Università
Ca'Foscari
Venezia

DEPT. OF SCIENZE AMBIENTALI, INFORMATICA E STATISTICA
Master's degree programme in Computer Science

Final thesis

Visual narrative of Venice through the centuries

Supervisor:
Chiar.mo Prof.
Andrea Torsello

Graduand:
Gaia Orsini
Matriculation Number 832985

Academic Year:
2016 – 2017

To my family, who patiently funded all this knowledge,
and to Matteo, who assisted me in the pursuit of it.

And to all my enemies: I will destroy you.

Abstract

Visual narrative of Venice through the centuries is an important topic for art historians all over the world. Computer vision can be an useful help for their work. In this thesis the historic images of Venice by Francesco and Giacomo Guardi are used to analyze how the representations of Venice change through the centuries. We built a tool that analyzes these images using a semi-supervised learning approach and performs topic modeling in order to understand the distribution of objects and themes represented in these images.

Contents

Chapter 1

Introduction

In this chapter it is explained what is the purpose of this thesis and how we propose to fulfill it.

1.1 The project

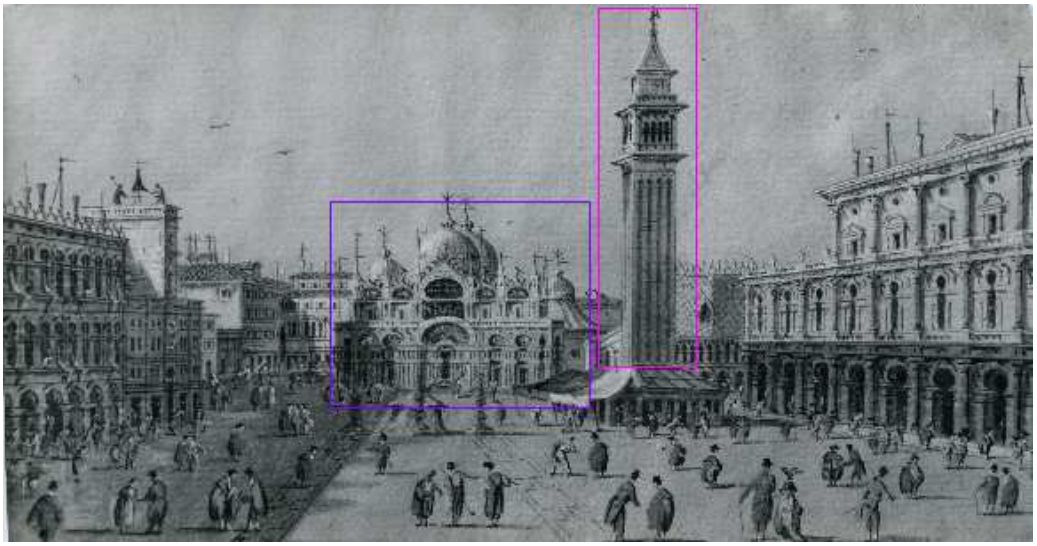
The work of this thesis is based on the University project *Visual narrative of Venice through the centuries*. The goal of this project is to study the representation of Venice through the centuries, using a database of textual and photographic material from *Fondo Morassi*.

Fondo Morassi is an art collection originally belonging to Antonio Morassi, an internationally famous art historian whose collection was obtained by Ca' Foscari in 1980. This collection is divided in sections (Artists, Collections, Various, Big, Notes) and is organized by artist or by locality. The images analyzed by this thesis are the representations of Venice by Francesco Guardi and his son Giacomo.

Antonio Morassi

The images are collected in a database. The points of interest of the images has been identified and tagged by art historians.

Given various visual representations of the city of Venice and its most prominent landmarks, the aim of this thesis is to define and extract the relevant features of the images and classify these images according to said features using topic models. These classifications will be useful in studying the changes of the representation of Venice through the centuries.



As an example, here we can see a view of *Piazza San Marco*. In the second image, the regions of interest (*chiesa, campanile*) are tagged. Every image in the corpus is tagged likewise.

1.2 Structure of the thesis

The thesis is organized as follows:

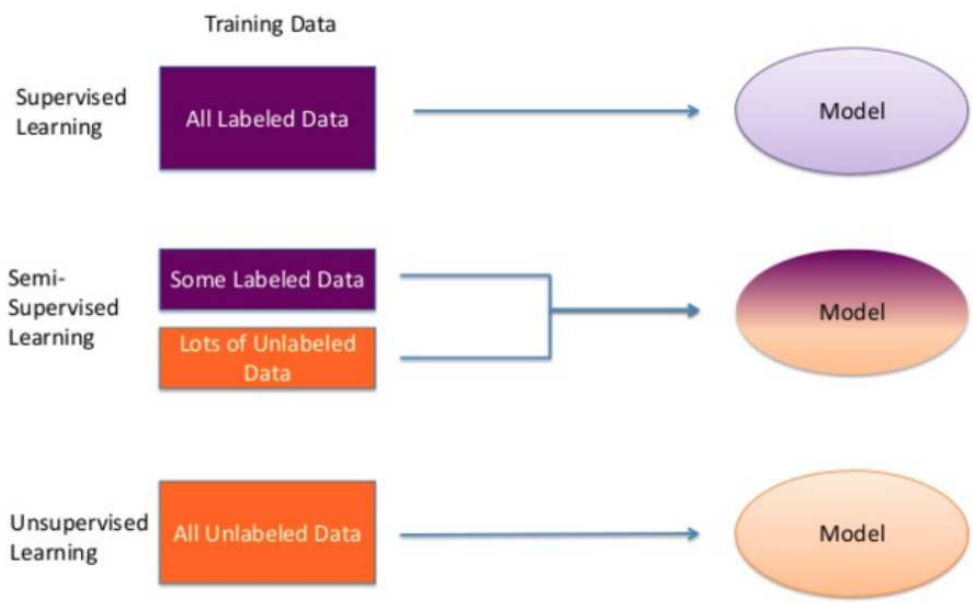
- in *Chapter 2* we present a brief survey about the state of the art of image classification and semi-supervised learning methods;
- in *Chapter 3* we discuss about feature extraction. The theoretical description is heavily influenced by the documentation of *VLFeat*[**?**], which will be used for the implementation;
- In *Chapter 4* we discuss multiple instance learning, describe the MILES algorithm, chosen for the implementation, and explain the creation of a codebook with the selected features;
- In *Chapter 5* we provide the reader an overview of topic models and their applications to computer vision;

- In *Chapter 6* we discuss the implementation of the project, focusing on the different phases and explaining the implementation choices;
- In *Chapter 7* we discuss and analyze the results;
- In *Chapter 8* we presents the conclusions of the work.
- In the *Appendix* we dwelwe more into some arguments that weren't fully discussed in the main corpus of the thesis. Also, some more figures are presented.

Chapter 2

The State of Art

In Computer Vision, image classification is the name for the task of extracting information classes from images. There are three different kinds of learning used for image classification:



1. *Supervised classification*: the learning algorithm takes some input $(x_1, ..., x_n)$ and some target output $(y_1, ..., y_n)$ and needs to find a function that approximates estimation. The outputs are the *labels* on which the classifier is trained

in order to generalize on the approximation.

2. *Semi-supervised classification*: the learning algorithm performs function estimation on both labeled and unlabeled data.

3. *Unsupervised classification*: the learning algorithm takes some input $(x_1, ..., x_n)$ but not target outputs.

Since the method we aim to implement is a semi-supervised method, we will now focus on this kind of classification.

The oldest semi-supervised learning methods are the so-called *Generative models*[**?**]. These methods assume a model $p(x, y) = p(y)p(x|y)$, where $p(x|y)$ is some identifiable picture distribution. This approach often is combined with the *Expectation-Maximization* algorithm. In fact, even if the mixture model assumption is correct, the components are effectively identified by EM algorithms.

There are also methods based on the concept of label propagation over a graph, whose nodes represent data examples and edges reflect their similarity. The labels that are maximally consistent with the supervised class labels and the graph structure are considered the *optimal* labels. One well-known example of this method is the *Eigenfuction*[**?**]. Eigenfunctions method are usually used to resolved locally-biased problems.

Other methods focus on the unlabeled data to regularize the classifying functions. The *SVM*[**?**] family of methods is the most representative of this category.

SVM is an acronym that stands for *support vector machines*. Given a training set, the elements of which are marked to belonging to one category, the SVM training algorithm builds a model that assigns a test set (new, unlabelled inputs), to one of the two categories.

The most notable methods that belong to this category are *transductive SVM*, *semi-supervised SVM* and *semi-supervised random forest*

A fourth family of methods, on which we will expand in chapter 4, is the family of *Multiple Instance Learning* methods, in which the learning algorithm takes in input a set of labeled bags, where each bag contains a set of instances and is labeled positive if there is at least one positive instance, and negative otherwise.

Chapter 3

Feature Extraction

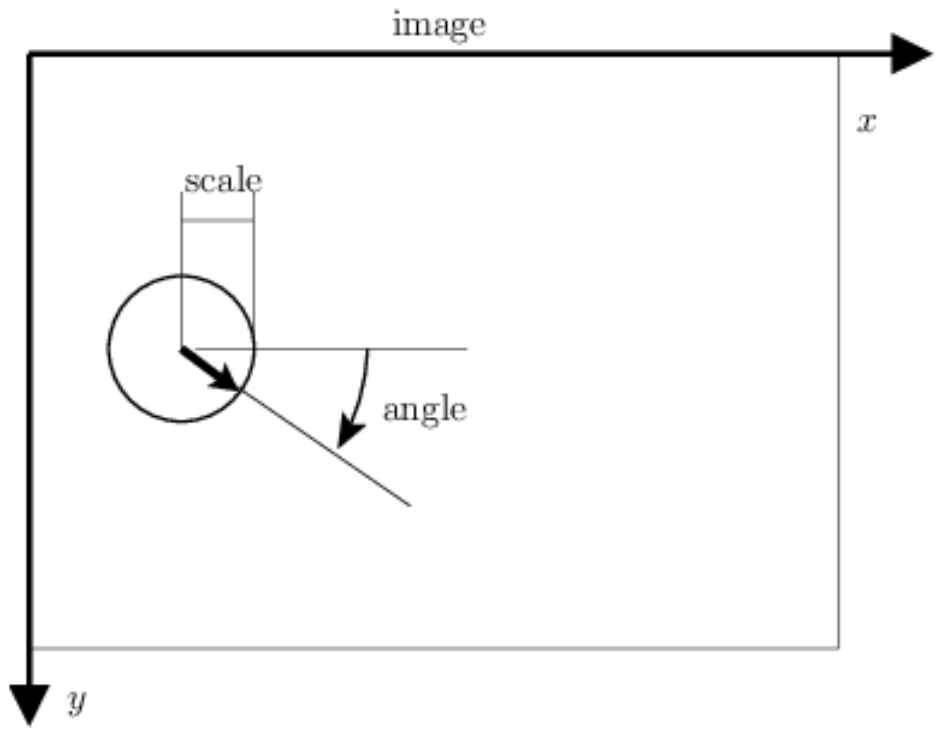
Feature extraction is the act of building a set of *features* from a given image. For the purpose of this thesis, the features are extracted using the dense version of the Scale invariant feature transform algorithm.

3.1 Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT), developed by David Lowe[**?**], are image descriptors very useful for image matching and object recognition. SIFT descriptor are very strong because they are:

1. invariant to translations, rotations and scaling transformations in the image domain;
2. robust to moderate perspective transformations and illumination variations.

A SIFT feature is a circular and oriented region of an image, called *keypoint*, composed by four parameters: the coordinates of its center, the scale (radius of the circle) and the orientation (angle expressed in radiant). At each SIFT feature is associated a descriptor vector.



These key points are selected at maxima and minima of a difference of Gaussian function applied in scale space [1].

The gradient vector field computed at the scale σ is denoted by:

$$J(x, y) = \nabla I_\sigma(x, y) = [\frac{\partial I_\sigma}{\partial x} \frac{\partial I_\sigma}{\partial y}] \tag{3.1}$$

A SIFT descriptor is a 3-D spatial histogram of the image gradients used for characterize the appearance of a key point. Samples are represented by the gradient at each pixel and are weighed by the gradient norm.

The histogram captures the distribution of $J(x, y)$, and is stacked as a (usually) 128-dimensional vector composed as follows:

- four bins for the spatial coordinates x and y;
- eight bins for the gradient orientation.

[1] In Dense SIFT, the detection phase of the standard SIFT algorithm is not performed: in fact, we select the locations of the key points using a parameter called *binsize*, that represent the fixed number of pixel used as *step* for the selection process.

The histogram, in fact, has $N_\theta \times N_x \times N_y$ bins[2], which are indexed by a triplet of indexes t, i, j. Their centers are given by

$$\theta_t = \frac{2\pi}{N_\theta} t, t = 0, .., N_\theta - 1 \tag{3.2}$$

$$x_i = i - \frac{N_x - 1}{2}, i = 0, .., N_x - 1 \tag{3.3}$$

$$y_i = i - \frac{N_y - 1}{2}, i = 0, .., N_y - 1 \tag{3.4}$$

The *binning functions* weigh contributions and compute the histogram:

$$w(z) = max(0, 1 - |z|), \tag{3.5}$$

$$w_{ang}(z) = \sum_{k=-\infty}^{+\infty} w(\frac{N_\theta}{2\pi} z + N_\theta k). \tag{3.6}$$

The gradient vector field is transformed in a three-dimensional density map of weighed contributions.

$$f(\theta, x, y) = |J(x, y)| \delta(\theta - \angle J(x, y)) \tag{3.7}$$

The histogram is localized in the keypoint support by a Gaussian window of standard deviation σ_{win} and it is given by:

$$h(t, i, j) = g_{\sigma_{win}}(x, y) w_{ang}(\angle J(x, y) - \theta_t) w(x - x_i) w(y - y_j) |J(x, y)| dxdy \tag{3.8}$$

The most convenient approach is to compute the descriptor directly in the image frame, denoting two kind of quantities:

- quantities relative to the canonical frame, denoted with a hat;

$$x = \begin{bmatrix} c\hat{x} \\ y \end{bmatrix} \tag{3.9}$$

[2]usually 4 x 4 x 8, as said before.

- quantities relative to the image frame, denoted without a hat.

$$x = \begin{bmatrix} cx \\ y \end{bmatrix} \tag{3.10}$$

And we can assume that canonical and image frame are related by an affinity:

$$x = A\hat{x} + T \tag{3.11}$$

In this way, all the quantities can be computed directly in the image frame. The updated function for the computation of the descriptor (either in the image or canonical frame) is thus:

$$h(t, i, j) = \int g_{A\hat{\sigma}win}(x - T) w_{ang}(\angle J(x)A - \theta_t) w_{ij}(A^{-1}(x - T)) |J(x)A| dx. \tag{3.12}$$

where the product of the two spatial binning functions is defined by:

$$w_{ij}(\hat{x}) = w(\hat{x} - \hat{x}_i) w(\hat{y} - \hat{y}_j) \tag{3.13}$$

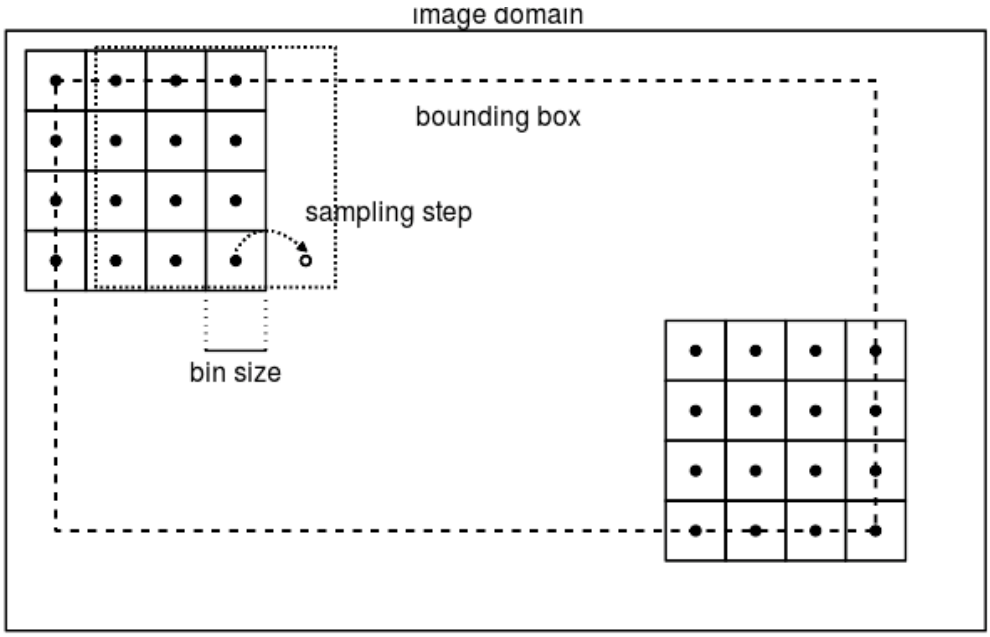
For a standard SIFT-detected keypoint of center T, scale σ and orientation θ, the affine transformation (A, T) reduces to the similarity transformation

$$x = m\sigma R(\theta)\hat{x} + T \tag{3.14}$$

where:

- $R(\theta)$ is a counter-clockwise rotation of θ radians;
- $m\sigma$ is the size of a descriptor bin in pixels;
- m is the descriptor magnification factor which expresses how much larger a descriptor bin is compared to the scale of the keypoint σ.

