 79 images, people, and animal representations.
- 7: 30 images, animals, and angels.
- 8: 274 images, big walls.
- 9: 259 images, big spaces.

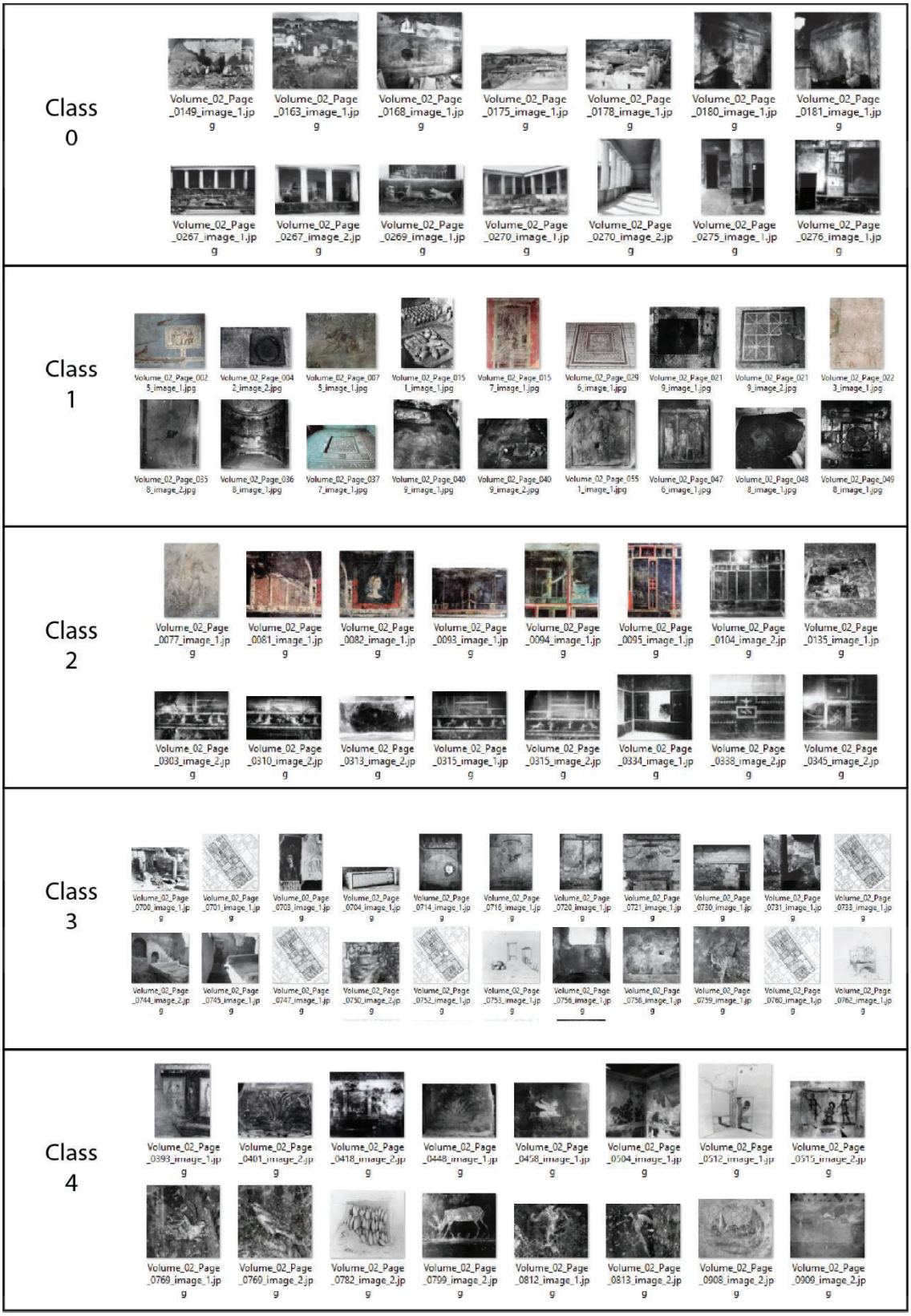

Figure 32: KNN Results Classes 0-4.

Figure 33: KNN Results Classes 5-9.

The CNN was clearly picking out some patterns. For instance, all the maps were successfully put in the same class, although with other types of images. We concluded that these feature vectors were relevant to be used as a descriptor vector for further analysis of the images.

Hence, we prepared the feature vectors to be used inside the research tool by calculating a matrix containing how close each vector was to one another using the cosine similarity (see chapter 6).

7.2.2 Image Similarity Search

Within the software research tool, the user has access to the similarity calculation area by clicking on either the image or caption on one of the image/caption pairs returned.

