Abstract. This paper report on an ongoing research in improvement of WebSOM document clustering and presentation techniques. Several modifications of clustering techniques have been suggested and their impact on clustering of documents within the WebSOM framework has been investigated.

1 Introduction

The increasing number of documents returned by search engines for typical requests makes it necessary to look for new methods of representation of contents of the results. Nowadays, simple ranked lists, or even hierarchies of results seem not to be adequate for some applications. Within a broad stream of various novel approaches, we would like to concentrate on the well known WebSOM project, producing two-dimensional maps of documents [11, 13]. A pixel on such a map represents a cluster of documents. The document clusters are arranged on a 2-dimensional map in such a way that the clusters closer on the map contain documents more similar in content.

The WebSOM like document map representation is regrettably time and space consuming, and rises also questions of scaling and updating of document maps.

In our research on application of Bayesian networks and artificial immune systems we want to deal with these challenging issues. Similarly to WebSOM, the goal of the project is to create personal tools for exploration of free text documents by creating a navigational 2-dimensional document map in which geometrical vicinity would reflect conceptual closeness of the documents. We extend, however, WebSOM’s goals by a multilingual approach, new forms of geometrical representation and we consider also various modifications to the clustering process itself.

For purposes of soft classification of documents and creation of concept closeness graph being a basis for unsharp similarity measures of documents Bayesian networks will be used [9]. For optimization of parameters of map representation immuno-genetic algorithms are envisaged [19].

Our research targets at creation of a full-fledged search engine (with working name Beatca) for small collections of documents (up to several millions) capable of representing on-line replies to queries in graphical form on a document map. We follow the general architecture for search engines, where the preparation of documents for retrieval is done by an indexer, which turns the HTML etc. representation of a document into a vector-space model representation, then the map creator is applied, turning the vector-space representation into a form appropriate for on-the-fly map generation, which is then used by the query processor responding to user’s queries.

Subsequently, we concentrate on some preliminary results concerning various optimization attempts of individual processes. In particular, we report on our efforts to optimize the dictionary size, our reference vector optimization, and our new approach to map visualization.

2 System architecture

The architecture of our system has been designed to allow for experimental analysis of various approaches to document map creation. Software consists of essentially five types of modules, cooperating via common data structures. The types of modules are: (1) the robot, collecting documents for further processing, (2)
indexer, transforming documents into vector space representation, (3) optimizer, transforming the document space dictionary into more concise form, (4) mapper, transforming the vector space representation into a map form (5) search engine, responding to user queries, displaying the document maps in response to user queries.

The data structures, interfacing the modules, are of type: (1) HT Base (Hypertext base), (2) Vector base, (3) map, (4) Base Registry.

A HT Base is the result of a robot activity. We have currently two types of robots, one collecting documents from the local disk space, and another for the Web. A robot collects the hypertext files walking through links connecting them and stores them in a local directory and registers them in an SQL (currently MySQL) database. Standard information like download (update) date and time, original URL, summary (if extractable), document language and the list of links (together with information if already visited) is maintained by the robot.

A HT base can be processed subsequently by an indexer and possibly an optimizer to form a Vector base for the document collection. A Vector Base is a representation of a document space spanned by the words (terms) from the dictionary where the points in space represent documents.

A Vector base is then transformed to a document map by a mapper process. A map is essentially a two-level clustering of documents: there are clusters of documents and clusters of document clusters. Document clusters are assigned a graphical representation of elementary pixels in a visual representation, whereas clusters of document clusters are assigned areas consisting of pixels and are labeled by appropriate phrases.

Note that the same HT base may be processed by various indexers and optimizers so that out of a single HT Base many Vector bases may arise. Similarly one single Vector base may be processed by diverse mappers to form distinct maps. To keep track of the various descendants of the same HT Base, the Base Registry has been designed. The search engine makes use of all the maps representing the same HT Base choosing the one most appropriate for a given user query.

