Document Analysis - Report

Exercise 3

Aebischer Nadia (nadia.aebischer@unifr.ch)
Luyet Gil (gil.luyet@unifr.ch)

Introduction

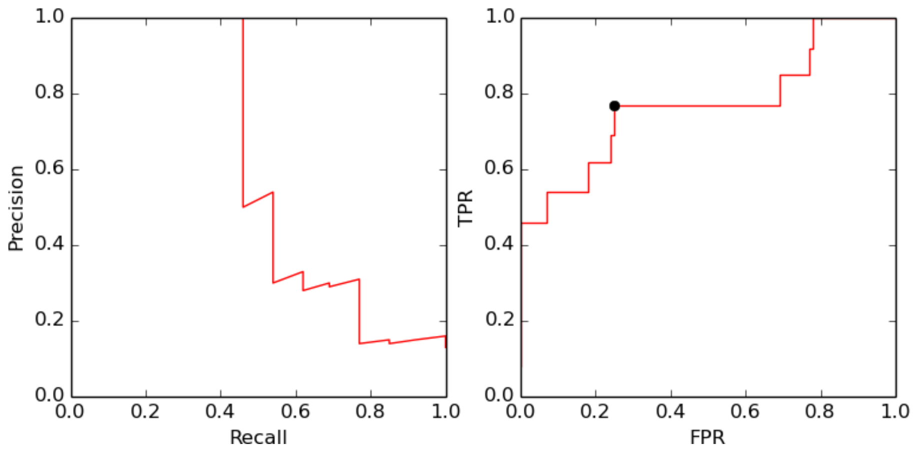
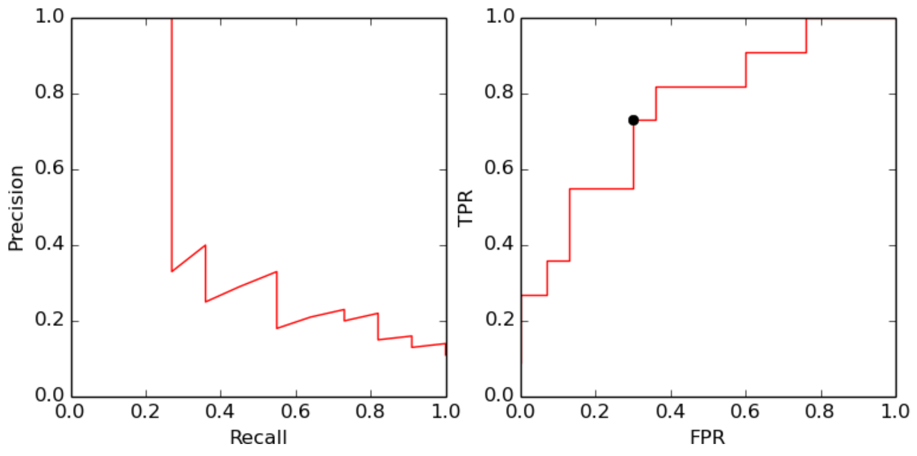
For this exercise session, we are going to learn about word spotting for handwritten documents. This exercise is splitted in two parts. In the first part, we are going to compare a keyword with a thousand of single words and find matchings. These matchings will be ranked with respect to the computed value returned by the system. In the second part, we are going to spot a keyword in a given text line.
For those two parts we are going to compute some values to determine the precision of our solution, the Receiver Operating Characteristic curve and Precision-Recall curve will be the most important of them. In addition we also compute the Equal-Error Rate and the Average Precision.

