Effective Implementation of Basic Operations for Information Retrieval

Julian Szymański
Department of Computer Systems Architecture
Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology, Poland
julian.szymanski@eti.pg.gda.pl

JDIM
Journal of Digital Information Management

Abstract. *In the article we describe the approach to parallel implementation of elementary operations for textual data categorization. In the experiments we evaluate parallel computations of similarity matrices and k-means algorithm. The test datasets have been prepared as graphs created from Wikipedia articles related with links. W also present the approach to computing pairs of eigenvectors and eigenvalues for visualizations of the datasets. The implemented basic operations: computing similarity matrix, data clustering and spectral analysis have been used in our system for visualization of the Wikipedia categories on SOM as well as in a system for categorization search results in Wikipedia.*

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]; Clustering I.2.7 [Natural Language Processing] Text analysis

General Terms:
Text Processing, Datasets, Visualization

Keywords: Documents Categorization, PCA, SOM, Text clustering, Information Retrieval

Received: 18 July 2012, Revised 12 September 2012, Accepted 21 September 2012

1. Introduction

In the face of growing sizes of text repositories effective processing of large text collections became a very important challenge. One of the most effective ways of their processing is to analyze the texts as a graphs. The graphs can be represented as a set of n points in n-dimensional Euclidean space. The position of a point is described according to its relation to other points and is coded as a vector containing a list of the nearest neighbors. In the simplest case it is a binary matrix where 0 denotes the lack of a reference and 1 indicates a relation between two nodes. The representation space is a n – dimensional hypercube and all the points cover only $\frac{n}{2n}$ of accessible space and they can form theoretically $2^{(n\,2)}$ different combinations. In practice the graphs, especially the linguistic ones, have low density, so the number of relations is much smaller. Exploiting that fact it is possible to introduce some optimizations into algorithms and data structures. If the text documents are considered their representation vectors are constructed with a binary coding of particular words occurrences or the presence of inner relations, such as bibliographic or hyper references.

Processing textual data involves additional issues related to the scale of the problem. The analyzed matrices, that represent text collections, are huge and for real data they can not be loaded into memory, also their processing on single a processor takes too much time. In our experiments we use data taken form Wikipedia. This electronic encyclopedia has different databases for different languages so it allows to rise crossbar size of processed data easily. Actually the size of Wikipedia data bases for different languages may differ even by two orders of magnitude. In our experiments we use the data, that while represented as graphs, contain thousands and hundreds of thousands nodes. Our first dataset is constructed from Simple English Wikipedia, and it contains nearly 100 thousands of articles. The second dataset is built from PolishWikipedia data and it approaches 1 million of articles. Processing such large datasets to be finished in reasonable time requires effective parallel algorithms that can be run on multiple processors. Also it is important to take into consideration the size of available operating system memory as well a disk storage. The effective imple mentation of the algorithms and their successful evaluation on our test datasets will be the basis for performing computations on a larger scale – English Wikipedia that contains over 3.5 millions of articles.

Wikipedia	File size	Articles number (n)	References number (m)	Density [%]
Simple	8.9 MB	61822	2172829	0.0569
Polish	106 MB	712734	26403853	0.0052

Table 1. The characteristics of the datasets

2. The Data

From the point of view of Natural Language Processing (NLP) Wikipedia is a very interesting material. It covers very vast area of human knowledge that can be represented in the form of interconnected concepts described in the articles. The analysis of the content of the articles is a interesting task in itself, but in this paper we focus only on the structure formed by connections between the articles.

The backup of Wikipedia dumps is available online for all languages. We use them to create Wikipedia articles representation based on their references. In that approach each of the articles is represented as a binary vector where 1 denotes presence of hyperlink between two articles. This structure can be stored in the form of asymmetric matrix of neighborhood.

The datafile containing that matrix has a size $O(n^2)$, where n is a number of nodes (articles). The final size depends on how relation is coded eg. in case of small integer the value should be multiplied with 2 bytes. Storing the representation matrix in the form of neighbors list it is possible to generate the size $O(n + m)$, where m is the number of graph edges (references). The lists of neighbors form a representation space where each article corresponds to a n-dimensional point. The lists are sorted so their zero coordinates can be easily omitted during computations.

The characteristics of the two test datasets used in the experiments constructed from Simple English and Polish Wikipedias have been given in Table 1. The Table contains information about the size of input datafile, the number of the articles (graph nodes), the number of references (edges) and graph density – the average number of references between two articles expressed in %.

3. The computations

In this paragraph we describe three main computations we perform on Wikipedia data. We describe the basis of the algorithms as well as their optimizations and the methods of their parallelization.

The first task is to compute a similarity matrix. According to the used distance measure we obtain square similarity matrix that can be used further as a metrics representation space and also is used for computing clusters. The second task – clustering can be performed in many ways [1]. In our approach we use k-means – one of the simplest algorithms that allows us to estimate computational cost of identifying groups of similar elements. The clustering may be performed on raw data, using metric space (constructed from similarity matrix) as well as using data transformed in spectral mapping constructed with eigen analysis [2], [3] which is our third implemented operation for large matrices.