3.2 Dense Scale Invariant Feature Transform



Dense SIFT descriptor geometry

DENSE SIFT are a variation of standard SIFT with dense keypoints. It makes three important new assumptions:

1. The location of each keypoint comes from a predesigned location, not from the gradient feature of the pixel;
2. The scale of each key point is predesigned and it is the same;
3. The orientation of each keypoint is always zero.

When key points have null rotation and differ only by their position, we can simplify further:

$$x = m\sigma\hat{x} + T \tag{3.15}$$

$$h(t,i,j) = m\sigma \int g_{\sigma win}(x-T) w_{ang} \angle J(x) - \theta_t) w(\frac{x-T_x}{m\sigma} - \hat{x}_i) w(\frac{y-T_y}{m\sigma} - \hat{y}_i) |J(x)| dx \tag{3.16}$$

Many different values of T are sampled, hence we can use a separable convolution. We need to translate in

$$x_{ij} = m\sigma(\hat{x}_i, \hat{y}_i)^T \tag{3.17}$$

and we use the symmetry of the various binning and windowing functions in order to write:

$$h(t,i,j) = m\sigma \int g_{\sigma win}(T' - x - x_{ij}) w_{ang} \angle J(x) - \theta_t) w(\frac{T'_x - x}{m\sigma} - \hat{x}_i) w(\frac{T'_y - y}{m\sigma} - \hat{y}_i) |J(x)| dx \tag{3.18}$$

$$T' = T + m\sigma \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3.19}$$

We then define *kernels*

$$k_i(x) = \frac{1}{\sqrt{2\pi}\sigma_{win}} exp(-\frac{1}{2}\frac{(x - x_i)^2}{\sigma_{win}^2}) w(\frac{x}{m\sigma}), \tag{3.20}$$

$$k_i(x) = \frac{1}{\sqrt{2\pi}\sigma_{win}} exp(-\frac{1}{2}\frac{(y - y_i)^2}{\sigma_{win}^2}) w(\frac{y}{m\sigma}), \tag{3.21}$$

We finally obtain:

$$h(t,i,j) = (k_i k_j * J_t)(T + m\sigma \begin{bmatrix} x_i \\ y_i \end{bmatrix}), \tag{3.22}$$

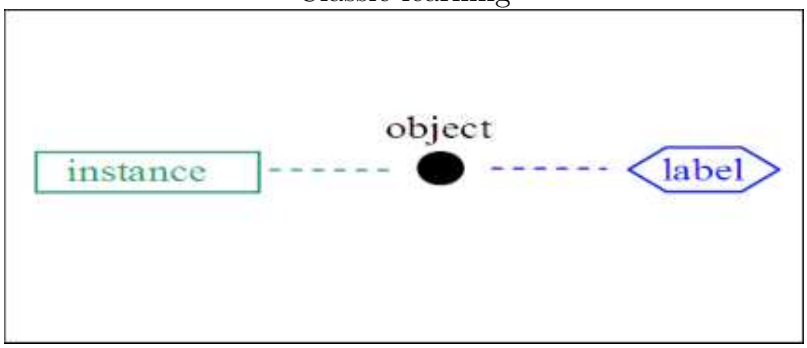
$$J_t(x) = w_{ang}(\angle J(x) - \theta_t)|J(x)| \tag{3.23}$$

Chapter 4

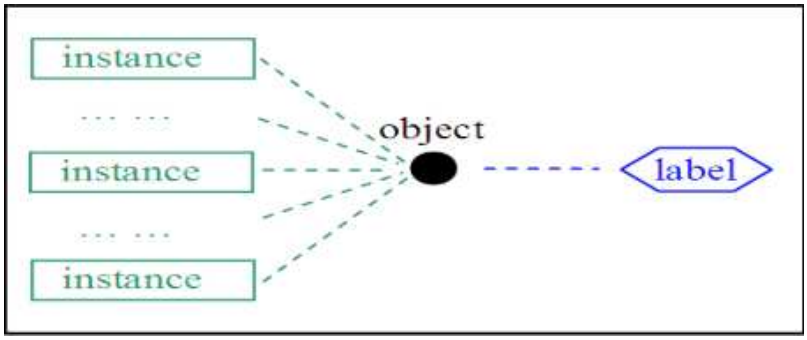
Multiple Instance Learning and Feature Selection

In this chapter multiple instance learning is explained and an algorithm for its implementation is described. The codebook creation is also discussed, more specifically the k-mean method for vector quantization.

Classic learning



Multiple Instance Learning



4.1 Multi-instance Learning: a Basic Introduction

Multiple Instance Learning (MIL) is a variation of supervised learning in which the labels are only assigned to bags of instances. The aim of MIL is to classify yet unseen bags or individual instances based on the labeled bags as the training data. The default case is the *binary* case, in which a bag is labeled positive if at least one instance in that bag is positive, and the bag is labeled negative if all the instances in it are negative. Individual instances have no labels. For this reason, a very important problem of MIL is the ambiguity caused by not knowing which of the patterns in a positive bag are the actual positive instances and which ones are not.

Formally, is given a set of input patterns $x_1, ...x_n$ grouped into bags $B_1, ..., B_m$, where $B_I = \{x_i : i \in I\}$ for given index sets $I \subseteq \{1, ..., n\}$[1].

Each bag B_I has a label Y_i associated. The interpretation of these labels is as follows:

- if $Y_I = -1$, then $y_i = -1$ for all $i \in I$, as to say that no instance in the bag is a positive one;
- if $Y_I = 1$, then at least one instance in the bag $x_i \in B_I$ is a positive one.

The relation between instance labels y_i and bag labels Y_I can be expressed both as

[1]Typically non-overlapping.

$$Y_I = max_{i \in I} Y_i \tag{4.1}$$

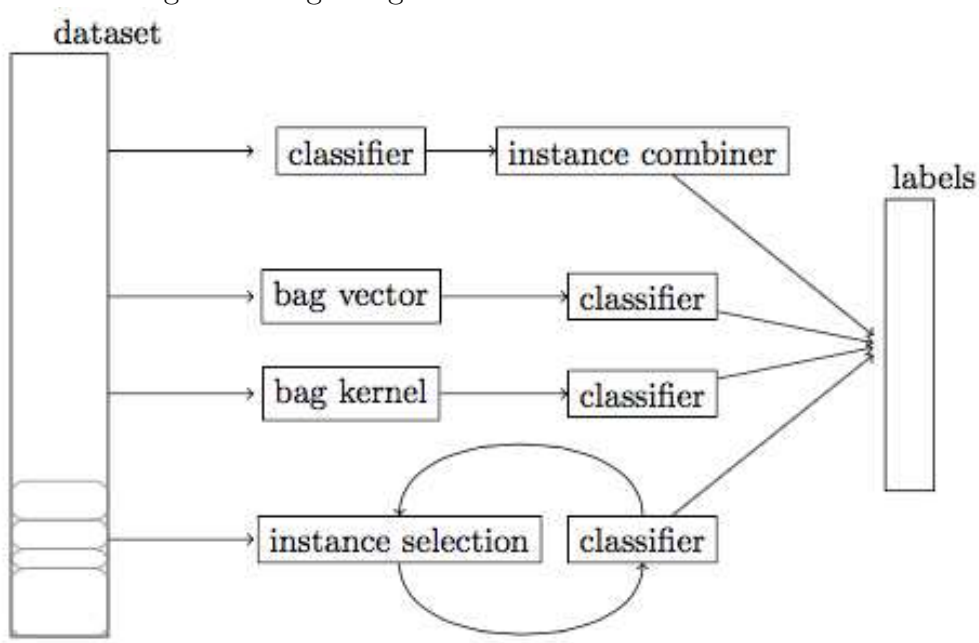
or as a set of linear constraints:

$$\sum_{i \in I} \frac{y_i + 1}{2} \geq 1, \forall I | Y_I = 1, \text{and } y_i = -1, \forall I | Y_I = -1 \tag{4.2}$$

Finally, a discriminant function $f : X- > R$ is MI-separating with respect to a multiple-instance dataset if, for all bags B_I, it holds $sgn\ max_i \in I\ f(x_i) = Y_I$.

4.2 Multi-instance Learning Algorithms

In our specific case, each instance is a *feature* vector, extracted with the DENSE SIFT algorithm. A bag of instances is a set of feature vectors: since we extract a feature vector for every relevant part of an image, we can consider all the feature vectors of an image as a single bag of instances.



The image, from [**?**], provides a general overview of MIL classifiers and descries the approaches that can be taken:

- Classify individually each instance using a standard classifier, and then for each bag combine the output of its instances into an overall output;
- Reduce the bags of instances to a feature vector and then apply a classifier to it;
- Reduce the bags of instances to a kernel matrix and then apply a classifier to it;
- Select the informative instances of each bags and trying a standard classifier: this can be performed in an iterative fashion.

4.2.1 MILES: Multiple Instance Learning via Embedded Instance Selection

MILES[**?**] is a learning method that converts the multiple instance learning problem to a standard supervised learning problem in which there is no assumption relating instance labels to bag labels.

MILES works by mapping each bag into a feature space defined by the instances in the training bags using an *instance similarity measure*. Since this mapping often produces a lot of redundant or irrelevant features, *1-norm SVM* is applied in order to select important features and train the classifier.

1-norm SVM can be formulated as a linear program, which can be solved from an optimization point of view without a relevant computational cost. Let's denote the class label variable with y, that can take values of $+1$ or -1.

We work in the feature space $\mathbb{F}_c$ and consider the classification problem of finding a linear classifier:

$$y = sign(w^T m + b) \tag{4.3}$$

used to distinguish between positive bags and negative bags, where $m \in \mathbb{F}_c$ represents a bag w and b are model parameters.

For solving this classification problem, SVM constructs classifiers based on hyperplanes by minimizing a regularized training error, $\epsilon_{training}$: ,

$$\lambda P[] + \epsilon_{training} \tag{4.4}$$

where:

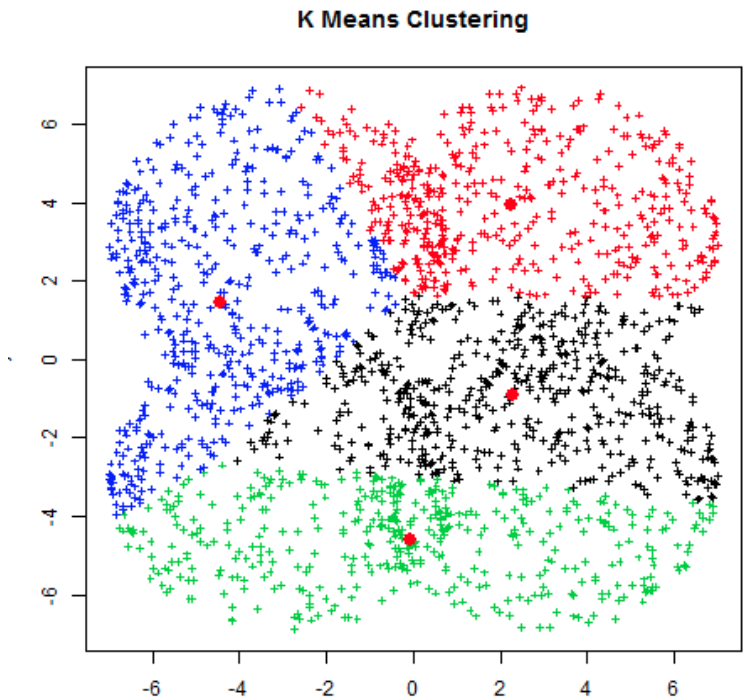
- $P[]$ is a regularizer;
- λ is the regularization parameter
- $\epsilon_{training}$ is a total of the loss that each bag introduces through a hinge loss function ξ [2]

The magnitude of w_k, component of the optimal solution w, indicates the significance of the effect of the kth feature in $\mathbb{F}_c$ on the classifier. The features that correspond to $w_k \neq 0$ are selected and used in the classifier.

4.3 Vector quantization and codebook creation

K means clustering is one of the simplest methods for performing vector quantization. The centroids found using the algorithm are the *codewords* necessary for the creation of a *codebook*. Once the codebook is created, the idea is to decode vectors assigning them to the centroid to which they are the closest.

[2] $\xi = max\{1 - y(w^T m + b), 0\}$



K means algorithm uses an iterative refinement technique, alternating between two steps.

Given an initial set of k means $m_1^{(1)}, ..., m_k^{(1)}$

1. *Assignment step*: Each observation is assigned to the cluster whose means yields the least within-cluster sum of squares.

$$S_i^{(t)} = \{x_p : ||x_p - m_i^{(t)}||^2 \leq ||x_p - m_j^{(t)}||^2 \forall j, 1 \leq j \leq k\}, \tag{4.5}$$

   where each x_p is assigned to one $S^{(t)}$.

2. *Update step*: calculate the new means to be the centroids of the observations in the new clusters.

$$m_i^{(t+1)} = \frac{1}{S_i^{(t)}} \sum_{x_j \in S_i^{(t)}} x_j \tag{4.6}$$

The algorithm converges when there are no longer changes to the assignments.

Chapter 5

Topic Models

In this chapter is explained in detail what a topic model is and in which way it can be useful to our purpose. Furthermore, here is explained how topic modeling can be applied to computer vision problems.

5.1 Topic models: a Basic Introduction

A *topic model* is a statistical model used for discovering the abstract topics that occur in a collection of documents. We can define a topic as a recurring pattern of co-occurring words. The goal of topic modeling is to identify these patterns in a corpus, grouping them together by a process of similarity.

5.1.1 Latent Dirichlet Allocation

The simplest topic model is *Latent Dirichlet Allocation* (LDA). LDA is a hierarchical model of count data originally developed to aid in the analysis of large collections of textual data. LDA models a document as a mixture of topics, where each topic is a multinomial distribution over words in a predefined vocabulary. The vector of topic proportions is thus the gist of a document.

LDA is part of the larger field of probabilistic modeling.

Let C be a collection of D documents, where each document d has a vector w_d associated denoting the sequence of words occurring in d. Let $w = (w_1, ..., w_D) =$

$(w_1, ..., w_N)$ denote the vector of all words in the collection.