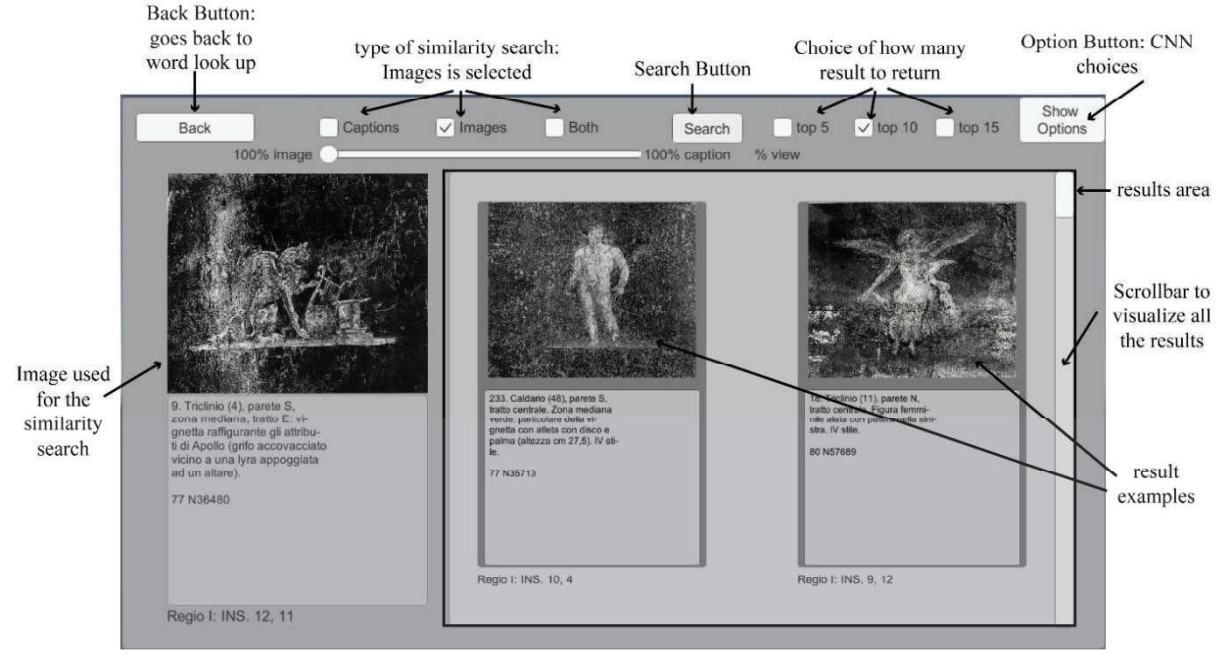

Figure 34: Image Similarity Search Overview. Overview of the similarity search area of the tool. The legend explains the different functionality of the tool. The image/caption displayed on the left is used for the image similarity displayed in the result area. The results showed on this figure are with the features from ResNet v2.

Once they have clicked on an image, they will be presented on a screen where they can perform similarity search on the image they have clicked on. Figure 34 is a tool's view of the similarity search area. The parts associated with the image similarity calculation have been emphasized.

With the image similarity, the user can choose either the CNN inception V3 [29] or Inception ResNet v2 [28] for the feature vectors to be used. We explore the difference between the two in chapter 9. Once they have picked how many results they want to visualize, and the CNN features they want to use, the tool displays image/caption pairs ranked by how similar they are to one another using the cosine method. Figure 35 is an example of a similarity results with the same image/caption pair as in Figure 31.

Figure 35: Image Similarity Example. Example of an image similarity search result with the search tool. The image/caption pair on the far left is the one used for the similarity search. the two image/caption pairs on the right are the top 2 results from an image similarity search using Inception ResNet v2.

The results are returned almost immediately because the distance matrix has been calculated beforehand. What takes the longest in the tool is loading the images which are big. Although we shrunk the images down in size to increase the tool's speed, loading the images can still slow it down.

8 Combination of word and image similarity

In the previous two chapters, we presented a way for a user to look for image/caption pairs using either caption similarity or image similarity. In is chapter we study a way to return similar image/caption pairs using both the image and text information.

8.1 Combination Similarity Search Overview

Within the software research tool, the user has access to the similarity calculation area by clicking on either the image or caption on one of the image/caption pairs returned.

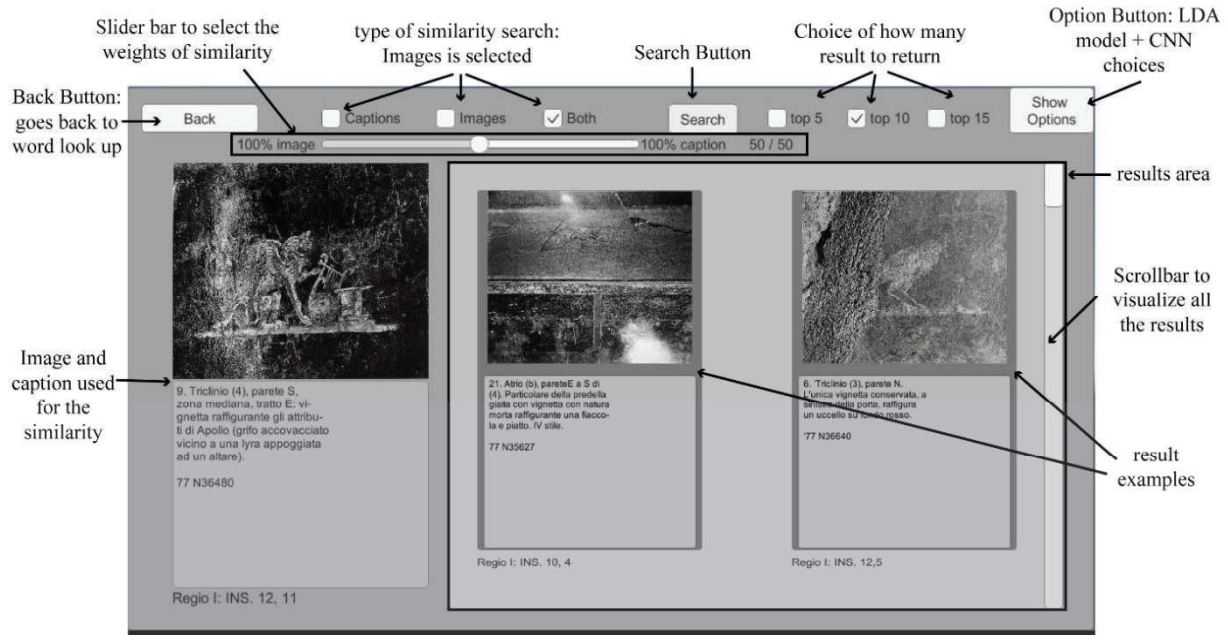

Figure 36: Overview Combination Similarity Search. Overview of the similarity search area of the tool. The legend explains the different functionality of the tool. The image/caption displayed on the left is used for the combination similarity displayed in the result area. The results showed on this figure are from the features from ResNet v2 and an LDA model with 100 topics. The similarity search has been done using weights of 0.5 (50 percent).

Once they have clicked, they will be presented on a screen where they can perform similarity search on the image/caption they have clicked on. Figure 36 is a tool's view of the similarity search area. The parts associated with the combination similarity calculation have been emphasized.

We wanted to allow the user the additional option of returning results that took into account the similarity of the image and the caption at the same time. An image similarity search returns results with similar images without considering the information in the caption. Vice versa, a caption similarity search does not take into consideration the similarity of their corresponding image. We decided to use weights to calculate how much the score of one similarity would matter to an overall combined score. The weights can be chosen by the user. We have implemented a slider for the users to pick the weight they would like. Putting the slider at 50% would return results where the similarity with the image matters as much as the similarity with the caption. This slider is visible in the overview of the combination similarity in Figure 36 and it is displayed up close in Figure 37. The user defines the exact weights by moving the slider handle along the bar. The percentages or weight amounts are displayed on both sides of the bar, see Figure 37 for illustration.

Figure 37: Slider Bar. Slider in the interactive research tool. The slider value determines the weight values for the image similarity and caption similarity. The numbers on the right side of the slider display the exact values of the weight percentages according to the handle position. The example displays a value at 42percent for the image similarity and 58 percent for the caption similarity.

