The Venice Atlas

A Digital Humanities atlas project by DH101 EPFL Students

était
sans
elle
Julien
Julien
bien
avait
dit
dans
était
Julien
bien
était
Julien
était
dit
avait
dit
bien
bien
était
Elle
était
était

Semi-Automatic Transcription Tool for Ancient Manuscripts

In this article, we investigate various techniques from the fields of shape analysis and image processing in order to construct a ***semi-automatic transcription tool*** *for ancient manuscripts. First, we design a* ***shape matching procedure*** *using shape contexts, as introduced in [1], and exploit this procedure to compute* ***different distances*** *between two arbitrary shapes/words. Then, we use Fischer discrimination to combine these distances in a single* ***similarity measure*** *and use it to naturally represent the words on a* ***similarity graph****. Finally, we investigate an* ***unsupervised clustering analysis*** *on this graph to create groups of semantically similar words and propose an* ***uncertainty measure*** *associated with the attribution of one word to a group. The clusters together with the uncertainty measure will form the core of the semi-automatic transcription tool, that we will test on a dataset of 42 words.*

Introduction

Shape matching is an attempt to operationalize our intuitive notion of *similarity* between shapes: if we consider for example the two handwritten letters in Figure 1, they appear rather similar to a human observer, while being radically different if compared as vectors of pixels.

Figure 1: Example of handwritten letters: very similar to a human observer, but radically different in terms of pixels.

One way to address this issue could be to design a **notion of distance between two shapes** : then, two shapes that match pretty well will be close in terms of this distance. More precisely, we will design three distances all based on the following shape matching procedure.

Description of the Shape Matching procedure

In the approach we investigate here, we will, as suggested in [1], treat an object as a point set and we assume that the *shape of an object is captured by a finite subset of its points*. In practice, we uniformly sample the internal and external contour of the object using an edge detector (see Figure 2). This gives us a set $\mathcal{P} = \{p_1, \dots, p_n\}, p_i \in \mathbb{R}^2$, of n points.

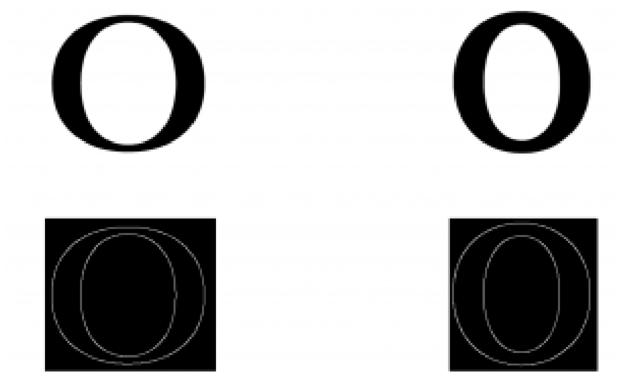


Figure 2: Edge detection performed on two "o" letters from different fonts. Top: original letters, bottom: edge detection of the letters.

Then, we face a **correspondence problem**: for each point p_i on the first shape, we want to find the "best" matching point q_i on the second shape. This matching can be efficiently performed by using **rich local descriptors** of the shape, such as **shape context**, introduced in [1]. The idea is the following :

Given a point p_i on the shape, we consider the set of vectors originating from a point to all other sample points on a shape. These vectors express the configuration of the entire shape with respect to the reference point (see Figure 3). Obviously, as the number of points increase, the description of the shape becomes more precise. But the full set of vectors is much too detailed to be used as a shape descriptor, therefore we choose a more compact descriptor: we consider the densities of these vectors in the log-polar space.

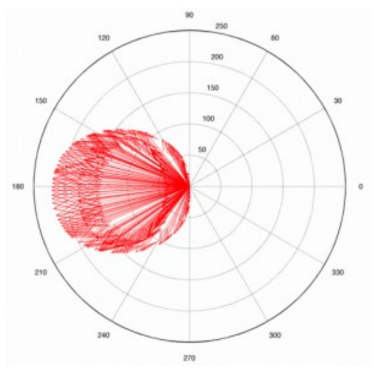


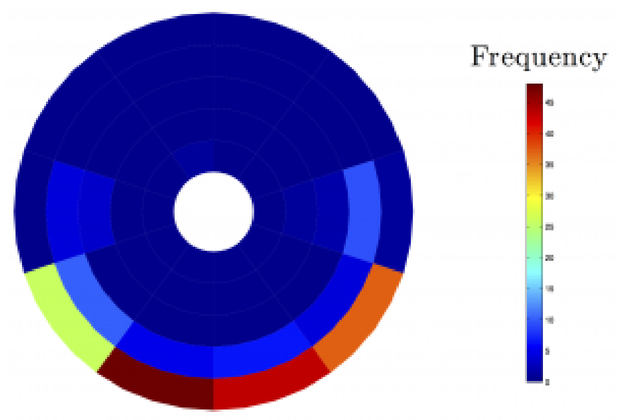
Figure 3: Set of vectors originating from a point to all other sample points on the "o" shape.