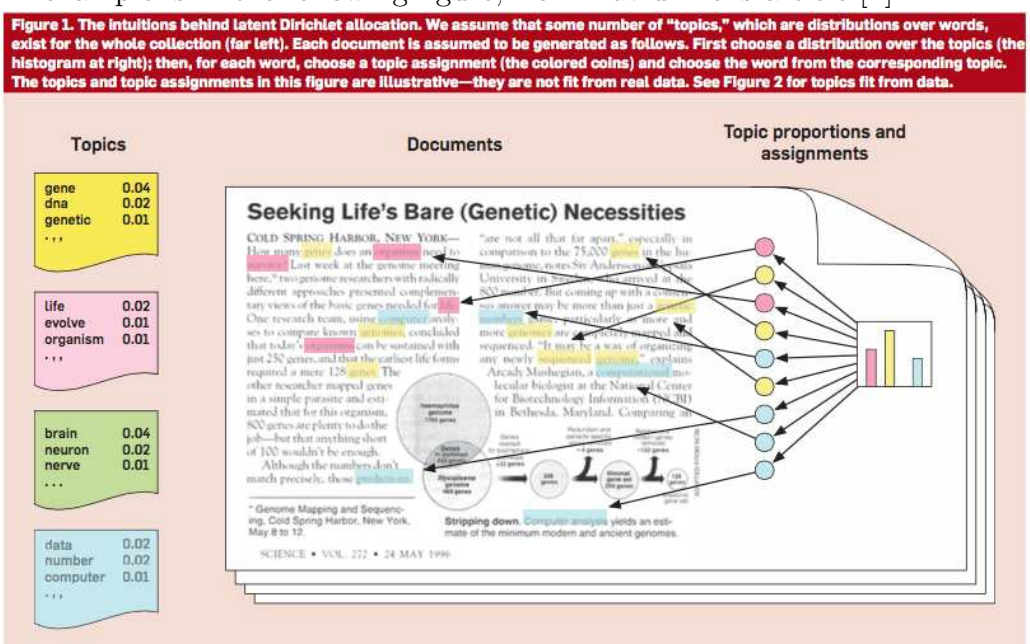
The model equation is given by:

$$P(w) = \sum_{d=1}^{D} \int p(\theta_d) \sum_{i=1}^{N_d} \sum_{j=i}^{T} P(z_i = j|\theta_d) \int P(\phi_j) P(w_i|z_i, \phi) d\phi_j d\theta_d \tag{5.1}$$

where

- N_d is the number of words in the dth document;
- $p(\theta_d)$ is the distribution over topics of the dth document of the collection;
- z_i are the topics of each word $w_i \in d$ and are distributed according to the mixture of topics given by θ_d.

An example is in the following figure, from David Blei's article [?]



The words in the article are divided according to the topics: for example *life* or *organisms* are words about *evolutionary biology*. A topic is a distribution over a fixed vocabulary: the *evolutionary biology* topic has words about evolutionary biology with high probability.

The model assumes that the topics are generated before the document. The words are generated for each document in the collection with the following process:

1. Randomly choose a distribution over topics
2. For each word in the document
   - Randomly choose a topic from the distribution over topics in the first step;
   - Randomly choose a word from the corresponding distribution over the vocabulary.

It's important to note that all the documents in the collection share the same set of topics, but those topics have different proportion in each document.

5.1.2 Gibbs sampling

For obtaining samples from complicated probability distributions, we can use Markov chain Montecarlo, which is a procedure that allows a Markov chain to converge to the target distribution then drawing samples from the Markov chain.

A Markov chain has a series of states, where each state represents an assignment of values to the variables that are being sampled. Transitions, in Markov chains, follows a very simple rule. In Gibbs sampling, for example, the next state is reaching by sequentially sampling all the variables from their distribution when conditioned on the current values of all other variables and the data.

The complete probability model is :

$$\begin{gathered} w_i|z_i, \phi^{(z_i)} \sim \text{Discrete } (\phi^{(z_i)}) \\ \phi \sim \text{Dirichlet } (\beta) \\ z_i|\theta^{(d_i)} \sim \text{Discrete } (\theta^{(d_i)}) \\ \theta \sim \text{Dirichlet } (\alpha) \end{gathered}$$

Hence, the conditional posterior distribution for z_i is given by

$$P(z_i = j|z_{-i}, w)\alpha P(w_i|z_i = j, z_{-i}, w_{-i})P(z_i = j|z_{-i}), \tag{5.2}$$

5.2 Topic models in computer vision

Often, in computer vision, we are interested in finding some latent structure in an image that allow us to match similar images. While they where originally developed to handle large collections of textual data, Topic Models may be very useful in these processes of image classification.

5.2.1 LDA in computer vision

The main question when using LDA for computer vision is how to define *words* and *documents* in the domain.

A good correspondence may be the following: the image is considered as a document, and image features are considered as words.

The steps to follow in order to work with this analogy are the following:

- Identify *a priori* what is considered to be a relevant feature;
- Extract the features from the images;
- Quantize the feature vectors in order to form a codebook.

There is another question to consider when porting LDA topic models in the computer vision domain. LDA assumes that a document is a *bag of words*[1], causing spatial and temporal structures among visual words to be ignored. However, while spatial and temporal structures are meaningless in a language model, they are very important in many computer vision problems.

Both of these problems for a direct application of LDA in computer vision problems may be solved with some modifications to the model.

[1]The Bag of Words model will be explained in more detail in the appendix

Chapter 6

Implementation

6.1 Dataset

The dataset is a repertoire of historical images of Venice belonging to *Fondo Morassi*. These images are contained in a database and are divided in two categories: front and back. Each representation, in fact, has a back page in which there are annotations of the art historians. In the front page, instead, there is the actual image and a ruler with the color scale.

We are interested only to the front page: more specifically, our objects are only the actual images, without the contours. The first step of the algorithm is, in fact, to select all and only the front pages and crop the actual images using the reference coordinates provided by the art historians.

6.2 Feature Extraction

We chose the Dense SIFT algorithm implemented by VLFeat open source library.

This algorithm select a grid of keypoints where each keypoint is separated by a fixed number of points, and then compute a SIFT descriptor for each of these key points.

Dense SIFT specifies the descriptor size by a single parameter, size, which controls the size of a SIFT spatial bin in pixels. In the standard SIFT descriptor, the bin size

is related to the SIFT keypoint scale by a multiplier, denoted as *magnification*.

This algorithm does not compute a Gaussian scale space of the image: instead, the images are pre-smoothed before the computation of the descriptors, since one of the biggest strength of the SIFT algorithm is that the results are independent from the scale of the image.

6.3 Vector Quantization and Codebook creation

Vector quantization is performed using the function *vqsplit.m*. Given in input a matrix each column of which is one of the selected vectors and the codebook size, the algorithm return the codebook in the form of the centroids of the clusters. Furthermore, the algorithm also returns the weight of each cluster and a vector containing the overall distortion of each iteration.

At the end of this step, we have created a codebook where the words are the relevant objects for the whole ensemble of images. For completing the codebook one last step is necessary. We now add to each entry of the codebook (that represents an object relevant for our analysis) the literal tag from the dataset. This is done using a vector whose elements are only 1 or 0. In every vector, there is only one 1, corresponding to the category of the object.

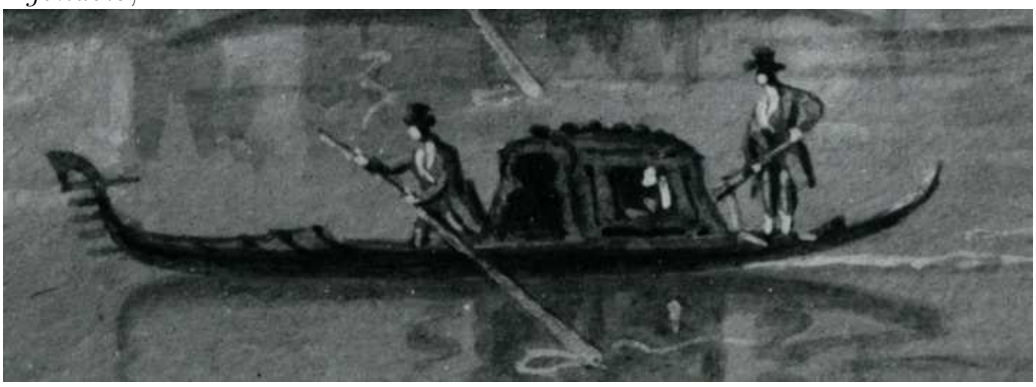
6.4 MILES classifier

For building a classifier and perform feature selection, we chose the MILES algorithm implemented by MIL toolbox[**?**], by Tax et al. This algorithm train the miles learner on a MIL dataset, where the bags are represented by the maximum similarity to all other instances. The similarity is measured by a kernel[1]function using diverse density proximity. A sparse linear classifier is then trained on this dissimilarity representation.

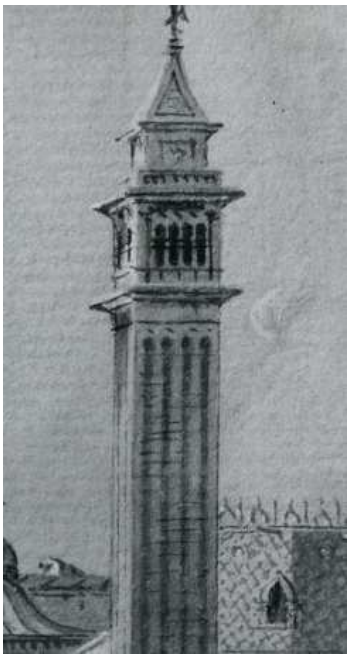
We need to build a classifier for each class of objects in the dataset. More specifically, some of the objects that we analyze are the following:

[1]the radial basis function

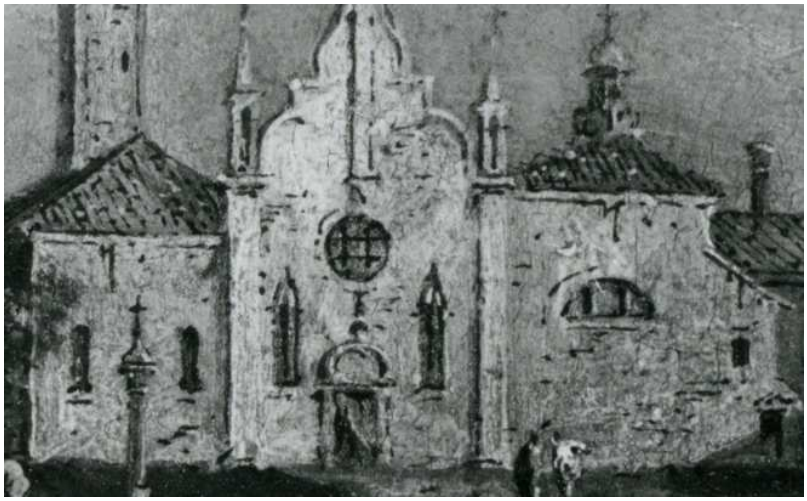
- *gondole*;



- *campanili*;



- *chiese*;



- *ponti*;

The steps of the feature selection for each class of objects are the following:

1. Generate a MIL dataset, where the feature vectors are considered as the instances and the considered image region as the bags;

2. Label the bags as positive or negative;

3. Build the classifier with these *training* bags;

4. Select the feature vectors where $w \geq 0$.

At the end of this phase, we have a big vector-quantized codebook that has been made on the whole corpus of images, and for each class of objects a code that represents the selected more relevant features for that object.

Here ends the phase of tools *building*, and begins the phase of *analysis*.

6.5 Topic Modeling

The phase of image analysis strongly relies on topic model analysis. For each image that we wants to analyze, we need to first extract its features and build the codebook relative to that image. Then, topic modeling is performed running the *Gibbs sampler* for the LDA model on a bag of words data: as to say, the code of the image we want to analyze and the codebook built before.

The outputs of topic modeling, that we need to study in order to understand the analysis, are:

1. A sparse matrix of size *vocabulary size x number of topics*, that contains the number of times a word has been assigned to a topic;

2. A sparse matrix of size *number of documents x number of topics*, that contains the number of times a word in a document has been assigned to a topic;

3. The topic assignments for each token.

These data can be analysed in many different ways. We will talk more about this in the following section.

Chapter 7

Results

The goal of image analysis is to take different images, belonging to the same class or to different classes, and perform topic modeling using:

- The *Image* or *Images* as document;
- The *Codebook* as the words of interest.

For each image or group of images analyzed, we find the distribution of topics. The number of times a word is assigned to a topic can be represented with an *histogram*, in order to have a visual representation of the results.

7.1 Parameters

Topic modeling has a set of parameters which values need to be tuned in order to perform a correct analysis:

1. *alpha* is an hyper parameters on the dirichlet priors for the topic distribution. The standard value of alpha is $50/T$, where T is the number of the topics;
2. *beta* is an hyper parameters on the dirichlet priors for the topic-word distribution. The standard value for beta is $200/W$, where W is the number of the words;

3. N is the number of iterations for which to run the Gibbs sampler. A good setting for this parameters depend on the number of the topics and the complexity of the problem. Usually, the number of iterations is set in the range $500 - 200$;

4. T is the number of topics we intend to extract. A little number of topics make them more broad, while a big number of topics make them more specific. I tried different parameters and then settled for $T = 20$;

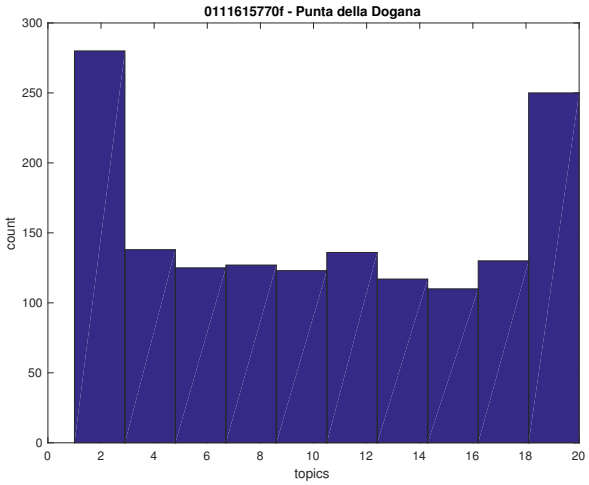
5. *seed* sets the seed for the random number generator

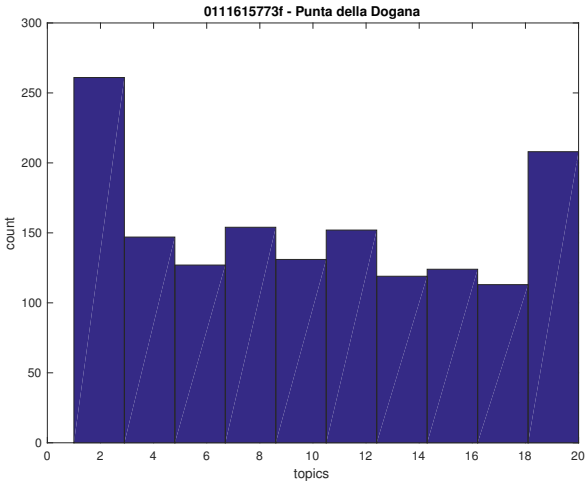
7.2 Analysis

One interesting analysis may be to study the number of times a topic has been assigned to a specific word in the document.

Of particular interest may be the analysis of two images that represents the same view. Here, for example, we have two images that are very similar: both are by Giacomo Guardi and represents Palazzo Ducale with view from Punta della Dogana.

Here we compare the histograms from these two images:





We can see that the histogram of the topics distribution of these images are very similar: this is what we expected, since the two images are practically identical.

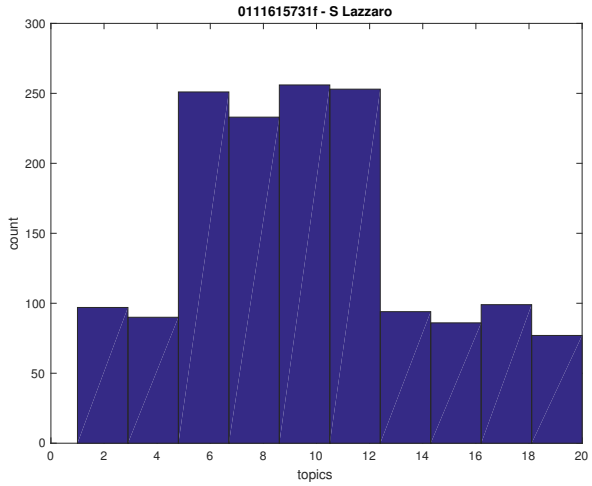
Another interesting analysis may be performed on two images that represent the same scenario but are not so similar.

