Report

Implementation of NBC algorithm

Michał Komorowski

2010-06-13

CONTENTS

INTRODUCTION

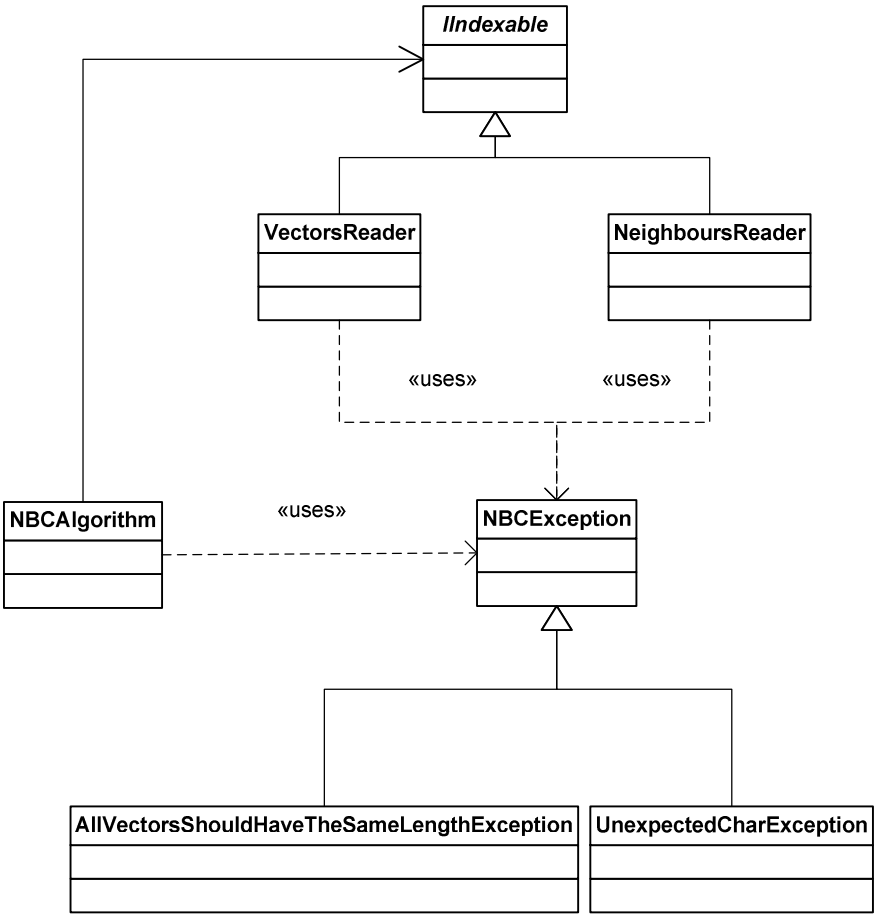
The goal of the project was to implement *Neighborhood-Based Clustering* algorithm that was proposed by Shuigeng Zhou, Yue Zhao, Jihong Guan and Joshua Huang and is used to cluster multidimensional vectors of real numbers. Algorithm was developed in C++ programming languages. The results of implementations are dynamic library *NBCLib.dll* and console application *NBC.exe* that use this library.

DESIGN AND IMPLEMENTATION

The main class in the project is a one that contains the implementation of the algorithm. It is called *NBCAlgorithm.* All other classes are responsible for other things like logging or measuring execution time. The following table contains short description of each classes.

Class Name	**Description**
NBCAlgorithm	The implementation of the algorithm
IIndexable	The base abstract class for data readers.
NeighboursReader	The class responsible for reading neighbours for vectors from a input file.
VectorsReader	The class responsible for reading vectors form a input file.
Logger	The class responsible for logging messages to a log file and to a console.
Timer	The class responsible for measuring execution time and writing results to the logger.
Utils	The helper class containing methods being used in the project.
NBCException	The base class for exceptions generated in the project.
UnexpectedCharException	The class of the exception that is thrown when unexpected char is found in the file
AllVectorsShouldHaveTheSameLengthException	The class of the exception that is thrown when not all vectors have the same length
WithComma	The helper class used to write decimal numbers to a output file with a comma instead of a point .