More specifically, for each point p_i on the shape, the **shape context** of p_i is defined by the histogram h_i (see Figure 4):

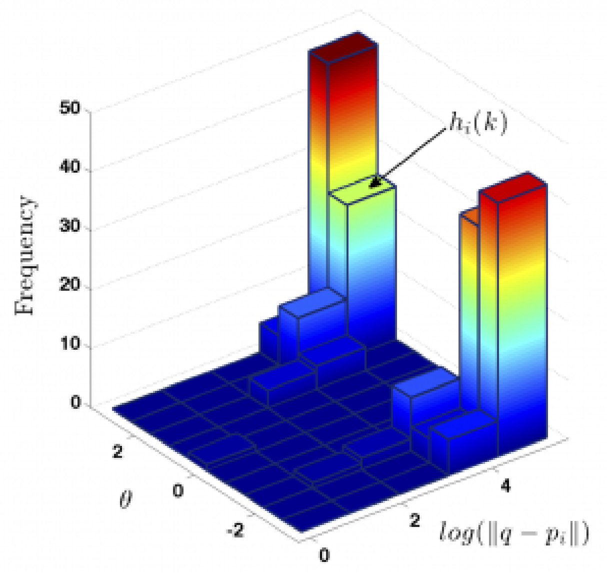
$$h_i(k) = \#\{q \in \mathcal{P}, q \neq p_i : (q - p_i) \in bin(k)\},$$

where the bins $bin(k)$ are taken uniformly in the log-polar space.

Figure 4: Shape context of a point p_i on the letter "o".



Histogram in the log-polar space.



Alternative representation of the same histogram.

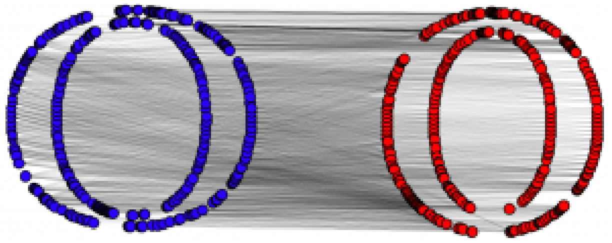
Then, the problem is to compare the shape context of two points p_i on the first shape and q_i on the second shape. We use the χ^2 test statistic C_{ij} to determine wether or not two histograms (i.e. shape context) are significantly different:

$$C_{ij} = C(p_i, q_i) = \frac{1}{2}\sum_{k=1}^{K} \frac{(h_i(k)-h_j(k))^2}{h_i(k)+h_j(k)}.$$

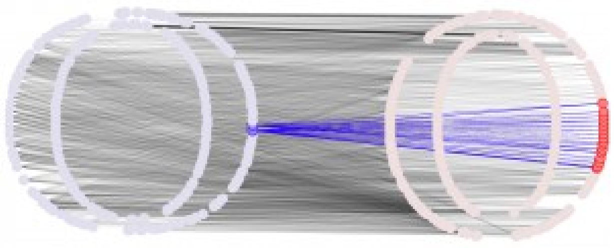
This statistic can be interpreted as the **cost of matching** p_i and q_i.

Then, we draw a bipartite graph, with nodes the sampled points of the first and second shape. Then, every points p_i and q_i are linked by an edge with weight C_{ij} if we cannot reject the hypothesis that their two shape context are the same (see Figure 5). This gives us for every point on the first shape, a set of potential match on the second one, and respectively (see Figure 5).

Figure 5: The problem of shape matching can be transformed in a problem of bipartite graph matching



We build a bipartite graph matching connecting points of each shape which shape context are not significantly different



Potential matching points on the second shape for a point on the first shape.