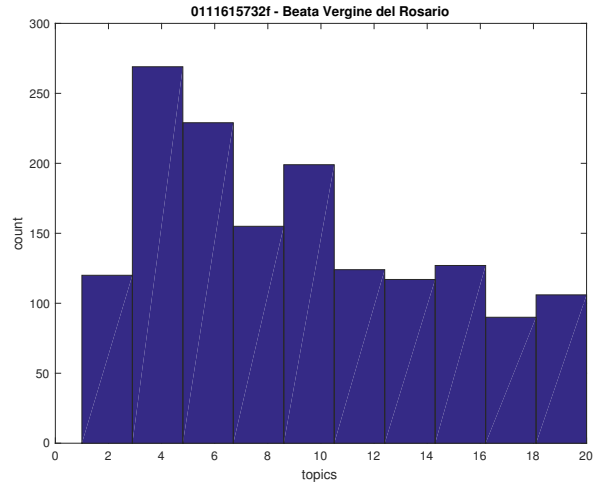
Here for example we compare two images that both represents two isles with a church or convent.

San Lazzaro

Beata Vergine del Rosario

The most significant difference between the two images is the bell tower (which corresponds to the keyword *campanile*), but both images have an isle, boats (*gondole*) and a church. We expect the topic distribution to be similar but not identical...



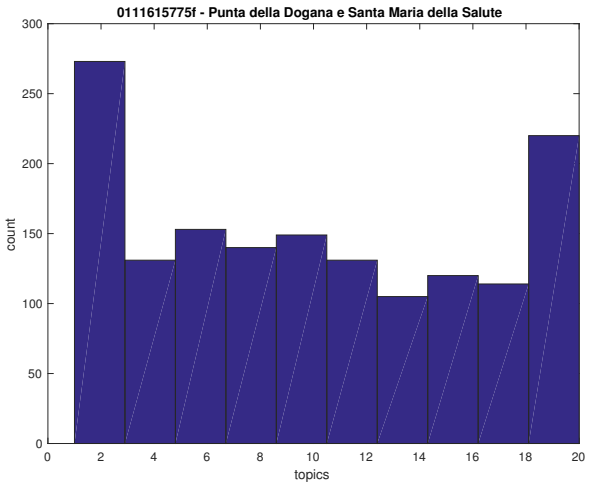


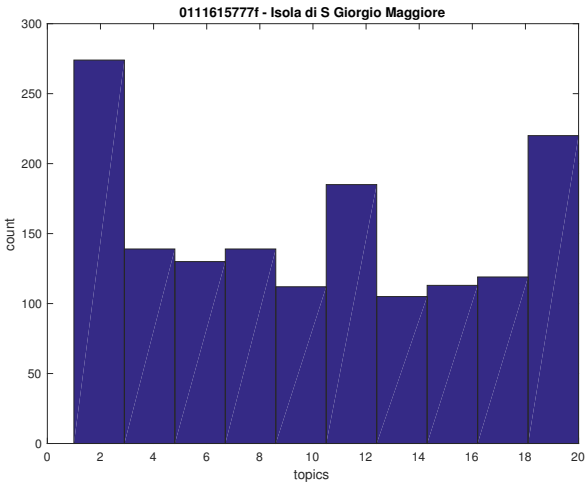
More evidence of this theory is presented by these other couple of images. Here we present two different locality, San Giorgio Maggiore e Santa Maria della Salute.

San Giorgio Maggiore

Punta della Dogana e Santa Maria della Salute

The histogram of these two images are pretty similar, as expected. The elements present in these images are in fact pretty similar: we have boats, a dome, the lagoon and a church.



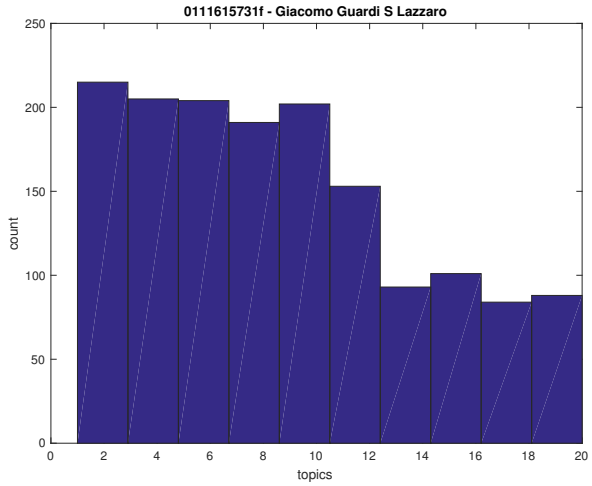


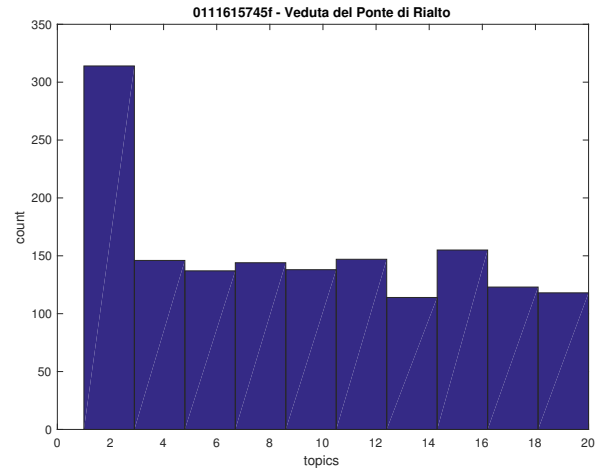
The analysis of course is not useful if we don't try the some variation. Here we present two images that are pretty different. The first image represents San Lazzaro, that we have already seen, and the second image is a view of the bridge of Rialto (let's recall that also bridge (*ponte*) was a keyword).

San Lazzaro

Ponte di Rialto

The histograms of these two images are fairly different. This is what we expected: the two images itself are different the one from the other. The first image, in fact, is the representation of a church on the island of San Lazzaro; the second image, instead, is a view of the famous bridge of Rialto. The only element in common between the two images are the boats.





7.3 Histogram Distances and Clustering

An interesting analysis that can be performed is the distance between two or more topic distributions. A classic way for measuring this distance is the so called *Jensen-Shannon divergence*

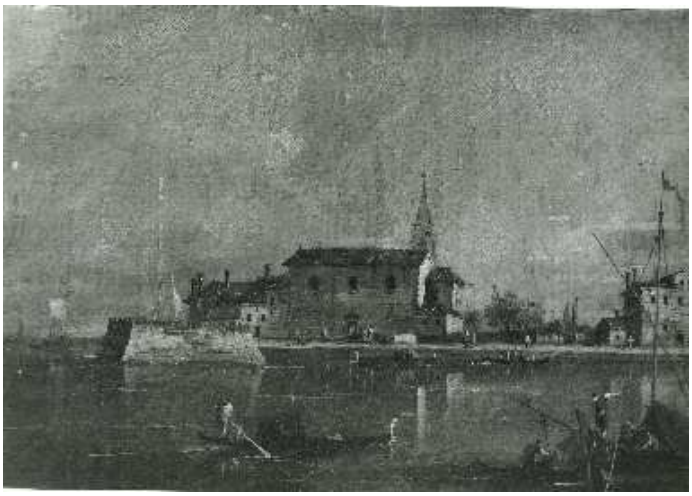
According to Jensen-Shannon divergence, the distance between the distributions P and Q ($JSD(P||Q)$) is defined by:

$$JSD(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M) \tag{7.1}$$

where $M = \frac{1}{2}(P + Q)$. This index has a range of values between 1 and 0, where 1 is the maximum similarity and 0 the maximum divergence.

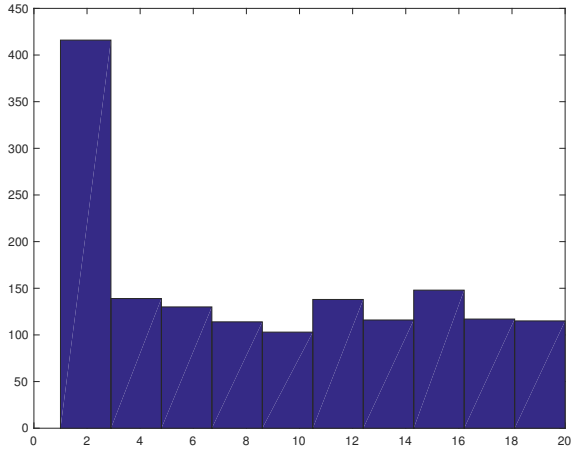
This distance is an interesting measure for comparing the pairs of histogram shown above. For example, these images:

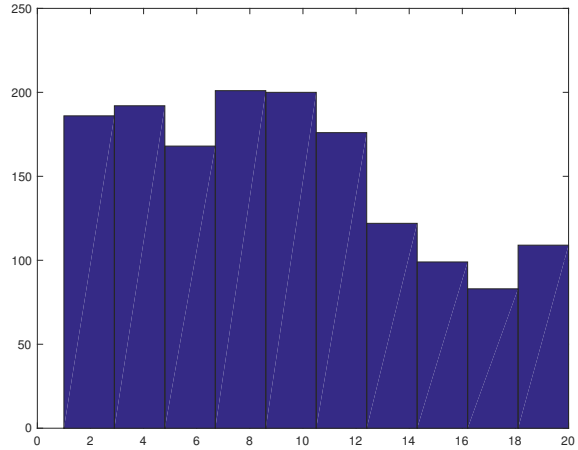
Bacino di San Marco



Veduta di Isola

Produces the following histograms;





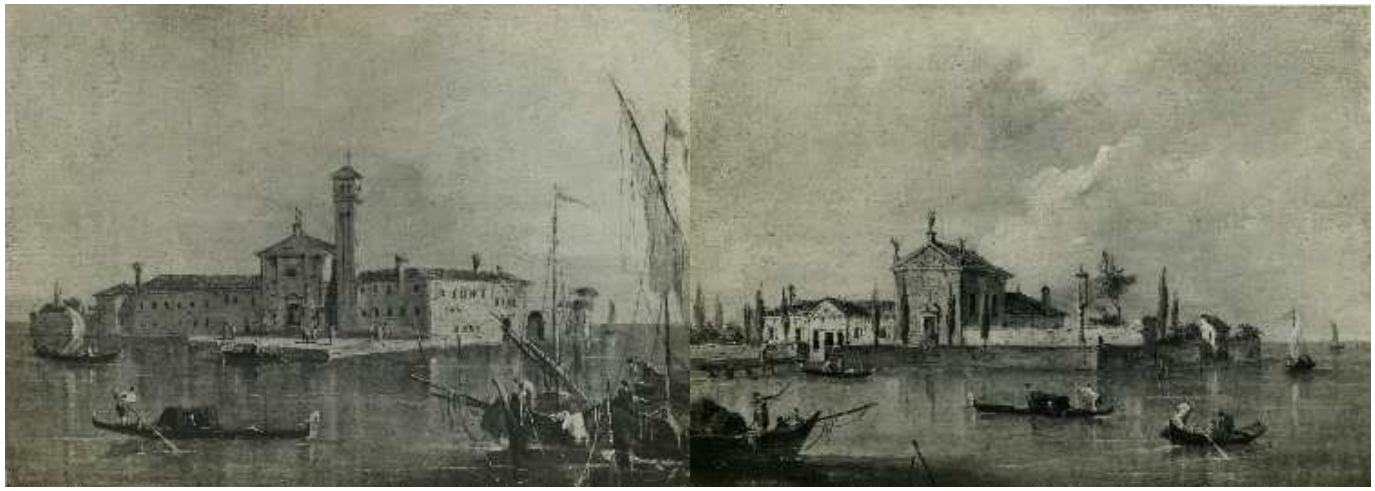
The Jensen Shannon Divergence between these histogram is JSDiv = 0.1025, meaning that they are very different.

The distance may be useful also for clustering the results. In fact, if we consider the histogram as points to be clustered and we build a similarity matrix where the distance between two points is the Jensen-Shannon divergence between two histograms, we can try to perform clustering on the images in order to find if the topics are related to visual similarity between the images.

If we try to cluster the previous histograms using the Jensen Shannon distances and Dominant Sets clustering[1], the images are grouped like this:

[1]More about Dominant Sets clustering will be said in the appendix.

First Group



Second Group

The clustering on the topics produces the expected results. We can in fact see the ratio between the two different groups. The similarities are coherent with the histograms and we can also see a visual similarity between the images grouped together.

Since Topic Modeling studies the distribution of the topics among the images, it's pretty easy to see how the first three images contains many similar elements: for example, there are a lot of gondole, represented in a similar style. Also the buildings are very similar: they have a lot of windows and vertical grooves.

At the contrary, in the second group of images we have fewer boats and more slender buildings, with less detailed facades.

7.4 Topics Analysis and Principal Component Analysis

In order to fully understand the scope of this analysis, we also need to study the topics as a group of probability distributions and as singular entities.

If we perform topic models using 20 topics, it's interesting to see a visual representation of the order in which the topics are drawn: i.e the probability of the

presence of a topic in the whole image corpus. Most likely topics are always the last extracted.

In order to discover the most likely topics, the probability distribution over each document is calculated. For each topic pair, the symmetrized *Kullback Leibler* distance between the two topic distributions is calculated.

Kullback Leibler (KL) divergence is, likewise the Jensen Shannon divergence, a measure of the difference between two probability distributions P and Q. The main differences between KL and JS divergence is that KL is

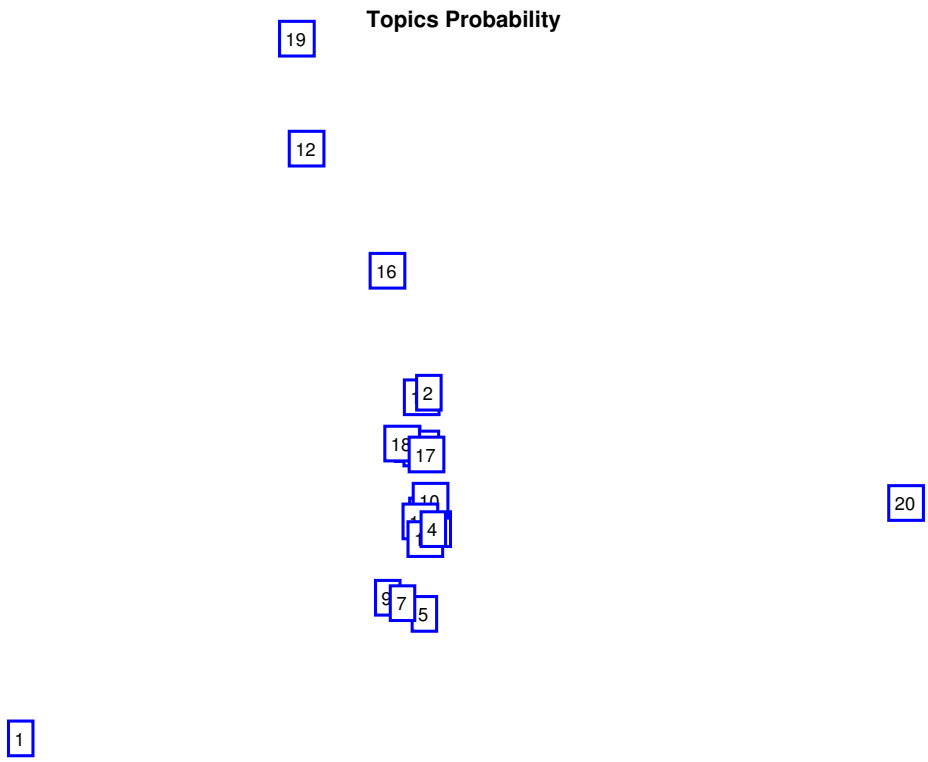
- asymmetric;
- may be a not finite value.

Formally, the Kullback-Leibler divergence from Q to P is defined to be

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \tag{7.2}$$

Once all the distances are calculated , they are subjected to multidimensional scaling in order to represents them on a 2-dimension plane.