The result for combination similarity with the image/caption pair from Figure 31 and Figure 35 gives the same top 2 results as the ones in the image similarity search (Figure 35). The results are calculated fast but not as fast as for the caption or image alone similarity. The calculation for the combined score is still minimal compared to the time for the software to display the images. Pulling up the images and rendering in the tool is what takes the longest when loading the results.

8.2 Combination Score Calculation

We used the previously calculated distance matrices between each feature vector to return the combination results. For a given image feature vector, we have the distances between that vector and all the other image feature vectors. We also have that information for the caption feature vectors. This information can be merged into a single vector using weights. This final vector contains the final score to consider for returning the similarity search results.

$$\overrightarrow{d_{combination}} = w_{image} * \overrightarrow{d_{image}} + w_{caption} * \overrightarrow{d_{caption}} \quad (12)$$

$\overrightarrow{d_{image}}$ is the vector of distances between the feature vector of the picked image for the similarity search and all the other image feature vectors of PPM. $\overrightarrow{d_{caption}}$ is the vector of distances between the feature vector of the picked caption for the similarity search and all the other caption feature vectors of PPM. w_{image} is the weight for the image distance chosen by the user. $w_{caption}$ is the weight for the caption distance chosen by the user. The similarity results are returned in order from the higher score to the lowest score taken from their values in $\overrightarrow{d_{combination}}$.. Another way to visualize the calculation is described in Figure 38.

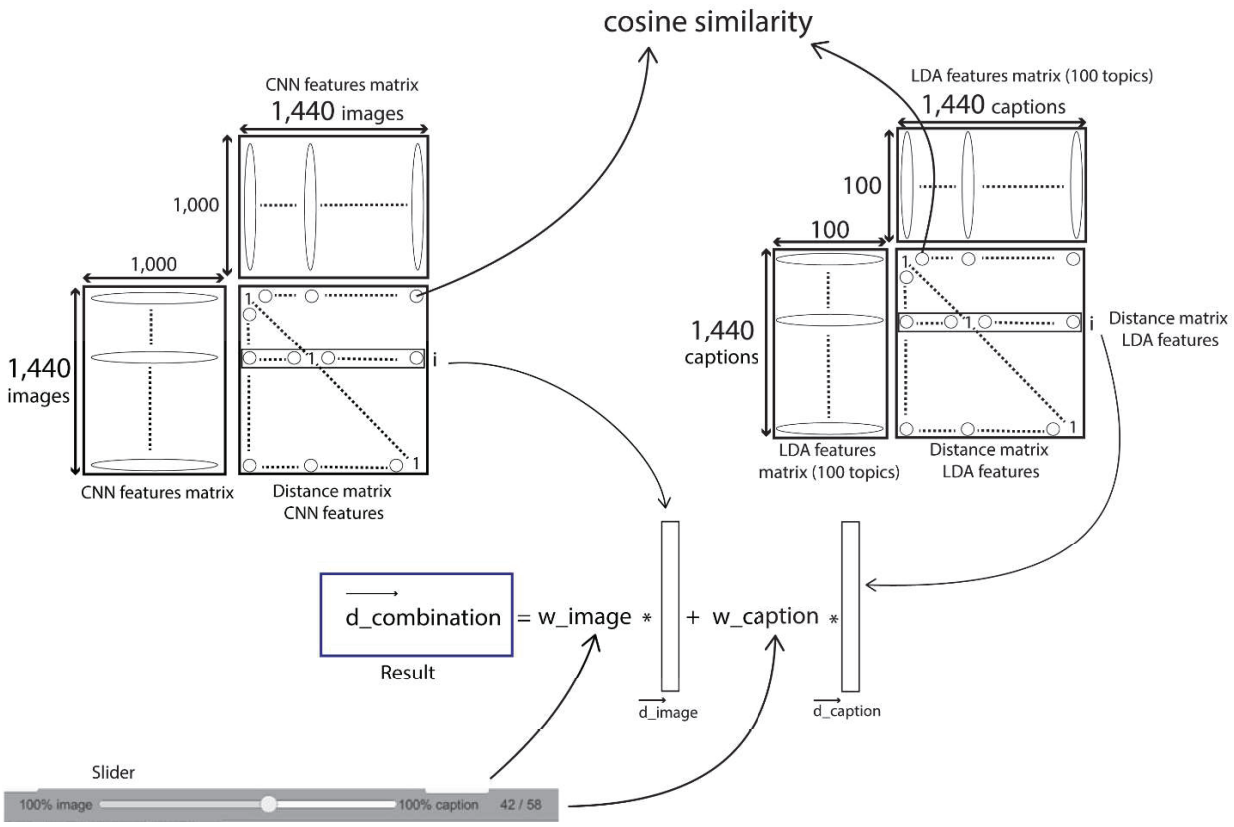

Figure 38: Distance combination vector calculation. Visual on the distance combination vector calculation. The distance matrices between each feature vector are calculated in advance using the cosine similarity function describe in chapter 6. The "i" represents the index of the image and caption that the user has clicked on. The weights are extracted using the user input from the slider. The result contains the scores used to rank the image\caption pairs to be returned. The vectors in the distance matrices are the ones used for the caption only similarity and the image only similarity.

9 Evaluation

In the tool we can look up words to return image/caption pair from volume 2 of PPM. A user can click on one and perform similarity searches on the image and/or the caption. To evaluate the functionalities and different methods described in the previous chapters, we have conducted an experiment with 8 experts. This chapter will discuss the experiment results and address various research questions.

Each expert had accessed to the tool and was asked to perform six tasks. The first five tasks introduced the functionality of the tool and the last task was a freedom question where they could use the tool as they pleased. The instructions were given in the form of a survey, after they were done with one task, they had to answer a few questions and leave comments.

9.1 Task 1: TF vs TF-IDF

In the task 1 the user was asked to look for the word “uccello” and observe the result with the TF and then the TF-IDF. The question in the survey asked them which one they preferred. The following list summarizes the results of the task with numbers:

- 75% picked TF-IDF better.
- 50% of the experts did not notice a difference between the two results.
- 50% of the experts thought that TF-IDF was returning more precise results.

From these results, we can conclude that the TF option could be removed from the tool and that TF-IDF does present a noticeable difference in performance when looking for terms.

Comparing Euclidian and Cosine similarity was not part of the test, but we know that the Euclidean distance is more relevant when searching with one term. The two methods should be compared when doing a multiple search term query.

