Mieczysław A. Kłopotek,

Sławomir T. Wierzchoń,

Krzysztof Ciesielski, Michał Dramiński,

Dariusz Czerski

Conceptual Maps of Document Collections in Internet and Intranet Coping with the Technological Challenge

Instytut Podstaw Informatyki
Polskiej Akademii Nauk

Warszawa 2007

Redaktor Naukowy Instytutu Podstaw Informatyki PAN
Prof. dr hab. Antoni Mazurkiewicz

Pracę do druku opiniował
Prof. dr hab. Jacek Koronacki

Adres autorów:
Instytut Podstaw Informatyki PAN
ul. Ordona 21
01-237 Warszawa

ISBN 978-83-922508-2-1

Preface

Visual presentation of document collections can provide us with new insights into their content. A number of research projects have been launched to create document maps, like WebSOM, Themescape and other.

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

In this book we describe some approaches we have found useful in coping with these challenges. Solutions have been verified by creating a full-fledged visual search engine for document collections (up to a million). We have extended WebSOM's goals with a multilingual approach and new forms of geometrical representation, and have also experimented with various modifications to the basic WebSOM document map creation process itself.

We also present certain map quality evaluation techniques, along with experimental results confirming the validity of our approach.

Categories and Subject Descriptors:
H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval
H.3.5 [Information Storage and Retrieval]: Web-based Services
H.5.3 [Information Interfaces and Presentation]: Web-based Interaction
Keywords: Web services, Search engines, Personalization, Bayesian networks, Neural networks, Artificial immune systems, Competitive learning, Information visualization

Contents

Chapter 1

Introduction

The increasing number of documents returned by search engines for typical requests forces us to look for new methods for representation of query results.

Nowadays, simple ranked lists, or even hierarchies of results seem inadequate for some applications.

That is why document maps have gradually become more and more attractive as a way of visualizing the contents of a large document collection. Within a broad stream of various novel approaches, we would like to concentrate on the well-known WebSOM project, producing (via extensive clustering) two-dimensional maps of documents (research by Kohonen and co-workers). A pixel in such a map represents a cluster of documents. Document clusters are arranged on a 2-dimensional map in such a way that the closer the clusters are to each other on the map, the more similar are the documents they contain.

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

In this work we describe some approaches we have found useful in coping with these challenges. Among others, techniques like Bayesian networks, growing neural gas and artificial immune systems will be discussed. Based on the above-mentioned techniques, we have created a full-fledged search engine for collections of documents (up to a million) capable of representing on-line replies to queries in a graphical form on a document map, for exploration of free text documents by creating a navigational 2-dimensional document map, where geometrical vicinity would reflect conceptual closeness of the documents. We have extended WebSOM's goals with a multilingual approach and new forms of geometrical representation. We have also experimented with various modifications to the underlying WebSOM clustering process itself.

The issue of crucial importance for the user's ability to understand the two-dimensional map is the clustering of its contents and appropriate labeling of the clustered map areas. It has been recognized long time ago that clustering techniques are vital to information retrieval on the Web. We will discuss several important issues that need to be resolved in order to present the user with an understandable map of documents. The first issue is the way of clustering the documents. In the domains, like e.g. biomedical texts, where the concepts are not sharply separated, a fuzzy-set theoretic approach to clustering appears to be a promising one. The other one is the issue of initialization of topical maps. Our experiments have shown that the random initialization performed in the original WebSOM may not lead to emergence of a meaningful structure of the map. Therefore, we have proposed several methods for topical map initialization, based on SVD, PHITS and Bayesian network techniques, which we will explain in the book.

We also consider selected optimization approaches to dictionary reduction, so-called reference vector optimization and new approaches to map visualization.

We will also report on an experimental study on the impact of various parameters of the map creation process on the quality of the final map.

The process of mapping a document collection to a two-dimensional map is a complex one and involves a number of steps, which may be carried out in multiple variants. In our search engine BEATCA[1] [15, 12, 13, 14, 41], the mapping process consists of the following stages (see Figure 2.1): (1) document crawling (2) indexing (3) topic identifi-

[1]http://www.ipipan.waw.pl/$\tilde{\ }$klopotek/BEATCA

cation, (4) document grouping, (5) group-to-map transformation, (6) map region identification (7) group and region labeling (8) visualization. At each of theses stages, various decisions can be made, implying different views of the document map.

For example, the indexing process involves dictionary optimization, which may reduce the document collection dimensionality and restrict the subspace, where the original documents are placed. Topics identification establishes the basic dimensions for the final map and may involve such techniques, as SVD analysis [3], fast Bayesian network learning (ETC [37]) and others. Document grouping may involve various variants of growing neural gas (GNG) techniques, [26]. The group-to-map transformation, used in BEATCA, is based on the SOM ideas, [43], but with variations concerning dynamic mixing of local and global search, based on diverse measures of local convergence. The visualization involves 2D and 3D variants.

With a strongly parameterized map creation process, the user of BEATCA can accommodate map generation to his particular needs, or even generate multiple maps covering different aspects of the document collection. Chapter 2 provides with more details on architecture of the search engine.

The overall complexity of the map creation process, resulting in long run times, as well as the need to avoid "revolutionary" changes of the image of the whole document collection, necessitate an incremental process of accommodating new, incoming documents into the collection.

Within the BEATCA project, we have devoted much effort to enable such a gradual growth. In this study, we investigate vertical (new topics) and horizontal (new documents on current topics) growth of document collection and its effects on the map formation capability of the system.

To ensure intrinsic incremental formation of the map, all the computation-intense stages involved in the map formation process (crawling, indexing, GNG clustering, SOM-clustering) need to be reformulated in terms of incremental growth.

In Chapter 4 we briefly mention our efforts to create a crawler that can collect documents (from the Internet) devoted to a selected set of topics. The crawler learning process runs in a kind of horizontal growth loop while it is improving its performance with the increasing amount of documents collected. It may also grow vertically, as the user can add new topics for search during its run time. In Chapter 5 we describe how the indexer is constructed in order to achieve incremental growth and optimization of its dictionary with the growing collection of documents. In Chapter 7 we describe how we explore the clustering properties of growing neural gas to accommodate new documents to the current clustering framework. At various stages of the overall process Bayesian networks are used, hence we briefly introduce in Chapter 3 the concept of Bayesian networks and in Section 3.3 we show how they are used in our search engine.

To evaluate the effectiveness of the overall incremental map formation process, we have compared it to the "from scratch" map formation in our experimental Chapter 8. We also present topic-sensitive approach, which appears to be a robust solution to the problem of the map generation process scalability (both in terms of time complexity and memory requirements).

In Chapter 9 we briefly characterize our map viewing system. Chapter 10 reports on related work by other researchers.

The conclusions from our research work can be found in Chapter 11.

Chapter 2

General Architecture

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

2.1 Modular Structure

The architecture of our system has been designed to allow for experimental analysis of various approaches to document map creation. The software consists of essentially five types of modules, cooperating via common data structures. The types of modules are as follows (see Figuire 2.1):

1. robot (spider, crawler), collecting documents for further processing,

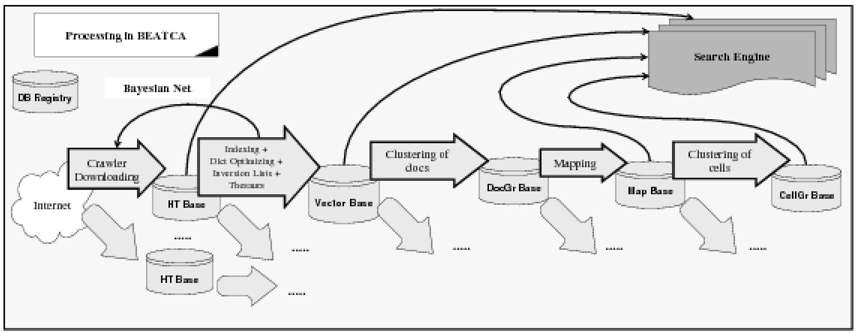


Figure 2.1: BEATCA system architecture

2. indexer, transforming documents into a vector space representation,

3. optimizer, transforming the document space dictionary into more concise form,

4. document clustering, identifying compact groups of documents sharing similar topics,

5. mapper, transforming the vector space representation into a map form

6. search engine, responding to user queries, displaying the document maps in response to such queries.

Additionally, we have an experiment management module, that can be instructed to configure the search engine process out of selected modules, to repeat the execution of some parts of the process, and to collect various statistics about the execution of the other modules and on the quality of the final and the intermediate results.

2.2 Data Structures

The data structures interfacing the modules are of the following types:

1. HT Base [hypertext documents],
2. Vector Base [vector space representations],
3. DocGR Base [thematical document groups]
4. Map Base [repository of various maps],
5. CellGR Base [map areas (groups of cells)]
6. Base Registry [registry of all databases, parameters and evaluation results].

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 from 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 (actually 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 – the 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 (stored in DocGR Base) and clusters of document clusters (stored in Map Base). Document clusters are assigned a graphical representation in terms of elementary "pixels" (labeled by appropriate phrases) in a visual representation, whereas clusters of document clusters are assigned "areas" consisting of "pixels". Note that in our approach we use a kind of multilevel maps, where higher levels "pixels" are "expanded" into maps/map fragments at a detailed level.

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.

The search engine has been explicitly designed as a test-bed for various algorithmic solutions to constituent search engine components. Hence an important additional feature is a database keeping track of results of experiments (constituting of selections of process components and data sets as well as quality evaluation procedures). The database of experiments is filled (and used in case of continued experiments) by the special experiment management module.

Chapter 3

Efficient Bayesian Networks

Bayesian networks (BN) [52] encode efficiently properties of probability distributions. Their application is spread among many disciplines. A Bayesian network is a directed acyclic graph (dag), the nodes of which are labeled with variables and conditional probability tables of the node variable, given its parents in the graph. Let capital letters $X_1, ..., X_n$ denote variables, and lower case letters $x_1, ..., x_n$ their particular instances. The joint probability distribution is then expressed by the formula:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | (x_1, \ldots, x_n) \downarrow \pi(X_i)) \tag{3.1}$$

where $\pi(X_i)$ is the set of parents of the variable (node) X_i, and $\downarrow$ is the projection of the left-hand vector onto the right-hand subspace.

