28462000 - Bourez Nicolas
25862000 - Loncour Romain

LDATA2010
18th of December 2023

LDATA2010 - Project report

1 Introduction

This report will embark on the journey of exploring our vizualisation dashboard. It will help you understand how to use it, what are its capabilities, where it strives and where it lacks and most importantly provide a good explanation of all our design choices.

Before opening our dashboard, make sure to follow the instructions in the README.md as they ensure that our project works.

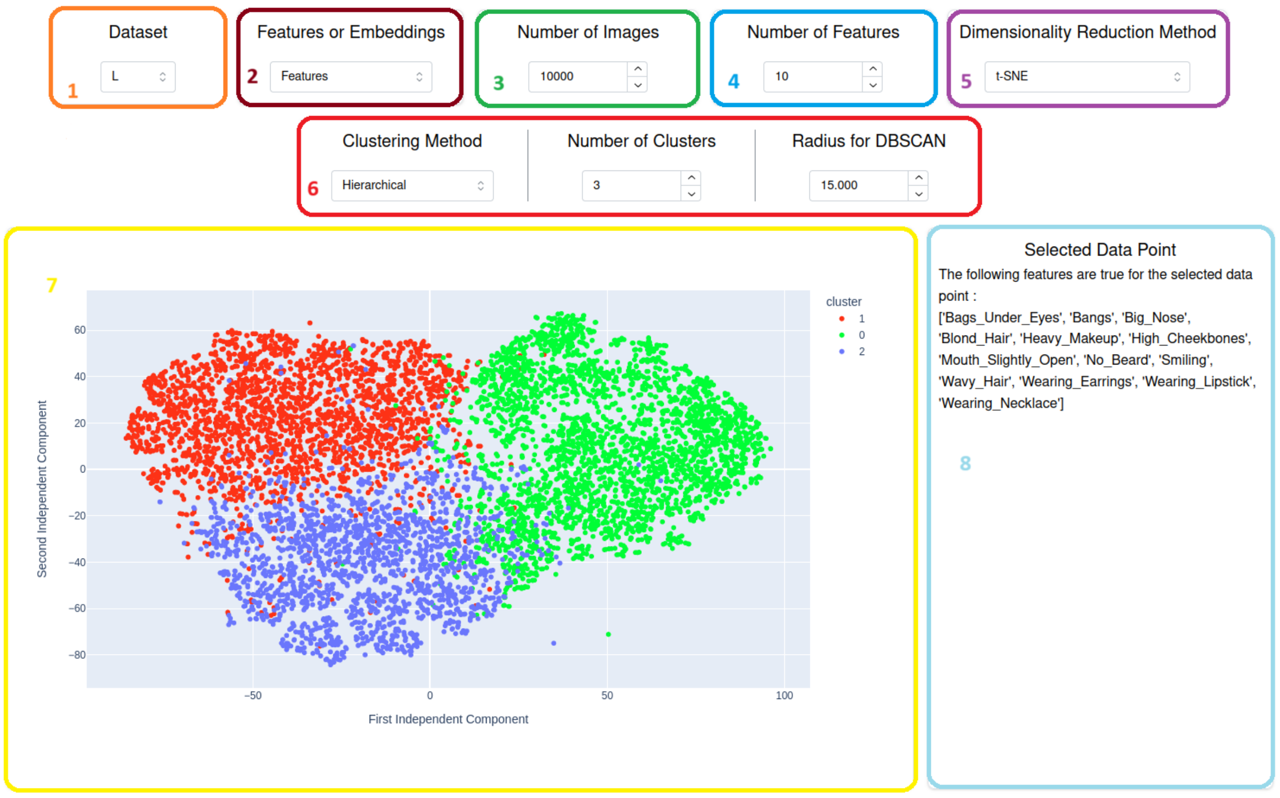
When opening our dashboard, you are greeted by an introduction page that details how the dashboard is structured and contains global information about the dashboard. You can start by reading what is written on this page as a good introduction to its main capabilities. However, everything written on that page, is also present in this report. After reading the introduction page, you can see on the left side of your screen a column containing our names, the university, the different pages you can browse and a brief paragraph that contains general information about the dataset as well as some small pieces of advice useful to get our interface to run somewhat smoothly.