3 Indexer

3.1 General outline

Indexer/analyzer in our search engine accepts plain text and html documents. Our solution allows to use documents in English, German and Polish languages. Document analyzer proceeds as follows:

- Recognizes language of document.
- Removes html tags from document (if needed).
- Retrieves single words from document.
– Removes stop words.
– Stems words and stores their base forms.
– Calculates frequency of terms for each document and builds dictionary for whole document set.

English "stop words" list was taken from [21], German "stop words" list was taken from [22], Polish "stop words" list was taken from [23]. All lists are kept in separate text files, therefore there is possible to add new words to one of the actual list or use completely new one. To create our indexer we used some existing stemmers. English stemmers come from Porter[17] and Lovins[25,14]. Polish stemmer comes from [24]. A stemmer for German words is based on the report [26]. Language recognition is based on counting the frequency of stop words (most common words for a given language). For each language we dispose list of stop words and we can compare how many words occur in a current document from each list. Additionally system counts unique chars for these three languages. German and Polish have set of chars that are not present in any other language.

Beatca as an experimental platform has implemented five different stemmers for three languages and any new language can be easily added as a new module. Implementation of indexer allows to run it as multi-threaded indexing process to gain performance on multiprocessor machine. For testing, we used Pentium IV 3.2GHz HT machine (running Windows XP) and documents from [27]. We had 22% performance gain using multi-threaded indexing in comparison to one thread. There is no design limitation how many threads indexer can use (number of threads should depend on the number of available processors).

Typical html documents (around 10Kb each) indexing rate is about 80 per second on typical AMD Athlon 2000+ machine. The speed of indexer is limited by MySQL server (running on the same machine).

3.2 Dictionary Optimization

Lagus [13] and other authors pointed at the fact that the speed of the process of map creation and map visualization depend heavily on the dictionary size. Therefore we attempted to reduce the dictionary size, however trying different methods than e.g. random projection or LSA.

The idea of our dictionary optimization is based on filtering words that are useless for clustering purposes. Clustering is based on similarity of objects (in our case objects are documents) therefore features (in our case words/terms) that distinguish clusters are common within the clusters and are not common outside the clusters. It means that these features (words/terms) need to be characteristic for set of objects - group of documents. After this short introduction it is easy to imagine two possible extreme situations. The first one is that considered word occurs only in a few documents – therefore cannot be used to measure similarity to other documents which do not contain this word. The second extreme situation is that considered word is common for whole set of documents and does not distinguish any subset of them. For considered set of documents we would like to keep in dictionary words that are common for subset of documents and ignore words that are very rare or occur very often. Our goal in dictionary optimization is decreasing number of words in dictionary (by ignoring useless words) as well as decreasing time of clustering process. Most of text analyzers use "stop words" list to delete from dictionary common words for considered language. Some approaches use deleting of words that occurred in one document only. In our analyzer we did exactly the same but we also noticed that the approach may be more general. It is useful to have some quality measure that can express quality of term for specified set of documents. By quality of term we mean quality from clustering point of view. The quality should be based on frequency of term in each document and frequency of documents that contain the term. All measures use an entropy of the term. Normalized entropy equals 0 if term occurs only in one document and 1 if term is uniformly distributed over the documents (no matter how many of them). Fraction of the documents that contain considered term gives additional information about representation of the term in set of the documents. In this paper we propose the following measures:

\[ Q_1(t_i) = \frac{N_i}{N} \times \frac{-\sum_{j=1}^{N} \frac{N_{ij}}{N_i} \log \frac{N_{ij}}{N_i}}{\log N_i} \]

\[ Q_2(t_i) = \frac{-\sum_{j=1}^{N} \frac{N_{ij}}{N_i} \log \frac{N_{ij}}{N_i}}{\max (\sum_{j=1}^{N} \frac{N_{ij}}{N_i} \log \frac{N_{ij}}{N_i})} \]
\[ Q_1(t_i) = \begin{cases} \sqrt{\frac{N_i}{N} - 1} \cdot \prod_{i=1}^{N} \frac{N_{i+1} \log N_{i+1}}{N_{i} \log N_i} & \text{if } \frac{N_i}{N} > 0.5, \\ \frac{2 \cdot N_i}{N} \cdot \prod_{i=1}^{N} \frac{N_{i+1} \log N_{i+1}}{N_{i} \log N_i} & \text{if } \frac{N_i}{N} \leq 0.5. \end{cases} \]

where \( N_{ij} = Freq(t_i, d_j) \) is the number of occurrences of term \( t_i \) in document \( d_j \), \( N_i = Freq(t_i) \) is the number of documents that contains term \( t_i \) and \( N \) is the total number of documents. Factor \( \frac{N_i}{N} \) means fraction of the documents that contain considered term.