On the one hand, BNs allow for efficient reasoning, and on the other hand many algorithms for learning BNs from empirical data have been developed [36]. When one can expect a particular form of the underlying dag structure, the learning algorithm can be quite simple. In particular, this is the case with the Chow/Liu [10] algorithm. The said algorithm is essentially a tree-spanning algorithm, where the weight of an edge between variables X, Y is calculated using the so-called DEP function:

$$DEP(X, Y) = \sum_{x,y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \tag{3.2}$$

We start with a "dag" consisting of the nodes representing variables, and containing no edges. Then we add an edge between the nodes X, Y for which the $DEP(X, Y)$ is maximal. This constitutes our initial tree. In subsequent steps we join other nodes one by one, and select the one node X from outside the tree, for which for some node Y in the tree $DEP(X, Y)$ is maximal at the given step. In this way we obtain an undirected tree, which we then orient arbitrarily, but in such a way that no edges meet head-to-head at any node. Note that the DEP function can be considered as a kind of "strength" of relationship between two variables: the higher its value, the stronger a mutual predictive relationship exists.

Other algorithms, learning a dag structure from data which assume a more general dag form are much less trivial.

A well-known problem with Bayesian networks is the practical limitation for the number of variables for which a Bayesian network can be learned in a reasonable time. In a comparative experiment using the BNLEARN system [36], for a network with 8 nodes and 8 edges and the sample size 15,000, several known algorithms had the following computational speeds: Chow/Liu [10]: 2s, Pearl [52]: 2s, PC [56]: 12s, K2 [19]: 7s, SGS [56]: 107s. For a network of 210 nodes and 301 edges (7 times replicated ALARM network) and the sample size 60,000, same algorithms had execution the following times: Chow/Liu: 40 min., Pearl: 40 min, PC: 12 hours, K2: 11 hours, SGS: could not be determined. The Chow/Liu [10, 11] algorithm learning tree-like Bayesian networks (and its derivative Pearl algorithm learning poly-trees) performed consistently better than the other ones. However, also this algorithm has an important limitation, related to the time and space consumption. The time and space required are both quadratic in the number of variables. This may prove also prohibitive for high dimensional data.

3.1 Types of Bayes Net Algorithms

The so-called "naive Bayes" classifier has been used for comparison in this study because it has been successfully applied in text categorization previously [62]. It may be viewed as a primitive form of a Bayesian network (a decision node connected to all other variables, which are not connected themselves).

The so-called TAN classifier (Tree Augmented Naive Bayes [25] is a combination of a naive Bayes classifier with Chow/Liu tree-like Bayesian network. The TAN classifier is constructed as follows: we create a tree-like Bayesian network using the Chow/Liu algorithm with the *DEP* function modified to:

$$DEP(X,Y|C) = \sum_{x,y,c} P(x,y,c) \log \frac{P(x,y|c)}{p(x|c)P(y|c)} \tag{3.3}$$

where C is the category variable.

We add the node C to the network connecting it with all the other nodes with arrows pointing away from C.

In our approach we use Bayesian multinets. They differ from TAN as follows: TAN assumes an identical structure of dependencies between non-category variables. A multinet-classifier allows for different structures of dependencies between variables for each level of the category variable, so it learns a separate Bayesian tree-like network (in our case using the ETC algorithm, see Section 3.2). For classification purposes we need to identify the a-priori probabilities $P_C(c)$ of the categories c (values c of the categorical variable C).

Classification with a Bayesian multinet is carried out by identifying the category c which maximizes $P(c|x_1,\ldots,x_n)$. The latter probability is calculated according to the Bayes rule as

$$P(c|x_1,\ldots,x_n) = \frac{P(x_1,\ldots,x_n|c)P(c)}{P(x_1,\ldots,x_n)} \tag{3.4}$$

As $P(X_1,\ldots,X_n)$ is not category-label dependent, we can simplify the above to:

$$P(c|x_1,\ldots,x_n) = \eta P(x_1,\ldots,x_n|c)P(c) \tag{3.5}$$

where η is a positive constant.

3.2 ETC Algorithm- Edge Tree Construction

The ETC Algorithm [37] may be considered as an accelerated algorithm for construction of a Bayesian network dag structure in the case when it can be assumed to be tree-like (either the true probability distribution is "tree-like", or a "tree-like" underlying dag is a sufficient approximation to the intrinsic probability distribution structure). The idea of the algorithm is based on growing tree of edges. Consider a Bayesian network with the dag structure in form of a tree. The tree of edges (edge tree) is a binary directed tree labeled with (undirected) edges of this dag such that if the dag consists of two nodes connected by an edge, then the edge tree consists of a single node labeled with this edge. Otherwise the edge tree describes the dag as follows: if we remove from the dag the edge labeling the edge tree root, then the dag splits into two subtrees which are described by the corresponding subtrees of the edge tree below the root. Thus, an edge tree is a kind of a complementary description of an ordinary tree.

In each iteration, one node (in our case describing a term from the dictionary of the document collection) is added to the tree, which in turn results in insertion of a new edge between that term and term already existing in the structure (see figures 3.1 and 3.2). In the second phase, the Edge Tree is converted to a BN and marginal probabilities are calculated.

As a foundation for the edge insertion process, DEP functions in the style of the Chow/Liu algorithm are used (see preceding section) to evaluate similarity of terms labeling the nodes of the dag we used in our experiments the DEP functions given below:

$$DEP_0(A,B) = \sum_{a,b} P(a,b)\log\frac{P(a,b)}{P(a)P(b)} \tag{3.6}$$

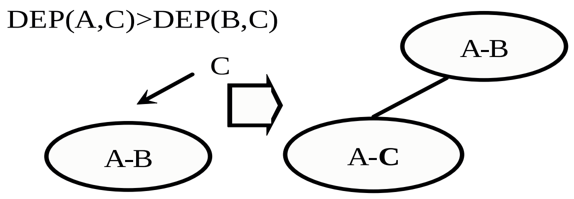


Figure 3.1: Adding a new term to the Edge Tree

$$DEP_1(A,B) = (P(a|b) - P(a|\bar{b}))^2 + (P(b|a) - P(b|\bar{a}))^2 \tag{3.7}$$

$$DEP_2(A,B) = (P(a|b) - P(a|\bar{b}))^2 \cdot (P(b|a) - P(b|\bar{a}))^2 \tag{3.8}$$

(The DEP function can be chosen as a parameter of the system). Note that the first equation is due to Chow/Liu [10]. The other ones were invented to overcome some problems with transitive similarity (see [37]). Calculation of DEP in the original Chow/Liu algorithm is necessary for each pair of terms. However in a massive collection of documents it is time consuming.

Therefore, in the ETC algorithm, the Edge Tree is built instead. First of all, terms are randomly permutated. The root node of the edge tree is composed of the two first terms. Each node in Edge Tree is constructed from two terms that represents one edge. In each iteration, a single subsequent term is inserted, constituting a new edge in the tree. For each new term, we start from the root node and calculate DEP between the new term and the left and right-hand side terms in the considered node. If the new term is more similar to the left-hand side term in the node, it goes left-hand side branch. If it is more similar to the right-hand side term, it goes. After selection of the branch, the term is repeatendly tested one level lower until it reaches the lowest level.

The ETC algorithm maximizes[1] the value of the DEP function of two terms in particular nodes. This means that edges are created on

[1]Strictly speaking, it can be proved mathematically that if the real dependence structure between variables is tree-like, and we take DEP_2, maximization is achieved. For DEP_0, and DEP_1, for a wide range of values it has been ex-

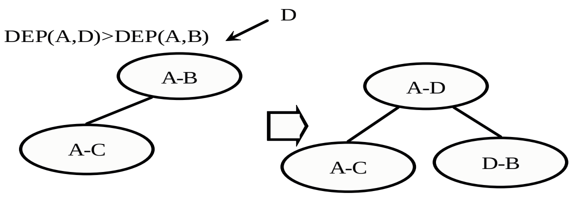


Figure 3.2: Replacement of a term in the node

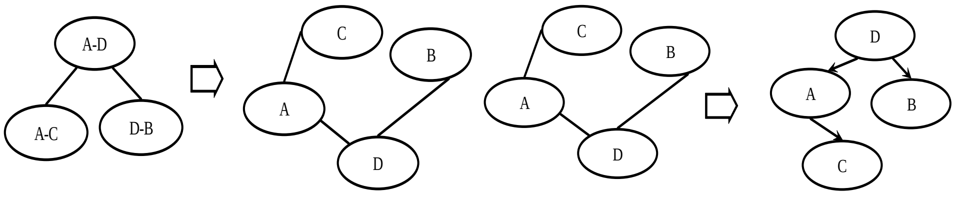


Figure 3.3: (a) Conversion from the edge tree into an undirected graph (b) Setting a direction inside the undirected graph

the terms that are as similar as possible. If the new term has a DEP greater than the DEP between the left and the right members of the node, it replaces the term to which it is more similar. Creation of a new node on a lower level of the tree is based on one of the terms from the top node and the new term.

Conversion from Edge Tree into BN is immediate. First of all, ET is converted into an undirected graph. Each term represents one node in BN and nodes are connected by edges taken from the edge tree (Figure 3.3).

The relation between the parent node and a child node (edge direction) is set randomly. First of all, a randomly selected node from the net becomes a root in the Bayesian tree, all the neighbors of root node become their children and so on. Finally, the marginal probabilities are calculated, i.e. the probability of occurrence of a term when the parent term is present (co-occurrence) and the probability when it is absent (exclusion) in a document.

perimentally verified that maximization preconditions hold. However, neither a mathematical proof nor a counter-example has been found.

Why is ETC applicable to large Bayesian networks? This is due to its reduced complexity, resulting from the use of the Edge Tree. When inserting a new variable into the Edge Tree, only $\log n$ comparisons with (at most n) variables already in the Edge Tree are needed. So, as proven in [37], only $n \log n$ comparisons (that is computations of DEP) are needed. For this reason, the algorithm scaled well for hundreds of thousands of variables.

Details of the ETC algorithm can be found in [37].

3.3 Bayesian Nets in the BEATCA System

In the BEATCA search engine, BNs are used at a few critical moments. We have found them very useful for initial clustering of documents set (see Section 7.5). For the map creation phase, a couple of clearly separated clusters are calculated. Such clusters proved to be especially useful for topical initialization (topics identification) of a clustering model and the SOM projection model, which we describe in section 7.5.