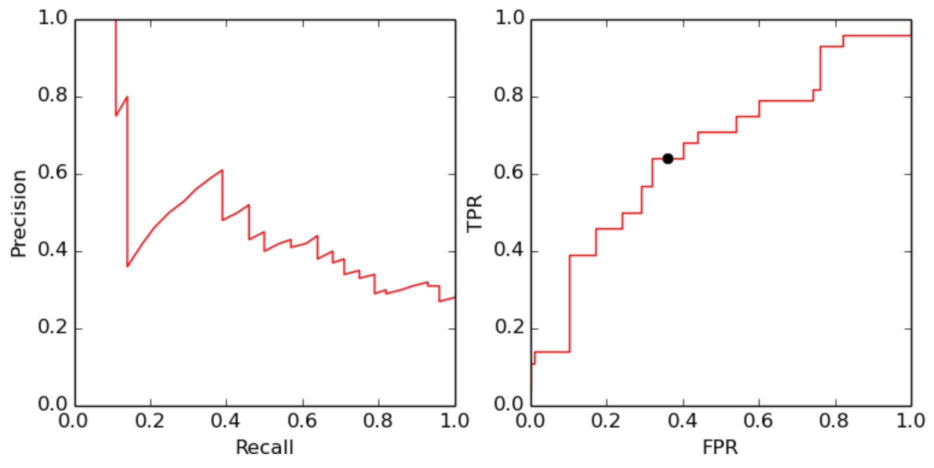
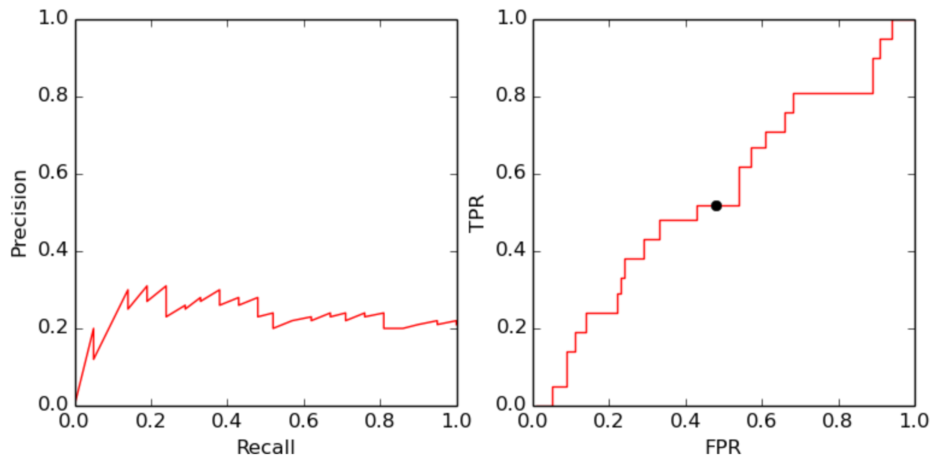
Method

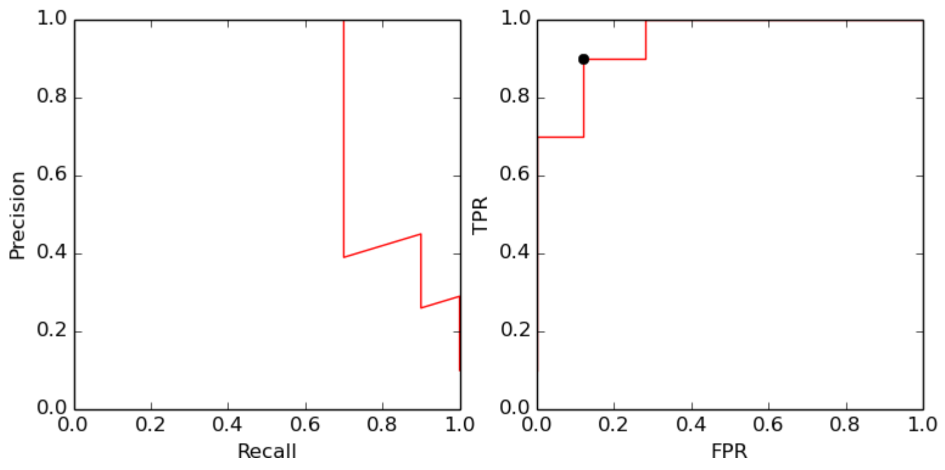
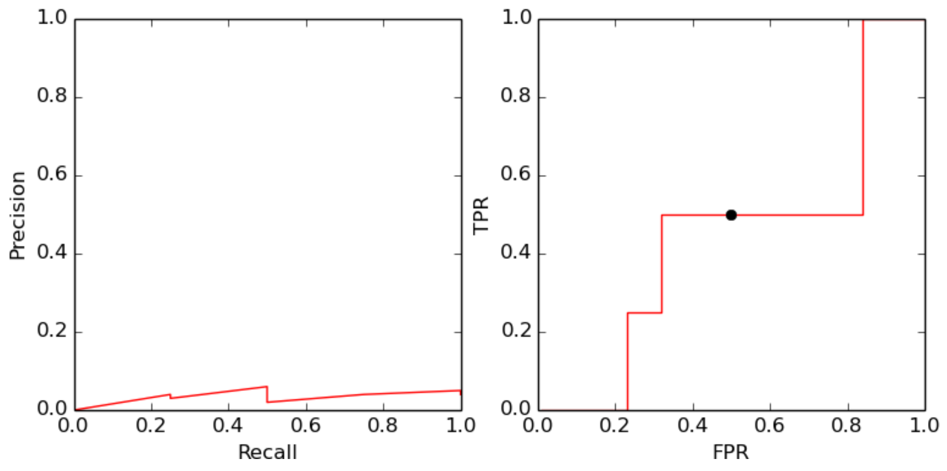
For this implementation we use the **method suggested by Rath et al. [1].** This method consists on a sliding windows of 1 pixel on which we compute four features. Those features are the **projection profile** (number of black pixels in one window divided by the height of the window), the **word profiles** (also known as the upper and lower profiles) which consist on taking the number of white pixels until the first appearance of the first black pixel (from top and from bottom) and the **number of black/white transitions**. For the word profiles, in some case there can be some holes in the word because of space between two characters being too big. To deal with this problem, we set the number of white pixel counted for the current window to half of the image height. Those features allow us to characterize a given matrix of pixels (word) and to determine if a given matrix of pixels is more or less similar to another (a sequence of text is similar to another sequence). In our implementation, we first compute for all the given words/lines their four features per column given by the method of Rath et al. [1] discussed earlier. When we try to perform keyword spotting on these words/lines we compute the four features for each keyword and compare them with those of the words/lines. This comparison is done via **Dynamic Time Warping with a particular cost function** (in the case with lines). We introduce free-cost for horizontal transitions. This free cost is used to allow to somehow parse the whole line with the keyword at zero cost. After applying dynamic time warping to all our features we end up with a vector that characterize the distance between a keyword and a given image to compare. We store all those distances into an array that we sort according to their euclidean norm. The best matches are the ones with the **smallest euclidean distances**. We then compute the results based on the **top 100 retrieved results** (we completely ignore what comes after the 100th rank, even if it's a matching).