For dictionary optimizer we defined two parameters minimum threshold \( \text{minTres} \) and maximal threshold \( \text{maxTres} \). For each measure there are different default \( \text{maxTres} \) and \( \text{minTres} \). After couple of experiments we decided to use \( Q_1 \) with \( \text{minTres} \) set on 0.01 and \( \text{maxTres} \) set on 0.95.

All words that are below \( \text{minTres} \) and above \( \text{maxTres} \) are ignored during building the map.

In the future we would like to combine quality with weight of term to use one measure for optimization, map learning, clustering and map labeling.

Initial experiments show that after dictionary optimization we dispose one tenth of words in dictionary (decreasing by 90%) and map learning process is over 30% faster with exactly the same map of documents (see Fig. 2).

## 4 Document Map Generator

One of main goals of this project is to create 2D document map in which geometrical vicinity would reflect conceptual closeness of documents in a given document set. Additional navigational information (based on hyperlinks between documents) is introduced to visualize directions and strength of between-group topical connections.

### 4.1 Original WebSOM approach

Our starting point was widely-known Kohonen’s Self-Organizing Map (SOM) principle [11], which is an unsupervised learning neural network model, consisted of regular, 2D grid of neurons. Regression of neurons (represented by reference vectors \( m \in R^n \)) onto the space of document vectors \( x \in R^n \) can be iteratively computed as:

\[
m_{i}(t + 1) = m_{i}(t) + h_{c(x),i}(t) \cdot ||x(t) - m_{i}(t)||
\]

where \( i \) is the neuron index, \( t \) is the iteration number, \( c(x) \) is index of the winning (closest to \( x(t) \)) reference vector and \( h(t) \) is neighborhood function, usually Gaussian one:

\[
h_{c(x),i}(t) = \alpha(t) \cdot \exp \left( \frac{-||r_i - r_{c(x)}||^2}{2 \cdot \sigma^2(t)} \right)
\]

where \( \alpha(t) \) is the learning rate (monotonically decreasing with the number of iterations \( t \)), \( r_i, r_{c(x)} \in R^2 \) are map locations of corresponding reference vectors, and \( \sigma(t) \) is kernel-width function (also monotonically decreasing). It should be noted that distances between map locations in \( R^2 \) are Euclidean, while similarity between document and reference vector in original document space is computed as cosine of angle between corresponding vectors:

\[
||x - m_{i}|| = 1 - \frac{x \cdot m_{i}}{\|x\|\|m_{i}||}
\]

### 4.2 Our optimizations

On each level of processing (dictionary building, document representation, map learning, fuzzy segmentation, labeling and visualization) we have introduced and implemented a number of alternative solutions to compare their efficiency and performance both as free-text grouping and document groups visualization method.
Our design principle was to separate the map itself from map processing. It enabled to examine various combinations of map topologies with map processing methods as well as incremental learning of the map with combinations of learning algorithms.

First step was to implement above mentioned Euclidean map, projected on torus surface. Documents in the document space were represented as vectors of standard TFxIDF term weights, i.e.:

\[ w_{ij} = w(t_i, d_j) = Freq(t_i, d_j) \times \log \left( \frac{N}{Freq(t_i)} \right) \]

where \( Freq(t_i, d_j) \) is the number of occurrences of term \( t_i \) in document \( d_j \), \( Freq(t_i) \) is the number of documents containing term \( t_i \) and \( N \) is the total number of documents. As we have to deal with text documents represented in very high-dimensional space, we applied previously mentioned methods to reduce the dimensionality of document-space.