3.1 Matrices of distances and similarity

The basic approach to compute similarity of two points in geometrical space is to calculate Euclidean distance (then the similarity is its reverse). When we operate on high dimensional vectors the alternative to Euclidean distance is the cosine angle between vectors formed between the origin of the coordinates and the points.

To reduce the size of final similarity matrix (that is n^2) typically given a priori threshold p is used that allows to filter small values of similarities and treat them as 0. Then final matrix is stored as a list of neighbors. This approach allows to considerably reduce the final matrix, that is used in other computations, where small values are not very significant. Note eg. for a graph with 1 million nodes creating full similarity matrix (which elements coded as integers) the final file will be near 1 TB. For our experiments we empirically set up the threshold that, while similarity expressed in % is < 0.05, allows to reduce outcoming files by $67\% \pm 15\%$.

In our experiments we use two distance measures: Cosine and Euclidean. These two metrics complete each other, and they find different applications. Cosine distance is known to be better for computing similarities between sparse vectors that are typical while representing graph nodes. It is also suitable for visualization tasks. Euclidean measure givesmore precise information about distances between points especially when their mutual cosine similarity is zero. This information is necessary eg. in k-means algorithm (section 3.2) when only small fraction of all similarities are non zero.

As we operate on high-dimensional data their processing causes also additional problems called *curse of dimensionality* [4]. The main issue here is the fact that, while increasing dimensions, the distribution of the distance averages concentrates around their expectations [5]. This requires to apply special approaches: dimensionality reduction and dimension (eg. with discussed further PCA) based on weighting distances.

The next problem with processing natural language data is that the vectors of the representation are sparse. Due

to that we use Cosine similarity in preprocess phase. Transforming data into metric space is known to be a good technique for high dimensional space vectors [6].

While calculating distances the most computation intensive operation is matrix multiplication with complexity $O(n^3)$. It has been optimized and implemented in parallel environment using MPI. As we operate on vectors that represent graph nodes in the form of neighbors lists average complexity of each operation is $O(\frac{m}{n})$ which finally reduces the algorithm of matrix multiplication to $O(n \,.\, m)$. Additionally we exploit the fact that the vectors are binary which allows to replace the floating points operations with logical ones. In that way Euclidean distance between vectors can be computed as square root of their Hamming distance and vectors dot product is the number of their non-zero common elements. Due to the fact the final matrices are symmetric and sparse we compute and store only one part of them, without main diagonal, in the compressed form of sorted lists (containing only non zero elements).

In our implementation of computing similarities we use data distribution that can be run on any number of processors The distribution algorithm divides the data between processors in such a way that the size of the data packages (constructed from matrix triangle) is the same on all processors. Note we assume here the uniform distribution of data density which may not be true and thus cause some problems, which is discussed further.

The distribution of the source matrix D we base on selecting areas from triangle matrix (built form matrix D formed with its diagonal) which has been shown in Figure 1A.

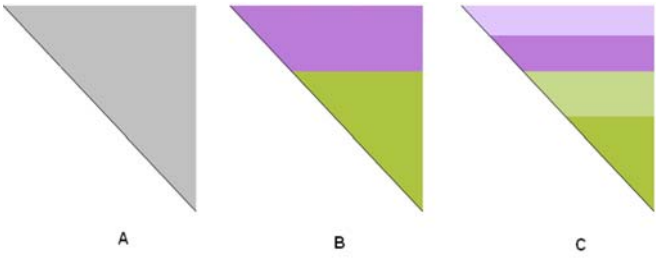

Figure 1. Creating data packages from data matrix for growing processors number

If we want to distribute the data between two processors we divide matrix D (given in Figure 1.A) into trapezoid and triangle in such a way that they have the same fields, which has been shown in Figure 1.B. To distribute the data between four processors the trapezoid and the triangle are divided again in two figures with equal fields what has been shown in Figure 1.C.

This approach does not require master process because using the information about the first index of the free range and the number of processors used for computations, each of processors is able to determine the size of the processed data by itself.

The method presented in Figure 1 can be generalized for o processors and data package is delimitated with one of o areas of matrix D that have uniform fields. Using equation 1 we calculate the number of rows y of $n \,.\, n$ similarity matrix that should be multiplied by the number of remaining rows to make given portion of data x.

$$y = \lfloor (1 - \sqrt{1 - x})\,(n - 1) \rfloor \qquad (1)$$

If the resources are limited Equation 1 allows us to compute sequentially only part of the matrix. The processing results can be joined easily into one final matrix by aggregating mediate files in proper sequence. It should be also noticed the approach allows to perform computations on computer clusters, as well as sequentially. Due to that the size of processed data that can be computed and the number of processors that can take part in computations are unlimited.

3.2 Parallel clustering with K-means

Clustering is a data mining method of unsupervised learning. It allows to find groups of the most similar objects in the dataset. The similarity is determined by a measure that describes distance between points.