Finally, we want to obtain the best bipartite graph matching, namely, we want to minimize the total cost matching,

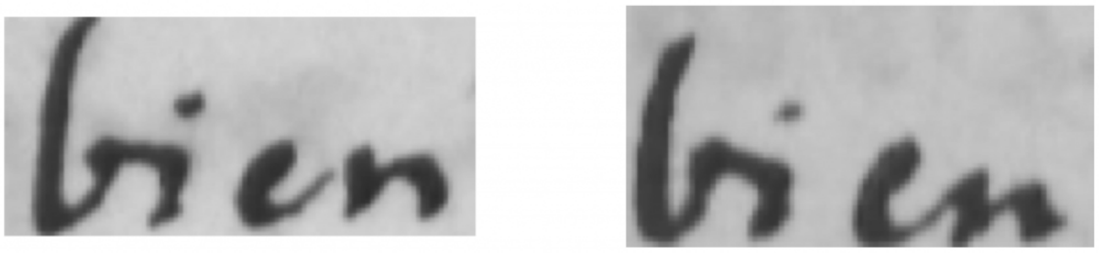
$$H(\pi) = \sum_i C(p_i, q_{\pi(i)}),$$

with the constraint that the matching be one-to-one, π is a permutation on the neighbors of p_i.

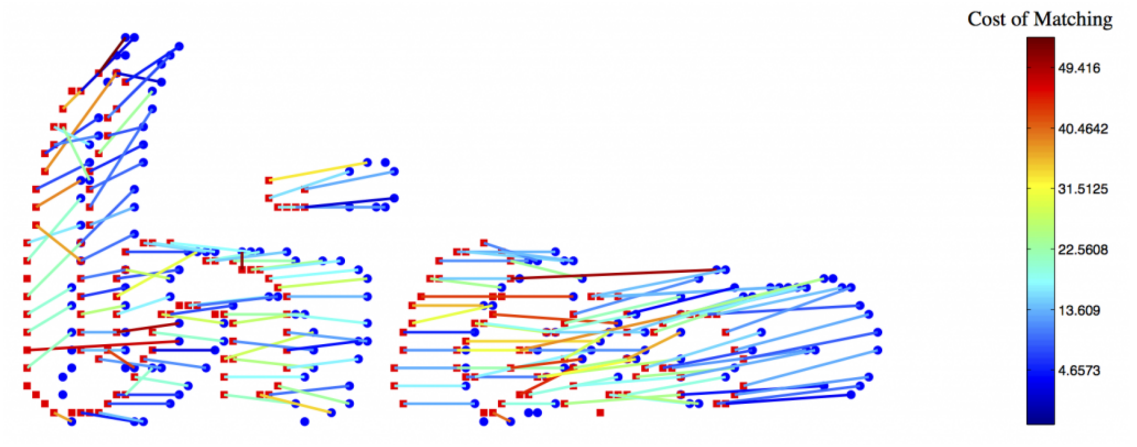
Construction of Shape Distances

Given two arbitrary shapes, we can now obtain potential correspondences between points on these two shapes by using the shape matching procedure described above. For example, correspondences between points on the two words in Figure 6 a) are given by the best bipartite matching in Figure 6 b).

Figure 6: *Shape matching procedure on two arbitrary words*



a) The two words on which the shape matching procedure is performed



b) Best bipartite matching between these two shapes and cost of this matching.

Then, we can use this finite set of correspondences to estimate a transformation of the plane $T : \mathbb{R}^2 \to \mathbb{R}^2$, allowing us to map any point on the first shape to a point on the second shape, and therefore extending the discrete set of correspondences to the continuous level.

To this end, we use two **thin plate spline** transformations (see [1]) for each coordinates:

$$T(x, y) = (f_x(x, y), f_y(x, y)).$$

More precisely, we choose a *source* and a *target* between the two shapes and then we solve two 1D interpolation problems.

Roughly speaking, we seek for the transformation that *minimizes the bending energy necessary to map points on the source shape to their corresponding matches on the target shape.*

In the particular example of the two words in Figure 6, we can see on the subsequent video the effect of applying the transformation T to the first word.

Thin plate spline transformation applied to sha...