Results

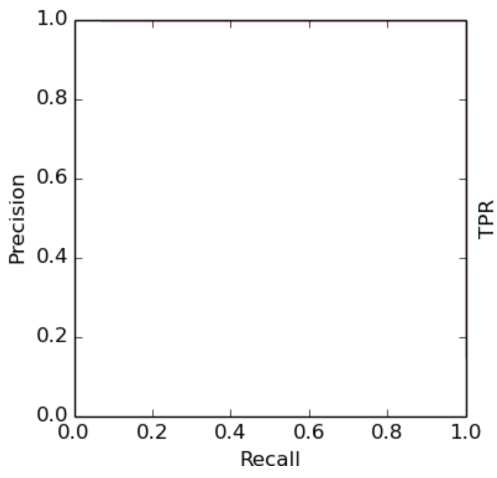
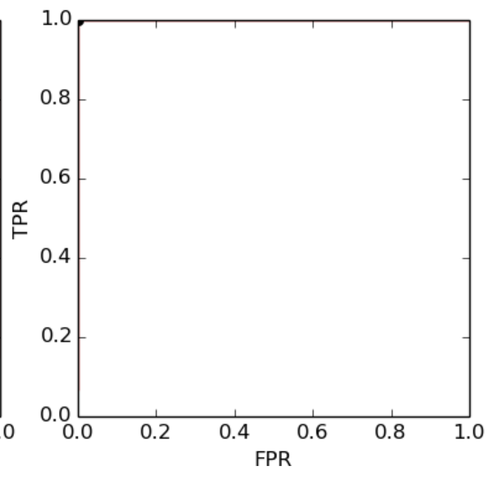
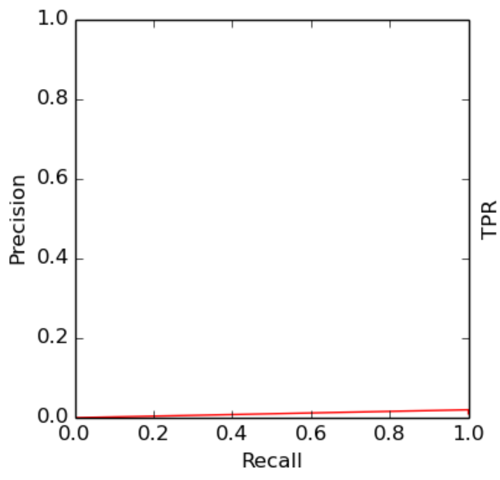
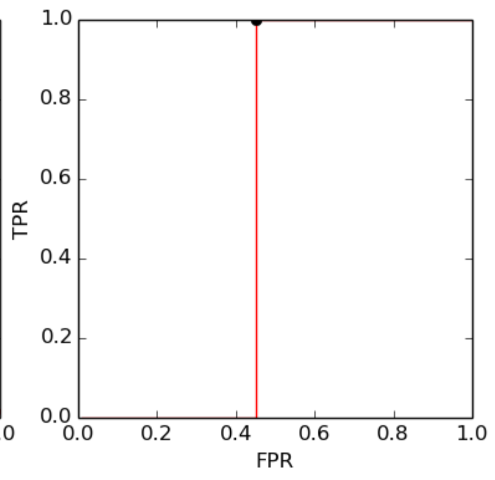
WashingtonDB