The following diagram shows relations between the most important classes. Classes *Timer, Logger, Utils* and *WithComma* were omitted because either there are not very important or there are used by many other classes and it will clutter the diagram. For example, the class *Timer* is used by classes: *NBCAlgorithm, VectorsReader*, and *NeighboursReadre*



The following lists shows how to use *NBCLib* in order to perform calculations:

1. Initialize logging mechanism by calling *Logger*::*init* method and passing as the argument a path to a log file.
2. Create instance of the *NBCAlgorithm* class.
3. Read input data with the *VectorReader* or provide your own class that is derived from *IIndexable< const vector<double>& >*.
4. Call the method *NBCAlgorithm.setData.*
5. [Optional] Read neighbours with the *NeighboursReader* or provide your own class that is derived from *IIndexable< const vector<unsigned int>& >.*
6. [Optional] Call the method *NBCAlgorithm::setNeighbour.*
7. Start calculations by calling *start* method of the *NBCAlgorithm* class and passing as a argument a value of the K parameter of the algorithm.
8. [Optional] Get the number of determined clusters by calling *NBCAlgorithm::getNumberOfCluster.*
9. [Optional] Write cluster to a output file by calling *NBCAlgorithm::writeClustersToFile.*
10. [Optional] Write neighbours to a output file by calling *NBCAlgorithm::writeNeighboursToFile.*
11. [Optional] Write complete algorithm data to a output file by calling *NBCAlgorithm::writeCompleteAlgorithmDataToFile.*

CONSOLE APPLICATION

The console application has following parameters:

Parameter	Mandatory	Type	Description
k	Yes	Integer	The K parameter of NBC algorithm. Do not use with parameters **from, to, step**.

from	Yes	Integer	The initial value of **K** parameter. Use with **to, step**.
to	Yes	Integer	The end value of **K**. Use with **from, step**.
step	Yes	Integer	The step of experiment. Use with **to, from**.
input	Yes	Path	The path to a file with input data.
output	Tak	Path	The path to a file with output data.
neighbours	No	Path	The path to a file with neighbours for vectors.
generateNeighboursFilesNames	No		If this parameter is use algorithm generates file names and try to read neighbours from them. For instance, if parameter **neighbours** is equal to *results* following file names can be generated: *results_10_neighbours, results_18_neighbours* etc.
writeNeighbours	No		If this parameter is use neighbours for vectors are written to the output file.
writeCompleteData	Nie		If this parameter is use algorithm data for vectors like NDF, neighbours are written to the output file.

INPUT DATA

The file with input data must contain one vector per each line. The coordinates of the vector must be separated by **^** char and as the decimal separator the coma must be used. For example:

```
664,159^550,946^665,845^557,965

597,173^575,538^618,6^551,446

635,69^608,046^588,1^557,588

…
```

The format of the file with neighbours is as follow. Line N in the file contains identifiers of neighbours for a vector in the line N in the input file. The identifier is a number of line in which vector appears in the input file (in numerations starting from 0). For example:

```
4005 57968 46629 32085 14913 70245 5834 29992 42630 19836

79019 90334 67275 37594 55053 834 9744 18537 80155 89324

49424 43549 78202 54658 47470 23963 95578 79718 78001 85631

…
```

OUTPUT DATA

The algorithm, depends on the parameters, can create four files with output data.

File name	**When will be created?**	**Description**

results	Always	Log
results_k	Always	The results of clustering. Line N contains identifier of the cluster for a vector in the line N in the input file. If a vector is a noise -1 value is used.
results_k_neighbours	**-writeNeighbours**	See Input Data
resulsts_k_completeAlgorithmData	**-writeCompleteData**	The complete data of the algorithm. Line N contains the complete data for a vector in the line N in the input file.

Example of a file with complete algorithm data:

CLUSTER_NUMBER NDF RKNB NUMBER_OF_NEIGHBOURS (NEIGHBOURS)

0 1.1 11 10 (id4005c0 id57968c0 id46629c0 id32085c0 id14913c0 id70245c0 id5834c0 id29992c0 id42630c0 id19836c0)

-1 0.1 1 10 (id79019c0 id90334c0 id67275c0 id37594c0 id55053c0 id834c0 id9744c-1 id18537c-1 id80155c-1 id89324c-1)

0 3.1 31 10 (id49424c0 id43549c0 id78202c0 id54658c0 id47470c0 id23963c0 id95578c0 id79718c0 id78001c0 id85631c0)

…

EXAMPLES

- NBC.exe -k 10 -input big -output big_res

  The algorithm will be run with the parameter K equal to 10, input data will be read from file *big*, the log will be written to the file *big_res* and the results of clustering will be written to file *big_res_10*.