One of the typical clustering algorithms is K-means [7]. The algorithm divides the data points into k subsets (clusters) where each of them joins the closest points (in sense of used metric). Typically the inverse of Euclidean distance is used as a similarity metric but any other measures can be employed here, which allows to capture particular aspects of data similarities [8]. The k-means algorithm has some drawbacks: it requires predefined k parameter – the number of clusters. It produces clusters that have convex shapes, and is randomly initialized, what may lead to obtaining different results each time. This issues can be eliminated with the usage of more sofisticated algorithms [9] but K-means is a base-line within clustering algorithms domain and its implementation allows to estimate computational complexity of the data analysis problem.

K-means for input data D (in the form of matrix $m \times n$) and for a given number of k groups (clusters) returns vector R (of the size m) whose elements describe assignments between each of m objects and one of k clusters. In our practical approaches we transform input data D into metric space (typically using Cosine distance) which allows to operate on $m \times m$ matrices. It reduces the size of input data because in our cases $m << n$, and allows to use two different distance measures (one while building metric space and the second in clustering algorithm) which positively influence final results.

Typical k-means algorithm is run in following steps:

1. Random initialization of clusters centers w.
$w\,[k, n] = rand\,()$

2. Calculate distances between each of k clusters and data points.
$r\,[m, k] = dist\,(w\,[k, n], D\,[m, n])$

3. Partition the data in such a way to minimize distances between points and clusters.
Each of the data point is assigned to the one group, represented by nearest cluster center.
$p[m, 1] = min\,(r\,(m, :))$

4. Compute centroids of each of the groups. The centroids are calculated as averages of most extreme centroids coordinates values.
$c\,[k, n] = (max\,[\,p, k] - min\,[p, k]) / 2$

5. Move each cluster center in direction of computed centroids.
$w = (w - c) / 2$

6. Repeat steps $2 - 5$ till clusters do not moves.

Because K-means is computations intensive, especially step 2 (where data are partitioned in such a way as to minimize distances between points and clusters) and step 3 (where centroids are computed) it has been written in parallel version in C++ using MPI. To warrant repeatability of the results we add optional parameter – random numbers generator seed that determines selection of the initial clusters. Due to random initialization the algorithm may not find optimal solution. The improvement is to perform several *test runs* and to obtain final results, starting from points that are averaged positions of the clusters achieved during *test runs*. It raises the cost of the computations proportionally to the number of *test runs*. The same improvement can be applied to find optimal number of k. Performing *test runs* with different k parameter allows to estimate its proper value (according to chosen criterion). Typically the evaluation of the clusters quality is performed according to internal criteria that analyze their internal consistency [10].

Due to ununiform distribution of data points in representation space initial clusters are not randomly selected from n-dimensional unitary hypercube, because it creates empty clusters. Instead of it we randomly select data points which improves the results considerably. Because the density of large textual graphs (eg. Wikipedia references) is usually very small (Table 1) most of their coordinates have zero values. Computing the distances between all data points and centroids, straightly from the original formula, requires to repeat many times the same operation of computing sum squares of centroids coordinates. In view of the fact it is a crucial part of the algorithm with complexity $O(n)$ and it is unacceptable, because it leads to complexity of one iteration $O(k\,.\,n^2)$. We introduce an improvement that allows to calculate distances in $O(\frac{m}{n})$ thus complexity of one iteration decreases to $O(k\,.\,m)$.

The improvement is based on compute and store squares of all distances of all k centroids at the beginning of each algorithm iteration. Then in phase where points are assigned to clusters the distances between points and centroids are computed using Formula 2.

$$|C - P| = \sqrt{C^2 - P^2 - 2\,(C \circ P)}$$

where C is centroid, P is a data point and $\circ$ is dot product. Square of a centroid distance C^2 is computed at the very beginning of iteration, the number of neighboring points of given P^2 point is given and $C \circ P$ has average complexity $O(\frac{m}{n})$.

Parallel K-means has been made with dividing the data points between processors equally and performing computations locally for the same clusters in each process. This approach requires to exchange local information between processors to obtain global values. Due to thatmain algorithmloop is executed synchronically. Howether it does not lead to idle times within main loop because computations are equally divided between all processors.

All the processors in the algorithm are equal except the initial and the final phase where one of them sets up seed for the random numbers generator for all others with the MPI Bcast function. After the algorithm finishes, the one processor using MPI Reduce collects local assignments of the points to the clusters and stores whole clusters in separate files on a disc. At the end of assignment phase the MPI Allreduce function is executed to test the global stop condition. To compute new centroids as a means of each clusters in estimation phase MPI Allreduce is used. It computes global sum of the points in each cluster and the number of points in these cluster.

3.3 Eigenvectors and Eigenvalues

One of the areas of applied algebra is data analysis with linear transformations. Most of these approaches is based on calculation of eigenvectors and eigenvalues [11] of data matrix D. These calculations are related to solving characteristic equation of input data $det\,(D - \lambda\,I) = 0$, where I is identity matrix and solutions to this equation are the eigenvalues λ_i.