0:00 / 0:14

With this mapping, we can already imagine two distances between two arbitrary shapes $\mathcal{P}$ and $\mathcal{Q}$. The first one is a *generalization to the continuous level of the total cost matching* H introduced previously:

$$D_{mc}(\mathcal{P},\mathcal{Q}) = \frac{1}{n}\sum_{p\in\mathcal{P}} \operatorname{argmin}_{q\in\mathcal{Q}} C(p, T(q)) + \frac{1}{m}\sum_{q\in\mathcal{Q}} \operatorname{argmin}_{p\in\mathcal{P}} C(p, T(q)),$$

with $T(\cdot)$ the thin plate spline transformation and $C(\cdot,\cdot)$ the cost of matching two points introduced in the previous post. We can also compute another distance, D_{bend} corresponding to the *amount of deformation* necessary to align the two shapes. This distance is naturally set equal to the bending energy I_f:

$$I_f = \int\int_{\mathbb{R}^2} \left(\frac{\partial^2 f_x}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 f_x}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 f_x}{\partial y^2}\right)^2 dxdy.$$

Finally, we introduce a last distance D_{ac} capturing the *appearance information* (texture and intensity information of the two grayscale images). To this end, we apply the thin plate spline transformation to the source image (see Figure 7) and compute the difference (pixels by pixels, see Figure 8) between the resulting warped image and the target image.

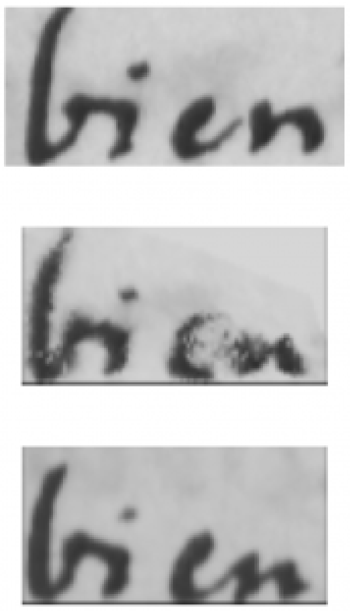
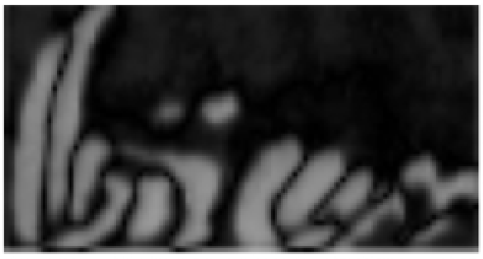


Figure 7: ***Top:*** *Source image,* ***Middle:*** *Source image warped with thin plate spline transformation,* ***Bottom:*** *Target image.*

Then, we compute the squared mean and the standard deviation of the resulting image (interpreted as a matrix), and set the third distance D_{ac} equal to the sum of these two quantities.



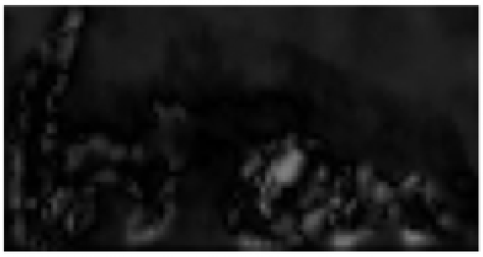


Figure 8: ***Top:*** *Difference between the two original images,* ***Bottom:*** *Difference between the two images after transformation of the source image*

Construction of a Similarity Measure

From the three distances previously introduced, we wish to design a **similarity measure**, expressing how close is a shape from another. To this end, we first build a single distance D that *separates the better the data*.

For this, we consider a *training set of 42 words*. In this training set, we know which words are *semantically* the **same** and which words are *semantically* **different**. Then, we compute the three distances D_{mc}, D_{bend}, D_{ac} between each pair of words. For each pair, we obtain a point of $\mathbb{R}^3$ which coordinates correspond to the respective values of the three distances. Additionally, we color this point in **red** if the two words compared are semantically the same and in **blue** if the two words compared are semantically different. We obtain the following plot:

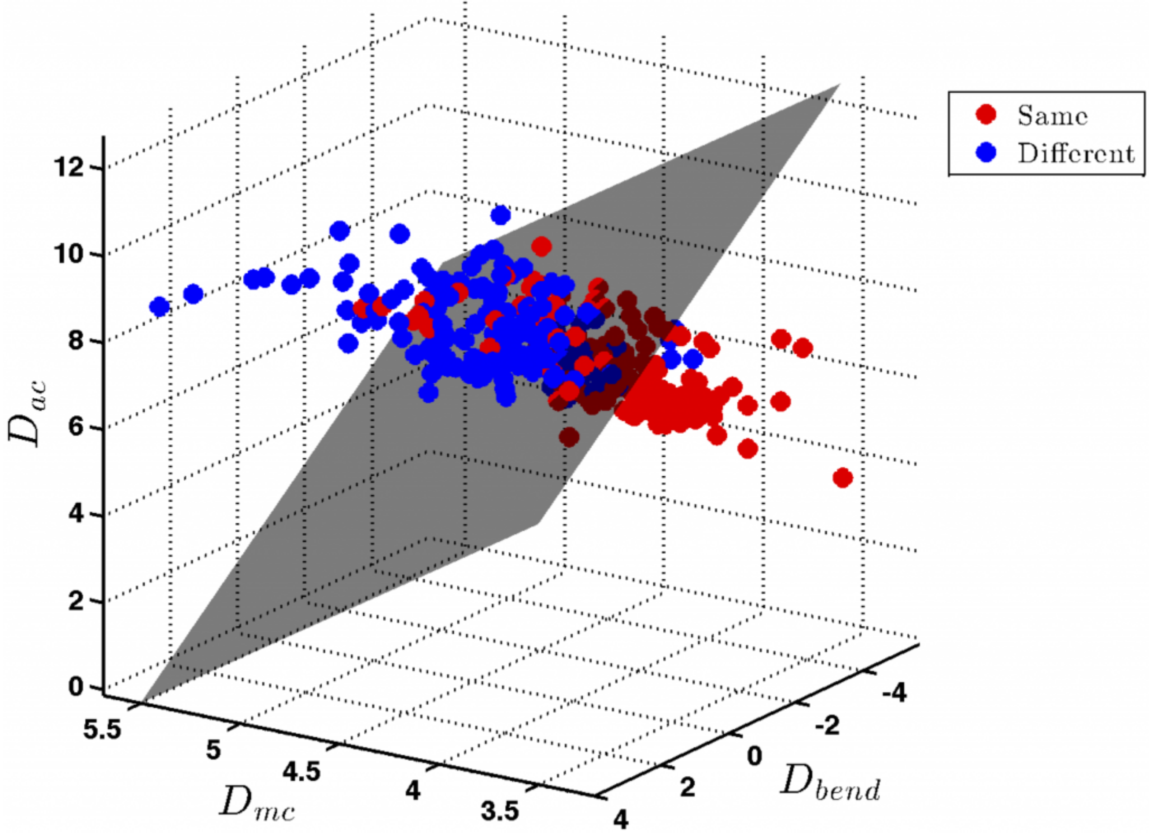


Figure 9: Plane separating the better the two groups.

The idea is now to find the **plane that separates the better the two groups**. Then, by projecting onto the vector normal to this plane, we obtain a unique distance that separates the best the two groups. This can be done using **Fischer discrimination**, and we obtain the following distance:

$$D = 2.68 \times \log(D_{mc}) + 0.68 \times \log(D_{bend}) + 0.89 \times \log(D_{ac}).$$

From this unified notion of distance, we introduce the **similarity** between two shapes $\mathcal{P}$ and $\mathcal{Q}$:

$$S(\mathcal{P}, \mathcal{Q}) = \frac{1}{\mathcal{D}(\mathcal{P}, \mathcal{Q})}.$$

If two shapes are very close, then their distance will be close to 0, and the similarity close to 1. Conversely, shapes very different will have a similarity close to 0. Note that the similarity measure is always well defined because we never compare strictly identical shapes.

Clustering Analysis with k-Nearest Neighbors

This measure of similarity will allow us to perform **clustering analysis** using the **k-nearest neighbors** methodology. To this end, we build an undirected fully-connected graph: each node correspond to a shape/word and edges between the nodes are weighted by the similarity function between the two nodes (shapes). Such a graph is called a **similarity graph**. An example of it is presented on Figure 10. In this example, we compared a database of *42 handwritten words, with 11 semantically different words*.