- NBC.exe -k 10 -input big -output big_res -writeNeighbours -writeCompleteData

  The algorithm will create *big_res_10_neighbours* file with neighbours and *big_res_10_completeAlgorithm*Data file with data of the algorithm.

- NBC.exe -k 10 -input big -output big_res -neighbours big_res_10_neighbours

  The algorithm will read neighbours from the file *big_res_10_neighbours*.

- NBC.exe -from 10 -to 20 -step 2 -input big -output big_res

  The algorithm will be run with the parameter K equal to 10, 12, 14, 16, 18, input data will be read from the file *big*, the log will be written to the file *big_res* and the results of clustering will be written to the files: *big_res_10, big_res_12, big_res_14 etc*.

- NBC.exe -from 10 -to 20 -step 2 -input big -output big_res -writeNeighbours

The algorithm will create *big_res_10_neighbours, big_res_12_neighbours, big_res_14_neighbours* etc. files with neighbours.

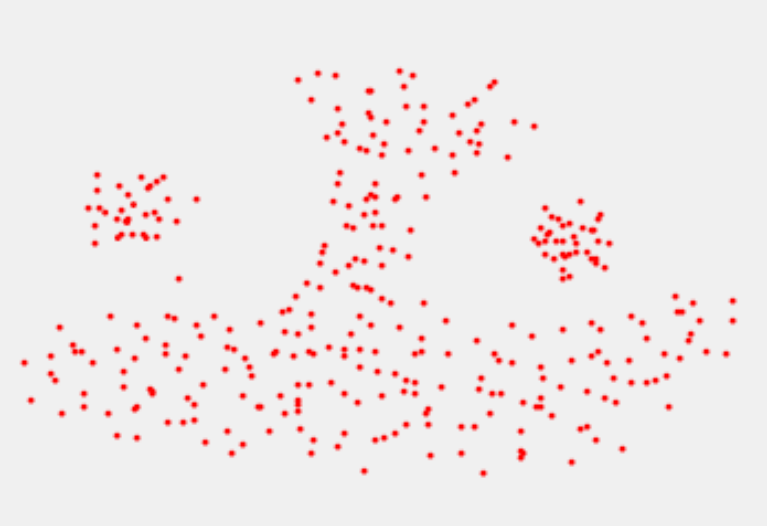
- NBC.exe -from 10 -to 20 -step 2 -input big -output big_res -neighbours big_res –generateNeighboursFilesNames

  The algorithm will read neighbours from the files: *big_res_10_neighbours, big_res_12_neighbours, big_res_14_neighbours* etc.

TEST DATA

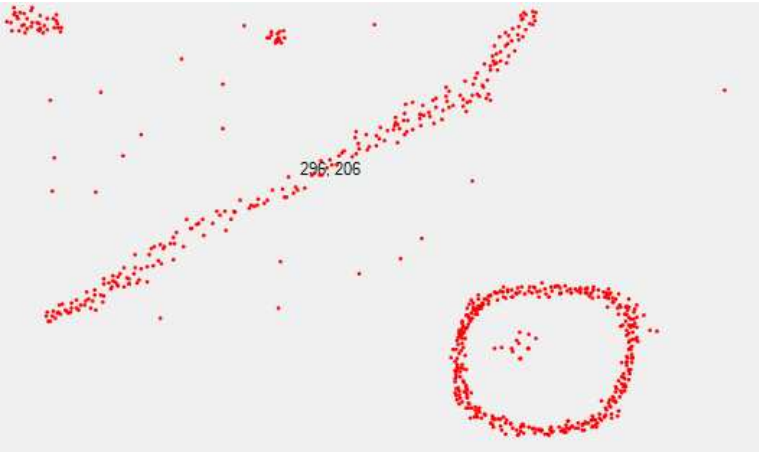
CUSTOM DATA 1

Custom two dimensional data with 341 vectors.



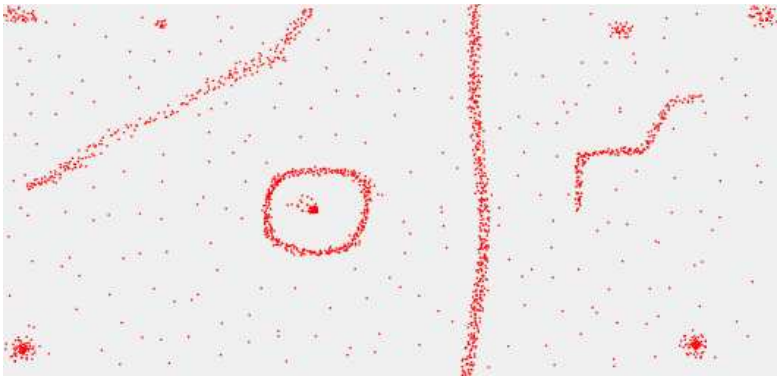
CUSTOM DATA 2

Custom two dimensional data with 702 vectors.