Eigen analysis finds applications in spectral clustering where data partitioning is related to the second highest eigenvalue [12] or creates eigenspace where typical clustering is performed [2]. Using eigenvectors, high dimensional input data can be reduced and presented in less dimensional space. If reduced space is 2,3 - dimensional it is possible to make visualizations of the data. One of the most well known methods to do that is Principal Component Analysis PCA [13] where projection of the initial data is performed into reduced space by a given number of the highest eigenvectors.

Solving characteristic equation is a tough mathematical problem. Also their implementations are hard to be implemented as parallel. Due to the characteristic of our data (that are high dimensional and sparse) it is possible to apply iterative Lanczos algorithm [14] implemented in library *arpack++* [15]. The algorithm calculates sequentially eigen pairs starting from the one with highest corresponding absolute eigenvalue.

In this case performing eigen analysis is made

sequentially. Using computed earlier clusters we are able to partition the data whose computations of eigens are computed on single processor for a separate group of the data. Due to that we are able to perform locally parallel data analysis and we are able to make visualization within subspaces formed by clusters. Sequential computations of eigenvectors do not allow to make global analysis of the data thus it is impossible for now to create eigenspace where spectral clustering is performed.

4. Results of Scalability

The main goal of the experiments we made here was an evaluation of scalability of the computations in a function of the data size and used processors. The size of the data used in the experiment has been shown in Table 1.

The scalability results of computing distance matrix in function of available processors have been shown in Table 2. The achieved results for SimpleWiki suggest the existence of some sequential factor. The implementation excludes such case, and the reason is in sequential access to the disc while storing results of computations. Also it is influenced by a non-uniform data distribution in source data – the number of referencesbetween articles is not uniform and it varies in each data package. The lack of the results for 2-8 processors for Polish Wikipedia is caused by the size of the input data, which did not allow to compute them in reasonable time while fewer than 16 processors were used.

In Tables 3 and 4 we show the time results achieved by our k-means clustering algorithm in function of available processors and for different k parameters for Simple English Wikipedia and Polish Wikipedia respectively. What can be seen the clustering process is almost linear as well for k parameter as for the growing number of used processors. The small deviations from linearity are caused by the costs of communications between processors and final aggregation of ununiform distributed data within single process.

While evaluating the results of eigen analysis we can consider only scalability according to the problem size, because as a basic computation block we use its sequential implementation. As we can see in the results presented in Table 5 there exists some sequential factor that does not depend on the number of outcoming pairs vector-value. It is caused by conversion of the input data to *Compressed Sparse Column* SCC format that is required by *arpack++* library. The results we present for two datasets – the first is a cluster of $n = 981$ elements formed within Polish Wikipedia. The second is full simple English Wikipedia.

5. Eigen space visualizations and clustering quality

One of the applications of spectral analysis is visualization of the data. Presenting the input data in a reduced subspace allows to obtain rough view of the dataset.

Wikipedia	Time according to processors number					Size of the data file
	2	4	8	16	32	
Simple	15 m 26 s	9 m 32 s	6 m 27 s	5 m 19 s	3 m 25 s	1.4 GB
Polish	-	-	-	225 m 50 s	154 m 59 s	265 GB

Table 2. Scalability of computing distance matrix in function of available processors

Number of clusters (k)	Time according to the number of processors						Number of iterations
	1	2	4	8	16	32	
10	3 m 31 s	1 m 47 s	59 s	29 s	17 s	11 s	14
20	7 m 36 s	3 m 47 s	2 m	1 m	33 s	20 s	16
40	49 m 51 s	25 m 11 s	13 m 25 s	6 m 20 s	3 m 13 s	1 m 51 s	54
80	49 m 45 s	24 m 49 s	13 m 11 s	6 m 11 s	3 m 11 s	1 m 52 s	27
160	96 m 58 s	50 m 5 s	26 m 13 s	13 m 4 s	6 m 43 s	3 m 38 s	26

Table 3. Scalability of clustering Simple English Wikipedia.

Number of clusters (k)	Time according to the number of processors				Number of iterations
	4	8	16	32	
10	45 m 3 s	23 m 18 s	12 m 21 s	6 m 43 s	35
20	65 m 11 s	34 m 39 s	17 m 34 s	9 m 44 s	28
40	164 m 3 s	90 m 36 s	46 m 31 s	24 m 57 s	39
80	498 m 16 s	258 m 8 s	139 m 12 s	72 m 21 s	59
160	1847 m 7 s	1023 m 36 s	564 m 36 s	311 m 1 s	107

Table 4. Scalability of clustering Polish Wikipedia

n	Type of input matrix format	Size of input data	Time [s] according to number of pairs x					
			1	2	4	8	16	32
981	list of neighborhoods	132 kB	0.055	0.075	0.089	0.118	0.151	0.352
	full similarity matrix	3.7 MB	0.35	0.41	0.50	0.68	0.85	1.69
61822	list of neighborhoods	8.9 MB	5.9	6.5	7.3	9.7	10.9	20.4
(Simple Wikipedia)	full similarity matrix	1.4 GB	183	188	219	315	333	524