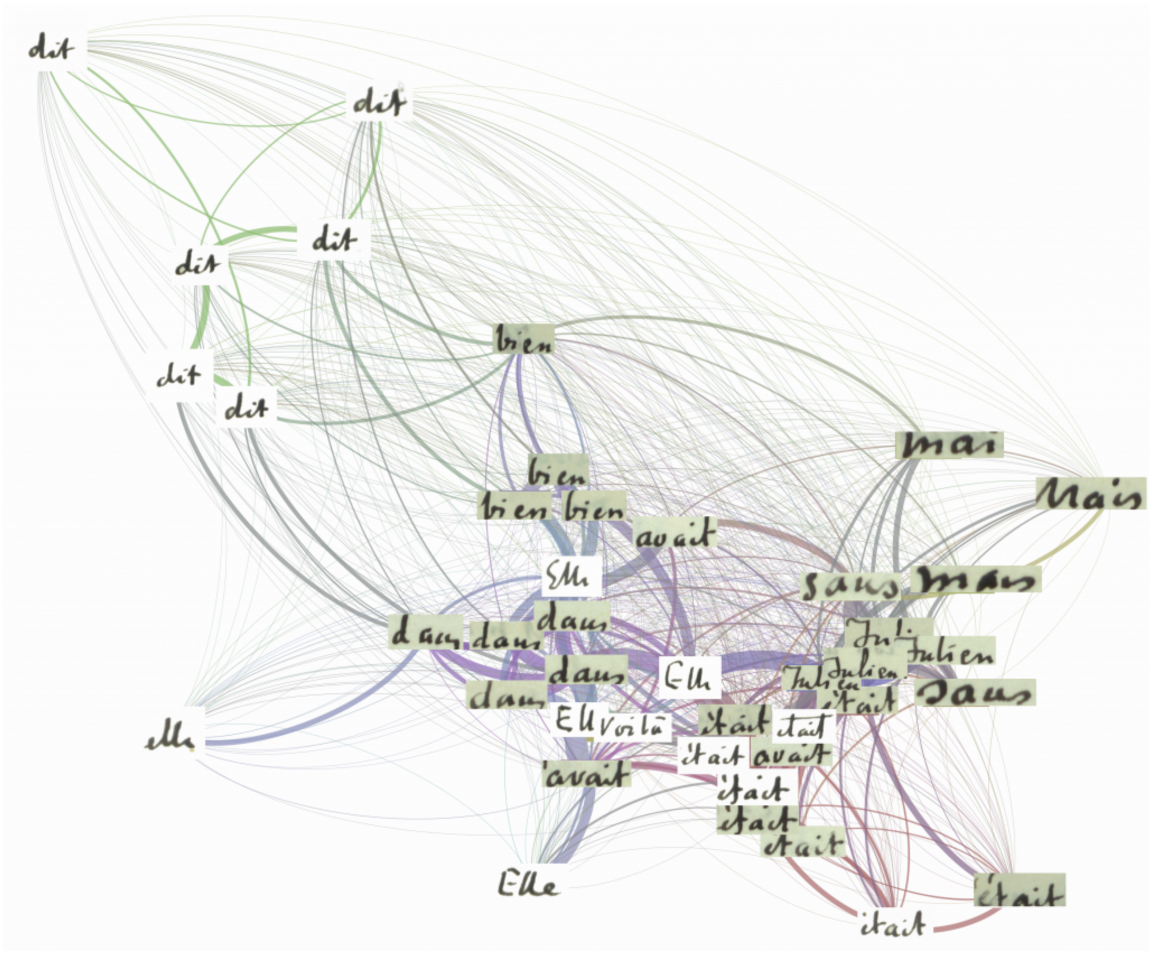


Figure 10: *Fully connected similarity graph of 42 handwritten words.*

Then, the idea of the k**-nearest neighbors** is to first reduce the number of edges in the similarity graph: two vertices x_i and x_j are connected if x_j is among the k nearest neighbors of x_i, with k a parameter to choose. Once the number of edges reduced, *the goal is to separate the graph in n components (clusters), so that the the sum of the weights of the edges we have to cut to disconnect these components is minimized*.

If we apply this procedure to our similarity graph, with $k = 4$ and $n = 9$ we obtain the clusters shown on Figure 11. On this figure, each color corresponds to a cluster, and nodes have been colored according to their respective cluster. We additionally added an **uncertainty measure**, represented by the following color code:

- **Green:** The nodes which only neighbors are members of their cluster: these nodes being only connected with their pairs, it is really unlikely that they have been misclassified.
- **Pink:** The nodes which have more neighbors in their cluster than outside of it: these nodes are strongly connected with their cluster, but still they could relate with some other nodes outside of it, so we might wonder if they have been correctly classified.
- **Red:** The nodes which have more neighbors outside their cluster than inside: these nodes are better connected with the rest of the graph than to the cluster they belong to. Therefore we have to be very careful in the trust we put in these nodes: it is very likely that they have been misclassified and that they should belong in reality to another cluster.