O-c-t-o-b-e-r	**Words**  EER = 0.25,0.77 Average precision = 0.611 **Lines**  EER = 0.3,0.73 Average precision = 0.453

t-h-a-t	**Words**  EER = 0.36,0.64 Average precision = 0.498 **Lines**  EER = 0.48,0.52 Average precision = 0.238181818182

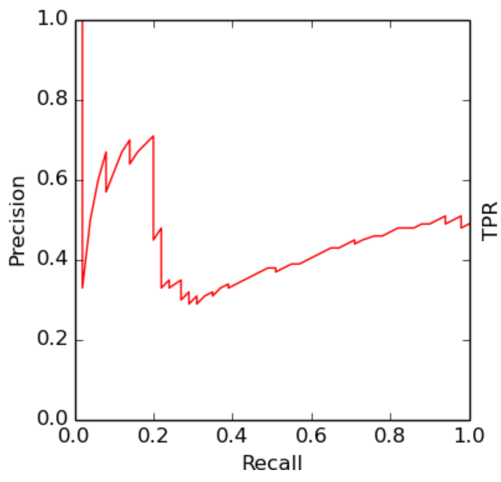
s-o-o-n	**Words**  EER = 0.12,0.9 Average precision = 0.816 **Lines**  EER = 0.5,0.5 Average precision = 0.038

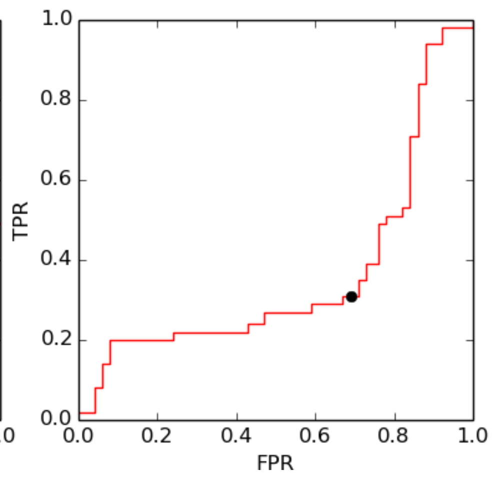
ParzivalDB

A-r-t-v-s	**Words**   EER = 0.0,1.0 Average precision = 1.0 **Lines**   EER = 0.45,1.0 Average precision = 0.01