![Fig. 2. Computation time vs dictionary and reference vectors optimizations](image)

Keeping in mind that target application is to group and visualize as much as one million documents, the only structure which can be stored in RAM is the map itself. We have to represent it in a way which is both compact and efficient from algorithmic point of view. While clustering structure emerges during learning process, individual reference vectors become gradually sparser, attempting to approximate different subspaces of original document space. We represent these sparse vectors as balanced dictionaries (based on red-black trees [4]), additionally restricting their size by imposition of tolerance level, below which given term’s weight is assumed to be insignificant and corresponding dimension is removed from the dictionary.

Above mentioned approach resulted in fast computation (with linear expected complexity with respect to the number of significant dimensions) of two crucial parts of any map learning algorithm: similarity computation and map updates in vicinity of the winning (most similar) cell (Figure 2; \( \text{tolerance} = 10^{-6}, 0.01 \leq \text{quality} \leq 0.95 \)). It should be noted here that similarities computations during winner search and map updates are performed concurrently to take the advantage of multi-processor environment.

---

1 one should expect that the document space and - consequently - each reference vector will have dimension in order of a dozen thousand

2 all experimental results depicts average performance of five system runs
Furthermore, in order to restrict the total amount of computations needed, winner search is realized by joining two widely-used methods: in the first few iteration global search is performed. In this phase, high computational cost of global search is counterbalanced by incremental map size growth from some initial, moderate value, towards postulated target one. After global similarities between groups of documents emerge, we start next phase, in which local relations between documents have to be approximated. Thus, for each neighborhood width, a single phase of global search, followed by phases of local search (in each iteration starting from the winner of previous one) are executed. When convergence criterion is met, the learning neighborhood width $\sigma(t)$ is decreased and whole procedure is repeated. We called this strategy joint winner search.

Since neighborhood topology alterations in subsequent iterations are more smooth than at the very beginning of the learning process, local search in this phase moves the document only slightly and average length of document movement path (which influences computation time) is significantly less dependent on map size (Figure 3). It can also be seen (on the right-hand side of Figure 3) that after global similarities emerge, computation time is even slightly lower for bigger maps. In case of a sparser map (having lower average number of documents in a single cell) clusters are more disjoint, what in turn results in sparser reference vectors representing smaller subspaces of the original document space. These observations lead us to further optimization of winner search strategy: full global search is performed only in the first iteration of the learning process. Since we can estimate upper bound $U(t)$ of document movement path on the basis of path lengths in previous phases, only semi-global search is performed when neighborhood is decreased. During semi-global search only map cells whose distance from the winning cell of previous iteration is less or equal to $U(t)$ are taken into account.

Simultaneously, joint winner search strategy has produced better maps (comparing with local strategy) with respect to clustering error computed as average cosine angle between each document and its nearest reference vector (Figure 4). One can note that during few middle iterations learning process convergence (with respect to clustering error) is disturbed. This effect is a result of incompatible $\alpha(t)$ and neighborhood width $\sigma(t)$ functions. Simultaneously, our initial research shows that changing learning factor and neighborhood width consistently and at proper rate can accelerate convergence and improve quality of the final result. For instance, reciprocal (i.e. inversely proportional to iteration number) $\alpha(t)$ function behaves better than linear one (Figure 5).

---

3 visible computation time peaks are single global search iterations when neighborhood width $\sigma(t)$ is decreased
Fig. 4. Convergence vs winner search methods

Fig. 5. Convergence vs $\alpha(t)$ decreasing strategies
In particular, above mentioned winner search strategy enables to build map hierarchy starting from the bottom of the hierarchy (i.e. most detailed map). In such case, only the most detailed map is being learned, the rest of the hierarchy can be quickly obtained by merging original map cells and computing centroids (weighted by cluster density) of the corresponding reference vectors. Nevertheless, for unbalanced cluster structure (i.e. document sets consisted of topical groups varying in density) we plan to adopt also top-down approach and to build refined map of subclusters only for most dense topical groups. We also plan to compare maps obtained by different possible strategies [5, 12].