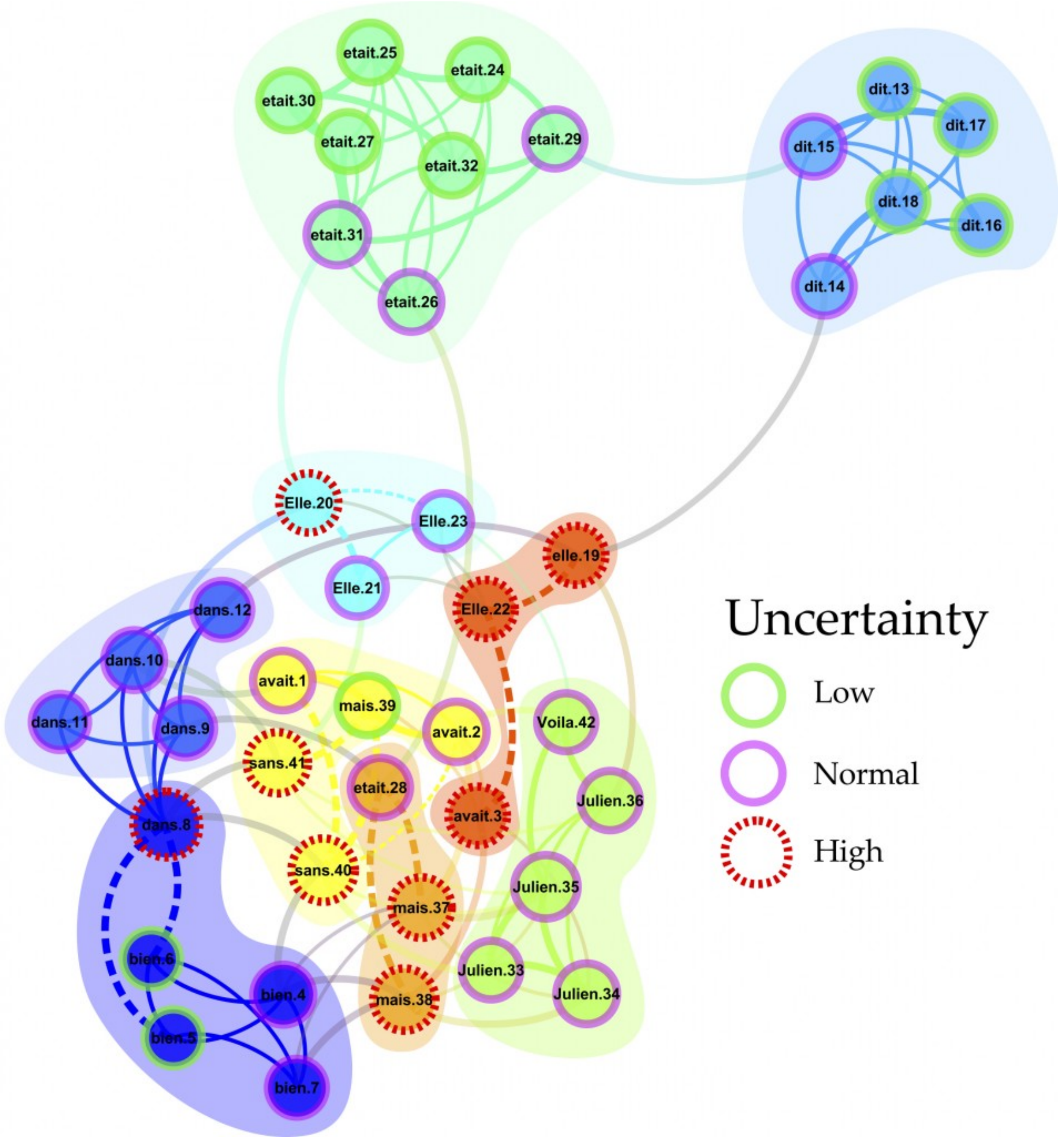


Figure 11: Result of the clustering analysis. Each color corresponds to a cluster.

The results of the clustering analysis are quiet good: **86%** (on average) of the words have been correctly grouped together. More precisely, **98%** of the green nodes have been correctly classified, against **87%** for the pink nodes and only **35%** for the red (so the uncertainty measure seems to be a good indicator).