A BN is also used as a thesaurus in our system. After the indexing phase in BEATCA, a special dedicated BN is built on all terms in the dictionary[2]. Having collected the relevant set of documents on a given subject, a joint information stored in the BN and in the clustering model (described in Chapter 7) will constitute a context-dependent thesaurus. There is no room for the details, so we only notice that such a thesaurus is used to expand user queries for the purpose of more precise search in BEATCA search engine [41].

For building such a net, we use the ETC algorithm, which is effective and can hold a huge number of nodes (terms). A thesaurus needs to keep a thousands of terms and relations between them. After creation of the BN via the ETC algorithm, we obtain a special form of a net - a tree. Each term in our net has exactly one parent (except root node) and a set of children. The relation between the neighbor nodes is unidirectional (parent-child) but this direction does not determine a

[2]excluding terms of low clustering quality [39]

strict implication between the terms. The parent-child relation gives an information what is the probability of the child occurrence if the parent term does or does not occur. It is important in such a tree that the terms, which are close to each other in the net structure, are close conceptually. We would like to use relations between terms to complete a user query in order to restrict too general queries. Let us consider the user query "sheep". The answer to such a query can have a smaller number of results and can be more specialized if the query is extended by terms, which are the parent and children of "sheep". Initially, for small document sets we decided to build a Bayesian tree on terms which have frequency in the dictionary greater than 1. It means that term occurring in only one document will not be used to build thesaurus. This simplifies final Bayesian tree, decreases the number of terms to be processed and reduces the number of terms of lesser importance. However, the Bayesian net still will consist of dozens of thousands of nodes.

A special field of BN application in the BEATCA system is intelligent, topic-sensitive crawling, described in Chapter 4.

Chapter 4

Intelligent Topic-sensitive Crawling

Let us mention briefly our efforts to create a crawler collecting documents from the Internet devoted to a selected set of topics. The crawler learning process runs in a kind of horizontal growth loop while it improves its performance with the increasing amount of documents collected. It may also grow vertically, as the user can add new topics for search during its run time.

The ultimate goal of intelligent crawling [1] is to grasp documents which belong to certain topics and, obviously, to do it as efficiently as possible. Often it is particularly useful not to download each possible document, but only those which concern a certain subject. In the original approach, Aggarwal et al. [1] relied on versions of Naive Bayes estimates of the probability of a page being interesting based on the terms of the pointing pages, URL structures, sibling pages contents etc. In our approach we restrict ourselves to term occurrence; we do not assume term independence, but instead of this use Bayesian networks (BN) and the HAL[1] algorithm to predict relevance of documents to be downloaded.

[1]HAL, Hyperspace Analogue To Language, [47] is a text processing technology based on the psychological theory claiming that the meaning of a word is a function of contexts in which it appears, and the words sharing contexts have similar meanings. For its mathematical form and usage in a search engine see Section 4.2.

Our topic-sensitive crawler begins processing from several initial links, specified by the user. To describe a topic we are interested in, we use a query document. This special pseudo-document contains descriptive terms with predefined weights, which are later used to calculate the priorities of the crawled documents. During the crawling of the first few hundred documents, crawler behavior depends on the initial query document only. In the subsequent cycles, BNs or HAL models are built using the ETC algorithm (see Section 3.2)) or the HAL algorithm. Each of those nets is assumed to be a more accurate approximation of terms co-occurrence in the predefined topical areas. Subsequent BNs/HALs are constructed in increasing time intervals; the number of documents between the subsequent nets is calculated as $\delta \cdot i^2$, where δ is the initial number of documents and i is the net index.

Once the BN/HAL is built, we use it to expand the query (the pseudo-document) with new terms and to calculate weights for further documents.

We expand the pseudo-document by the adding parent and children nodes of the BN/HAL terms which are already present in query document, obtaining a set of extended query terms. New terms are assigned weights proportional to the product of the likelihood of their co-occurrence and the weight of the original term. We can also have negative weights, to exclude some terms which are unlikely to appear, calculated on the basis of extremely low marginal probabilities in BN/HAL.

4.1 Bayesian Net Document Query Expansion

At increasing subsequent time intervals, we build Bayesian Nets via ETC learning algorithm [37] to approximate term co-occurrence in topical areas. We use them to expand query and to calculate priorities for further documents links. We expand query by adding parent and children nodes of BN terms, which are already present in the query.

New terms get weights proportional to the product of the likelihood of their co-occurrence and the weight of the original term.

For each query term t_i we determine weights wz_{ij} for terms $t_j \in PC$, where PC is the set of parent and children terms taken from BN model:

$$wz_{ij} = \frac{p_{ij}}{\sum\limits_{k \in PC} p_{ik}} \cdot tqfidf_i \tag{4.1}$$

where $tqfidf_i$ is the product of query term frequency and inverse document frequency and p_{ij} is the probability of term i on the condition of occurrence of term j (taken from BN).

We can also have "negative" weights, to exclude some terms which are unlikely to appear. Final document links priorities are calculated by modified cosine measure between new expanded query document and document containing those links:

$$cos(q, d) = \frac{\sum_{t \in q} wd_t \cdot wq_t}{\sqrt{(\sum_{t \in q} wq_t^2) \cdot (\sum_{t \in q} wd_t^2)}} \tag{4.2}$$

where wd_t is the weight of term t in document d, wq_t is the weight of term t in query q. It should be noted that all sums are restricted only to terms appearing in q.

4.2 HAL Document Query Expansion

To expand query document we also use HAL model [47]. the ideology of HAL roots in the assumption that the context determines the meaning of a word. From computational perspective, HAL model can be represented as a matrix H in which each cell h_{ij} corresponds to a similarity measure of terms i and j.

Briefly speaking, if $s = (t_1, ..., t_k)$ is a sentence (ordered list of terms), then h_{ij} is the sum (over all sentences in a collection of documents) of co-occurrences of terms i and j. The scoring value of co-occurrence is defined as $max(0, K - p)$ where K is the predefined size of the scoring window and p is the number of terms which separate terms i and j in a given sentence. The main problem in this simple algorithm is obviously huge size of the matrix, which is equal to the number of distinct terms.

Similarly to the BN case, we build and iteratively update HAL term co-occurrence table. Next, we expand our pseudo document by adding k best co-occurrence terms $t_j \in W = \{t_1, ..., t_k\}$ for the terms already present in the query document. New terms get weights proportional to the product of their co-occurrence score and the weight of the original term:

$$wz_{ij} = \frac{h_{ij}}{\sum\limits_{k \in W} h_{ik}} \cdot tqfidf_i \tag{4.3}$$

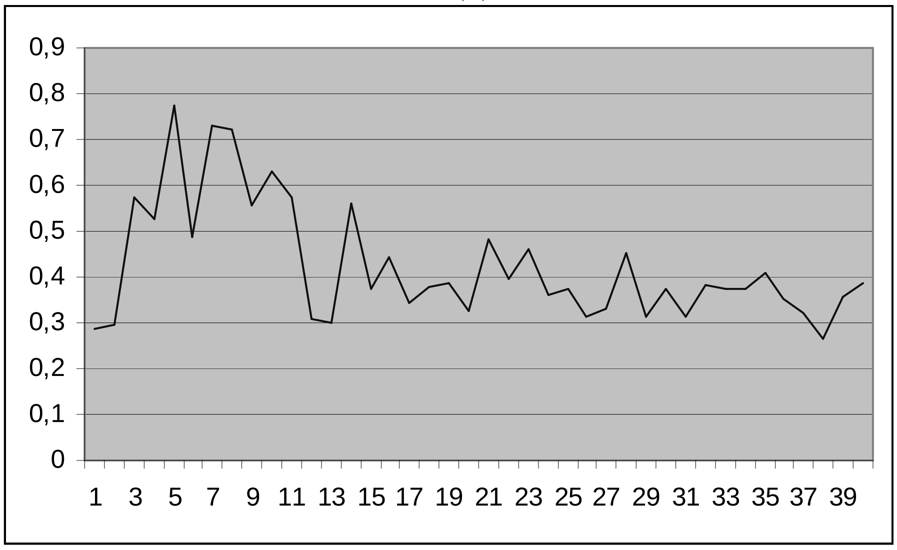
where $tqfidf_i$ is defined as in Equation 4.1, h_{ij} is the weight of term j taken from the HAL table.

Like in the Bayesian Net algorithm, the final priorities of the document links are calculated using a modified cosine measure between the new expanded query document and the document containing those links that is according to (4.2).

4.3 Experiments

To evaluate effectiveness of the presented topic-sensitive crawling, we have conducted two experiments, the first for the Bayesian Net algorithm and the other for the HAL algorithm. In both cases, the crawler starts from three seed links [`http://java.sun.com/j2ee/index.jsp`, `http://java.sun.com/products/ejb/`, `http://www.javaskyline.com/learning.html`]. A pseudo-document (a query) contains six descriptive terms with the corresponding weights, treated as occurrence frequencies [java(20) documentation(30) ejb(100) application(50) server(50) J2EE(30)]. Figure 4.1(a) presents the results for the crawler based on the Bayesian Net algorithm, and Figure 4.1(b) presents the results for the crawler based on the HAL algorithm.