d-a-z

Words

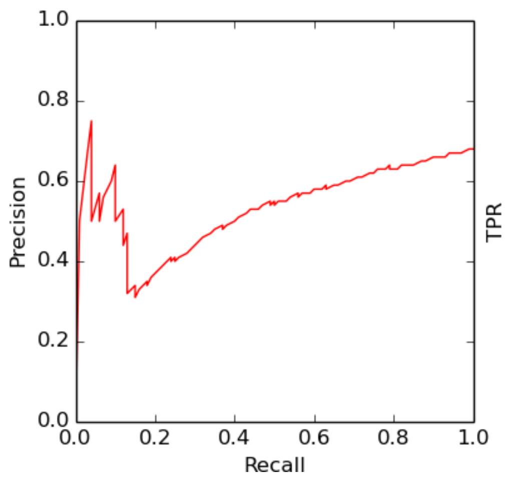


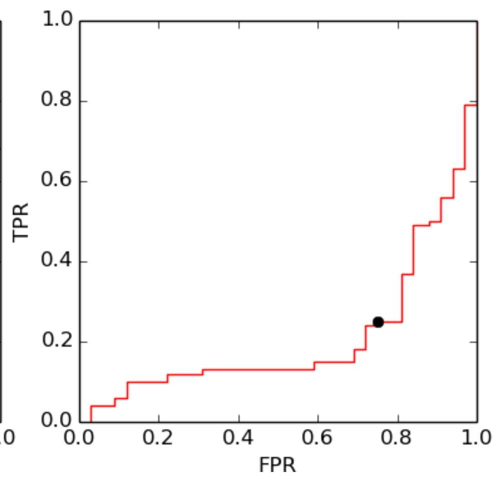


EER =   0.69,0.31

Average precision = 0.47

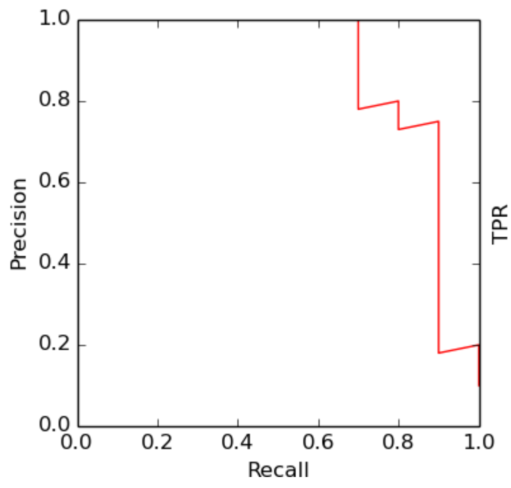
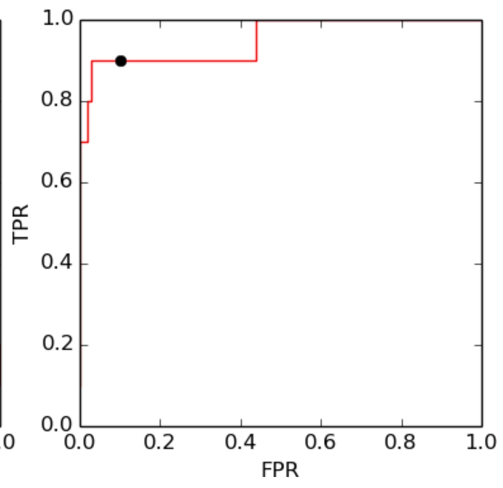
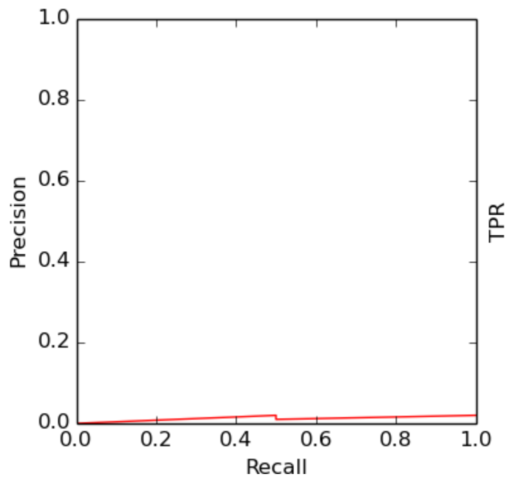
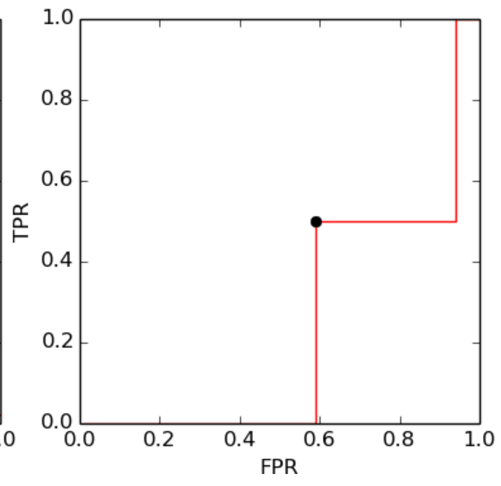
Lines