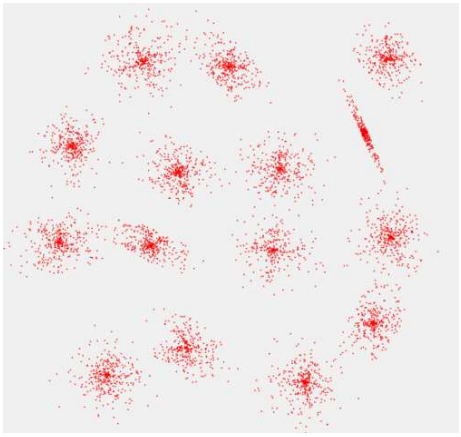
CUSTOM DATA 3

Custom two dimensional data with 1840 vectors.



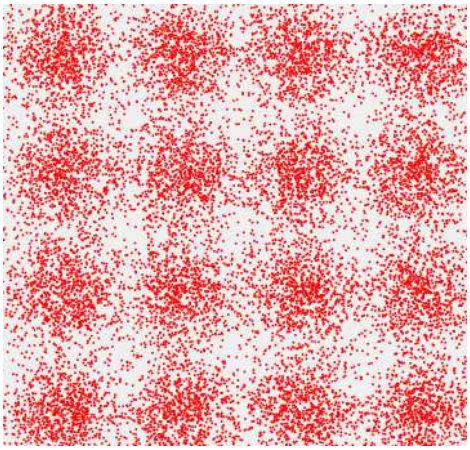
S1

Synthetic two dimensional data with 5000 vectors and 15 Gaussian clusters with different degree of cluster overlapping. P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems", *Pattern Recognition*, 39 (5), 761-765, May 2006



BIRCH1

Synthetic two dimensional data with 100 000 vectors and 100 clusters. Zhang et al., "BIRCH: A new data clustering algorithm and its applications", *Data Mining and Knowledge Discovery*, 1 (2), 141-182, 1997.



Based on this dataset the Birch1B dataset was prepared. This new dataset contains 10% of vectors from original dataset.