9.2 Task 2: Inception V3 vs ResNetV2

During task 2, the experts were asked to perform a similarity search using the image/caption pair of Figure 39. The image represents a bird on a branch with a dark rectangular background. The following list summarizes the results of the task with numbers:

- 87.5% preferred the results of the Inception V3, because the ResNetV2 first results did not contain birds. The one person who picked ResNetV2 agreed with the previous statement but found the other results (ranked lower) about birds of ResNetV2 more relevant than the result in Inception V3.

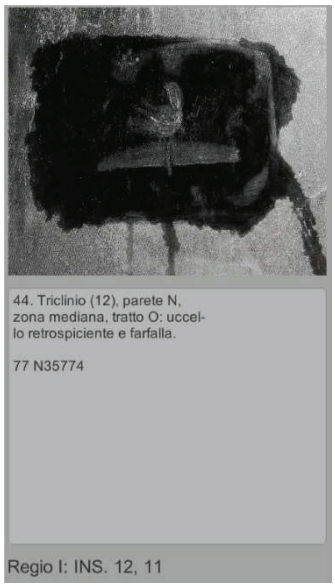

Figure 39: Image/Caption pair. Image/caption pair used to perform the similarity search for task 2, task 3 and task 4 of the evaluation experiment.

We can conclude that for this particular image the Inception V3 was more relevant. Task 6 does give more input on the comparison of Inception V3 and ResNetV2. It seems that the InceptionV3 does a better job at comparing similarities for images with geometric shapes and the overall global shape of the image whereas the ResNetV2 does a better job finding similarities between images with curvy motifs.

We performed three other tests with images where we asked five individuals (four of them with archaeological background) to choose a group of ranked images they found the most similar given a specific image (no access to the captions). Figure 40 present the test 1, 100% of the experts chose the results provided by ResNetV2. Figure 41 presents the test 2, 100% of the experts chose the results provided by Inception V3. Those result help identify the usefulness of the CNNs and where to they excel the most. Figure 41 is a close up of a fresco motif corner, the Inception V3 features pick up on it more than ResNetV2 features. Indeed, we can observe the cornet of another motif being return higher up. In the test 1, the global look of the images returned by InceptionV3 do look alike the picked one, but it neglects the faint image of the woman in the center. For these type of images, ResNetV2 features are more useful. When presented with a scene with many elements as shown in test 3 (Figure 42), the InceptionV3 features and ResNetV2 features are closer competitors and both provide somewhat relevant results, 40% of the individuals picked the ResNetV2 features choice. The hypothesis that Inception V3 is better at recognizing geometric shapes and that ResNetV2 is more tuned to details within the images seems to be valid according to the results of the 3 test images and observing the user behavior from task 6.

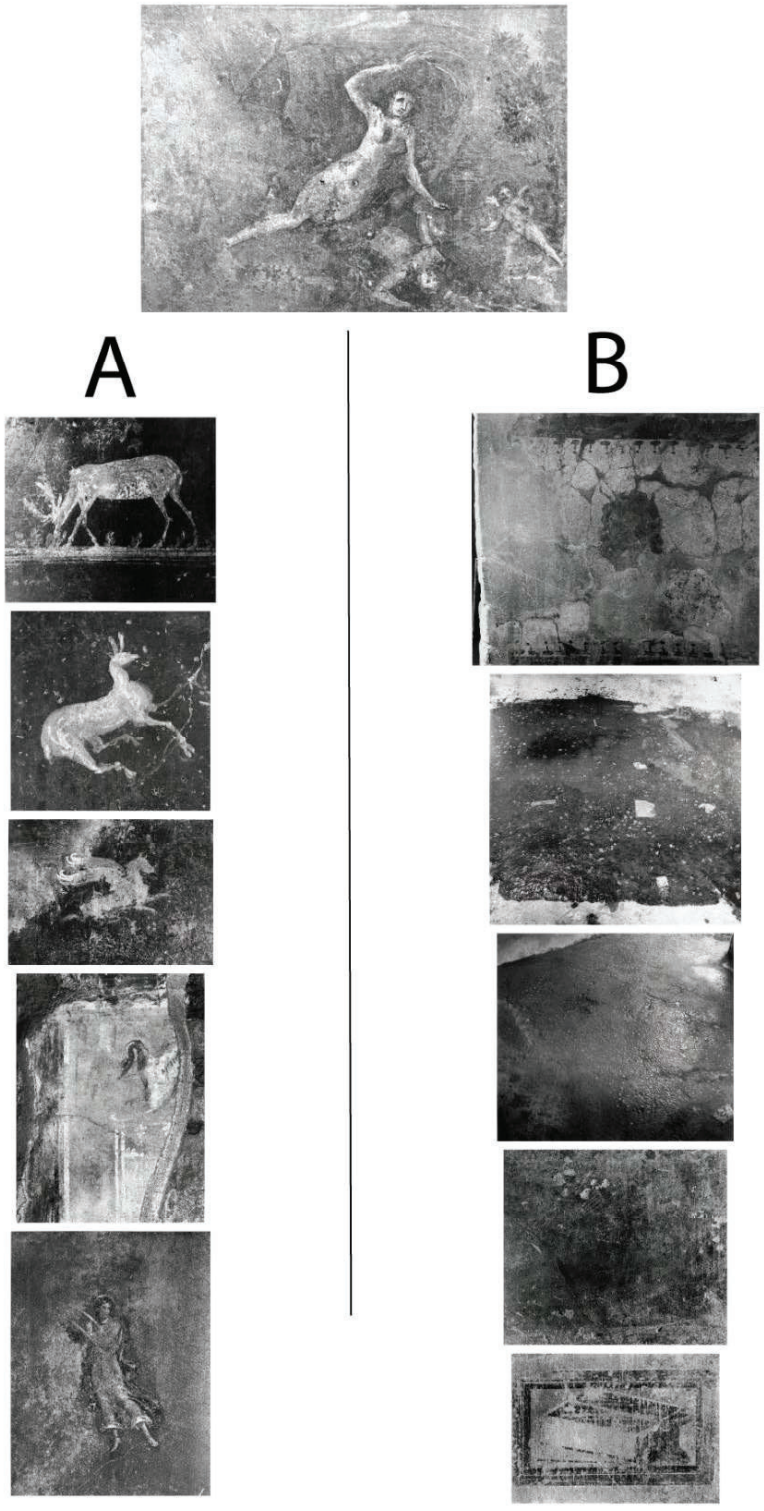

Figure 40: Test 1. This image was presented to experts, they were asked to pick a group of image A or B in regard to the image on the top of the page. In this test, group A was the top 5 results from the image similarity search performed with ResNet v2. Group B was the top 5 results from the image similarity search performed with Inception V3.