Table 5. Scalability of computing x most significant eigenvalues and eigenvectors pairs

Clustering quality, especially textual data may be measured according to the external criteria [16]. One of the popular measures is *Purity* that is calculated by counting the number of correctly assigned data points and dividing it by N (total their number), when each cluster c is assigned to the class ω that is most frequent in the cluster. It can be described with the Formula 3.

$$\text{Purity}(\Omega, C) = \frac{1}{n}\sum_{k}\max_{j}|\omega_k \cap c_j| \qquad (3)$$

To be able use this measure we need to possess referential set Ω. In our application we useWikipedia category system and for each cluster we select the category that binds together most of its elements.

The results of presented here experiments have been made on Polish Wikipedia data set. At the beginning we compute clusters with k-means clustering and $k = 10$. The results have been shown in Table 6. Respective times to obtain these results can be found in Table 4. What can be seen in the results, there have been formed nine relatively small clusters with high *Purity* and one very big cluster where all other articles, that couldn't be separated from others have been put. Next we may run again clustering with highest k parameter (which would lead to highest partitioning). Alternatively we may process the data within formed clusters separately what will produce hierarchical structure.

To make visualizations from among clusters presented in Table 6 we choose the cluster with number eight. It contains articles that fall into general subject *biology* that is well defined generic Wikipedia category. The cluster has small density so additional groups of data should be easily extracted.

In Figure 2 we show visualization of the *Biology* cluster represented with cosine similarity and projected into 2 dimensional space using two highest principal components. What can be seen, the data points tends to lay along straight lines. We evaluate the semantic interpretation of the points along lines manually and it shown some semantic similarity of the objects laying along the same lines. The object tends to group elements such as birds, reptiles, amphibians but *Purity* of such partitioning is low. There is much noise and it would be hard to make such semantic interpretation automatically. It is caused by the fact that the data representation based on links is a relatively weak approach for capturing text semantic. The second reason is the visualization which is performed in 2D. Note it is very big limitation that represents a complex category like a Biology using only two axes, even if they correspond to linear combinations of most distinctive features (what indeed performs PCA).

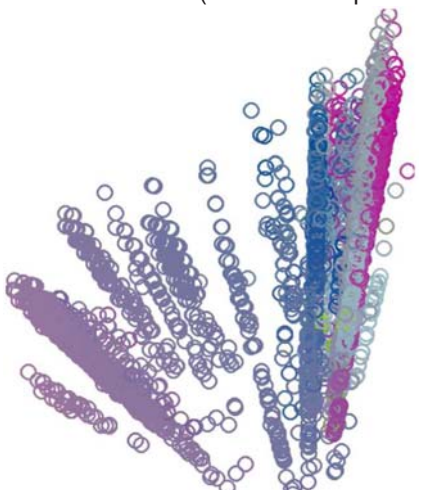

Figure 2. Visualization of similarity matrix for 8 Polish Wikipedia cluster

Cluster	Subject matter	Purity	Articles number(n)	Number of references (m)	Cluster Density [%]
1	Variety	0.1	713373	19139451	0.0038
2	Poland	0.82	125	12945	82.8
3	Accounts	1	6	19	52.7
4	Planetoids	0.69	2278	82083	1.58
5	Ireland	0.91	43	1677	90.6
6	Computers	0.88	53	2213	78.7
7	People	0.61	53489	2256391	0.078
8	Biology	0.79	22555	376728	0.074
9	Alliances	0.91	116	13144	97.6
10	Railways	0.93	425	175978	97.4

Table 6. Clustering results

6. Applications

6.1 SOM Visualization

Hierarchical systems of categories allow to effectively navigate over large datasets. The hierarchy allows to present data on different abstraction levels thus it is the one of most functional ways for organizing them. The main problem with the hierarchies is that if we process the complex data they tend to form more complicated structures than tree-like. The other issue is there can be more than one suitable structure for organizing the data especially if we want to see different aspects of the data.

The categories are initially considered as tree-like structures that form hierarchies that organize the data using their generalizations represented as abstract concepts. As some concepts can be related to more than one parent the relations between categories form a graph. Thus its presentation with folder-like approach as it is used in eg. file systems misses many aspects of the data. Additional issues related to the cycles should be also considered here. To solve these problems for presentation of Wikipedia categories we use Self Organizing Maps [17] that allow to present topological similarities of the processed data. The SOM approach and its extension in the form of hierarchical and growing self organizing maps [18] has been used for documents organization [19], eg. in the WEBSOM [20].

Self organizing map (SOM) allows to present topological relations between categories that come from the same conceptual level. To efficiently calculate the SOM as a starting values of the weights we use eigen vectors calculated for space visualization (section 5) The SOM presentation takes into consideration distances between categories. The distances are used to represent the similarities between categories. In Figure 3 we present a top Self Organizing Map for categories from the highest Simple Wikipedia hierarchy level. As different categories may activate the same neurons in Figure 3 instead of inserting around each of the neuron labels we mark them with colors specified to the categories.