After having seen and read all of that, we can now dive into the main attraction of our interface : The Main Page.

2 Main Visualization

Upon arrival at the main page, you can see that it contains one big plot and a lot of ways to modify it. While building this interface, we really wanted to have one main visualization that was very customizable. So that's what we did. This main visualization represents by default 1000 images chosen at random among the `celeba_s` dataset. The dimensionality reduction used to project on a 2-dimensional plane is the PCA as it is standard and relatively easy to compute and there is no clustering. This first representation uses the categorical values as data but it is very easy to toggle to embeddings with the selection bar on the right. It is also very easy to toggle from one dataset (S) to the other (L) with a selection bar on the top left. Something quite important about our dashboard is that it is possible and highly recommended to change the amount of images shown in the representation. Having too many images can clog the representation and increase drastically the computation time. We therefore advise the user to limit the number of images to a maximum of 5000 at most but our interface works well for a dataset comprising 2000 images, it is a good balance between efficiency and reactivity of our dashboard and expressiveness of the representation. Under this main plot are two minor data representations that will be detailed later in the report.

2.1 Overall Eagle View


Dataset: L
Features or Embeddings: Features
Number of Images: 10000
Number of Features: 10
Dimensionality Reduction Method: t-SNE
Clustering Method: Hierarchical
Number of Clusters: 3
Radius for DBSCAN: 15.000

Selected Data Point
The following features are true for the selected data point :
['Bags_Under_Eyes', 'Bangs', 'Big_Nose', 'Blond_Hair', 'Heavy_Makeup', 'High_Cheekbones', 'Mouth_Slightly_Open', 'No_Beard', 'Smiling', 'Wavy_Hair', 'Wearing_Earrings', 'Wearing_Lipstick', 'Wearing_Necklace']

Figure 1: Main Part of our Visualization

2.1.1 Dataset Choice

The first element we'll present here is the dataset toggle bar. The images comprising the dataset have been encoded using two different neural networks from *Insightface* so there are two different datasets to be chosen from. We won't go any deeper as to how these embeddings are generated. By default, the interface uses the `celeba_s` dataset.

2.1.2 Features or Embeddings

At first, we wanted to only use the embeddings of the images for the visualization but realized that the results were not the best at first glance, the 2-D outputted plot was usually very cluttered and it was difficult to interpret anything tangible. Therefore, we added to possibility of toggling from the categorical features (*Features*) to the embeddings (*Embeddings*) so that the user can choose the representaiton they please. We also introduced this functionality because the features dataset contains less columns than the embeddings dataset does so the calculations are faster. By defaut, the interface is toggled on the *Features* dataset.

2.1.3 Number of Images

The third element is one of the most important ones. It allows the user to change the amount of images that will be represented on the main graph. It is strongly advised to reduce the number of images to under 3000 as some methods are quite hard computationally. The images are chosen at random among the 30.000 data points present in the DataFrames. The change in the number of images is automatically taken into account by the two other representations. We advise the user to firstly explore the dataset using at most 500 points and then increase the number of points to validate its intuitions. By default, the interface uses 1000 data points.

2.1.4 Number of Features

This interface allows for feature extraction, we realized that there were a lot of embeddings which created a huge amount of dimensions which, in turn, outputted a messy visualization when using those. We therefore allowed for the possibility of using a PCA that would project the data on a maximum of 39 principal components. We chose 39 because it is the number of features in the dataset and feature extraction can also be done when using the *Features* dataset. We thought that being able to extract more than 39 principal components would defeat the purpose of extracting features in the first place. Be default, the interface doesn't do feature extraction.

2.1.5 Dimensionality Reduction Method

There are multiple dimensionality reduction methods that can be used, they will be detailed further in subsection 2.2. By default, the interface uses PCA as it is relatively quick to compute.

2.1.6 Clustering Method

There are multiple clustering methods that we implemented, they will be detailed further in subsection 2.3. There are two different parameters which can be changed in this interface, the number of clusters that is used for hierarchical clustering and K-Means and the radius that is used by DBSCAN. By default, there is no clustering that is done.