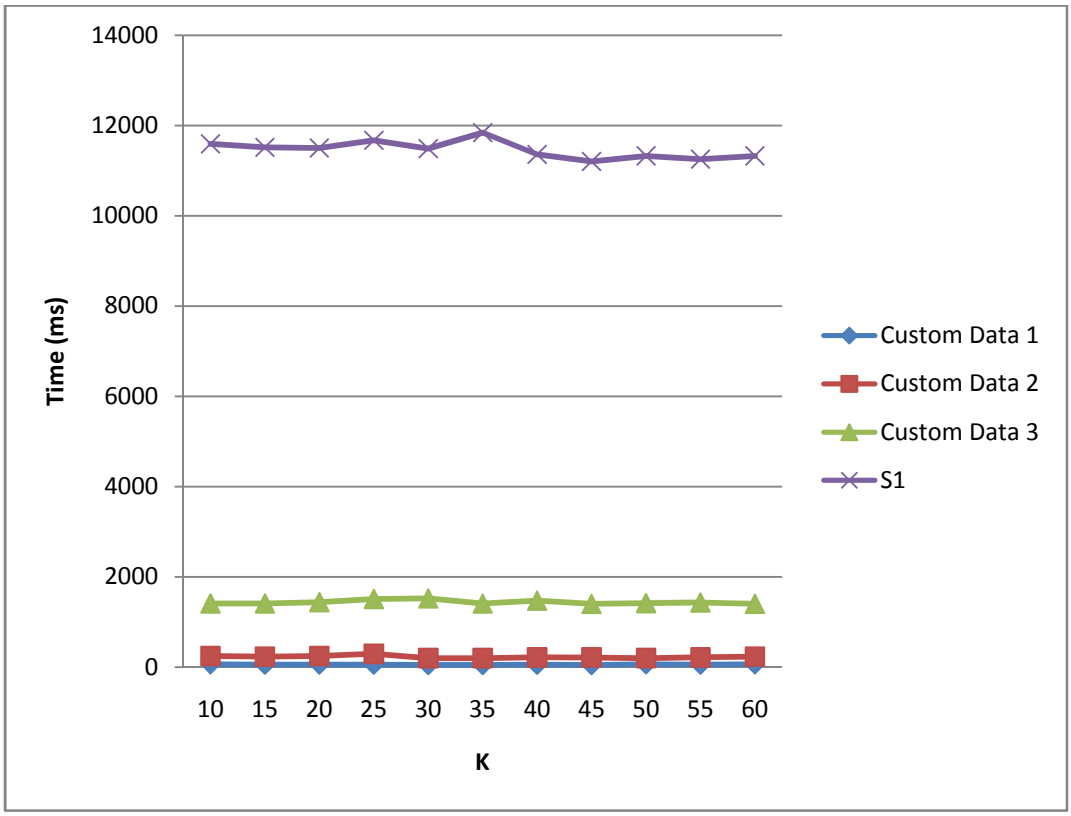
KDD CUP 98 DATA

Multidimensional data with 96367 vectors. Based on this dataset new datasets with 10% of vectors of original dataset and lower number of dimension were created.

EXPERIMENTS

Experiments were conducted on the machine with 4 GB of RAM memory and Intel Core Duo 2,53 GHz processor.

Firstly, it was tested how the value of parameter K affects the time of calculations. In this test *Custom Data 1,2,3* and *S1* were used. The results are shown below. As it can be notices the time of calculations does not depend on the value of parameter K and it only differs slightly for different values of K.

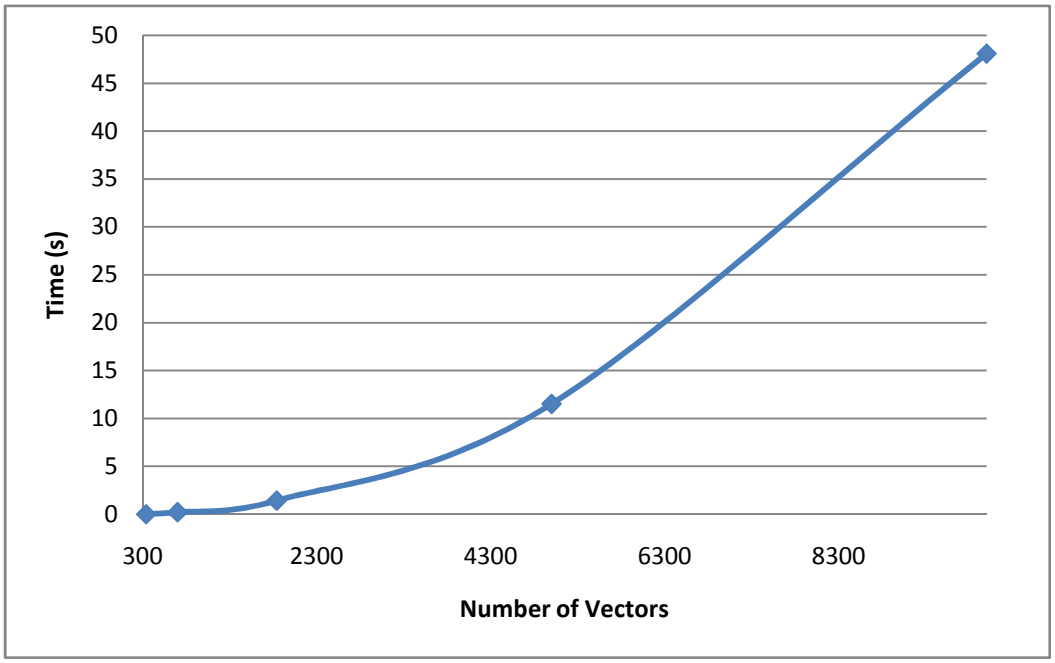


Secondly it was tested how time of calculations depends on number of vectors. In this test *Custom Data 1,2,3, S1, Birch1* and *Birch1B* were used. The parameter K was equal to 15. The results are shown below.

Based on the results it can said the algorithm has polynomial $O(n^2)$ complexity against number of vectors.