The place where the category label is displayed has been delimited with the center of the shape formed from the neurons activation for a particular category. The centroid has been calculated according the formula 4.

$$m_k = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_j \qquad (4)$$

where m_k denotes position of the centroid for k category and Si is the values of neuron activations related to that category.

The resulting map describes here common-feature similarities between categories that can be easily changed introducing different ways for calculating the distance. Eg. we can strengthen particular features and in this way obtain different visualizations that show other aspects of the similarity of the categories [21]. Beside visualizations the SOM method allows also to narrow the number of processed objects. The main idea is to narrow the search results (presented on SOM) by introducing additional features obtained by the interaction with the user [22], what is the our plan for the future visualizations development.

6.2 Clustering search results

The parallel version of k-means algorithm evaluated in section 3.2 we used in our prototype system namedWikiClusterSearch. It automatically organizes the results of searching Wikipedia for a given keyword. In the system, user may specify a searched phrase and the articles containing it are organized into clusters in the fly. It allows to present directions in which user may continue his or her search. Our initial implementation has been created for Polish Wikipedia. WikiClusterSearch (WCS)

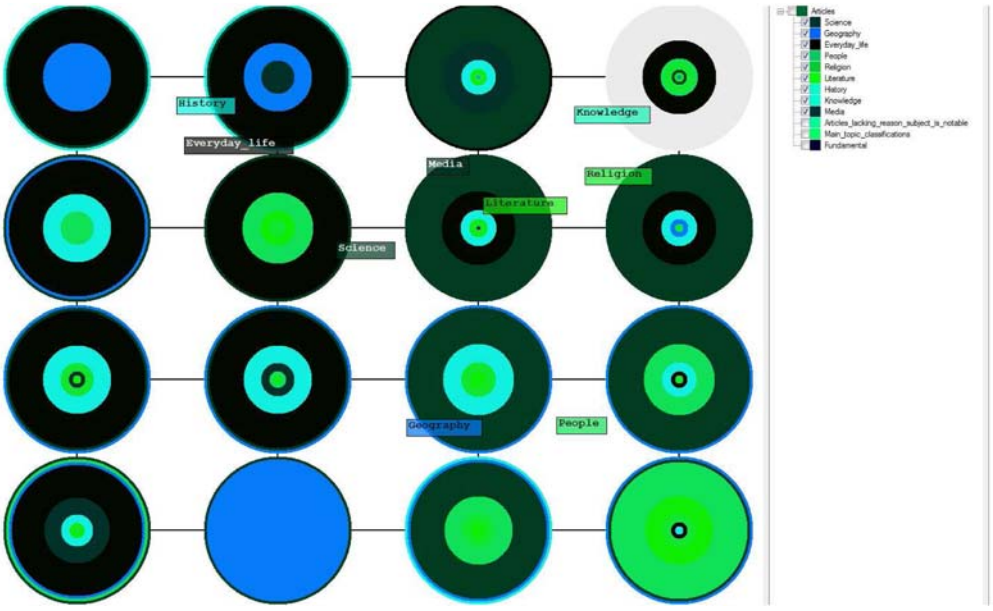

Figure 3. SOM for top level categories

has demonstrated the proposed approach can be used to obtain a good quality hierarchy of clusters. The system is available on line under http://kask.eti.pg.gda.pl Universal Search.

WikiClusterSearch is written in C# in .NET 3.5 environment. Client side uses the advantage of ASP.MVC 2.0 technology combined with jQuery JavaScript library. The system is built in modular architecture, each one is responsible for performing a different task. Main modules are presented in Figure 4. Modular architecture allows to encapsulate functionalities and test different approaches to system key elements:

– Data acquisition – the module is responsible for providing the data. The data may be used from off line version of Wikipedia when local databases are used. This approach ensure high efficiency of data processing as they can be accessible immediately. The drawback of this approach is the data may be in actual. The other approach is to use on-line data that are for each user query obtained from the Internet. This may cause efficiency problems while queries that index large amount of articles are processed. Due to that we use here local cache that is periodically cleaned up.

– Data preprocessing – the module is responsible for crating computational representation of the articles. In current implementation we used joint representation based on words and article references. As this step is crucial for obtaining good results of automatic categorization in future we plan to extend it with usage more advanced methods of extracting characteristic text features.

– Data clustering – the module provides algorithms for grouping the data. In our current implementation we used k-means algorithm that is performed in the space reduced with selecting the most significant eigenvectors. In future we plan to test density based approach [23] and clustering with passing the messages [24].

– Cluster labeling module allows to implement a method for adding the labels to clusters. Here we can apply many approaches eg. usage most frequent words in a cluster. In our implementation we employ the fact that we operate on Wikipedia articles and we use more frequent category that binds together articles in a cluster.