(a)



(b)

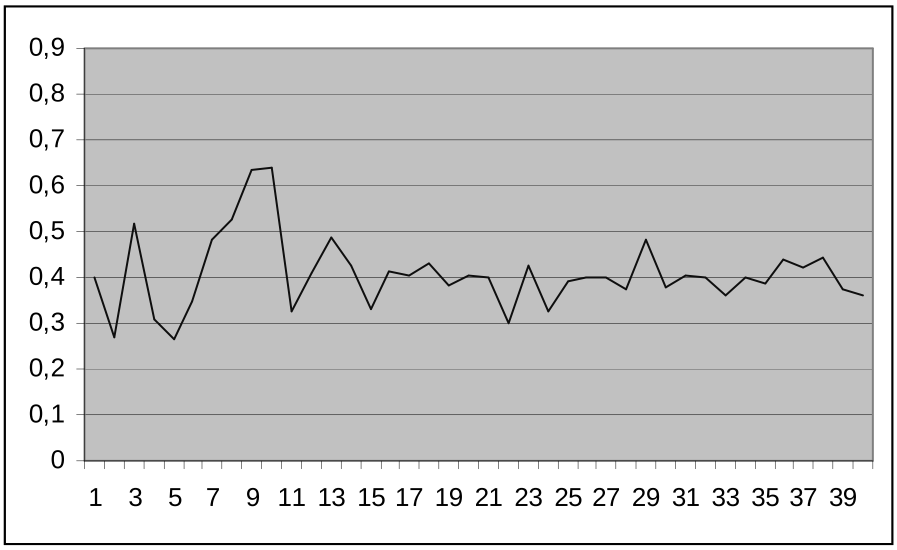


Figure 4.1: Crawler evaluation - average relavance measure (Y axis) after a number of iterations (X axis, one iteration - 500 documents downloaded): (a) Bayesian Net algorithm (b) HAL algorithm

The quality measure is the average relevance measure (4.4), computed after downloading each new 500 documents. The relevance is equal to the modified cosine measure (4.2), but only for the terms which are present in the initial user query ($q = q_0$).

$$relevance = cos(q_0, d) = \frac{\sum_{t \in q_0} wd_t \cdot wq_t}{\sqrt{(\sum_{t \in q_0} wq_t^2) \cdot (\sum_{t \in q_0} wd_t^2)}} \tag{4.4}$$

Both methods gave similar results; the average cosine measure was about 0.4. This appears to be a satisfactory result, which shows that the crawler did not lose the predefined topic during the crawl. The Bayesian Network proved to be the faster of the two methods. However, its disadvantage is the requirement to stop whole process in order to rebuild the BN model. The HAL table can be built during the crawl, but it requires more computations.

Chapter 5

Indexer

5.1 Functionality

The indexer/analyser in our search engine accepts plain text and html documents. Our solution allows for using documents in English, German and Polish. The document analyzer proceeds as follows:

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

The English "stop words" list was taken from [67], the German "stop words" list - from [68], and the Polish "stop words" list - from [69]. All lists are kept in separate text files, so new words can be added to one of the current list or a completely new one can be used. To create our indexer we used certain existing stemmers for the three languages. The English stemmers come from Porter [53] and Lovins [71, 49], the Polish stemmer comes from [70], and the stemmer for German words is based on [72]. The structure of the experimental platform BEATCA allows for an easy inclusion of further languages into the analysis as new modules. Inclusion of a new language requires incorporation of appropriate stemmers because of the importance of appropriate word stemming for human friendly performance (document clustering, response to queries), and the significant differences between various languages prohibits creation of a common stemming algorithm.

Our implementation of the indexer allows for running it as multithreaded indexing process to gain performance on a multiprocessor machine. For testing, we used a Pentium IV 3.2GHz HT machine (running under Windows XP) and documents from [73]. We achieved a 22% performance gain using multithreaded indexing in comparison to a single thread. There is no design limitation on the number of threads that the indexer can use (the 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).

5.2 Dictionary Optimization

Lagus [46] 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 [5] or LSA [21].

The idea of dictionary optimization is based on filtering words that are useless for clustering purposes. Clustering is based on similarity of objects, i.e. documents; therefore the best cluster separating features (i.e. words/terms) are those that are common for objects (documents) within one cluster and rarely occur outside that cluster. It means that these features need to be characteristic for set of objects - group of documents. Hence one needs to exclude from dictionary words occurring only in a few documents as well as those occurring in the vast majority of them – as they poorly split the set of documents. Our goal in dictionary optimization is to decrease the number of words in dictionary (by ignoring useless words) so that the time of clustering process is diminished without seriously deteriorating the quality of the search engine processes. Most of text analyzers use "stop words" list to delete from dictionary common words for considered language. Some approaches use deleting the words occurring in one document only. In our analyzer we did exactly the same but we also noticed that the approach may be generalized. Below we explain this generalization..

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. We proposed and use the following measures:

$$Q_1(t_i) = \frac{N_i}{N} \cdot \frac{-\sum_{j=1}^{N} \frac{N_{ij}}{N_i} \cdot \log \frac{N_{ij}}{N_i}}{\log N_i} \tag{5.1}$$

$$Q_2(t_i) = \frac{-\sum_{j=1}^{N} \frac{N_{ij}}{N_i} \cdot \log \frac{N_{ij}}{N_i}}{max_{k;t_k \text{is a term}}(-\sum_{j=1}^{N} \frac{N_{kj}}{N_k} \cdot \log \frac{N_{kj}}{N_k})} \tag{5.2}$$

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. The factor $\frac{N_i}{N}$ means fraction of the documents that contain considered term.

For dictionary optimizer we defined two parameters: minimal threshold (*minTres*) and maximal threshold (*maxTres*). For each measure there are different default (*maxTres*) and (*minTres*). After couple of experiments we decided to use Q_1 with (*minTres*) set on 0.01 and (*maxTres*) set on 0.95. All words that are below *minTres* or above *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.

The initial experiments showed that after dictionary optimization we are left with one tenth of the words in the dictionary (decreasing their number by 90%) and the map learning process is over 30% faster with exactly the same map of documents (see Fig. 6.1).

5.3 Incremental Indexer

The indexer has been designed in order to achieve incremental growth and optimization of its dictionary with the growing collection of documents.

In our database, a document can be labeled as 'indexed', 'downloaded' and 'updated'. After each new portion of documents downloaded by crawler, indexing is run on the 'downloaded' or 'updated' documents (marked by crawler). For recently updated documents indexer runs reindexing process. In the first step of this process indexer analyzes vector of terms and decreases corresponding frequencies in the dictionary. In the second step vector of terms is dropped down and document is indexed from scratch. Process described above is a simple solution, however it has very strong advantage since it minimizes cost of the database communication.

After the indexing phase, afore-described (section 5.2) dictionary reduction techniques are applied.

5.4 Summarizing Documents

General idea of document summary in our search engine is to select the most valuable sentences from document. Summary of a document is based on user's query which can be presented as set of terms:

$$\{t_1, t_2, \ldots, t_n\} \tag{5.3}$$

that influence on later selection of sentences. Let us assume that each sentence ends with '.' or '!' or '?' and in given sentence there are m terms. For such sentences the algorithm calculates frequencies of each term from the query using stemmed forms of terms.

$$\{f_1, f_2, \ldots, f_n\} \tag{5.4}$$

To measure importance of sentence we calculate the following weight for each sentence:

$$i = \mathit{entropy} + \mathit{intensity} + \mathit{size} \tag{5.5}$$

where:

$$\mathit{entropy} = \frac{-\sum_{i=1}^{n} \frac{f_i}{F} \cdot \ln \frac{f_i}{F}}{\ln n}, \qquad F = \sum_{i=1}^{n} f_i \tag{5.6}$$

$$\mathit{intensity} = \frac{F}{m} \tag{5.7}$$

$$\mathit{size} = \begin{cases} \frac{s}{m}, & if s \leq m; \\ \frac{m}{s}, & if s > m. \end{cases} \tag{5.8}$$

Third factor in i is a penalty function for too long or too short sentences. In initial experiments we used parameter s set on 15. Final order of the sentences in the summary is the same as in document. Abstracts below come from medical documents [73]. For example we present two abstracts for user query *'Sleep Disorder'*:

- *'MEDICAL PROBLEMS AFFECTING SLEEP First, the bad news: Older people are likely to suffer both medical disorders that may disrupt sleep and specific sleep disorders. One sleep disorder combines dreams with movement: REM sleep behavior disorder. Most sleepers are virtually paralyzed during REM or dreaming sleep; people with REM sleep behavior disorder do not have this motor inhibition and literally act out their dreams.'*

- *'Each year, there are about 40 million people in the United States who suffer from sleeping disorders. Recent research suggests* *that if sleep deprivation is long-term–whether because of lifestyle choices or sleep disorders–it may increase the severity of age-related chronic disorders such as diabetes and high blood pressure. director of the National Center on Sleep Disorders Research, part of the National Heart, Lung, and Blood Institute.'*

Use use dynamic abstracting techniques, that is the actual abstract is generated on the fly in response to user query. The proper indexing phase concentrates on preparation of static structures and weights needed during the dynamic phase.

Chapter 6

Mapper

One of the main goals of our BEATCA project is to create a multidimensional document map where 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 inter-group topical connections.

At the heart of the overall process is the issue of clustering documents. Clustering and content labeling are the crucial issues for user's understanding of the two-dimensional map. It was recognized long time ago that clustering techniques are of vital importance for information retrieval on the Web [38]. We started our research with the WebSOM approach, which, however, was a bit unsatisfactory for us, as both speed and clustering stability were not very encouraging.

We guess that the basic problem with WebSOM lies in the process of initialization of so-called reference vectors, being the centroids of the clusters to grow. In the original WebSOM, they are initialized randomly, to be corrected later on in the clustering process. Such an initialization may lead to instability during clustering, because the WebSOM learning process possesses a "learning speed" parameter α which may turn out to be too low to ensure convergence for a particular initialization. Another problem lies in the general concept of clustering. In WebSOM, it is tightly coupled with a (non-linear) projection from a multidimensional to a two-dimensional space. As there may be infinitely many such projections with equal rights, one really needs a sense of goal to select the appropriate one.

The first issue we have tackled was dictionary optimization strategies and their speed-up effects to cope with the complexity issue (see Section 5.2. Another research direction was to obtain better clustering via the fuzzy-set approach and immune-system-like clustering, [39]. Our approach to document clustering is a multi-stage one:

- clustering for identification of major topics (see [14], [13])
- cellular document clustering[1] (see [15])
- cellular document clusters to WebSOM map projection (see [12])
- cell clusters extraction and cell labeling (see [39])

In order to obtain a stable map, we need to fix the perspective from which the document collection is viewed. This can be achieved if we identify major topics of the document collection. This is done during the step "clustering for identification of major topics". In Section 6.3 (see also [15]) we suggest a Bayesian approach, which was the result of our investigation into the behavior of the algorithm [32]. Alternatively, different initialization techniques could be used: in [14] we described an approach to major topic identification based on LSI/SVD (Latent Semantic Indexing/Singular Value Decomposition), and in [16] we described usage of a version of Fuzzy-ISODATA algorithm for that purpose. Having identified the major topics, we can initialize the map in a more definite way, as described in Section 6.4.