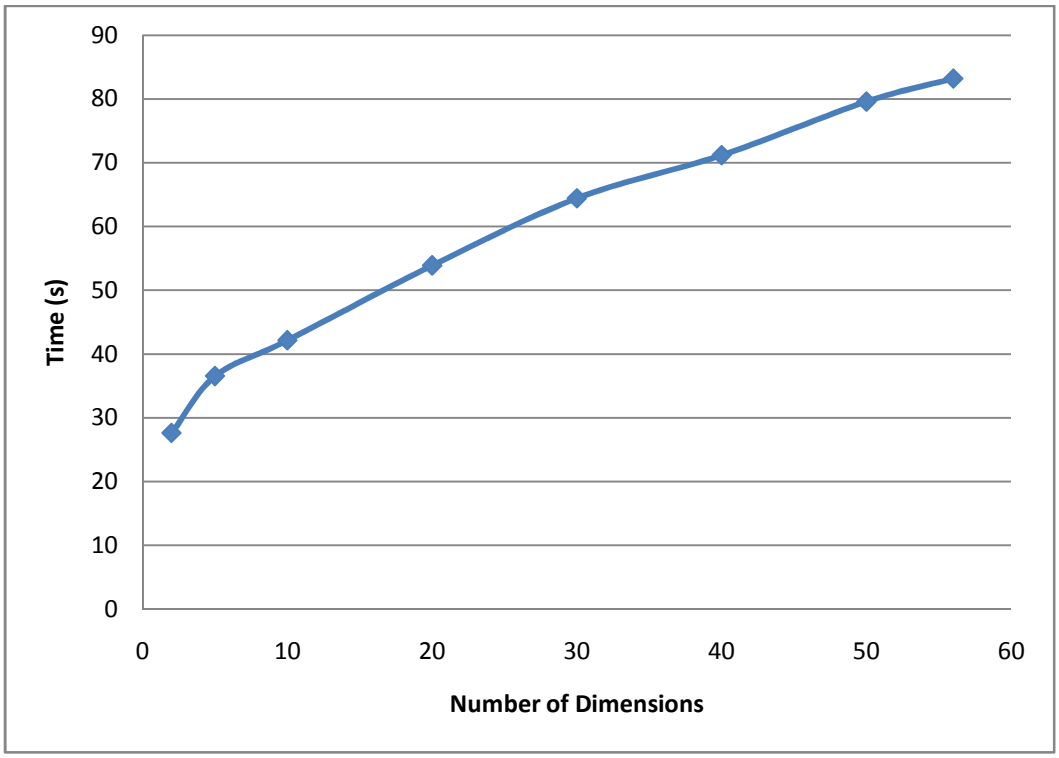
Number of Vectors	Time (s)
341	0,0057
702	0,212
1840	1,412
5000	11,513

10000	48,064
100000	5861



Then, it was evaluated how time of calculations depends on the number of dimensions. In this test new datasets created on the basic of KDD Cup 98 dataset were used. The parameter K was equal to 15. The results are shown below.

Based on the results it can said the algorithm has linear complexity against number of dimensions.



Then, it was evaluated which values of the parameter K give the best results. The results are shown below.

Dataset Name	Number of vectors	Range of values of K when results of clustering were accepted
Custom Dataset 1	341	9-25
Custom Dataset 2	702	16-40
Custom Dataset 3	1840	19-40
S1	5000	24-240
Birch1B	10000	9-50
Birch1	100000	N/A
		16-25

Finally, the application was run under control of performance profilers. According to results more than of 90% of execution time falls to sorting distances between vectors in order to calculate *k*-Neighborhood.

CONCLUSION

1. The efficiency of the algorithm does not depend on the parameter K.
2. The algorithm has polynomial $O(n^2)$ complexity against number of vectors.
3. The algorithm has linear complexity against number of dimensions.
4. The time of clustering in comparison to time of calculating NDF measure can be neglected. When number of vectors in input data is equal to 100 000 the time of clustering is equal to ~40ms!
5. There is no universal value of K parameter and it should be chosen separately for each dataset. However, in majority of cases the value chosen from range (16-25) should give proper results.
6. The more clusters differs and the more vectors is in the dataset the higher values of parameter K give proper results.
7. The number of cluster is inversely proportional to value of parameter K.
8. The more symmetrical the clusters are the better are results of algorithm. For instance, it is easier to cluster vectors arranges in the circles than in the lines.
9. The way of calculating *k*-Neighborhood should be optimized. For instance, instead of sorting collection (vector) of distances the tree or heap data structure can be used.

REFERENCES

Shuigeng Zhou, Yue Zhao, Jihong Guan, and Joshua Huang. NBC: A neighborhood based clustering algorithm. In *Proceedings of PAKDD'05*, LNAI vol. 3518, pp. 361-371, Springer, Hanoi, Vietnam, May 2005.