In Figure 5 we present an example of system user interface. It shows clusters formed by WCS system for articles retrieved from PolishWikipedia for a query 'jdro' (kernel). As it can be seen clusters create different conceptual directions in which user may continue his or her search.

7. Discussion and Future Directions

Effective analysis of the large textual data is a basis for automatic text classification. The experiments presented here aim to research opportunities to use advances in clustering process from the usage of larger computational power.

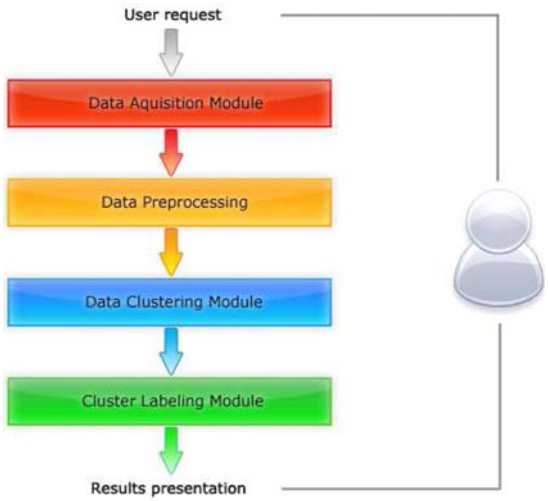

Figure 4. System architecture

Good scalability of clustering shows that the K-means algorithm can be made parallely and this process can be sped up with adding more processors. In future we plan to experiment with different similarity measures to obtain partitioning of the data according to selected (with the similarity measure) aspects of the data [8]. The k-means algorithm even calculated in eigenspace has many drawbacks so we plan to replace it with other more effective approaches like improved DBSCAN [25] and clustering with passing messages [24]. We think they will rise additional issues that should be solved to parallelize these algorithms.

In the results of our experiments we do not find straight relation between k and the number of k-means iterations. It is probably caused by high diversity of data density and random initialization of algorithm. In case of k-means round-robin approach for data partitioning was sufficient to balance the computations.

The situation in partitioning data differs while we compute distance matrix – partitioning the data into blocks does not allow to eliminate heterogeneity of data density. Some solutions can be introduced here eg. modifications of round-robin partitioning of data blocks that takes into account data densities. In the experiments we find the existence of some fixed sequential factors caused by sequential access to the disc storage. It appeared even when each process wrote to different files. Probably this could be changed in configurations of computational cluster.

For larger data sets eigen analysis should be parallelized to allow to obtain global view of the data. We also find other approaches for multidimensional scaling that use sequential computing of eigens [26]. This approach also requires to set up a parameter that constructs computation subspaces. Local Linear Embedding requires k nearest neighbors as an algorithm parameter while we have to define the number of k clusters.

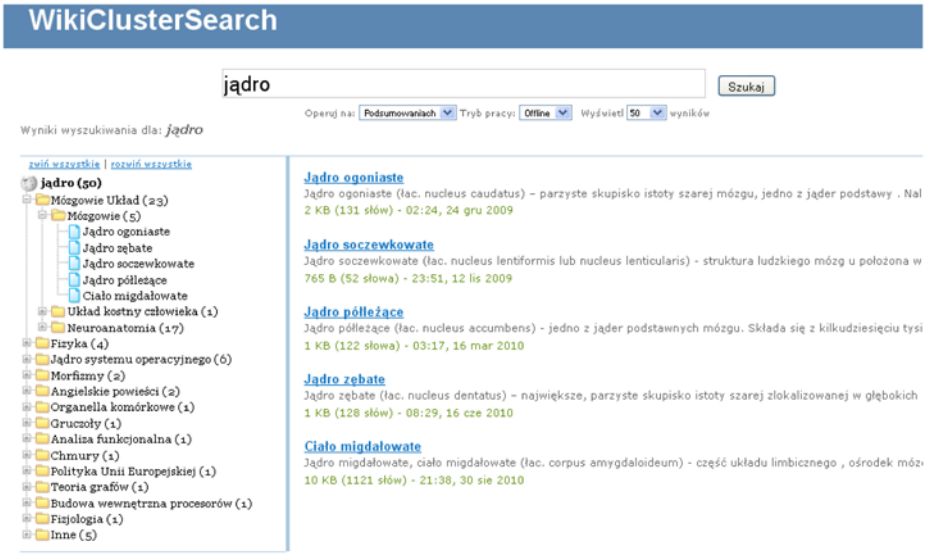

Figure 5. Example of user interface

Thus these approaches are conceptually similar. In future we plan to make parallel implementation of computation eigenvalues and eigenvectors to be able to make global data analysis. The challenge here will be to warrant effective access to the data on a disc storage and reduce interprocessor communication costs. We also want to compare our parallel implementations with implementations on GPU.

The effective implementation of basic operations for information retrieval:

– calculating the similarity

– clustering

– computing eigenvectors and values

have been used in our systems aiming at improving searching information inWikipedia. The initial implementation shows the SOM can be used as additional component for graphical presenting category structure. The clustering allowed to improve looking trough search results especially when the number of retrieved items were high.