4.3 Topical groups extraction

Once the two-dimensional map has been generated (the generation process converged with respect to clustering error measure to some optimum), it represents some (possibly fuzzy) structure of topical groups. It is often useful to extract borders (which may also be fuzzy) of such groups and to assign them to different areas of map. For instance, Probabilistic Latent Semantic Analysis (PLSA [7]) method is based on the assumption that there exists latent variable $Z$, whose levels $z_k, k = 1 \ldots K$ correspond to underlying topical groups. Conditional probabilities $P(z_k \mid d_j)$ can be computed by means of singular-value decomposition (SVD) of term-document frequency matrix $N_{ij} = \text{Freq}(t_i, d_j)$ or, alternatively, by maximizing log-likelihood function:

$$\ln L(z) = \sum_{i,j} N_{ij} \ast \log (P(t_i \mid d_j)) = \sum_{i,j} N_{ij} \ast \log \left( \sum_k P(t_i \mid z_k) \ast P(z_k \mid d_j) \right)$$

Alternatively to word-document representation we can use WebSOM to build a model of citation patterns. In such model, document is represented as sparse vector, whose $i$-th component represents path length from [via outcoming links] or to [via incoming links] to $i$-th document in a given collection. It is also possible to estimate a joint term-citation model [2, 3].

In our investigations we have applied fuzzy clusters extraction algorithms used in artificial immune systems. Numerous methods have been drawn up in the past, to mention only [20], [6], [1] or [18]. Unfortunately, none of them appeared to be directly applicable to free-text document maps. We encountered several difficulties:

1. extremely fuzzy structure of implicit clusters in SOM (due to the smooth similarity structure of topical groups)
2. necessity of taking into account both similarity measure in original, document space and in the SOM space (in order to obtain jointed clusters during visualization)
3. necessity of identification and exclusion of outliers (i.e. documents not representative of any topical group) during cluster formation process

Fuzzy-ISODATA [1] and mountain method [18] are both based on gradual approach to the nearest (with respect to fuzzy similarity measure) attractor and dynamic modification of similarity weights. For instance, in ISODATA (also known as Fuzzy C-MEANS algorithm) the goal is to maximize fuzzy partition quality measure:

$$J_m (\{\mu_i\}, \{\tilde{v}_k\}) = \sum_{k=1}^{V} \sum_{i=1}^{c} \mu_i^m (k) \ast d^2 (k)$$

where $V$ is a set of objects (vectors) to be clustered, $d_i(k)$ is a distance between $v_k \in V$ and $i$-th cluster centroid $\tilde{v}_i$, $\mu_i : V \rightarrow [0, 1]$ is an $i$-th cluster fuzzy membership function and $m$ is a small positive constant, defining clustering crispness (in our case $m = 1.5$).

A priori estimation of the number of topical clusters ($c$) in a given document set is usually non-trivial issue. Consequently, a family of graph-theoretical clustering algorithms based on minimal spanning tree (MST) construction has been proposed [20]. We adapted efficient implementation of Prim algorithm on Fibonacci heaps [4] to build MST for a lattice (graph connecting adjacent cells) of reference vectors, in which
edge weights represent similarities between corresponding vectors. In second step, edges whose weights are significantly higher than adjacent ones are assumed to be fuzzy cluster borders and removed from MST. Resulting connected graph components represent clusters.

Unfortunately, MST on original map reference vectors reveals all three previously mentioned problems and especially weight deviations among neighboring edges are not significant. For this reason, we have applied two-stage approach: in first step ISODATA algorithm is performed on original map reference vectors and using upper-bound estimate of the number of clusters. Next, cluster reference vectors are computed as weighted average:

\[ v_c = \frac{\sum_{k=1}^{N_c} \text{Freq}(cell_k) \ast v_k}{\sum_{k=1}^{N_c} \text{Freq}(cell_k)} \]

where \( v_k \) is a reference vector of \( cell_k \) (i.e. k-th cell on the map), \( N_c \) is a set of cells in the cluster and \( \text{Freq}(cell_k) \) is the number of documents assigned to k-th cell. Finally, MST is built on a lattice of cluster reference vectors and above described edge trimming algorithm is applied.