2.1.7 Main Graph

The main graph is a scatterplot that uses the different parameters chosen previously to display the main visualization. It is highly customizable but, in turn, may lack in reactivity. There is for example a 3-4 seconds delay before it is displayed when first opening the main page. By the way, to know if the visualizations have been changed, you can look at the name of the tab in your browser. If the tab is named `Updating...`, the interface is retreiving data. But, if the tab is named `Celebs Visualization with Dash Plotly`, the data has been shown. It is highly advised to wait for the tab to be done updating before sending another request (aka. modifying the parameters).

2.1.8 Selected Data Point

Some interactivity is built in our visualization. More precisely, it is possible to select a point by clicking on it and see which ones of its featurs are set. This frame contains the list of all of them. They are also represented on a following visualization known as the *Features Histogram*. The yellow bins are set and the blue ones are not. This can be useful to see if two data points that are close share or not common features (which would be an expected behaviour). This functionality is quite responsive. It is available for every combination of previous parameters.

2.1.9 Features Histogram

After this main visualization, we added two secondary plots that compliment the main one. The first one is a features histogram, it counts the amount of times a feature is set in the currently displayed dataset. As the data is relatively well distributed, the shape of this graph is usually the same. We didn't sort the bins from highest to lowest as the purpose of the graph is to have a feel of the distribution between the different runs so a reordering of the bins would not suit this purpose. But, most importantly, the point of this visualization resides in the color of the bins. When a point is selected on the main visualization, its selected features get colored on this plot which can be useful to compare neighboring points, for example. By default, the color of this graph is neutral since no point has been selected.

2.1.10 Correlation matrix

We have also decided to plot the correlation matrix without much interactivity. This tool is simply at the disposal of the user if they need it. We follow this philosophy of making things acessible even if they may not be the most interesting, some user may find it useful. We ensure that the representation is interpretable and well-presented, sober but clear.

2.2 Dimensionality Reduction

We have decided to implement three dimensionality reduction methods, PCA, ICA and t-SNE. At first, we wanted to only incorporate the PCA and t-SNE but thought that offering more options to the user would be more important so we decided to offer the possibility of using the ICA. We will here explain our choice of implementing the PCA and t-SNE.

2.2.1 PCA

The **PCA** stands for Principal Component Analysis and is a linear dimensionality reduction method that can be used for feature extraction or visualization purposes. In the context of our project, we used it to for both. The PCA reduces the number of dimensions by finding the n (usually 2 for visualization) axes that maximize the explained variance of the data. The first principal component is this first axis and the second principal component is the second one.

2.2.2 t-SNE

We also implemented a similarity-based dimensionality reduction known as **t-SNE** which is one of the best non-linear dimRed methods currently available. It is a statistical method commonly used to reduce the number of dimensions from a very high amount to 2 or 3. The t-SNE algorithm comprises two main stages. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects are assigned a higher probability while dissimilar points are assigned a lower probability. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map. In our case, t-SNE has been hard to use as it has a $O(n^2)$ complexity which has been proven to be difficult to manage even for datasets containing multiple thousands of points. We advise the user to use it with datasets smaller than 1000 images or after having extracted a smaller number of features.

2.3 Clustering

We implemented three clustering methods that are very distinct from one another : K-Means, Hierarchical with normal Linkage and DBSCAN.

2.3.1 K-Means

K-Means is a clustering method where points are separated in a set number K of clusters. This is an iterative method that always converges to a local solution. We start the method by plotting K

random points on the 2-D plane named centers. Iteratively, the points closest to each center will join the corresponding cluster. After every recruitment phase, the center is recomputed as the mean of every point. This algorithm converges in a finite amount of iterations.

2.3.2 Hierarchical Clustering with normal linkage

Hierarchical Clustering with normal linkage is an iterative clustering method that allows to, when all computations are done, change the number of clusters very easily. At every iteration, the points closest to one another based on the linkage are joined in a combined cluster until only one cluster remains. In this context, the linkage function is important as it chooses the order and which points are joined. The simple linkage is the one we decided to use as it is feasible computationally and provides interpretable results. Using average linkage could have been more precise in the representation but would have been extremely slow as it would increase the amount of computed operations by $O(n^2)$.