A Semi-Automatic Transcription tool (Demo)

We use the preceding procedures to develop a semi-automatic transcription tool of ancient texts.

Let say we have digitized an old handwritten book. Then, to be able to extract the information out of it, we need to transcript it, so that a computer could recognize the words. Usually, a specialist is paid to realize manually this tedious task. But his work could be considerably facilitated by the clustering analysis developed earlier: as soon as he types the transcription of a word, it automatically propagates to all the members of the cluster to which the word belongs. In the demo we present here, the actual numbers of words the user has to type in order to transcript all the 42 words is reduced by **70% on average**. Of course errors might occur, but we can notify the user of potential mis-transcriptions using the uncertainty measure previously introduced. In the demo we propose here, uncertain words are transparent and represented into dashed-line boxes (see figure 12). An alternative version representing uncertainty with a color code is also available here (**red:** high uncertainty, **blue:** normal uncertainty, **green:** low uncertainty).

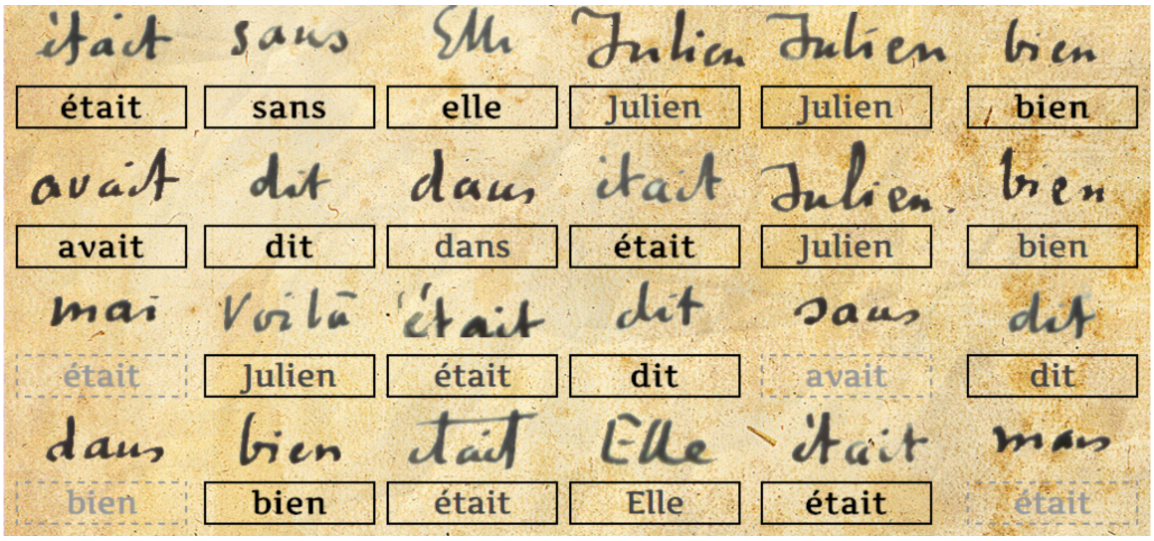


Figure 12: *Semi-automatic transcription tool on a training set of 42 words.*

Demo of the tool on a database of 42 words:

For optimal performances, please begin transcription by words in plain black boxes and correct potential errors in dashed-line boxes.

>Direct link to the demo<

Conclusion

The results obtained on the training set of 42 words are quite encouraging: the number of words to type during the transcription process is reduced on average by 70% and 86% of the words have been correctly classified by the clustering analysis. However, the current version is static, and doesn't update the clusters when the user is manually correcting a mis-transcripted word. It would be interesting to implement such a dynamic behavior, in order to improve even more the accuracy and efficiency of the tool. On the long range, we could even imagine a collaborative database storing the transcriptions of each user of the tool, so that the procedure would become more and more efficient and eventually tend to a fully automatic tool when the number of transcripted words in the database becomes big enough. Finally, there is still a lot to do in terms of computation: the running time of the procedure explodes with the number of words to process. It is necessary to optimize the code in C++ and maybe reduce the number of operations by applying pre-filters to the words, so that we do not compare words that are obviously different.

SIMEONI Matthieu

References

[1] S. Belongie,J. malik, J. Puzicha, *Shape Matching an Object Recognition using Shape Contexts*, IEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, April 2002.

[2] Daniel Kressner, *Computational Linear Algebra*, Lecture notes, EPFL, Spring 2014.

[3] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction,* Second Edition, February 2009.