Having map segmentation fixed, each topical group is labeled with most descriptive term, chosen among descriptors with highest weight in reference vectors assigned to corresponding area. Best label can be computed on the basis of between-group entropy:

\[ Fw(t_i, cell_k) = \text{Freq}(t_i, cell_k) \ast \text{Avg}(t_i, cell_k) \]

\[ w(t_i, N_c) = \frac{\sum_{k=1}^{N_c} Fw(t_i, cell_k)}{\sum_{k=1}^{N_c} \left( Fw(t_i, cell_k) \ast \left( 1 + \log \left( 1 + \frac{Fw(t_i, cell_k)}{\sum_{k=1}^{N_c} Fw(t_i, cell_k)} \right) \right) \right)} \]

It should be stressed that in case of hierarchical maps, this measure has to be recomputed on each level of the hierarchy. Another possibility is to compute measure based on inter-cluster term frequencies, average weights and standard deviations, for instance in a manner similar to previously mentioned reference vectors macro-averaging:

\[ w(t_i, N_c) = \frac{\text{Avg}^2_{cell_k \in N_c} (Fw(t_i, cell_k))}{\log (\text{Std}_{cell_k \in N_c} (Fw(t_i, cell_k)))} \]

and to label each cluster with descriptors which are highly rated (characteristic) with respect to above-given measure, but only for a given cluster.

5 Initialization of broad topics

Map-based approach, especially dealing with free-text documents, is very sensitive to initialization of cell reference vectors. To a large extent it is caused by overlapping topical clusters. Initialization method not taking into account main topics present in a document set and inter-topic similarities leads to obscure visualization. Besides, it negatively affects cluster extraction algorithms, which operate directly on resulting map reference vectors (e.g. Fuzzy-ISODATA).

Consequently, we have proposed following topic-based initialization method:

1. using PLSA, select K main topics in a given document set
2. select K map cells as the fixpoints for individual topics. Fixpoints are cells evenly spread over a centrally placed circle, whose radius is about 2/3 of the map dimension. Radius is chosen so that internal and external distances (on a torus surface) between opposite fixpoints would be equal.

\[ e.g. w_e > \text{Avg}(w_E) + k \ast \text{Std}(w_E), \text{ where E is a set of neighboring edges, k is small positive constant, avg is an average edge weight and std is a standard deviation of weights in the vicinity} \]
3. initialize selected fixpoints with K main topics
4. initialize all the remaining cells reference vectors using the following rule:

\[ \vec{v}_j = \frac{\sum_{i=1}^{K} d(c_j, c_i) \ast \vec{v}_i}{\sum_{i=1}^{K} d(c_j, c_i)} + \vec{\varepsilon}_j \]

where \( d(c_j, c_i) \) is the Euclidean distance on torus surface between given cell and i-th fixpoint (represented by vector \( \vec{v}_i \)) and \( \vec{\varepsilon}_j \) is a small, random disturbance.

Our experiments showed that maps initialized in this way are not only more comprehensible and stable - they are also easier and faster to learn. Since global similarities are determined during initialization phase, only local relations have to be learned. Consequently, such map can be learned with narrower learning kernel radius and lower value of learning coefficient \( \alpha(t) \). Moreover, in case of hierarchical maps, such initialization procedure can be repeated iteratively on each level of the map hierarchy.

In the next step, in addition to PLSA, we plan to utilize Bayesian approach to enrich topic descriptions with terms conditionally dependent on a given topic. Bayesian networks of terms could also be used to estimate between-topic relations: nearly orthogonal topics should occupy distant areas on a map.

A very intricate problem is the issue of map quality. We have developed the following procedure to check "objectively" the map properties. One of the maps is assumed as the "ideal" one. In case of our (dictionary) optimization methods it is the map without optimization. Then the map creation procedure is run with identical initialization for the "ideal quality" and the optimized procedure. The map quality is measured as the sum of squared distances of the location of each document on both maps. Initial results seem to be encouraging, though it is still too early to present a complete report.