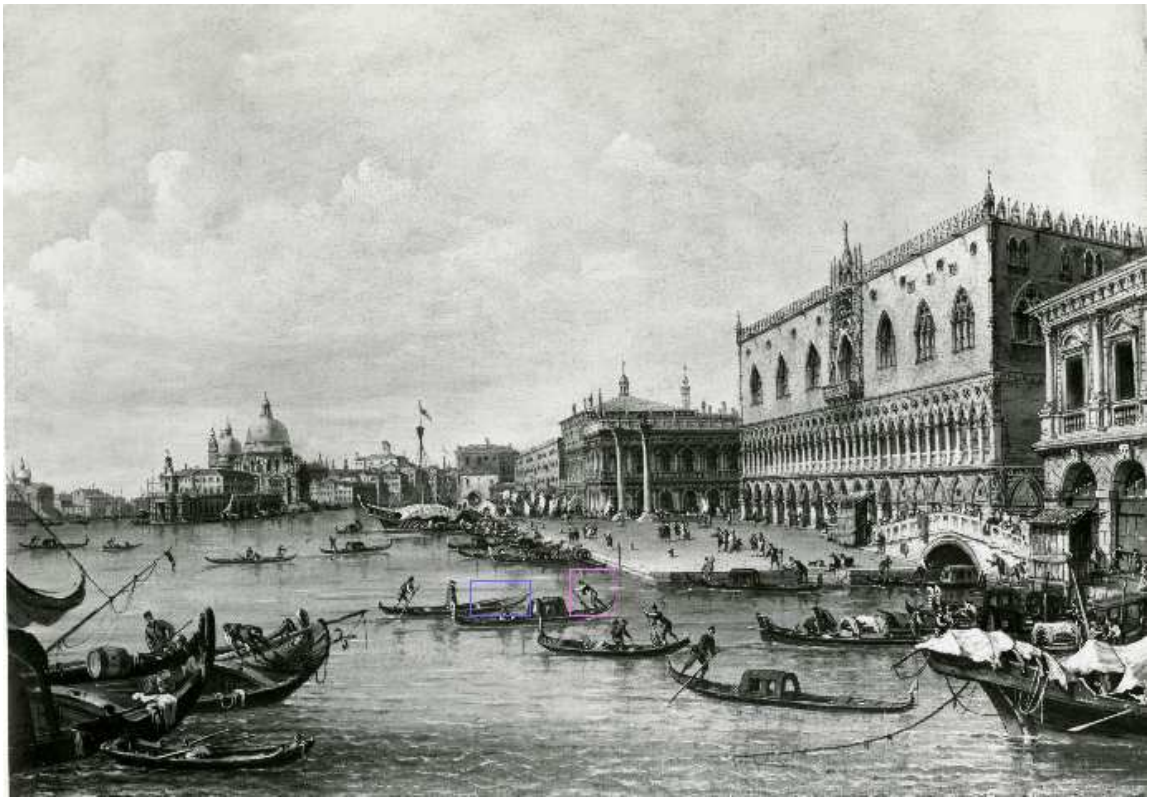
Multidimensional scaling is useful for visualize the level of similarity of individual cases in a dataset. It refers to the information contained in a distance matrix, that we created on the words linked to every single topic. The Euclidean distances between the features approximate a monotonic transformation of the corresponding dissimilarities.



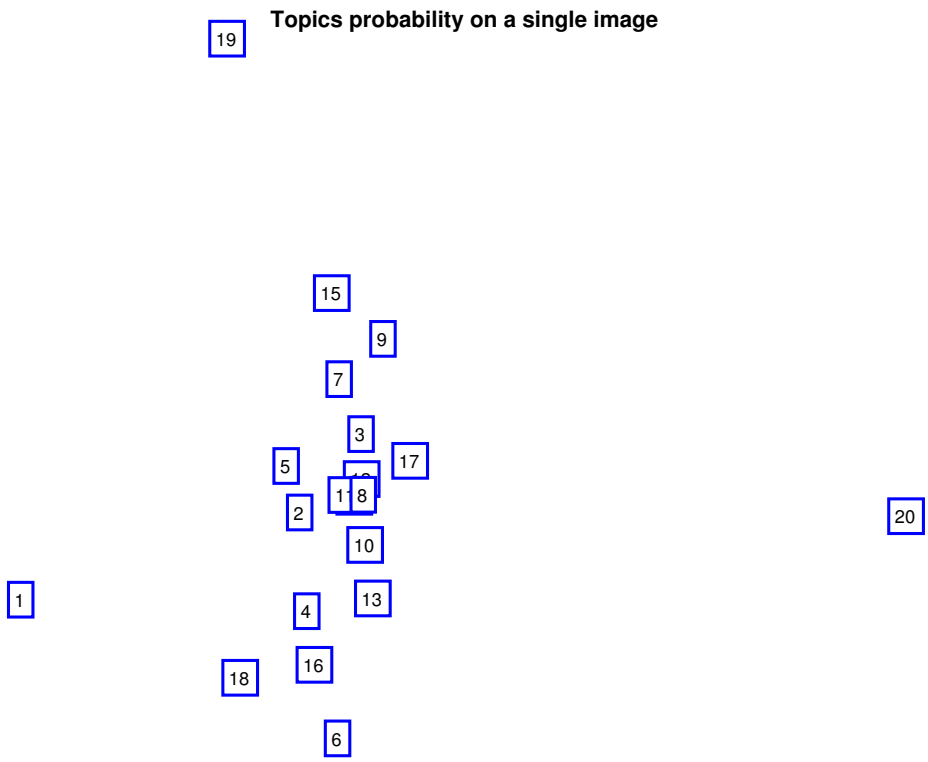
The topic order represented in the picture is the following:

```
Order = [ 1, 19, 12, 16, 2, 15, 6, 18, 13, 17, 14, 10, 3, 11, 4, 8, 7, 9, 5, 20]
```

Let's see now an example of a single image and its topic probabilities, in order to see if there is correspondence with the global order of drawn topics.



Punta della Dogana




```
Order = [ 1, 19, 18, 5, 4, 16, 6, 2, 15, 7, 13, 14, 10, 11, 8, 17, 12, 3, 9, 20]
```

We can see that the results are pretty similar: in this image the first topic is the first to be drawn, followed by the nineteenth topic, and then the results differ slightly. The last topic is again 20.

Another interesting analysis to perform is to study the principal components[2] of the topic distribution. It is the classic way for performing multidimensional scaling.

The interpretation of this analysis focuses on finding which original variables are correlated which the components. This can be seen from the table by looking at the largest numbers in magnitude (both positive and negative values).

[2]More about Principal Component Analysis will be said in the appendix.

Pca1	Pca2	Pca3	Pca4	Pca5	Pca6	Pca7
0,01469054	0,88260535	0,41191724	-0,1026406	0,05879215	-0,1061813	-0,0788393
0,06191754	0,0422711	-0,2932203	-0,0426128	0,06322466	-0,1779308	-0,2369132
0,07150835	0,10689628	-0,1309247	-0,1794603	-0,4263589	0,04266828	0,16928686
0,08481524	0,10605875	-0,2512514	-0,1308523	0,09352955	-0,1821476	-0,3131921
0,10314741	0,16250349	-0,0112897	0,60642509	-0,3856615	0,35140993	-0,1954503
0,04548268	0,06625836	-0,2521588	0,08214393	-0,0265856	0,05219371	-0,2368197
0,07176758	0,21112636	-0,2746014	-0,2414406	-0,000859	0,70648789	0,33571604
0,08512866	0,11226439	-0,1603501	0,00112618	0,10843723	0,00178427	-0,2029157
0,06276658	0,18611858	-0,3254441	0,50714717	0,53926548	0,00539015	0,1186067
0,07253806	0,09702888	-0,1754332	-0,1296149	-0,2005483	-0,2096807	0,33736782
0,07783321	0,12439693	-0,1558386	0,24987365	-0,0815343	-0,4148844	0,59307178
0,00469403	0,01627026	-0,1015198	-0,0024674	-0,1008754	0,04794933	-0,1014616
0,05616731	0,06329031	-0,138566	0,083645	-0,3115783	-0,0451968	-0,1008697
0,06558374	0,11864366	-0,3592641	-0,3672827	0,24086165	0,07114276	-0,0378282
0,05293226	0,03897221	-0,1829188	0,09547045	-0,1028852	-0,0215448	-0,0532684
0,02030644	0,01728693	-0,1803269	0,05176163	-0,1033084	-0,0906197	0,03897874
0,06541108	0,06915226	-0,1887967	-0,0838448	-0,3237043	-0,1840354	-0,2337779
0,04314683	0,08178358	-0,1872199	-0,0811911	-0,0799248	-0,1692526	-0,0468727
0	0	0	0	0	0	0
0,96252061	-0,1319326	0,20741618	-0,042152	0,07375005	0,0027336	0,00445166

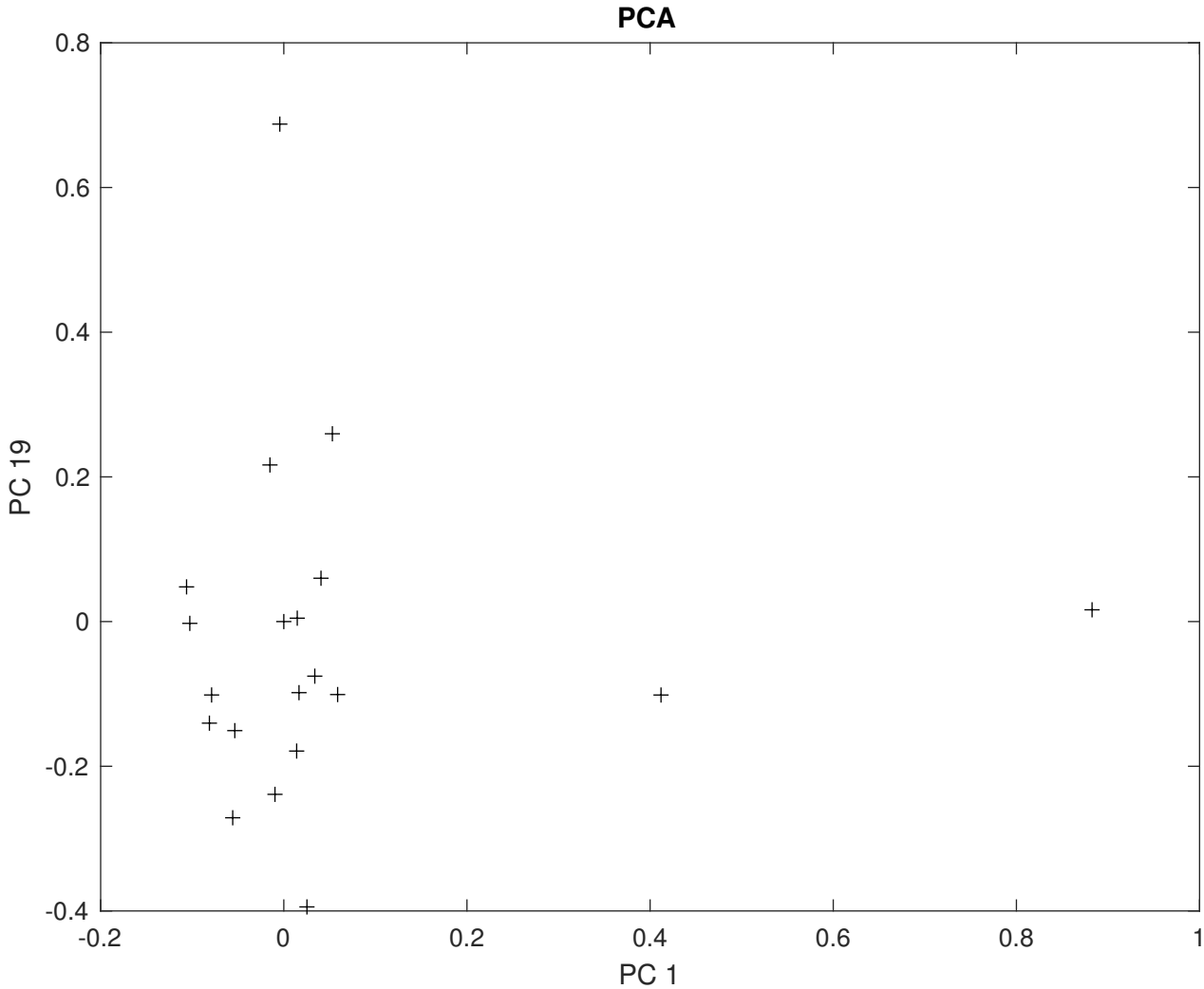
Pca8	Pca9	Pca10	Pca11	Pca12	Pca13	Pca14
0,05304243	0,03383957	-0,0812446	-0,0557769	-0,0535001	-0,0043628	-0,0151196
0,60719695	0,32403709	0,08043892	0,07172236	-0,2214706	-0,0811982	0,00929068
-0,2295083	0,58368379	0,45631592	-0,1663312	-0,1277101	-0,0060156	0,15793345
-0,1488408	-0,3800391	0,45790453	-0,1292727	-0,0640794	-0,055676	-0,3599964
-0,0342016	-0,0603701	0,14960345	-0,0241858	0,26732518	-0,0496302	0,06633961
-0,0031875	0,0400602	0,08155719	0,03948316	-0,1644443	0,04949834	-0,3971755
0,11932523	-0,2241383	-0,04395	0,19434186	-0,2256532	0,02628375	-0,120223
0,16336094	-0,228881	0,31587799	0,14506945	0,34846915	0,14763832	0,40872475
-0,1561618	0,34618326	-0,1093826	0,1282644	-0,0105962	0,02997302	-0,0053116
0,30941517	0,10909554	-0,091401	0,14697855	0,57994434	0,03803193	-0,3801597
-0,0003842	-0,3756672	0,19792251	-0,0176941	-0,2407775	-0,0315925	0,18254893
0,25958446	-0,0754481	-0,140404	-0,2711796	-0,1507353	0,68762849	0,2165014
-0,0319795	-0,0980662	-0,3051454	-0,1224199	0,14833589	-0,1628207	-0,1357234
-0,2333005	0,00266793	-0,1253534	-0,2602498	0,32417563	-0,2370492	0,34886706
0,01293088	-0,0591607	-0,3030408	-0,4635569	-0,2424536	-0,2927645	-0,0519756
0,12864922	-0,0383456	-0,2009622	-0,303936	0,06100143	0,06390791	0,12334712
-0,1170642	-0,0690369	-0,2817519	0,61194068	-0,2192491	-0,1739224	0,29840662
-0,48804	0,0494225	-0,1903634	0,09947774	0,03066138	0,53242099	-0,1718403
0	0	0	0	0	0	0
0,00056754	0,02232389	-0,0490968	-0,0125609	-0,0326564	0,02134331	-0,0108964

Pca15	Pca16	Pca17	Pca18	Pca19	Pca20
0,04061649	0,01659565	-0,009634	0,01403868	0,02528226	0
-0,1284568	-0,3721811	0,26095567	0,17027848	0,14716277	0
-0,081197	0,11711692	-0,1358185	-0,1108976	0,0160483	0
-0,2839653	-0,004528	-0,2485077	0,18059811	-0,2376769	0
0,05420213	-0,1970128	0,19016958	0,26248191	-0,1781274	0
0,74965772	0,17218526	0,00352978	-0,2128137	0,17231506	0
-0,1474972	0,00188699	0,00809368	0,05484178	0,09834501	0
-0,129049	0,35704395	0,16722945	-0,3517019	0,30907079	0
-0,1695292	0,09324212	-0,2564012	-0,0790191	-0,1014884	0
0,00384996	0,15958578	0,02760262	-0,0237301	-0,3050196	0
0,18435428	-0,2013406	0,09497895	-0,0438394	0,05346033	0
0,05990505	-0,098254	-0,2388139	-0,1789805	-0,3944208	0
-0,2142715	-0,3679507	-0,3884193	-0,4232971	0,40039871	0
0,32096282	-0,3080585	0,06190828	0,06703229	-0,1613548	0
-0,2092747	0,38800795	0,43847301	-0,232873	-0,2019943	0
0,05993234	0,38132283	-0,2928437	0,61771813	0,39185701	0
0,02665039	0,16288461	-0,1494675	0,07289969	-0,2457065	0
-0,165673	-0,1199418	0,4464387	0,13908852	0,22668771	0
0	0	0	0	0	1
0,02193602	0,00268363	-0,0218095	0,01017955	0,01038916	0

From this analysis we can see that no component is particularly bounded. The 20th component and the 19th topic, which are almost all zero, are pretty strange compared to the others. What can this means?