After the topics have been identified, the documents need to be assigned to these and the intermediate ones, and the relationships

[1]By cellular clustering we understand those clustering techniques which lead not only to obtaining a set of (disjoint) clusters, but also to imposition of an inter-cluster relationship based on content similarity between clusters. The clusters in such a technique will be called "cells". One example of such a technique is the WebSOM (Web Self-Organizing Maps) approach, where Web documents are assigned to cells in a rectangular grid of squares or hexagons. Other approaches in this class are growing neural gas (GNG) and artificial immune systems (AIS).

between the topics have to be identified. This process is called by us "Cellular document clustering", and leads to creation of a graph model of the document collection. Three different techniques may be used at that point: the WebSOM (plain or hierarchical) approach (see Section 6.1), the GNG approach (see Section 6.5), or the artificial immune systems (AIS) approach (see [42]).

The graph model of the document collection needs to be visualized in the form of a document map. Therefore, a step named "projection of cellular document clusters to the WebSOM map" is applied. It is redundant if the WebSOM algorithm (plain or hierarchical) is used to cluster the documents, but is necessary for more elastic topologies like the GNG model and the AIS model. The above step is described in Section 7.5.

Finally, for the purposes of better readability of the document map, cells need to be joined into larger, topically uniform areas, which is done in the step named "cell clusters extraction", described in Section 6.6. Also cells have to be labeled, as described in Section 6.6.

6.1 The Original WebSOM Approach

Our starting point was the widely-known Kohonen's Self-Organizing Map (SOM) principle [43]. SOM is an unsupervised learning neural network model, consisting of a 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:

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot h_{\zeta(x),i}(t) \cdot \|x(t) - m_i(t)\| \qquad (6.1)$$

where i is the neuron index, t is the iteration number, $\zeta(x)$ is the index of the winning[2] (closest to $x(t)$) reference vector, $\alpha(t)$ is the learning

[2]The process of establishing the winning reference vector is called winner search. If all the reference vectors are taken into account, when looking for the winner, as in the original WebSOM approach, we speak about *g*lobal winner search , while any heuristics that restrict the set of candidates to a single reference vector and its vicinity, is called *l*ocal winner search . The advantage of the global search is the correctness of identification of the winner, which is paid for by high computational

coefficient $h(t)$ is the neighborhood function (the kernel learning function), usually the Gaussian one [43], and the $\|x(t) - m_i\|$ operator measures the distance between the vector $x(t)$ representing the document and the vector m_i, being the model vector of the cell i.

$$h_{j,i}(t) = a \cdot exp[-\frac{d(j,i)}{2 \cdot (\sigma^2(t))}] \tag{6.2}$$

with $\sigma(t)$ called neighborhood width function.

It should be noted that the distances $d(j,i)$ between map locations in R^2 are Euclidean (Manhattan distances may also be considered), while the similarity between document and reference vector in the original document space is computed as the cosine of the angle between the corresponding vectors (in order to account for diverse document lengths).

6.2 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 a visualization method for documents groups. Our design principle was to separate the map itself from map processing. This allowed us to examine various combinations of map topologies with map processing methods as well as incremental learning of the map with combinations of learning algorithms.

The first step was to implement the above-mentioned Euclidean map, projected on a torus surface. The documents in the document space were represented as vectors of standard term frequency inverse document frequency ($tfidf$) weights:

burden. In local winner search strategies, this cost is reduced by radical reducing the number of reference vectors to compare to, but at the risk of missing the intrinsic winner and using only a locally best reference vector.

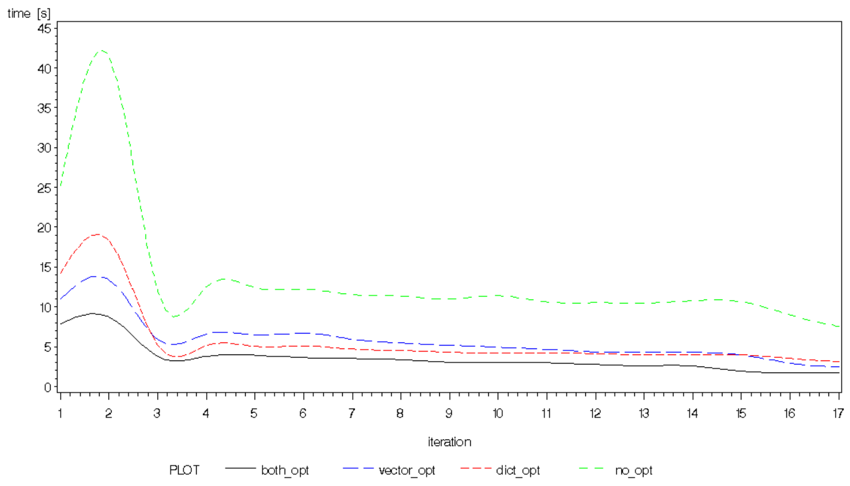


Figure 6.1: Computation time vs optimization of dictionary and reference vectors

$$tfidf_{ij} = w(t_i, d_j) = Freq(t_i, d_j) \cdot log\left(\frac{N}{Freq(t_i)}\right) \tag{6.3}$$

where $tfidf_{ij}$ is the term frequency inverse document frequency value for term t_i and document d_j, $Freq(t_i, d_j)$ 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 a very high-dimensional space, we applied the previously mentioned methods to reduce the dimensionality of the document-space (see Section 5.2).

Keeping in mind that the target application has 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 the algorithmic point of view. While the clustering structure emerges during the learning process, the individual reference vectors become gradually sparser, attempting to approximate different subspaces of the original document space. We represent those sparse vectors as balanced dictionaries (based on

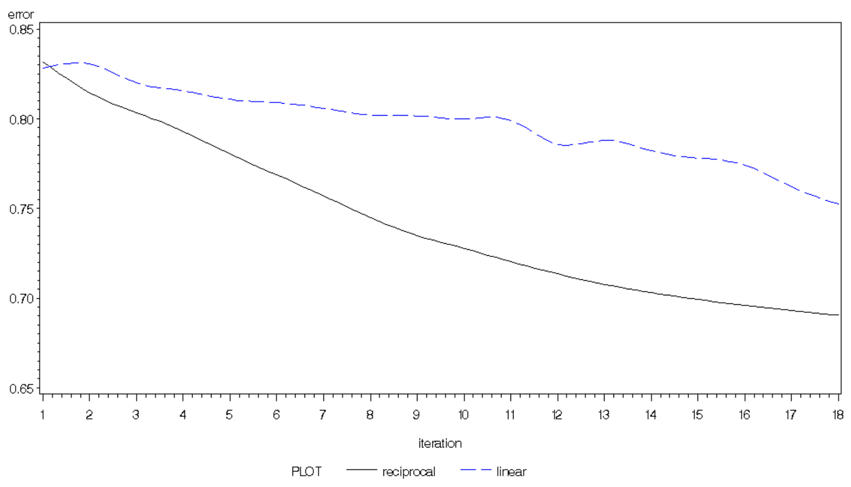


Figure 6.2: Convergence vs strategies for decreasing $\alpha(t)$

red-black trees [20]), additionally restricting their size by imposing a tolerance level, below which given term's weight is assumed to be insignificant and the corresponding dimension is removed from the dictionary.

The above mentioned approach resulted in fast computation (with expected linear complexity with respect to a number of significant dimensions) of two crucial parts of any map learning algorithm: similarity computation and map updates in the vicinity of the winning (most similar) cell (Figure 6.1[3]; $tolerance = 10^{-6}$, $0.01 \leq quality \leq 0.95$). It should be noted here that similarities computations during winner search and map updates are performed concurrently to take advantage of the multi-processor environment.

Furthermore, in order to restrict the total amount of computations needed, we combined in a creative way two approaches to winner search The strategy, that we call *j*oint winner search, is carried out by creatively combining two widely-used methods: *global winner search* and *local winner search* (see section 6.1). In the first few iterations global search is performed. In this phase, the high computational cost

[3]all experimental results depict the average performance of five system runs

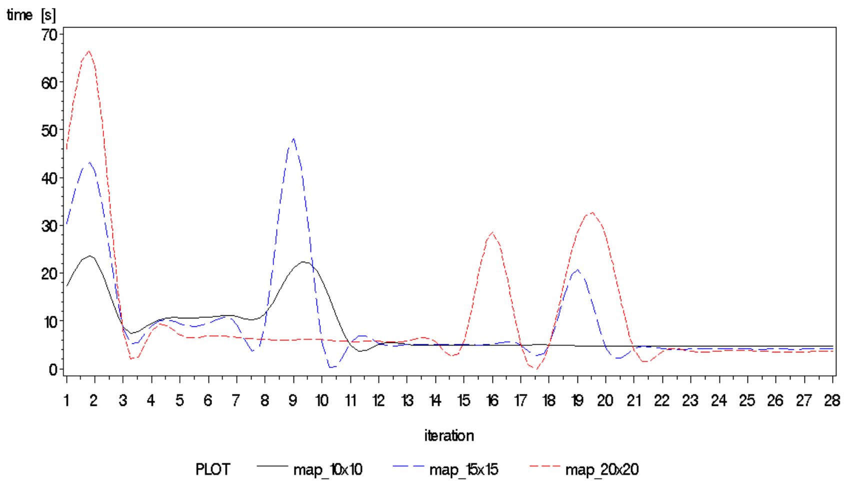


Figure 6.3: Computation time vs changing map sizes [joint winner search method]

of global search is counterbalanced by the incremental map size growth from some initial, moderate value towards the postulated target one. After global similarities between groups of documents emerge, we start the next phase, in which local relations between documents have to be approximated. Thus, for each neighborhood width, only one phase of global search is executed, and thereafter local search (in each iteration starting from the winner of the previous one) is only performed. When the convergence criterion is met, the learning neighborhood width $\sigma(t)$ is decreased and whole procedure is repeated. We call this strategy *joint winner search*.