6 3D Visualization

There is a variety of modifications to the basic SOM topology, having different clustering and visualization properties. In particular, we have applied Euclidean SOMs with quadratic and hexagonal cells, projected on torus surface and visualized in 2D. Well-known problem of SOMs in Euclidean space is limited number of cell neighbors (3, 4 and 6 - respectively for triangular, quadratic and hexagonal cells). SOM in hyperbolic spaces (HSOM) [15] is free from such limitation. In such space, document map is presented as a part of hyperbolic plane, and then its projection into the unit circle is computed (the type of projection determines type of model, e.g. Poincarre or Klein model [16]). What makes HSOM favorable approach is exponential growth of the number of cell neighbors with the distance from the center of the circle. Not only it influences visualization (so called “fisheye effect”), but also gives more accurate 2D approximation of (possibly complex) high-dimensional relations between groups of documents.

At present, in our search engine map of documents can be presented by 2 dimensional map (bitmap), following WebSOM’s paradigm [13], or by 3 dimensional cylinder. User can rotate map/cylinder to see it from each side. We decided that 3D cylinder is the best (more intuitive) representation of torus for typical user. For an example see fig. 6.

7 Conclusions and Future research

Our investigations show that in spite of immense successes in work on WebSOM project [13], there is still much space for improvements. We believe, that various optimization methods that have been presented here, may be combined with the results of other researchers giving a synergic effect. Also the visualization may be made still more attractive, however keeping a balance between visual effects and computer speed.

Next, we intend to integrate Bayesian and immune system methodologies with WebSOM in order to achieve new clustering effects.

Bayesian networks will be applied in particular to classify documents, to accelerate document clustering processes, to construct a thesaurus supporting query enrichment, and to keyword extraction. Immuno-genetic
Fig. 6. 3D map – cylinder

systems will be used for adaptive document clustering by referring to the mechanism of so-called metadynamics, for extraction of compact characteristics of document groups by exploitation of the mechanism of construction of universal and specialized antibodies, and for visualization and adjustment of resolution of document maps.

In the near future we envisage the number of further investigations, checking our ideas on:

1. Comparison of various map neighborhood topologies and visualization space properties: hexagonal neighborhood, hyperbolic spaces [15, 16], growing neural gas dynamically updated neighborhood based on minimal spanning trees computations (emerging local similarities are dynamically taken into account), growing SOM (dynamic change of map size during learning process)
2. Comparison of previously mentioned map hierarchy building approaches: top-down, bottom-up
3. Compassion of various document representations: weighted terms (basic, normalized, entropy-based quality measures $Q_1 - Q_3$), initial term and document groups obtained from Bayesian networks [9], joint model based on term and citation links representation
4. Comparison of dictionary reduction methods: dictionary optimization by quality measures $Q_1 - Q_3$, Bayesian networks of terms, Principal Component Analysis (PCA), random mapping [8]
5. Comparison of initial document space reduction methods: preliminary clustering by Bayesian networks of documents or artificial immune algorithms [19]
6. Comparison of map cells initialization methods and research into its impact on learning process convergence
7. Comparison of various map areas labeling methods: term extraction from Bayesian network, immune methods, labels induction from fuzzy rule-based classification
8. Selection of optimal (possibly dynamically modified) and coherent set of functions controlling learning neighborhood width, learning constant etc.
9. Introduction of dictionary, fuzzy clustering and map quality evaluation functions
References

21. English stop words
http://www.engin.umich.edu/caen/wls/software/oracle/text.901/a90121/astopsu2.htm
22. German stop words
http://www.engin.umich.edu/caen/wls/software/oracle/text.901/a90121/astopsu7.htm
23. An implementation of Finite State Automatons in Java, Dawid Weiss
24. An implementation of Finite State Automatons in Java, Jan Daciuk
http://www.eti.pg.gda.pl/~jandac/lsa.html