Each dimension represents mainly the topic which has the higher value for it. If we look at the table, every topic most significant for each component is circled with a pink line. For nineteenth topic, each principal component has value zero except the twentieth component, which is an astounding value compared to the rest of the table. This value means that the twentieth dimension represents exclusively the nineteenth topic.

Let's see alone the first two components plotted the one against the other, in order to capture most of the variance.



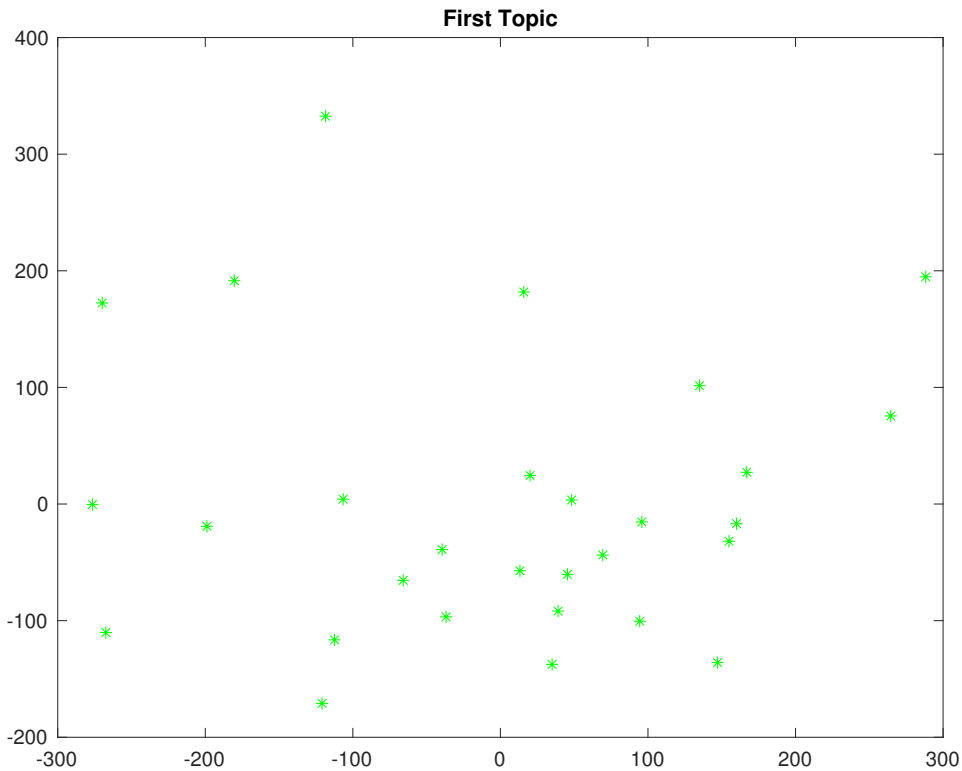
Looking at the dot out by itself to the right we may assume that this particular dot has a very high value for the first principal component. From the table we can see that the first component has an high value for the fifth topic.

Looking at the dot out by itself to the top we may assume that this particular dot has a very high value for the second principal component. From the table we can see that the first component has an high value for the seventh topic.

It can be interesting now to understand which words correspond to each topic.

The problem is that our words are not textual words, but visual words in the form of features. Performing multidimensional scaling, we can display the visual words corresponding to each topic reducing them to 2-dimensionality.

All the topics have associated words, but for brevity here we presents only the words associated with the first topic. The plot of the other words will be displayed in the appendix.



This kind of analysis, while it can tell us many things about the principal components of the words related to a topic, is not very useful in order to discern exactly which visual word corresponds to each topic.

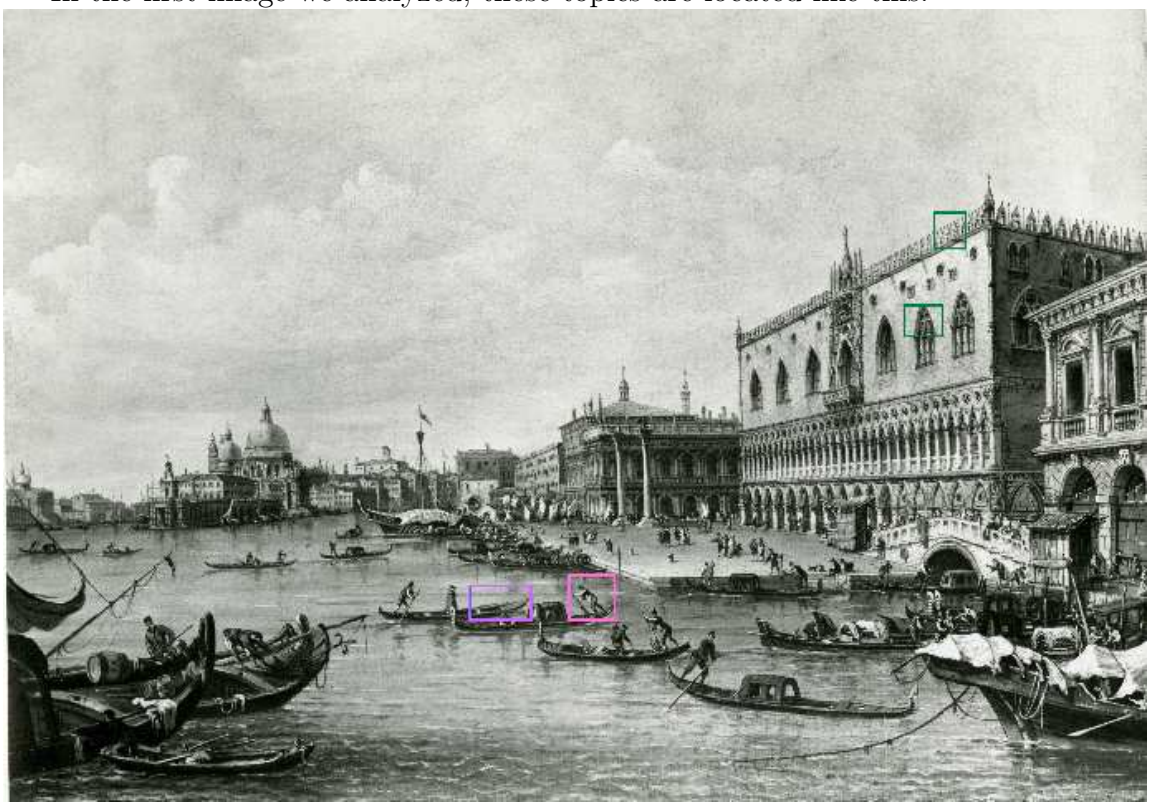
If we want to perform a more significant analysis, we can take the words associated to each topic and we can extract the image patches corresponding. For example, if we analyze the result of topic modeling on the first image we can see some visual word corresponding to topics:

- First Topic

- Fifth Topic

- Ninth Topic

In the first image we analyzed, these topics are located like this:



Let's now see the visual words on the topics when topic modeling is performed on the entire dataset: Let's analyze the dataset differently, exploring the correspondence when 10 topics are extracted

First, we compute the order of extraction with ten topics. Halving the number of topics extracted we increase the granularity of the topic modeling and we have a lot more of visual words associated to each topic. For brevity, here we present only the some of the words in this association. It's important to note that, changing the number of topic, also the precision/generalization of topic model and the specific association topic-words change.

The order from the *most probable topic* to the last is the following:

```
10, 7, 8, 5, 4, 6, 3, 2, 9, 1
```

Let's now have a look to the correspondences between topics and visual words:

- Tenth Topic

  This topics contain visual words bounded to the sky and the sea. We have a lot of sea and sky in our images, so as expected this topic is one of the most presents in the dataset. This is not the only topic associated to the textual words *sky* and *sea*, since the images have different quality and precision.

  We can define as textual words for this topic:

  `sky, sea, landscape, ground`

- Seventh Topic

The visual words bounded to this topics are the one related to architectonic elements. All these patches represents less or more blurred elements belonging to a bell tower, a church or a palace. In fact, we can see (and we know about!) the similarity in the window structure in the architecture of the whole city of Venice in the same historic period.

We can define as textual words for this topic:

`building, window, architecture, palaces`

- Eight Topic

The visual words belonging to these topics are portions both of the sky and the sea. The two elements are in fact pretty similar in the representation, changing granularity according to the quality and precision of the image. Also some building facade belongs to this category: in fact, in the points without windows, their structures are very similar to texture of sky and sea. These topic assignation is not very precise, even considering that there is another topic that share visual words pertaining to the same argument.

We can define as textual words for this topic:

`sky, sea, building`

- Fifth Topic

Here we can see visual words that represents vertical slender structures: for example, the cross on top of a church or a bell tower, some chimneys and also something that seems a *briccola* (a specific term of Venetian architecture that represents a pointed structure on the wooden docks scattered among the islands).

We can define as textual words for this topic:

`briccola, cross, chimney, pole`

- Fourth Topic

This topic pertains to the boats of Venice, especially *gondole*. In fact, both the curve of this specific kind of boat and its *pettine* (the particular figurehead similar to a golden comb) are very particular and easily recognizable. There are a lot of gondole in Venice, and their representations are so characteristics that every gondola is depicted almost identical to every other gondola.

We can define as textual words for this topic:

`boat, gondola, sea, pettine`

- Sixth Topic

The visual words belonging to these topics are sailors or bystanders, especially on boats. All the sailors represented on the gondole are depicted in the same position: a figure slightly bended over with an oar in hand. Also bystanders belongs to this topic, because of the similarity in their body structure and sometimes also in the position.

We can define as textual words for this topic:

`sailor, person, bystander,`

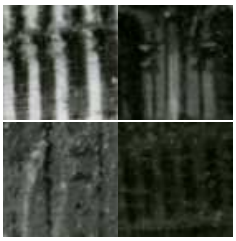
- Third Topic

The visual words associated to these topics are curved structures and parts thereof: more specifically, we can say that these topics represents bridges (there are a lot of bridges in Venice), arcs and domes (thanks to the islamic culture contamination, most churches have a dome-like structure, even the very famous San Marco).

We can define as textual words for this topic:

`bridge, arc, dome`

- Second Topic

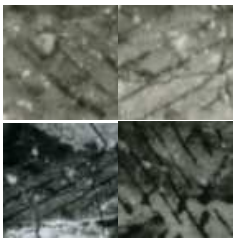


The visual words associated to this topic are architectural elements similar to the ones we already seen, but with a bigger focus on columns and grooved building. The majority of bell towers (and we have a lot of bell towers in Venice) has in fact a similar structure, with vertical grooves.

We can define as textual words for this topic:

`column, groove, architecture`

- Ninth Topic



The visual words belonging to this topic represents building facades and roofs. There are many topics associated to this particular words, since there are many ways in which the artists can represents architectural elements.

We can define as textual words for this topic:

`roof, building, architecture`

- First Topic

This visual words are pretty confused to a first glance: differently from the others, an observer isn't immediately able to understand what they represents. Analysing the images that contains these visual words, we are able to see that they depict what in art is called *capriccio*.

Capricci are imaginary landscapes created with a free combination of real or fantastical architectonical elements, that especially in the production of our corpus are represented in a more stylized and *postcard*inspired style. While the other images are in style of real paintings, many of these capricci are, in fact, more similar to drawings.

We can define as textual words for this topic:

`capriccio, building, architecture`

Chapter 8

Conclusions

The tool we discussed is useful to analyze big datasets composed of images in order to find visual similarities and differences among the pictures. From the analysis, we found that this tool actually works: working on the low-level features of the images, we found correspondences that are perceptible to the human eye. Furthermore, this procedure can also be useful in order to find *new* similarities that are yet undiscovered.

Clustering images starting from the topic distribution can be important for an art historian: the outputs from this analysis may offer many suggestion for new point of views in a study of a big corpus of images. The analysis of topics can also tell us what elements are the most important for the representation of something. In fact, the single topics are related to visual words that represents specific objects of interest. It can be interesting for an historian to analyze the *trends* in the history of the representation of Venice or every other location.

In this project the analysis is focused on the historic representations of Venice, but the same tool can be used on pictures of every different place in the world, by just changing the data images used for training.

8.1 Future Works

The work can be further expanded in the following directions:

- Try to work with other kind of topic modeling, for example dynamic topic models that take count of the documents order;
- Experiment more with the results of the various experiments, also changing the tune of the parameters;
- Adapt the tool in order to work with tags that have a hierarchical relationship.

Bibliography

[1] *VLfeat open source library*, 2007-13

[2] Xiaojin Zhu, *Semi-Supervised Learning Literature Survey*, 2008

[3] Kristin P. Bennett, Ayhan Demiriz, *Semi-Supervised Support Vector Machines*, 2004

[4] David G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, 2004

[5] Bing Han, Dingyi Li, Jia Ji, *People Detection with DSIFT Algorithm*, 2011

[6] Jun Yang, *Review of Multi-Instance Learning and Its applications*, 2012

[7] David M. Blei, *Probabilistic Topic Models*, 2012

[8] Denis Deratani Maua, *Visual Topics: The Latent Dirichlet Allocation Model in Computer Vision*, 2012

[9] Jun Yang, *Review of Multi-Instance Learning and Its applications*, 2012

[10] David Tax, *Multi-instance toolbox Manual*, March 2016

[11] Yixin Chen, Jinbo Bi, James Z. Wang, *MILES: Multiple Instance Learning via Embedded Instance Selection*, IEEE Transactions on PAMI, vol. 28, no 12, 2006

[12] Zijun Zhang, *K-means clustering Algorithms* 2012

[13] Yin Zhang, Rong Jin, Zhi-Hua Zhou, *Understanding Bag-of-Words Model: A Statistical Framework*

[14] Massimiliano Pavan, Marcello Pelillo *Dominant Sets and Pairwise Clustering* IEEE Transactions on Pami, vol. 29, no. 1, 2007

[15] Herve Abdi, Lynne J. Williams, *Principal Component Analysis*, 2010

[16] Forrest W. Young *Multidimensional Scaling*, Kotz-Johnson (Ed.) Encyclopedia of Statistical Sciences, vol. 5, 1985

Appendix A

Bag of Words

A *Bag of Words* model is a representation in which a text is represented as the bag of its words, taking in account multiplicity but without care for grammar or word order.

An example of a text modeling using bag of words is the following:

```
Gaia likes kittens very much. Roberta likes kittens too.

Gaia also likes to party with her friends.
```

With reference to these two documents, we construct the following list:

```
[ "Gaia", "likes", "kittens", "very", "much", "Roberta",
"too", "also", "party", "with", "her", "friends" ]
```

This list is what is called a bag of words. After its construction, we can calculate various measure in order to characterize and analyze the text. One of the simplest analysis (which is also useful for the purpose of this thesis) is to count the number of times a term appears in the text.

In topic modeling, like in the bag of words model, the topic are *independent* of word order! The technique of topic modeling, which is analyzed and used for this project, goes beyond the simple bag of words model. In fact, topic modeling not only count the distribution of words, but most of all it counts the topic mixture over a set of documents.

In this sense, we can see that topic modeling share its foundation with the bag of words model, but enrich it with the inference of the topics from the single words. Latent Dirichlet allocation, more specifically, assume a generative bag-of-word model and then backtracks from the documents to find a set of topics that are likely to have generated the collection.

Appendix B

Dominant Sets

A dominant set for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D. Dominant sets are used for pairwise, proximity-based data clustering. Using this approach, the data to be clustered is represented as an undirected edge-weighted graph with no self-loops.

Vertices in G correspond to data points, edges represent the relationships between neighbors, and edge-weights reflect similarity between pairs of linked vertices.

A cluster should satisfy two conditions:

1. it should have high internal homogeneity;
2. there should be high inhomogeneity between the entities in the cluster and the entities outside the cluster.