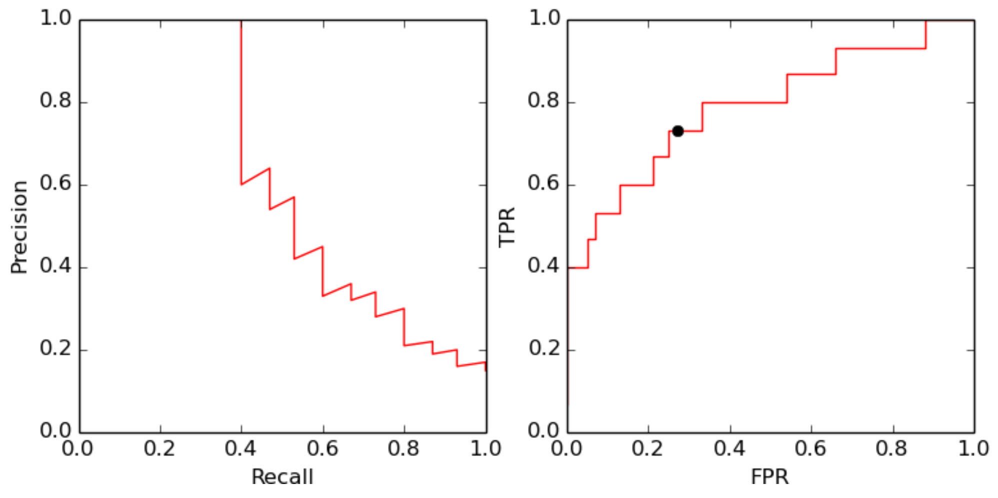
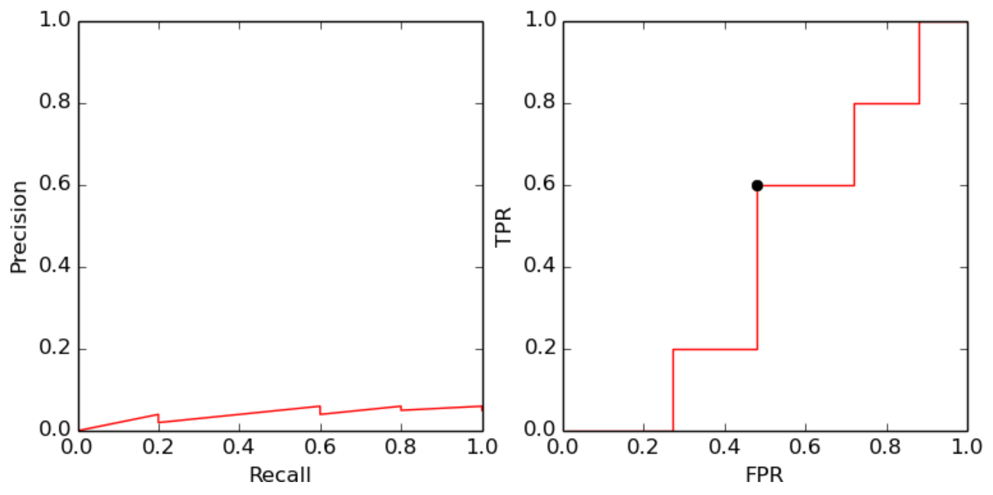




EER = 0.75,0.25

Average precision = 0.546

G-r-a-l-s	**Words**   EER = 0.1,0.9 Average precision = 0.875 **Lines**   EER = 0.59,0.5 Average precision = 0.013

k-v-n-e-g-i-n-n-e	**Words**  EER = 0.27,0.73 Average precision = 0.617 **Lines**  EER = 0.48,0.6 Average precision = 0.043

Conclusion

Our solution is quite good for word already segmented but we have problem with the whole lines. This come from the line splitting. As we are not splitting the lines but using DTW to emulate this behavior it is possible that the less expensive path from our keyword to a line is the line with the the minimal length.
Our features seem to work pretty well for the WashingtonDB data set but less good for the ParzivalDB data set. This might come from the fact that the writing style from the ParzivalDB is more squared and the characters are less distinguishable (often there are holes in characters) and thus our computed features are no more fine enough to take a good decision because the word profiles and the projection profile tend to be very similar.
But the fact that we have a perfect match for the word **“A-r-t-v-s”** is very strange and we still cannot explain this behavior.
To conclude, one can say that the method used works very well considering that they are only four features. It is clear that these seem not being decidable for all cases, but an extension or a modification may lead to better results.

References

[1] Rath, Toni M., and Raghavan Manmatha. "Word image matching using dynamic time warping." In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, pp. II-521. IEEE, 2003.