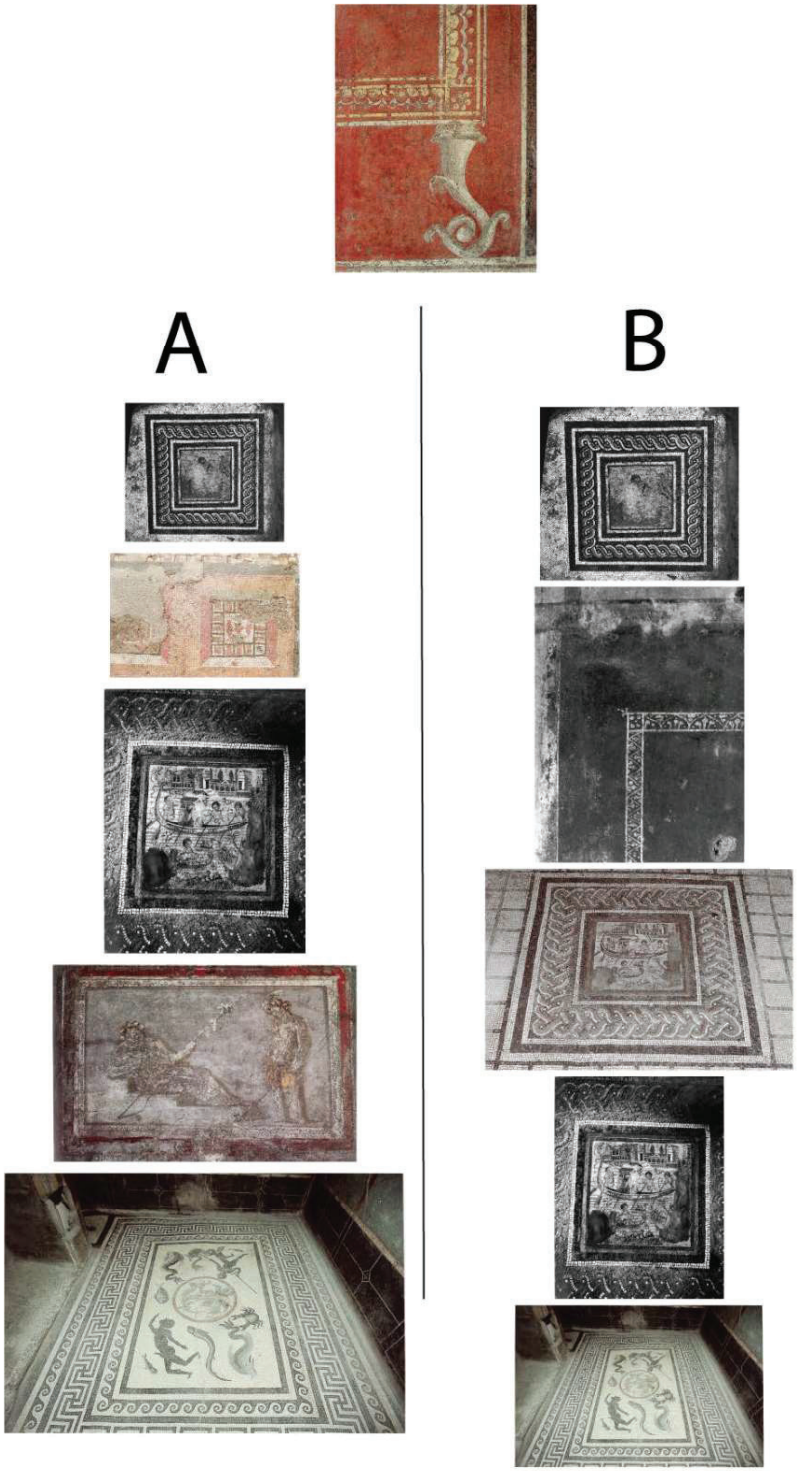

Figure 41: Test 2. This image was presented to experts, they were asked to pick a group of image A or B in regard to the image on the top of the page. In this test, group A was the top 5 results from the image similarity search performed with ResNet v2. Group B was the top 5 results from the image similarity search performed with Inception V3.

Figure 42: Test 3. This image was presented to experts, they were asked to pick a group of image A or B in regard to the image on the top of the page. In this test, group A was the top 5 results from the image similarity search performed with Inception V3. Group B was the top 5 results from the image similarity search performed with ResNet v2.

9.3 Task 3: 50 Topics vs 100 Topics (LDA models)

During task 3, the user is asked to switch the parameters and to do a caption similarity search on the same image/caption pair than from task 2 (see Figure 39). The following list summarizes the results of the task with numbers:

- 100% of the experts preferred the result from the 100 topics distribution.

We can conclude that we could eliminate the 50 topics distribution from the tool and experiment by building LDA models with a lot more hidden topics.

9.4 Task 4: Combination similarity vs Image similarity vs Caption Similarity

In task 4, the expert was asked to perform a similarity using 50/50 weights on image features and caption features, using the same image/caption pair as Figure 39. The options were Inception V3 and an LDA model of 100 topics. The survey then asked which one they preferred. The following list summarizes the results of the task with numbers:

- 75% of them liked the combination similarity search, they find the combination similarity search presented a wider and more precise selection.
- 25% liked all of them (image, caption, and combination search), they could see all similarity functionalities being useful depending on what they are looking for.

From these numbers and their comments, we can conclude that having all three possible searches is useful. If one user is looking for very precise results with much closeness to their image/caption, the combination search will be more useful. However, it is relevant for users to be able to not be as precise if they are looking to explore PPM; for instance when looking for similar global shape or color distribution.

9.5 Task 5: Filters

Task 5 introduces the filter functionality to the user, the survey asked them to go back to the word lookup area and to add the word “volo” in the filter (meaning “in flight” in English). This task does not have a specific question but has a comment section. 100% of them find this functionality useful and 25% of them expressed that having the filter implemented with the “AND” rather than “OR” would be more useful. Indeed, as of now, the filters return the captions if and only if they contain at list one of the words from the filters entered (“OR”).