Let $S \subseteq V$ be a nonempty subset of vertices and $i \in S$. The average weighted degree of i with regard to S is defined as:

$$awgdeg_s(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij} \tag{B.1}$$

where, when $i \in V, awdeg_{\{i\}}(i) = 0$.

Furthermore, if $j \notin S$:

$$\phi_S(i, j) = a_{ij} - awdegS(i) \tag{B.2}$$

We can see that, intuitively, $\phi_S(i,j)$ (this quantity can be either positive or negative) measures the relative similarity between nodes i and j with the respect to the average similarity between node i and its neighbors in S.

Hence, let $S \subseteq V$ be a nonempty subset of vertices and $i \in S$. The weight of i with regard to S is:

$$w_S(i) = \begin{cases} 1 & \text{if } |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(j,i) W_{S \setminus \{i\}}(j) & \text{otherwise} \end{cases} \tag{B.3}$$

And the total weight of S is defined to be $W(S) = \sum_{i \in S} W_S(i)$.

In fact, $W_S(i)$ fives us a measure of the overall relative similarity between vertex i and the vertices of $S \setminus \{i\}$ with respect of the overall similarity among the vertices in $S \setminus \{i\}$.

We can formalize the concept of a cluster in an edge-weighted graph as follows. A nonempty subset of vertices $S \subseteq V$ such that $W(T) > 0$ for any nonempty $T \subseteq S$, is said to be dominant iff:

1. $W_S(i) > 0$, for all $i \in S$,
2. $W_{S \cup \{i\}}(i) < 0$, for all $i \notin S$.

Consider now a similarity graph $G = (V, E, w)$ with n vertices, and its weighted adjacency matrix A. A common way to represent a cluster of vertices is to associate a (real- valued) n-dimensional vector to it. The component of this vector express the participation of nodes in the cluster: a small value corresponds to a node weakly associated with the cluster, whereas if it has a large value, the node is strongly associated to it. The values that are equal to zero corresponds to nodes not participating in the cluster.

We can express the cohesiveness of a cluster using the following quadratic form:

$$f(x) = x^T A x \tag{B.4}$$

This equation allows us to interprete the clustering problem as the problem of *finding a vector x that maximizes f.*

Hence, since the objective function is useless without some normalization of the components of x, we have the following standard quadratic program, which is a generalization of the Motzkin-Straus program:

- maximize $f(x)$
- subject to $x \in \Delta$

where

$$\Delta = \{x \in \mathbb{R}^n : x \geq 0 \wedge e^T x = 1\} \tag{B.5}$$

is the standard simplex of $\mathbb{R}^n$.

According to this formula, a maximally cohesive cluster correspond to a local solution of the program, and dominant sets are in correspondence with (strict local) solutions.)

A straightforward way to find local solutions of the standard quadratic program is given by the *replicator dynamics*, a class of continuous and discrete-time dynamical systems arising in evolutionary game theory. In this project, we will use the following model:

$$x_i(t+1) = x_i(t)\frac{(Ax)_i}{x(t)^T Ax(t)} \tag{B.6}$$

for $i = 1...n$, which corresponds to the discrete-time version of first-order replicator equations.

The simplex Δ is invariant under these dynamics, which means that every trajectory starting in Δ will remain in Δ for all future times. Furthermore, since A is symmetric, the objective function increases strictly along any nonconstant trajectory, and its asymptotically stable points are in one-to-one correspondence to strict local solutions of the program. These, in turn, correspond to dominant sets for the similarity matrix A.

Appendix C

Principal Component Analysis

Principal Component Analysis, or *PCA*, is a technique generally used to decrease the redundancy of features of a data set and to reduce the dimensionality of problems, facilitating the description and visualization of a data set with minimal loss of information. Formally, it can be defined in two different ways:

- Maximum variance formulation (MVF). PCA is the orthogonal projection of a certain data set onto a lower dimensional linear space, the so-called principal subspace, such that the variance of the projected data is maximized;
- Minimum error formulation (MEF). PCA is the linear projection of a certain data set such that the mean squared distance between the data points and their projections is minimized.

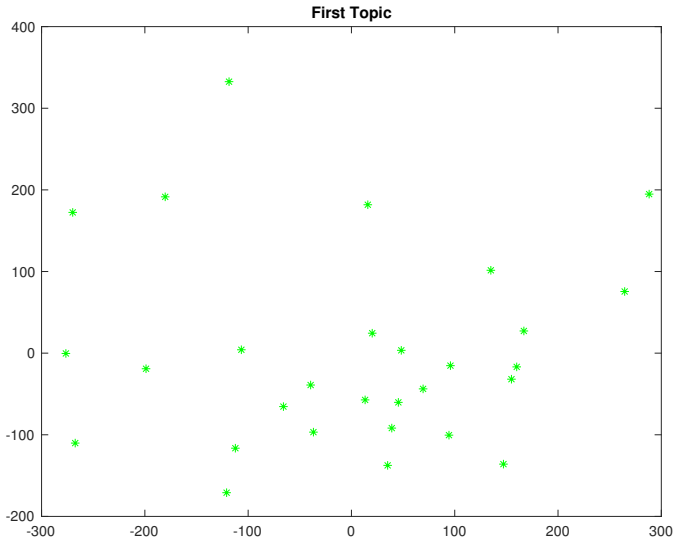
The goal of PCA is to find a new, lower-dimensional coordinate system for the original data set in which the axes represent maximum variability. To find such subspace, PCA transforms a given set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables, the *principal components*, which are always less or equal to the number of features of the original data set.

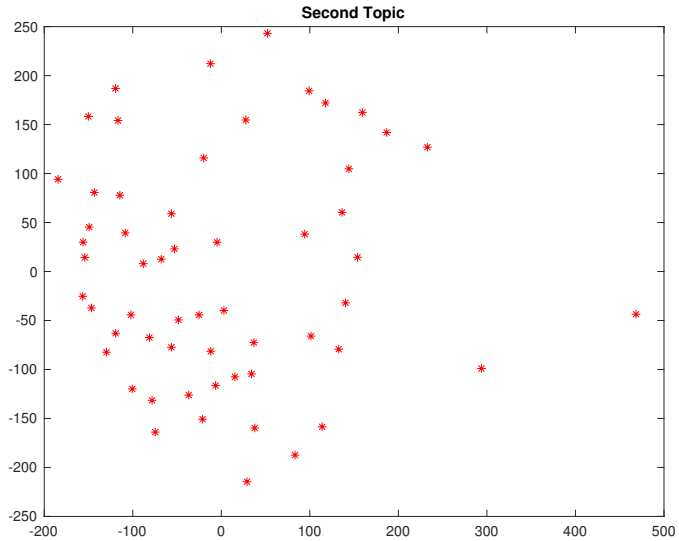
Principal components can be viewed as the directions where there is the most variance: the first principal component explains the greatest amount of variability of the data while the second principal component must be orthogonal to the first principal component in order to explain the variance in the data that is not captured

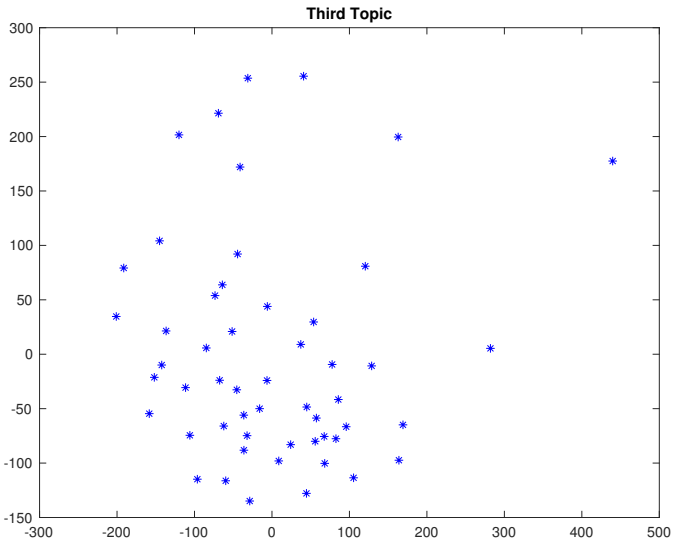
by the first principal component. The same applies to the remaining components, which must be orthogonal to the preceding components. To find such principal components we use eigenvectors and eigenvalues.

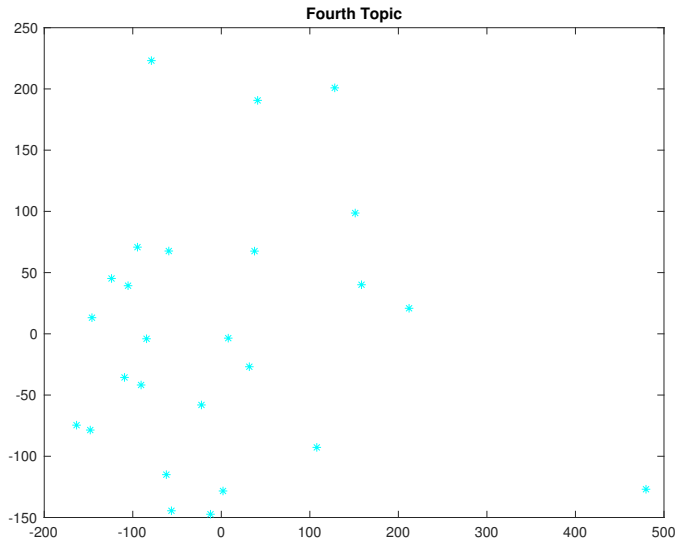
Appendix D

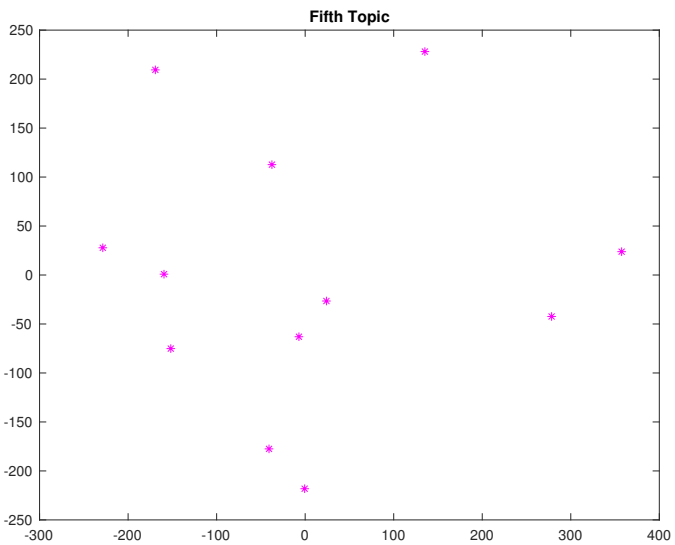
Topic-Word Correspondences

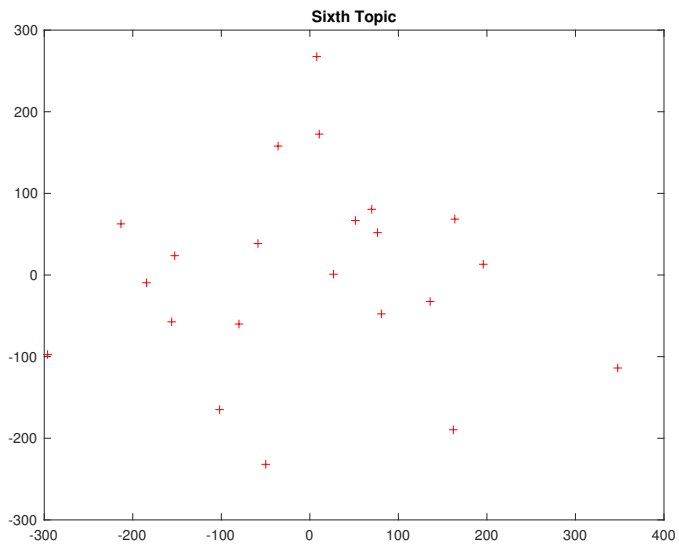


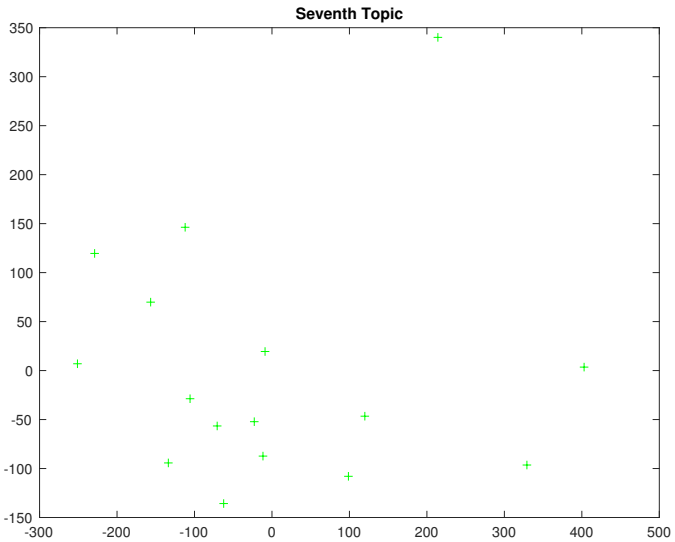


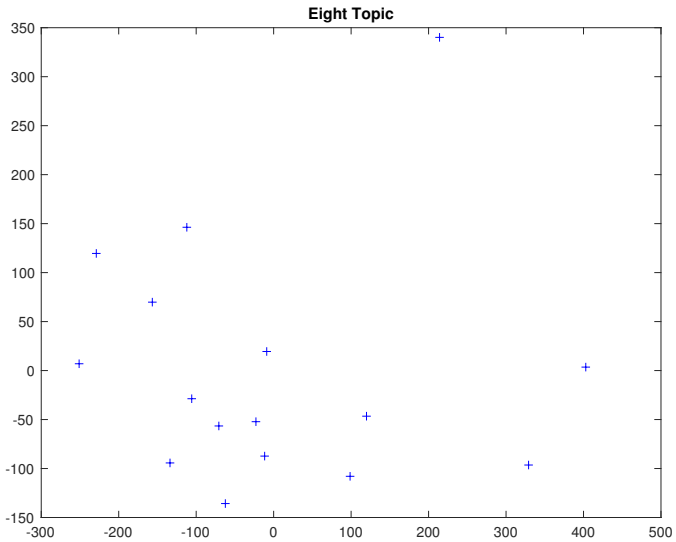

Third Topic
300
250
200
150
100
50
0
-50
-100
-150
-300
-200
-100
0
100
200
300
400
500

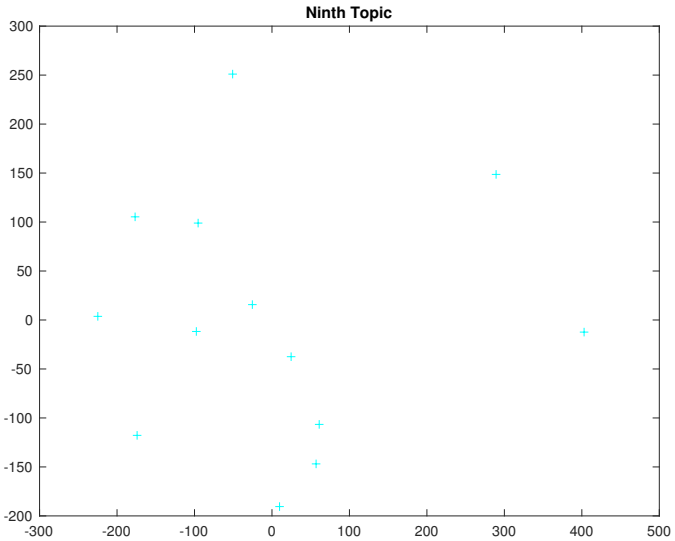

Fourth Topic
250
200
150
100
50
0
-50
-100
-150
-200
-100
0
100
200
300
400
500

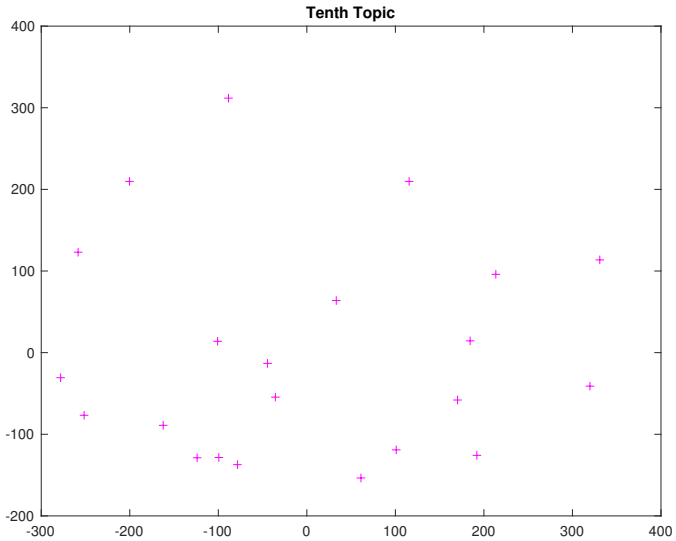

Fifth Topic
250
200
150
100
50
0
-50
-100
-150
-200
-250
-300
-200
-100
0
100
200
300
400