Since the neighborhood topology alterations in the subsequent iterations are more smooth than at the very beginning of the learning process, in this phase the lokal search moves the document only slightly and the average length of the document movement path (which influences the computation time) is significantly less dependent on the map size (Figure 6.3[4]). It can also be seen (on the right-hand side of Fig-

[4]the visible computation time peaks are single global search iterations when

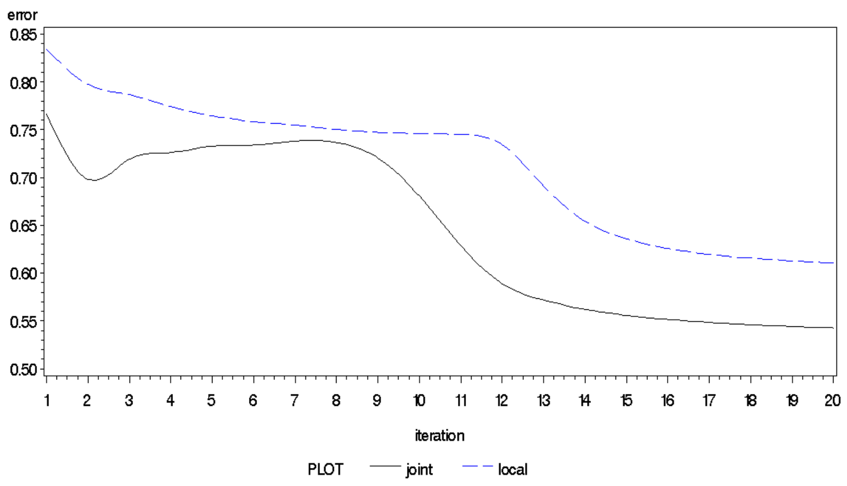


Figure 6.4: Convergence vs winner search methods

ure 6.3) that after global similarities emerge, the computation time is even slightly *lower* for bigger maps. In case of a sparser map (having a lower average number of documents in a single cell) the clusters are more disjoint (have fewer common terms), which in turn results in sparser reference vectors representing lower-dimensional subspaces of the original document space.

At the same time, the joint winner search strategy has produced better maps (compared with local strategy) with respect to the clustering error computed as the average cosine angle between each document and its nearest reference vector (Figure 6.4).

One can note that during few middle iterations the learning process convergence (with respect to the clustering error) is disturbed. This effect results from incompatibility of the learning speed funkcion $\alpha(t)$ and the neighborhood width funkcion $\sigma(t)$. Our initial research shows that the wrong choice of these functions can negatively impact the map quality. On the other hand, changing the learning factor and the neighborhood width consistently and at proper rate can accelerate

the neighborhood width $\sigma(t)$ is decreased convergence and improve the final result. For instance, a reciprocal (i.e. inversely proportional to the iteration number) function $\alpha(t)$ behaves better than a linear one (Figure 6.2).

6.3 Identification of Broad Topics

Our preliminary experiments showed that the random initialization performed in the original WebSOM may or may not lead to emergence of a meaningful structure of the map. It was known also in the past that the SOM map as a whole behaves in a highly chaotic way, and even a slight disturbance in the data or introductionel some random factor (which is always the case, for the document vectors are presented in a random order and the model vectors are initiated randomly at the beginning) can make the resulting maps dramatically different, although both may be of the same quality. Different parts of the map can be rotated, scaled and twisted in many ways, though they may still reflect a specific clustering structure that is enforced by the data set and the algorithm principles.

As a first step towards the map stability, we need to fix the perspective from which we look at the documents. The most natural way would be to identify the major topics of the document collection and to distribute these topics over the map. Topic identification is covered in this section, and the map initialization in the next one.

One of the well-known possibilities to identify topics in a collection of documents is the LSI/SVD (Latent Semantic Indexing/Singular Value Decomposition). For a detailed description please refer to [3]. Our implementation has been described in [14]. In brief, the idea is that the document-term matrix $N_{ij} = Freq(t_i, d_j)$ (with non-zero entries indicating presence of a term t_i in a document d_j) can be decomposed into a set of eigen vectors, with the eigenvectors associated with the largest eigenvalues being capable of a good approximation of the original matrix content. The principal eigenvector can be interpreted as the "major" theme of the collection, with components indicating which terms are characteristic for the topic. The second, third etc. eigenvector can be viewed in a similar way (as variations of the major topic). Some controversies about negative values of the non-principal vectors need to be resolved (see [14]). One possible interpretation is of a departure from the topic of the major eigenvector. The other one is to take documents closest to the axes spanned by the orthogonal eigenvectors as representative of the topic of an eigenvector and to derive term profile of the topic from these documents.

After having studied the LSI approach, a natural alternative seemed to be the investigation of the PLSA algorithm [32], based on the same general ideology as LSI, but on different, probabilistic principles. PLSA (Probabilistic Latent Semantic Analysis)

The PLSA algorithm seemed to be a good choice in principle, but some weaknesses came to sight. The advantage of the PLSA approach over e.g. LSI is the possibility of using fuzzy membership in each cluster (sum of membership levels need to be 1) and working on the frequency of terms in documents. However, there is a problem with algorithm convergence. We presume that the direct reason is a too high number of degrees of freedom and existence of more than one maximum for the likelihood function. Therefore, we decided to abandon the fuzzy clustering of PLSA in favor of a crisp cluster membership (like that used in Naive Bayes approaches). This approach gives us sharp clusters that are easily processed and stable.

The original PLSA algorithm [32] can be viewed as a combination of Naive Bayes reasoning with the well-known EM (Expectation Maximization) algorithm. PLSA views a document collection as a Naive Bayes network of variables: terms and documents, which are all dependent on a single, hidden (latent) topic variable. That is, given the knowledge of the topic, the probabilities of occurrence of topics are independent of each another. Similarly, given a topic, the probability, that a document belongs to this topic, does not depend on the probability of some other document belonging to this topic. The PLSA consists in finding the most probable distribution of topics, given a known distribution of terms in documents. The EM process in the PLSA algorithm consists - after a random initialization - in estimating cyclically the probability distribution of hidden topics given the term/document distribution, and then the conditional probabilities of terms and documents given the hidden variable distribution.

Our modification consist in assigning document to exactly one - most probable - topic. Furthermore, we do not assume that terms are independent, given the topic, but rather that there exists a Bayesian network of terms, given the topic (the BN is obtained via the discussed ETC algorithm, see section 3.2). In this way we take into account the known fact of the dependencies existing between terms.

To sum up, we have made a small but important change: for documents we do not calculate the degree of membership, but instead assign a document to the most probable class. After selection of natural clusters in our documents collection, the algorithm selects the description of each cluster. To describe clusters, it uses the terms from the search engine's dictionary.

A Naive Bayes classifier assumes that "events" (in our case documents) are described by a set of attributes (in our case terms) $A_1, A_2, ..., A_n$, and that these attributes have nominal values (in our case "present" or "absent") and are (statistically) independent of each other. Independency of attributes in our case means that a document contains terms without any order, and the presence or absence of each attribute depends on the cluster only. These assumptions simplify grouping of huge collection of documents. We need to remember that a dictionary which contains 100 000 terms is nothing unusual, and that each term is related to one attribute.

Given a test set D of documents, we estimate the probabilities of the individual values of all attributes for each decision concept $c \in C$ (in our case the cluster to which the document belongs). We need to estimate the a-priori probability of each concept too. Let us assume that there is a document $d \in D$ in the collection D of documents, and c is a concept identifier. The hypothesis $h(d)$ of document's d class membership is derived from probability distributions evaluated by Naive Bayes classifier as follows:

$$h(d) = argmax_{c \in C} \, P(c) \prod_{i=1}^{n} P(A_i(d)|c) \tag{6.4}$$

The number of clusters K in the algorithm is a parameter which can be set to any positive number. However, on this stage we have decided to group documents into 4 clusters for quadratic map cells and 6 clusters for hexagonal cells. This number of clusters results in a natural map initialization (see next section, also [13]). After random initialization of clusters, there is time for estimation of probabilities $P(c)$ that is, for each document d we compute how probable its membership to group c is; and $P(a_i|c)$ that is, how probable it is that the attribute A_i takes the value a_i for the object x, given the fact that the document d belongs to group c.

Next, we assign a new cluster for each document using Equation (6.4). This process is iterated: after each evaluation of a new cluster for all documents we re-calculate the probabilities $P(c)$and $P(a_i|c)$ for the whole document collection and so on. The number of iteration is a parameter but for the initial experiments was set to 10, as we found it satisfactory in terms of convergence for the data sets we used.

For the clustering phase each document is represented by a bit different vector than used at earlier stages. This vector has length equal to the size of the dictionary. Each value in the vector corresponds to a particular term in the dictionary. If a given term is present in the document, then the term value is set to 1; if it is not - the value is 0. For the initial clustering of documents, we do not take into consideration the frequency of the terms. This allows us to calculate only the following probabilities for each term $P(A_i = 1|c)$ and $P(A_i = 0|c)$.

After creation of clusters we need to find the terms that describe the created groups. To achieve this goal, we have implemented the following algorithm. For each group, we select a set of terms t_i that maximizes "representativeness" τ within the group c.

$$\tau_{i,c} = \frac{\log(\sum_j N_{ij,c}) \cdot avg_j(N_{ij,c})}{\sqrt{stddev_j(N_{ij,c})}} \cdot \frac{|D_{t_i,c}|}{|D_c|} \tag{6.5}$$

where $N_{ij} = Freq(t_i, d_j)$ is the number of occurrences of term t_i in document d_j. The index g denotes the terms that occur in considered group. The abbreviation *mean* - denotes arithmetic mean, *stddev* - standard deviation, $|D_{t_i,c}|$ - the number of documents that contain the term t_i, and $|D_c|$ - the number of documents in the considered group c.

The idea behind this "representativeness" measure is to promote terms that occur in a larger number of documents, occur frequently (with logarithmic order), and occur uniformly in the documents in a given group.

The computed value of $\tau_{i,c}$ is the basis for inclusion of term t_i into the set of descriptors of group c. To be included, the τ measure must exceed a predefined threshold. Furthermore, it can be included into the descriptor set of one group only, that is of that c, for which it reaches the highest value.

6.4 Fuzzy Initialization of Broad Topics on the Map

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

Therefore, we needed to create a clever initialization algorithm.

We have proposed the following topic-based initialization method:

1. using PLSA [32] (see previous sectionfor the modifications we applied), select K (our choice was $K = 4$) main topics in the given document set

2. select K map cells as the fixpoints for the individual topics. The fixpoints are cells evenly spread over a centrally placed circle whose radius is about $2/3$ of the map dimension. The radius is

chosen so that the internal and external distances (on a torus surface) between the opposite fixpoints be equal.