2.3.3 DBSCAN

DBSCAN is a density-based clustering method. It is the only clustering method that can mark a point as not part of a cluster if it is too far from the other. DBSCAN is a quite intuitive method, if a new point is close some other points, it enters the cluster created by these points. It is however a difficult method to use in high dimensions as all the points seem to be very far away due to the *curse of dimensionality.* Therefore, it is advised to combine DBSCAN with data extraction to bring the points somewhat closer. This approach may however break some relationships that existed in the data in higher dimensions. DBSCAN is somewhat slow as its complexity is $O(n^2)$ so it would maybe not be the best method for exploring the current dataset.

2.4 Limitations & Improvements

One big limitation of our dashboard is its lack of reactivity to changes, even the slightest. For example, one could think that changing the number of clusters in hierarchical clustering would be quick but the interface has to recompute everything. It can be impractical but a good way to bypass this limitation is to follow one of the previous two advices written in red, to use a smaller amount of images to 'try a little something' and then to use more images for validation of the intuition and to generate nice plots.

Another functionality that we didn't manage to implement was a way to select two datapoints on the main visualization to be able to compare them. It would have given a way more precise way to compare two points. The current method only gives a feel as to which features change from one point to another. It was however very difficult to implement in Dash.

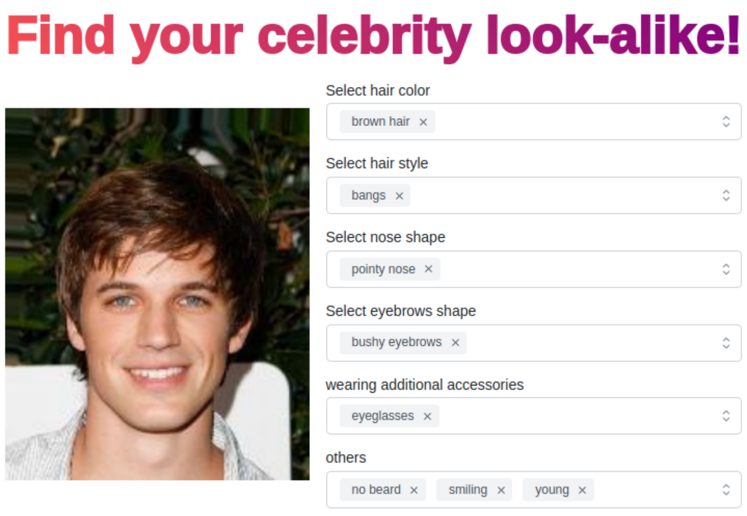
It would have been interesting to add the possibility of changing the linkage function when using hierarchical clustering.

After building this interface, we feel that we could have added more data representations such as a summary of the different features or being able to select multiple points but we judged that this would probably make our interface messier. There is a tradeoff between clarity and the number of representations.

3 Fun part

As indicated by its name, this part is not as serious as the previous one. The goal of this page is to show to the user a subset of the images by selecting the attributes they would like the celebrity to have. The title of the interface is "Find your celebrity look-alike", which gives a goal to the user and brings engagement. They have to select their own attributes (brown hair, no beard, etc.) and have to find the celebrity which shares the most features with they.

Here is an example of one of the author's celebrity look-alike :



Find your celebrity look-alike!

Select hair color: brown hair ×

Select hair style: bangs ×

Select nose shape: pointy nose ×

Select eyebrows shape: bushy eyebrows ×

wearing additional accessories: eyeglasses ×

others: no beard × smiling × young ×

3.1 Implementation details

Internally, the software is returning the images for which the sum of all selected dummy values are the highest. If I had selected `brown_hair, bangs` and `pointy_noise`, it would have calculated :

max(`df['Brown_Hair'] + df['Bangs']] + df['Pointy_Nose']`)

and return the image associated with it. The *generate another one* button just takes another picture from the subset of those that respect the *max* condition.

3.2 Improvement suggestions

As an improvement, we thought about letting the user upload a picture of himself and feeding this image to the same neural network as the one that the assistants used for the other images. Based on the embeddings we would have received, we could have found the closest image in the embeddings' space. Because we are not sure about which neural network had been used to get the embeddings, we didn't implement this functionality.