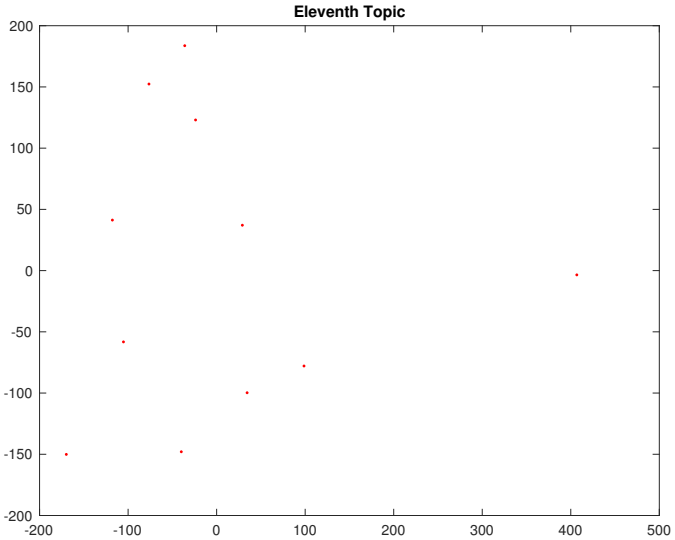

Sixth Topic
300
200
100
0
-100
-200
-300
-300
-200
-100
0
100
200
300
400

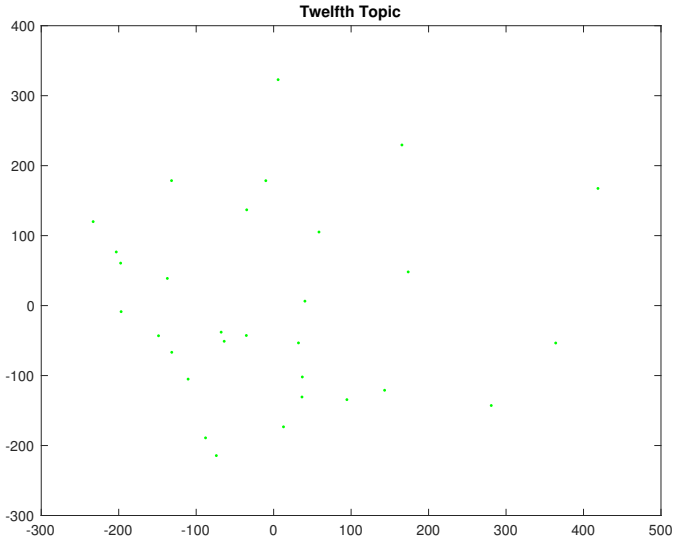

Seventh Topic
350
300
250
200
150
100
50
0
-50
-100
-150
-300
-200
-100
0
100
200
300
400
500

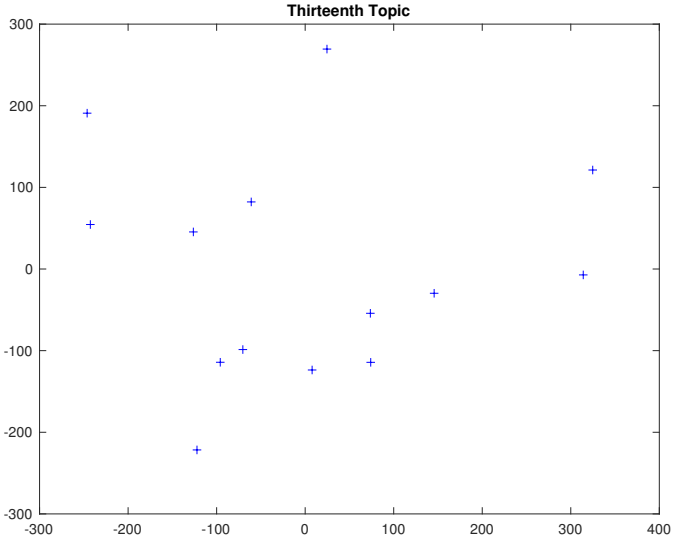

Eight Topic
350
300
250
200
150
100
50
0
-50
-100
-150
-300
-200
-100
0
100
200
300
400
500

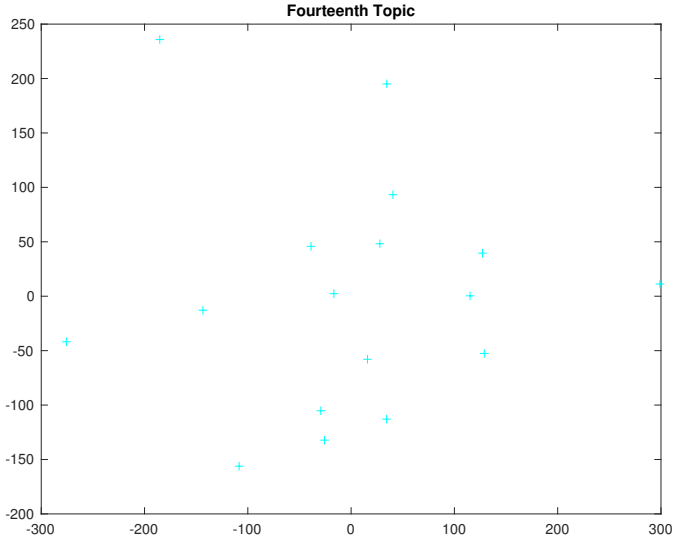

Ninth Topic
300
250
200
150
100
50
0
-50
-100
-150
-200
-300
-200
-100
0
100
200
300
400
500

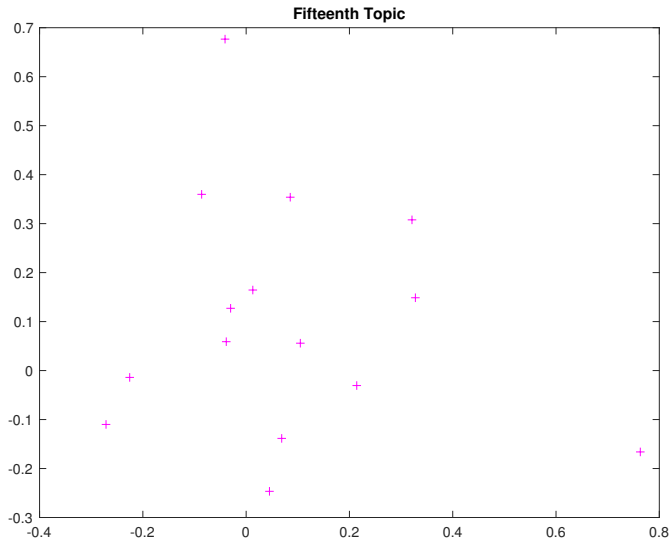

Tenth Topic
400
300
200
100
0
-100
-200
-300
-200
-100
0
100
200
300
400

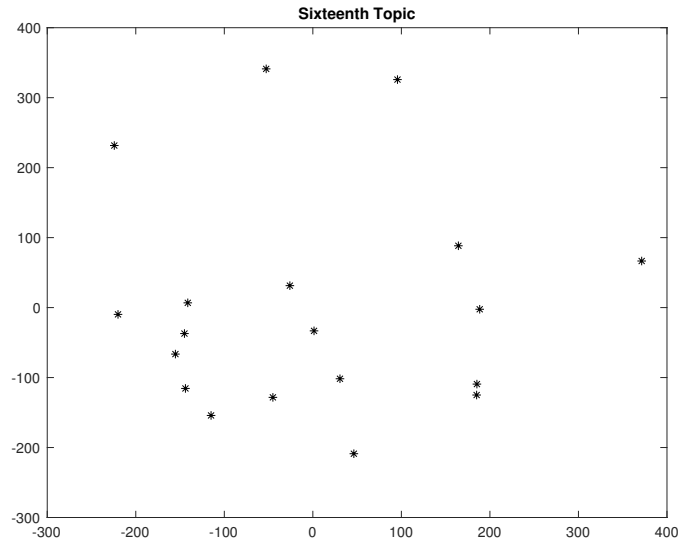

Eleventh Topic
200
150
100
50
0
-50
-100
-150
-200
-200
-100
0
100
200
300
400
500

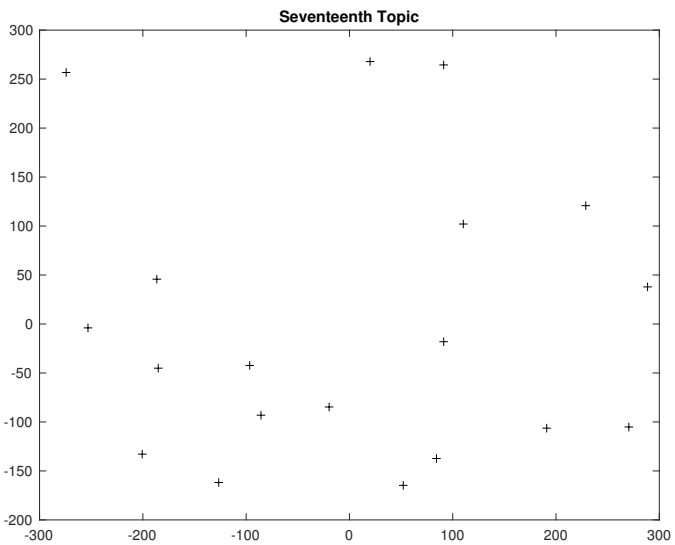

Twelfth Topic
400
300
200
100
0
-100
-200
-300
-300
-200
-100
0
100
200
300
400
500

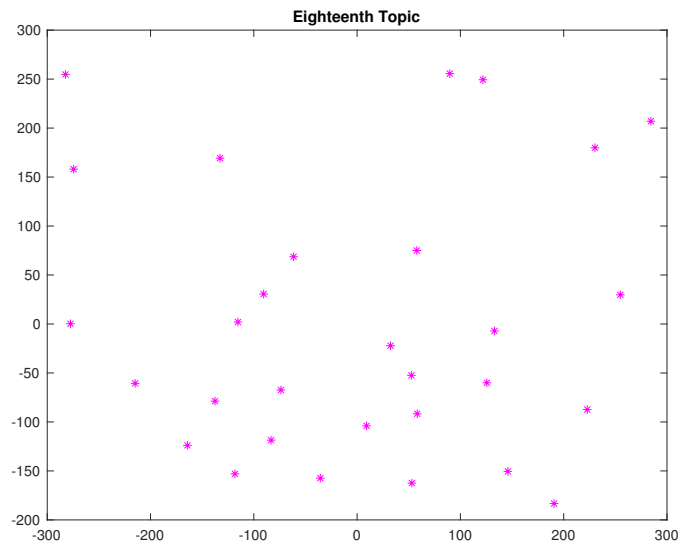

Thirteenth Topic
300
200
100
0
-100
-200
-300
-300
-200
-100
0
100
200
300
400

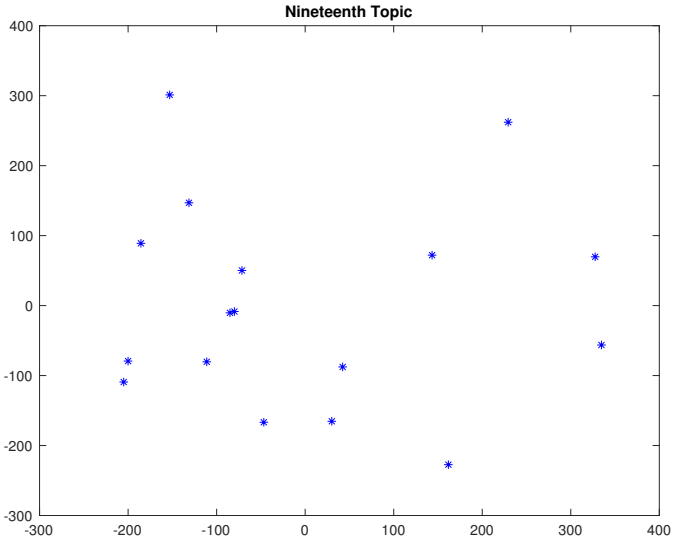

Fourteenth Topic
250
200
150
100
50
0
-50
-100
-150
-200
-300
-200
-100
0
100
200
300

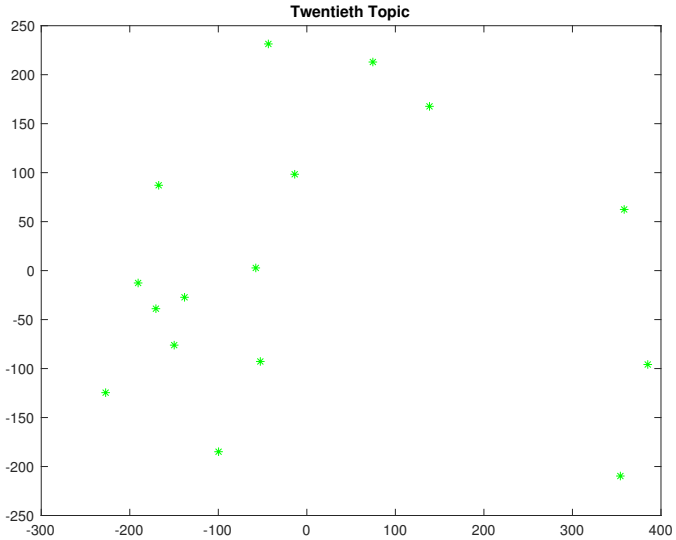

Fifteenth Topic
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.2
0
0.2
0.4
0.6
0.8


Sixteenth Topic
400
300
200
100
0
-100
-200
-300
-300
-200
-100
0
100
200
300
400


Seventeenth Topic
300
250
200
150
100
50
0
-50
-100
-150
-200
-300
-200
-100
0
100
200
300


Eighteenth Topic
300
250
200
150
100
50
0
-50
-100
-150
-200
-300
-200
-100
0
100
200
300


Nineteenth Topic
400
300
200
100
0
-100
-200
-300
-300
-200
-100
0
100
200
300
400


Twentieth Topic
250
200
150
100
50
0
-50
-100
-150
-200
-250
-300
-200
-100
0
100
200
300
400

Aknowledgements

Ringrazio innanzitutto il mio relatore Andrea Torsello, per la disponibilitá e la cura con cui mi ha seguito nella stesura della tesi. Ringrazio Andrea Gasparetto per avermi dato dei suggerimenti utili allo sviluppo del lavoro. Ringrazio i miei compagni di scuola: Roberta, Gianluca, Mattia, Matteo, Tommy, Mauro e molti altri, per aver condiviso con me questi anni di studio e di vita. Ringrazio tutti i miei professori, che mi hanno insegnato molte cose anche se a volte mi sono attivamente opposta.

Ringrazio Matteo, che mi é stato vicino dall'inizio di questo percorso e spero mi resterá vicino per ancora molto tempo. Ringrazio Michela, Irene, Giulia, Clarissa, e tutti gli altri amici senza i quali gli anni dell'universitá sarebbero stati di sicuro meno divertenti. Infine ringrazio la mia famiglia, che fortunatamente non ha fatto pressioni perché facessi il dottorato.