3. initialize selected fixpoints with K main topics

4. initialize all the remaining cells reference vectors using the following rule:

$$v_j = \frac{\sum_{i=1}^{K} d(c_j, c_i) \cdot v_i}{\sum_{i=1}^{K} d(c_j, c_i)} + \varepsilon_j$$

where $d(c_j, c_i)$ is the Euclidean distance on torus surface between given cell and i-th fixpoint (represented by vector v_i) and ε_j is a small, random disturbance vector.

Our experiments showed that maps initialized in this way are not only more comprehensible and stable - they are also easier and faster to learn. One example can be seen in Figure 6.5. We can see, for example, the closeness of the concepts of sleep disorders and general health conditions, or pregnancy and birth. There is, of course, no *a priori* knowledge encapsulated in the algorithm, but rather a reflection of the fact that people talking (writing) on both the subjects use a particular vocabulary more frequently than on average. There are, of course, many ill-defined clusters, but this should not be a surprise for news-groups data.

Since global similarities are determined during the initialization phase, only local relations have to be learned. Consequently, such a map can be learned with a smaller learning kernel radius and a lower value of the learning coefficient alpha. Moreover, in case of hierarchical maps, such an initialization procedure can be repeated iteratively on each level of the map hierarchy.

6.5 Growing Neural Gas for Document Clustering

Another possibility of extracting topical groups is offered by the Growing Neural Gas (GNG) networks proposed in [26]. Like Kohonen's

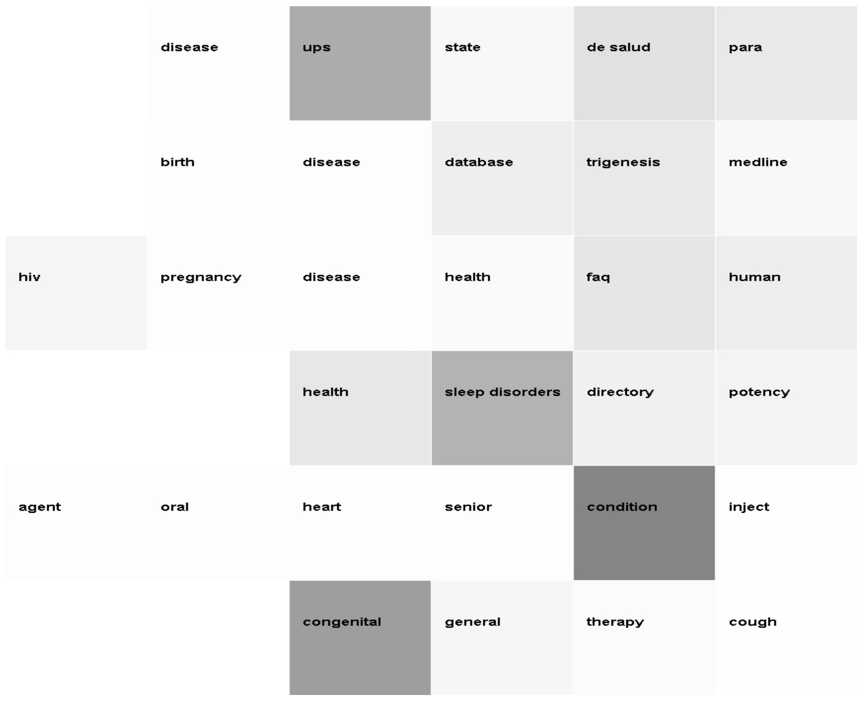


Figure 6.5: Themes identified by WebSOM

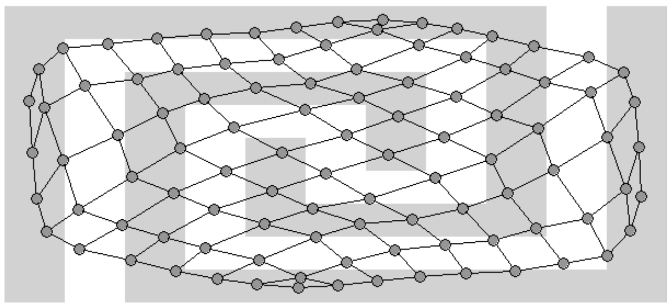


Figure 6.6: SOM map for a data distribution which is uniform in the shaded area (output of the DemoGNG applet [48])

(SOM) networks, the GNG can be viewed as a topology learning algorithm. More precisely, its aim can be summarized as follows: Given some collection of high-dimensional data, find a topological structure which closely reflects the topology of the collection. In a typical SOM, the number of units and the topology of the map are predefined. As observed in [26], the choice of the SOM structure is difficult, and the need to define a decay schedule for various features is problematic. A typical SOM produced by Kohonen's algorithm is shown on Figure 6.6. On the contrary, a GNG network does not require specification of the network size and the resulting network adapts very well to a given data topology - see Figure 6.7. If we treat a single GNG graph node as a cluster of data, then the whole network can be viewed as a meta-clustering structure, where similar groups are linked together by graph edges.

GNG starts with very few units[5] , and new ones are inserted successively every k iterations. To determine where to insert new units, local error measures are gathered during the adaptation process; a new unit is inserted near the unit which has accumulated the maximal error. Interestingly, GNG cells of the network are joined automatically by links, which results in a possibly disconnected graph, the connected components of which can be treated as different data clusters. Fig-

[5]the initial nodes referential vectors are initialized with our broad topic initialization method [39]. We briefly describe it in section 7.5.

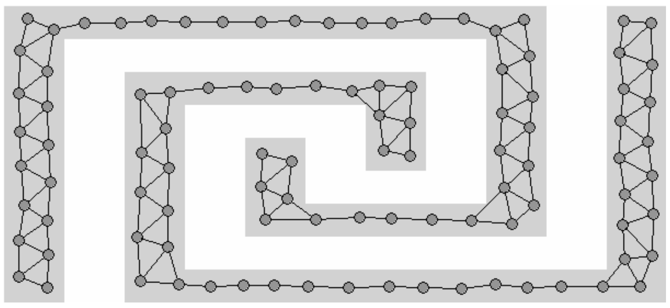


Figure 6.7: GNG network for a data distribution which is uniform in the shaded area (output of the DemoGNG applet [48])

ures 6.8 and 6.9 compare SOM and GNG networks for the well-known Wisconsin Breast Cancer set, consisting of 683 nine-dimensional data points representing two different cancer classes: malignant, and benign.

Both algorithms discover regions typical for each diagnostic class, and present regions where both the classes overlap. The complete GNG algorithm specification and its comparison to many other competitive soft methods can be found in [27] or [26].

6.6 Topical Groups Extraction

The SOM algorithm can be generally viewed as a clustering algorithm: map cells serve as a kind of containers for groups of similar documents. But the idea of WebSOM represents still something more. Not only the similarity within and dissimilarity outside a cell plays a role, but also the issue of similarity to the neighboring cells on the map. For this reason, also inter-cellar clusters may occur on the map of documents, making some groups of cells appear separated from the other ones. These clusters of cells will be called *topical groups* here, such a cluster is expected to contain documents concerned with some (more or less broad) topic. Recall that the number of cells, and hence the respective cellular clusters of documents may be quite large (thousands), so that in order to understand the whole collection of documents, a human

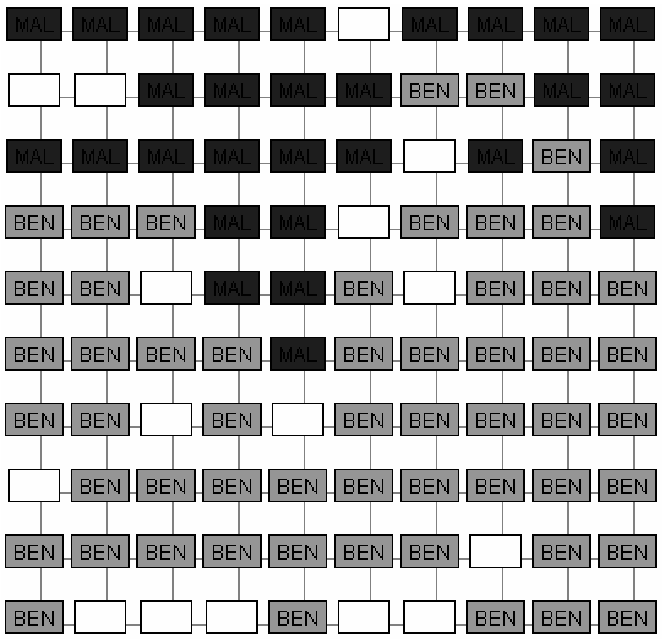

MAL MAL MAL MAL MAL MAL MAL MAL MAL
MAL MAL MAL MAL BEN BEN MAL MAL
MAL MAL MAL MAL MAL MAL MAL BEN MAL
BEN BEN BEN MAL MAL BEN BEN BEN MAL
BEN BEN MAL MAL BEN BEN BEN BEN
BEN BEN BEN BEN MAL BEN BEN BEN BEN BEN
BEN BEN BEN BEN BEN BEN BEN BEN
BEN BEN BEN BEN BEN BEN BEN BEN BEN
BEN BEN BEN BEN BEN BEN BEN BEN BEN
BEN BEN BEN BEN BEN

Figure 6.8: SOM map for the Wisconsin Breast Cancer data set

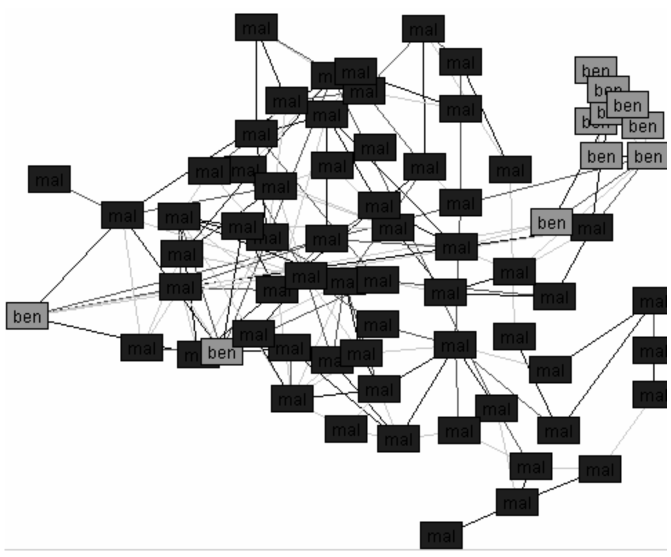


Figure 6.9: GNG network for the Wisconsin Breast Cancer data set

being, would not try to in identify the subject of each map cell, but he would rather look first at the content of the larger intercellular clusters. So, once the two-dimensional map has been generated (i.e. the generation process has converged, with respect to the 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.

