ITI8730 - Data Mining
Home Assignment 2

I want to attend the defense in class.

LEBARON Maëlie
Student code : 214322IV
maleba@ttu.ee

December 12, 2021

1 Text datamining

1.1 Dataset

I found the dataset on kaggle. It is a group of text about world war II. There is originally 8 texts.

1.2 Preprocessing the dataset

I performed a PCA on the dataset : the dimensionality was to high to be effectively represented, and I use the PCA to select some texts.

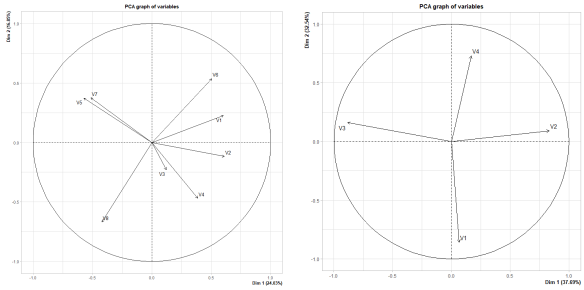

Figure 1: PCA before and after reducing the dimensionality

I then plotted the most common words of the dataset, to have an better insight.

1.3 Depict the centroids

Since some text were longer than other, and also the fact that the word *war* appears a lot more than other words, I used the term frequency and the inverse document frequency.

I then applied the k-means function from the first homework. The plotting is tricky since the dataset has a dimensionnality higher than 2.

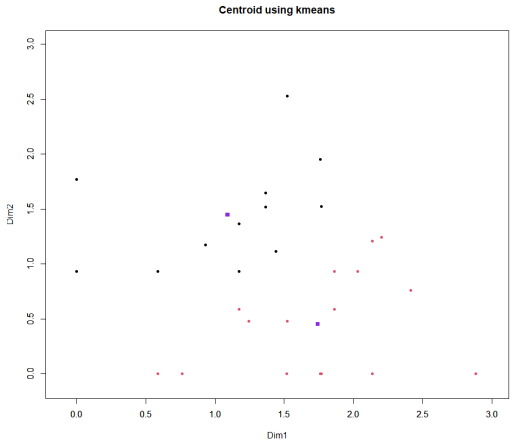

Figure 2: K-means applied to our dataset (centroids in red)

1.4 Decision boundary

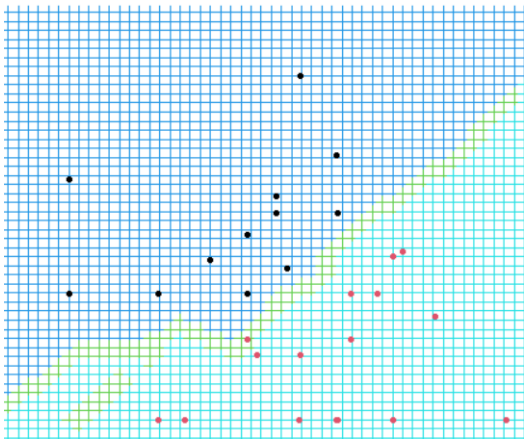

Figure 3: New PCA after selection the texts

To depict the decision boundary, I created a grid. Using the *knn* algorithm already implemented in R, I affected a label on each part of the grid. I created another label, with was the boundary label. I affected it to all the frame where the frame around had different value (using the variance of the labels around).

I then plotted the decision boundary grid with the labels.

2 Second exercise

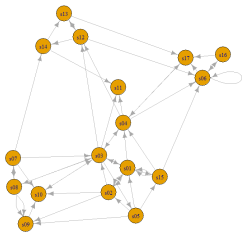

Figure 4: Representation of the graph used for this exercise

2.1 Usefull functions

To compute the functions, I created 2 function to help me :

- *get_node_neigh :* which return the neighbors of the node
- *node_distance :* which compute the distance from a node to another

2.2 Local clustering coefficient

The function return a matrix with the value of the local clustering coefficient for each node of the graph. For each node, I take all the neighbors, and then apply the formula using that list. When computing, we can see that *node 5* and *node 16* both have a coefficient of 1.0, because all their neighbors are interconnected.

2.3 Degree centrality

For this function I used the function to get the neighbors on each node. Using it, I had the number of neighbors and therefore can calculate the degree centrality. The node with the highest degree is the *node 3*, which makes sens when we look at the graph, because it is the node with the most connections.

2.4 Degree prestige

Same principle as for *degree centrality*, except that we use the paramter *"in"* when getting the neighbors. Again, *node 3* is the one with the highest degree. It is because it is the one the receive the more edge among the nodes.

2.5 Gregariousness of a node

The gregariousness is almost computed as degree prestige and degree centrality, expect using only the edge coming out of the node. The *node 11* is interesting because its greariousness is 0.0: it has not edge coming out of it.

2.6 Closeness centrality

Starting from a node, I computed the distance to each other nodes. When there was no path, I made the choice to just not count it (but I could also used a infinite value to represent the fact it is unreachable). According to the result, the *node 6* is really close from every other nodes.

2.7 Proximity prestige

The same principle as closeness, but this time with the *"in"* parameter. In this case, the *node 3* has the highest value. Which is logical because the node is very central to the graph and has a lot of edges that comes in.

2.8 Betweenness centrality

This describe the importance of the node in the graph, using the number of shortest path that goes though it. Therefore, we compute shortest paths and we count the number of time the node appears in it. The *node 3* is again at the center of our graph: it is the one which is in the most shortest path. If this node was not in the graph, the distance to goes from one node to another on average would augment a lot.

2.9 Common neighbor based mesure

To compute the common neighbors based mesure we compute the list of neighbors for both nodes, and we count the number of nodes that are in both list. For *node 1* and *node 2*, they have 2 common neighbors, which are *node 5* and *node 3*.

2.10 Jaccard mesure

To compute the Jaccard mesure we use the common neighbor based mesure, and we ponderate it with the sum of number of neighbors of both node.

For *node 1* and *node 2*, the Jaccard mesure is 0.2.