The system presented in section 6.2 clusters only search results retrieved with a specified keyword. We also plan to perform experiments on large scale clustering – it is on the whole Wikipedia. Here instead of performing clustering within limited dataset (with specified by the user keywords) we compute whole Wikipedia. The experiments using well tuned clustering algorithm will allow to improve category system of Wikipedia finding missing and identify wrong assignments articles to categories.

8. Acknowledgments

This work has been supported by the National Center for Research and Development (NCBiR) under research Grant No. SP/I/1/77065/10 SYNAT: "*Establishment of the universal, open, hosting and communication, repository platform for network resources of knowledge to be used by science, education and open knowledge society*". The author would like also to thank Adam Gaca for his contribution to the application development.

References

[1] Berkhin, P. (2006). A survey of clustering data mining techniques, *Grouping Multidimensional Data*, p. 25–71.

[2] Ng, A., Jordan, M., Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm, *Advances in neural information processing systems*, 2, 849–856.

[3] Szyma´nski, J. (2011). Categorization of wikipedia articles with spectral clustering, *In*: Proceedings of IDEAL, *Lecture Notes In Computer Science*, 2, 849–856.

[4] Pestov, V. (2000). On the geometry of similarity search: Dimensionality curse and concentration of measure, *Information Processing Letters*, 73 (1-2) 47–51.

[5] Lee, J., Verleysen, M. (2007). *Nonlinear dimensionality reduction*. Springer Verlag.

[6] Strehl, A., Ghosh, J., Mooney, R. (2000). Impact of similarity measures on web-page clustering, in *Workshop on Artificial Intelligence for Web Search* (*AAAI 2000*), p. 58–64.

[7] Hartigan, J., Wong, M. (1979). A k-means clustering algorithm, *Journal of the Royal Statistical Society C*, 28 (1) 100–108.

[8] Szymański, J., Duch, W. (2010). Dynamic Semantic Visual Information Management, *In*: Proceedings of the 9th *International Conference on Information and Management Sciences*, p. 107–117.

[9] Xu, R., Wunsch, D., Books, I. (2009). 24 x 7, *Clustering. IEEE Press.*

[10] Halkidi, M., Batistakis, Y., Vazirgiannis, M. (2002). Cluster validity methods: part I, *ACM Sigmod Record*, 31 (2) 40–45.

[11] Aldrich, J. (2006). Eigenvalue, eigenfunction, eigenvector, and related terms, *Earliest Known Uses of Some of the Words of Mathematics*.

[12] Verma, D., Meila, M. (2003). A comparison of spectral clustering algorithms, *University of Washington, Tech. Rep. UW-CSE-03-05-01*..

[13] Jolliffe, I. (2002). *Principal component analysis*. Springer verlag.

[14] Cullum, J., Willoughby, R. (2002). *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Theory*. Society for Industrial Mathematic.

[15] Gomes, F., Danny, C. (1997). ARPACK++: A C++ Implementation of ARPACK Eigenvalue Package (Draft Version).

[16] Eldridge, S., Ashby, D., Bennett, C., Wakelin, M., Feder, G. (2008). Internal and external validity of cluster randomised trials: systematic review of recent trials, *British Medical Journal*, 336 (7649) 876.

[17] Kohonen, T., Somervuo, P. (1998). Self-organizing maps of symbol strings, *Neurocomputing*, 21 (1-3) 19–30.

[18] Dittenbach, M., Merkl, D., Rauber, A. (2000). The growing hierarchical self-organizing map, *In*: Proceedings of the *IEEE-INNS-ENNS International Joint Conference on Neural Networks*. IEEE, 6, 5–19.

[19] Dittenbach, M., Merkl, D., Rauber, A. (2001). Hierarchical clustering of document archives with the growing hierarchical self-organizing map, *In*: Proceedings of the Artificial Neural Networks Conference ICANN 2001, p. 500–505.

[20] Kaski, S., Honkela, T., Lagus, K., Kohonen, T. (1998). Websom-self-organizing maps of document collections1, *Neurocomputing*, 21 (1-3) 101–117.

[21] Duch, W., Szymański, J. (2008). Semantic web: Asking the right questions, *In*: Proceedings of the 7th International Conference on Information and Management Sciences. California Polytechnic State University Press, p. 1–8.

[22] Deptua, M., Szyma´nski, J., Krawczyk, H. (2012). Interactive information search in big data collections, *Studies in Computational Intelligence Springer* (*in print*).

[23] Wang, C., Duo, C. (2007). An improved density-based dbscan clustering algorithm, *Journal of Guangxi Normal University Natural Science Edition*, 25 (4) 104.

[24] Frey, B., Dueck, D. (2007). Clustering by passing messages between data points, *Science*, 315 (5814) 972.

[25] Birant, D., Kut, A. (2007). St-dbscan: An algorithm for clustering spatial-temporal data, *Data & Knowledge Engineering*, 60 (1) 208–221.

[26] Roweis, S., Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding, *Science*, 290 (5500) 2323–2326.