9.6 Task 6: Freedom question

During task 6, the experts were free to use the tool as they wanted. Each expert searched different things such as:

- A specific element (swan, skull, Apollo) at a few different locations (center, upper, or lower zones on a wall).
- Mythological scenes.
- Facades and doorways.
- Snakes.
- Round shapes/motifs.
- Flowers and gardens.
- Motifs or elements (serpente, mercurio, grifo) in specific rooms
- Fountains

Then they were asked what they found the most useful for their search:

- 50% of the experts had the word look up and filters in their answer.

- 50% of the experts had the combination similarity in their answer.
- 37.5% had the image similarity search in their answer.

The caption similarity function was never mentioned as the most relevant for their freedom searches. The users were then asked what functionalities they wished the tool had. This part of the survey helped define the ideas described in chapter 10:

- 62.5% mentioned UI improvements.
- 37.5% mentioned being able to look for specific house/location.
- 25% mentioned word look up improvements.

10 Discussion and Future Work

This Chapter will bring a reflection over the entire work done and convey a summarized overall evaluation as well as explore potential future work for the project.

10. 1 Research questions and hypothesis

The goal of our research was to present and defend the theory that it is possible to help archeologist researchers to conduct and check hypothesis on historical findings using computed correlation between artifacts using image data, text data and a combination of both. To meet this objective, we answered the following research questions.

- Is it possible to find and retrieve image data and text data digitally from scanned documents with the same shape and form as PPM? Yes.
- Can the OCR algorithm run on whole pages without caring about the images being in the way? Or is pre-processing to remove text/image needed? We had to preprocess the page using the methods described in chapter 3 and 4.
- What OCR library works best for our application? We used the “tesseract” library, but we did not try to modify its source code or any other one.
- Is it possible to use text information to assist in the classification of the Pompeii images? we have detailed the work done to create various LDA models that helped with extracting features. We did implement a KNN and observed results using an LDA model of 10 topics. It was working well, that is why we went on using the other LDA models in the similarity search. The preliminary results of the LDA classification have not been detailed in this document because they were very similar to the results given by the classification of the images using KNN in chapter 7.

- Is it possible to create a computer vision algorithm capable of extracting and classifying images more efficiently than a human could? Thanks to our work we were able to extract automatically and accurately thousands of images that would have taken countless hours to do so manually. The map images were also automatically classified thanks to the work described in chapter 7.
- Is it possible to use the results of the classification to build a tool to look up Pompeii images? The classification of the map images did help in the making of the tool and was crucial to assign the images their correct indexes.
- Is it possible to detect multiple objects and/or features of an image when they are present? this a hypothesis that has not been tested. However, the image search similarity is able to return images containing similar objects. This has been proven to be true with amphora images in PPM among others (observable in KNN classification in chapter 7 and during experiments).
- Which framework or CNN architecture would be the best for each of the task that we are undertaking for image and text analysis? It seems that the evaluation from task 2 and the observation of the users from task 6 that the Inception V3 is better for recognizing global shapes and geometric patterns. ResNetV2 is better at extracting feature more detailed oriented as well as more curvy shapes. We did not try to use a CNN nor a NN to perform text analysis. However, we did conclude that, for the volume 2 of PPM, a LDA model with at least 100 topics performs better to extract features from captions/text.
- Is it better to use one CNN to classify the images all at once or to use different ones in phases? We have used different ones in phases in our work, one to classify maps and one that

we used to extract feature that we put into another classifier. We have not tested any architecture feeding input and output directly into one another.

- How can we save our content classification results? We have saved them by creating a database with an entry for a category, and table following the third normal form.
- How can we easily get content experts to evaluate our classified results? We created an interactive software tool. The tool is only on one machine for now. The software tool will be able to be online in the future. Building the tool on a WebGL platform is our plan. Tool, aka PPMExplorer, is detailed and illustrated throughout part II of the dissertation.
- How can we use the word information from the pages to help the image classification? We implemented the combination similarity search, we are able to use both the text features and the image features to ranked images by how close they are to one another, see chapter 8.
- How can we use the location information to help with the image classification? We have made sure that the LDA model was taking into account the cardinal information by preprocessing the text. However, the doorways, region and house number are not searchable at the moment.
- Can we combine the text information with the image information to get better classification results? According to the experiment with the experts, our combination similarity search does provide more precise result in regard to similarity, see details in chapter 9.
- Can we use location information to calculate similarities between images? The LDA does take into consideration the cardinal location of a wall's room or a wall area. We did not try to extract geolocation of images to calculate a distance. That is something that we believe as feasible. We could compute a distance matrix using meters of the location from each image to another. Those numbers could be used as an additional feature. It could be added to a

combination search with a new weight. That would be useful to collect a feature that describes the physical closeness of the images. The similarity in the kind of space the images are is contained in the LDA models, but it is not explicit. We could be more precise and define manually the space kind the images are in and then manually add those descriptors to the caption feature vector.

- Is there a correlation between the location of an artifact and its style? This is an archeological research question that we believe our tool can help answer.
- How can we do object detection or segmentation on images containing several different types of elements? This is something that we did not explore but only briefly thought about. In a sense, the features extracted through the CNNs potentially do object detection and segmentation but not explicitly. This is a relevant unexplored research question.
- What kind of feature vectors should we used to calculate similarities between images? We have decided to test two methods. One with Inception V3 and one with ResNetV2. We picked Inception V3 because we had done previous work with it and it was presenting good results. ResNetV2 was picked because it was bigger and had better results in the classification of the ImageNet dataset competition than Inception V3. Interestingly, our research showed that even though the ResNetV2 performed better in the ImageNet competition, it was not always outperforming Inception V3 in our PPM dataset.
- Can we use results from a “random” CNN as a feature vector for one image? We experimented with Inception V3 and ResNetV2, they both extracted relevant features.
- Should we separate an image that has several elements into different parts in order to analyze it by comparing feature vectors on each part, or should we use a CNN specialized in object detection? We have not experimented with that idea. However, with our work as groundwork

it could be possible to ask a user to select an area of an image, they like to perform a new type of similarity search on the area they have selected (possibly with a mouse select).

- Is it possible to build a tool to help content experts rapidly evaluate results of image classification? We did not build a tool to evaluate any classification results.

10.2 Potential Functionalities

While conducting this research, we identified several areas of future research and features to add to our PPM exploration tool. These are listed below (not ranked in any order).

- Add all the data from all the volumes to the tool. Adding all the other images and caption from the PPM volumes in PPMExplorer will create more correlation and more searchable data to the user.
- Perform regular testing and evaluation to improve user experience. Getting user feedback is essential to be aware of what archeologists and historians would want to be able to do with the tool. Then we can assess and maybe implement new functionalities as well as improvements on the user interface.
- Add the possibility to choose the type (AND, OR, NOT) of filter for each word added in the filter input field. This would help the user to be more precise when using the word search functionality.
- Add the possibility to a user to send feedback or a bug report within the tool.
- Have access to a suggested word list within the tool. This would help users to find inspiration on what word to look for as well as informing the lexical used the PPM volumes.

- Being able to copy and paste. User could copy and paste words they have in a separate document to search them or even take one from a displayed caption. This would make the word search functionality more practical.
- Add a map display of where the image/caption pairs are located in Pompeii (simplified map).
- Add the possibility to zoom in and out on the results. To be able to see how many of the returned results at once and to be able to see one image full screen.
- Add the functionality to look by houses (name or number) and region of Pompeii.
- Having the plural or words of the same family as part of the results or suggested search. This could be implementable using the techniques of the pre-processing of the text for the LDA.
- Fix UI elements such as: scrollbar, button and animation, user feedback on loading, resetting views, displaying number of available results, displaying score of similarities, displaying the page name of the image location within PPM.
- Add other known sources of the same element described in PPM. Images from the web.
- Have a colorized version of one image generated by either new technology algorithm used to automatically colorized image or images of the same element but colored within PPM or with another source (tourist photos? Web?).
- Make the tool able to be modifiable, to some extent, by the users to improve results and calculation as the tool gets utilized. For instance: caption OCR corrections, adding labels, marking images as irrelevant or relevant.

11. Conclusion

The thesis of this dissertation is that image processing combined with natural language processing, data mining and machine learning provide the means to explore archaeological and historical data in new ways. Our work opens doors to re-discover Pompeii by taking a new look into the *Pompei: Pitture e Mosaic* volumes. Part I explored information retrieval, from scanned paper version of the volumes to usable image data and text data. Part II examined data analysis, how to make best usage of the retrieved data and how to connect them in a smart manner that can help archeologists and historians test hypothesis and find new theories on the world's history.

During the information retrieval process, we favored image processing heuristic methods over machine learning methods. The heuristic methods have shown good and efficient results. Nowadays, there is a lot of hype around using big data to solve any problem. Data mining does present a lot of promising outcomes in a wide range of domains. However, one constraint to the use of these methods is having access to reliable and robust data in a huge bulk. This data is essential for training and testing architectures. Moreover, because of the exponential progress in computer power which provides us with almost unlimited mathematical capabilities, a lot of the outcomes relies on the quality and quantity of the data. This works shines lights on the fact that even with a modest data set of a few thousand images and captions data, it is possible to build a system that works and that is useful. Not only does is provide fruitful information but it also produces groundwork for a multitude of improvement possibilities. One of them being the preparation of an environment ready to receive more data, to grow, and to eventually be used for various data mining methods.

The data analysis part of our work has made use of transfer learning methods which was good way to use big data methods having little data. We did use heuristics functions such as KNN to check the integrity of the features data that we extracted from the machine learning methods. The results evaluated show the great potential of machine learning to be applied to analyzing archeological images.

Our research leads us to conclude that we are just standing at the start of the road towards finding even greater artifact correlations. We have proven that using a combination of heurist methods and data mining methods with combining both image and text information presents tremendous potential. There is still much to be done, but the journey ahead is an exciting one.

12. References

[1] Charalampos Georgiadis et al. “Fast and accurate documentation of archaeological sites using in the field photogrammetric techniques”. In: International Archives of Photogrammetry and Remote Sensing 33 (2000), pp. 28–32.

[2] Sang-Bum Kim et al. “Some effective techniques for naive bayes text classification”. In: IEEE transactions on knowledge and data engineering 18.11 (2006), pp. 1457–1466. Title Suppressed Due to Excessive Length 7

[3] Laurens van der Maaten et al. “Computer vision and machine learning for archaeology”. In: Proceedings of Computer Applications and Quantitative Methods in Archaeology (2006), pp. 112–130.

[4] L Van Der Maaten and E Postma. Towards automatic coin classification. na, 2006.

[5] Ziqiang Wang and Xu Qian. “Text categorization based on LDA and SVM”. In: 2008 International Conference on Computer Science and Software Engineering. Vol. 1. IEEE. 2008, pp. 674–677.

[6] Jingnian Chen et al. “Feature selection for text classification with Naïve Bayes”. In: Expert Systems with Applications 36.3 (2009), pp. 5432–5435.

[7] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee. 2009, pp. 248–255.

[8] Alex Krizhevsky and Geoff Hinton. “Convolutional deep belief networks on cifar-10”. In: Unpublished manuscript 40.7 (2010), pp. 1–9.

[9] Yuval Netzer et al. “Reading digits in natural images with unsupervised feature learning”. In: (2011).

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet classification with deep convolutional neural networks”. In: Advances in neural information processing systems. 2012, pp. 1097–1105.

[11] Xiaolu Yang et al. “An improved median-based Otsu image thresholding algorithm”. In: Aasri Procedia 3 (2012), pp. 468–473.

[12] Samih Al Rawashdeh et al. “Archaeological Documentation Based on Geomatics Techniques for Jerash Historical Site”. In: International Journal of Applied Science and Engineering 11.3 (2013), pp. 235–244.

[13] Nikolaos Aletras, Mark Stevenson, and Paul Clough. “Computing similarity between items in a digital library of cultural heritage”. In: Journal on Computing and Cultural Heritage (JOCCH) 5.4 (2013), pp. 1–19.

[14] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: Advances in Neural Information Processing Systems 26. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 3111–3119. url: http : / / papers . nips . cc / paper / 5021 -distributed-representations-of-words-and-phrases-and-theircompositionality.pdf.

[15] Rodrigo Moraes, Jo˜aO Francisco Valiati, and Wilson P Gavi˜aO Neto. “Document-level sentiment classification: An empirical comparison between SVM and ANN”. In: Expert Systems with Applications 40.2 (2013),pp. 621–633.

[16] Deodato Tapete et al. “Integrating radar and laser-based remote sensing techniques for monitoring structural deformation of archaeological monuments”. In: Journal of Archaeological Science 40.1 (2013), pp. 176–189.

[17] Vishwanath Bijalwan et al. “KNN based machine learning approach for text and document mining”. In: International Journal of Database Theory
and Application 7.1 (2014), pp. 61–70.

[18] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).2014, pp. 1532–1543.

[19] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: arXiv preprint arXiv:1409.1556 (2014).

[20] Bruno Trstenjak, Sasa Mikac, and Dzenana Donko. “KNN with TF-IDF based framework for text categorization”. In: Procedia Engineering 69 (2014), pp. 1356–1364.

[21] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: European conference on computer vision. Springer. 2014, pp. 818–833.

[22] Christian Szegedy et al. “Going deeper with convolutions”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, pp. 1–9.

[23] Bing Xu et al. “Empirical evaluation of rectified activations in convolutional network”. In: arXiv preprint arXiv:1505.00853 (2015).

[24] Plamen Angelov et al. “Advance in Computational Intelligence System”. In: Intelligence (2016).

[25] Federico Becattini et al. “Imaging novecento. a mobile app for automatic recognition of artworks and transfer of artistic styles”. In: Euro-Mediterranean Conference. Springer. 2016, pp. 781–791.

[26] Kaiming He et al. “Deep residual learning for image recognition”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 770–778.

[27] Ji Young Lee and Franck Dernoncourt. “Sequential short-text classification with recurrent and convolutional neural networks”. In: arXiv preprint arXiv:1603.03827 (2016).

[28] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: arXiv preprint arXiv:1602.07261 (2016).

[29] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 2818–2826.

[30] Mark Eramian et al. “Image-based search and retrieval for biface artefacts using features capturing archaeologically significant characteristics”. In: Machine Vision and Applications 28.1-2 (2017), pp. 201–218.

[31] Gao Huang et al. “Densely connected convolutional networks”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, pp. 4700–4708.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: Communications of the ACM 60.6 (2017), pp. 84–90.

[33] Sathit Prasomphan and Jai E Jung. “Mobile application for archaeological site image content retrieval and automated generating image descriptions with neural network”. In: Mobile Networks and Applications 22.4 (2017), pp. 642–649.

[34] Jun Zhou et al. “Identifying designs from incomplete, fragmented cultural heritage objects by curve-pattern matching”. In: Journal of Electronic Imaging 26.1 (2017), p. 011022.

[35] Abdelhak Belhi, Abdelaziz Bouras, and Sebti Foufou. “Leveraging known data for missing label prediction in cultural heritage context”. In: Applied Sciences 8.10 (2018), p. 1768.

[36] Simon H Bickler. “Machine learning identification and classification of historic ceramics”. In: Archaeology 20 (2018).

[37] Luke N Darlow et al. “CINIC-10 is not ImageNet or CIFAR-10”. In: arXiv preprint arXiv:1810.03505 (2018).

[38] Yohanes Gultom, Aniati Murni Arymurthy, and Rian Josua Masikome.“Batik classification using deep convolutional network transfer learning”. In: Jurnal Ilmu Komputer dan Informasi 11.2 (2018), pp. 59–66.

[39] Bashir Kazimi et al. “Deep Learning for Archaeological Object Detection in Airborne Laser Scanning Data”. In: 2nd Workshop On Computing Techniques For Spatio-Temporal Data in Archaeology And Cultural Heritage (COARCH 2018). Vol. 2230. 2018, pp. 21–35.

[40] Nada A Rasheed and Md Jan Nordin. “Classification and reconstruction algorithms for the archaeological fragments”. In: Journal of King Saud University-Computer and Information Sciences (2018).

[41] Claudia Engel et al. “Computer vision and image recognition in archaeology”. In: Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse. 2019, pp. 1–4.

[42] Alexander V Gayer, Yulia S Chernyshova, and Alexander V Sheshkus. “Effective real-time augmentation of training dataset for the neural networks learning”. In: Eleventh International Conference on Machine Vision (ICMV 2018). Vol. 11041. International Society for Optics and Photonics. 2019, p. 110411I.

[43] Nesreen Hamdallah Jboor et al. “Towards an inpainting framework for visual cultural heritage”. In: 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). IEEE. 2019, pp. 602–607.

[44] Hamed Jelodar et al. “Latent Dirichlet Allocation (LDA) and Topic modeling: models, applications, a survey”. In: Multimedia Tools and Applications 78.11 (2019), pp. 15169–15211.

[45] Uday Kulkarni et al. “Classification of Cultural Heritage Sites Using Transfer Learning”. In: 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM). IEEE. 2019, pp. 391–397.

[46] Peter McKeague et al. “Mapping Our Heritage: Towards a Sustainable Future for Digital Spatial Information and Technologies in European Archaeological Heritage Management”. In: Journal of Computer Applications in Archaeology 2.1 (2019), pp. 89–104.

[47] V Palma. “TOWARDS DEEP LEARNING FOR ARCHITECTURE: A MONUMENT RECOGNITION MOBILE APP.” In: International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences (2019).

[48] Cindy Roullet et al. “An Automated Technique to Recognize and Extract Images from Scanned Archaeological Documents”. In: 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW). Vol. 1. IEEE. 2019, pp. 20–25.

[49] Wei Wang et al. “Development of convolutional neural network and its application in image classification: a survey”. In: Optical Engineering 58.4 (2019), p. 040901.

[50] Taylor Arnold and Lauren Tilton. “Enriching Historic Photography with Structured Data using Image Region Segmentation”. In: Proceedings of the 1st International Workshop on Artificial Intelligence for Historical Image Enrichment and Access. 2020, pp. 1–10.

[51] Anamika Dhillon and Gyanendra K Verma. “Convolutional neural network: a review of models, methodologies and applications to object detection”. In: Progress in Artificial Intelligence 9.2 (2020), pp. 85–112.

[52] Natalia Dıaz-Rodrıguez and Galena Pisoni. “Accessible Cultural Heritage through Explainable Artificial Intelligence”. In: 11th Workshop on Personalized Access to Cultural Heritage. 2020.

[53] Hosni Qasim El-Mashharawi et al. “Grape Type Classification Using Deep Learning”. In: (2020).

[54] Farhana Sultana, Abu Sufian, and Paramartha Dutta. “Evolution of image segmentation using deep convolutional neural network: A survey”. In: Knowledge-Based Systems (2020), p. 106062.

[55] Melvin Wevers and Thomas Smits. “The visual digital turn: Using neural networks to study historical images”. In: Digital Scholarship in the Humanities 35.1 (2020), pp. 194–207.

[56] Motti Zohar, Ilan Shimshoni, and Fadi Khateb. “GIScience Integrated with Computer Vision for the Interpretation and Analysis of Old Paintings.” In: GISTAM. 2020, pp. 233–239.