Identification of such broad topics in document collections drew attention of a number of researchers. For instance, the issue was approached by the so-called PLSA [32], a method based on the assumption that there exists a 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 the term-document frequency matrix $N_{ij} = Freq(t_i, d_j)$ or, alternatively, by maximizing log-likelihood function:

$$\begin{aligned} lnL(z) &= \sum_{i,j} N_{ij} \cdot log\left(P(t_i \mid d_j)\right) \\ &= \sum_{i,j} N_{ij} \cdot log\left(\sum_k P(t_i \mid z_k) \cdot P(z_k \mid d_j)\right) \end{aligned} \tag{6.6}$$

As an alternative to the word-document representation we can use WebSOM to build a model of citation patterns. In a such model, a document is represented as a sparse vector whose i-th component represents the path length *from* [via outgoing links] or *to* [via incoming links] the i-th document in a given collection. It is also possible to estimate a joint term-citation model [17, 18].

In our own investigations we have applied algorithms for extraction of fuzzy clusters. Numerous methods were drawn up in the past, to mention only [64], [24], [4] or [58]. Unfortunately, none of them appeared to be directly applicable to free-text document maps, mainly due to extremely fuzzy structure of SOM clusters (very smooth similarity structure of topical groups).

Fuzzy-ISODATA algorithm [4] (called also Fuzzy C-Means, or FCM) is based on a gradual approach to the nearest (with respect to the fuzzy similarity measure) attractor and dynamic modification of similarity weights. The goal is to maximize the fuzzy partition quality measure:

$$J_m\left(\{\mu_i\},\{\bar{v_k}\}\right) = \sum_{k=1}^{|V|}\sum_{i=1}^{c}\mu_i^m(k)\cdot d_i^2(k) \tag{6.7}$$

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

Though numerous methods were drawn up in the past, none of them appeared to be directly applicable to free-text document maps. We have 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 the similarity measure in the 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 the cluster formation process

Fuzzy-ISODATA [4] is based on a gradual approach to the nearest (with respect to the fuzzy similarity measure) attractor and dynamic modification of similarity weights. A priori estimation of the number of topical clusters (c) in a given document set is usually a non-trivial issue. Consequently, a family of graph-theoretical clustering algorithms based on the minimal spanning tree (MST) construction has been proposed [64]. We have adapted an efficient implementation of the Prim algorithm on Fibonacci heaps to build the MST for a lattice (a graph connecting adjacent cells) of reference vectors, in which the edge weights represent the similarities between the corresponding vectors. In the second step, the edges whose weights are significantly higher [6] than the adjacent ones are assumed to be fuzzy cluster borders and are removed from the MST. The resulting connected graph components represent clusters.

Unfortunately, the MST on the original map reference vectors reveals the previously mentioned problem; in particular, the weight deviations the among neighboring edges are not significant. For this reason, we have applied a two-stage approach: in the first step, the Fuzzy-ISODATA algorithm is performed on the reference vectors, of the original map using the upper-bound estimate of the number of clusters (for visualization purposes, it is sufficient to set it to 10-15).

Next, cluster centroids (cluster reference vectors) are computed as weighted average:

$$\bar{v_c} = \frac{\sum_{k=1}^{|N_c|} Freq(cell_k) \cdot v_k}{\sum_{k=1}^{|N_c|} Freq(cell_k)} \tag{6.8}$$

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

Finally, the MST is built on the lattice of cluster centroids.

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

$$Fw(t_i, cell_k) = Freq(t_i, cell_k) \cdot w(t_i, cell_k) \tag{6.9}$$

[6]E.g., $w_e > avg(w_E) + k \cdot std(w_E)$, where E is the set of neighboring edges, k is a small positive constant, avg is the average edge weight and std is the standard deviation of weights in the vicinity

$$w(t_i, N_c) = \frac{\sum_{k=1}^{|N_c|} Fw(t_i, cell_k)}{\sum_{k=1}^{|N_c|} \left(\frac{Fw(t_i,cell_k)}{\sum_{k=1}^{|N_c|} Fw(t_i,cell_k)} \cdot \left(1 + log \left(1 + \frac{Fw(t_i,cell_k)}{\sum_{k=1}^{|N_c|} Fw(t_i,cell_k)} \right) \right) \right)} \tag{6.10}$$

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 a measure based on inter-cluster term frequencies, average weights and standard deviations, for instance, in a manner similar to the previously mentioned macro-averaging of reference vectors:

$$w(t_i, N_c) = \frac{avg^2_{cell_k \in N_c} \left(Fw(t_i, cell_k) \right)}{log \left(std_{cell_k \in N_c} \left(Fw(t_i, cell_k) \right) \right)} \tag{6.11}$$

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

6.7 Map Quality

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. The initial results seem to be encouraging, though it is still too early to present a complete report.

Chapter 7

Incremental Document Clustering

In this chapter we describe how we explore the clustering properties of growing neural gas and artificial immune networks to accommodate new documents into the current clustering framework.

Clustering and content labeling are the crucial issues for the user's understanding of the two-dimensional map. We started our research with the WEBSOM approach, which appeared to be unsatisfactory: both speed and clustering stability were not very encouraging.

One of the main goals of the BEATCA project was to create multidimensional document maps in which geometrical vicinity would reflect conceptual closeness of documents in a given document set.

In our approach, objects (text documents, as well as the graph nodes, described below) are represented in the standard way, i.e. as vectors of the dimension equal to the number of distinct dictionary terms. A single element of a so-called *referential vector* represents importance of the corresponding term and is calculated on the basis of the $tfidf$ measure (6.3) or our own context-sensitive w_{tdG} measure, which will be described later. The similarity measure is defined as the cosine of the angle between the corresponding vectors.

7.1 The Referential Approach - WebSOM

From the purely conceptual point of view, the SOM approach to document clustering (WebSOM) is incremental, as documents are presented one by one to the learning mechanism.

In Section 6.1 we have already described the SOM clustering algorithm. We have pointed out subsequently various shortcomings of that method, and suggested ways of overcoming them, e.g. the issue of slow winner search (Section 6.2) or non-reproducable initialization (section 6.3), too rigid structure (Section 6.5).

However, from the perspective of an incremental approach to map construction (which is essential for long-term time complexity reduction), further important deficiencies of SOM need to be mentioned and overcome (cf. [2]): (a) SOM is order dependent, i.e. the components of the final weight vectors are affected by the order in which training examples are presented, (b) the components of these vectors may be severely affected by noise and outliers, (c) the grid size, the step size and the neighborhood size must be tuned individually for each data-set to achieve useful results.

7.2 GNG Extension with the Utility factor

A typical problem in web mining applications is that the data being processed is constantly changing - some documents disappear or become obsolete, while other enter the analysis. All this requires models which are able to adapt their structure quickly in response to non-stationary distribution changes. Thus, we decided to adopt and implement the GNG with the utility factor model [28]. We have already characterized the basic GNG approach in Section 6.5, and will now concentrate on the utility factor extension.

A crucial concept here is to identify the least useful nodes and to remove them from the GNG network, enabling further node insertions in regions where they would be more necessary. The utility factor of each node reflects its contribution to the total classification error reduction. In other words, a node's utility is proportional to the expected error growth if that node were removed. There are many possible choices for the utility factor. In our implementation, the utility update rule of the winning node has been simply defined as $U_s = U_s + error_t - error_s$, where s is the index of the winning node, and t is the index of the second-best node (the one which would become the winner if the actual winning node did not exist). The utility of a newly inserted node is arbitrarily initialized to the mean of the two nodes which have accumulated most of the error: $U_r = (U_u + U_v)/2$.

After the utility update phase, the node k with the smallest utility is removed if the fraction $error_j / U_k$ is greater than some predefined threshold, where j is the node with the greatest accumulated error.

7.3 Robust Winner Search in GNG Network

Similarly to Kohonen's algorithm, the most computationally demanding part of the GNG algorithm is the winner search phase. Especially in application to web documents, where both the text corpus size and the number GNG network nodes is huge, the cost of even a single global winner search phase is prohibitive.

Unfortunately, neither the local-winner search method (i.e. searching through the graph edges from some staring node) nor the joint-winner search method (our own approach to SOM learning described in section 6.2, see also [39]) are directly applicable to the GNG networks. The main reason for this is that a graph of GNG nodes can be unconnected. Thus, the standard local-winner search approach would prevent document from shifting between the separated components during the learning process.

A simple modification consist in remembering the winning node for more than one connected component of the GNG graph[1] and conducting in parallel a single local-winner search thread for each component.

[1]In our experiments, two winners turned to be sufficient to overcome the problem of components separation

Obviously, this requires periodical (precisely, once for an iteration) recalculation of connected components, but this is not very expensive[2].

A special case is the possibility of a node removal. When the previous iteration's winning node for a particular document has been removed, we activate search processes (in parallel threads) from each of its direct neighbors in the graph.

We have implemented another method, a little more complex (both in terms of computation time and memory requirements) but, as the experiments show, more accurate. It exploits the well-known Clustering Feature Tree [65] to group similar nodes in dense clusters. The node clusters are arranged in a hierarchy, and stored in a balanced search tree. Thus, finding the closest (most similar) node for a document requires $O(log_t V)$ comparisons, where V is the number of nodes and t is the tree branching factor (refer to [65]). The amortized tree structure maintenance cost (node insertion and removal) is also proportional to $O(log_t V)$.

7.4 Artificial Immune Systems

An immune algorithm is able to generate the reference vectors (called antibodies) each of which summarizes basic properties of a small group of documents treated here as antigens[3] . This way the clusters in the immune network spanned over the set of antibodies will serve as internal images, responsible for mapping existing clusters in the document collection into network clusters. In essence, this approach can be viewed as a successful instance of exemplar-based learning giving an answer to the question "what examples to store for use during generalization, in order to avoid excessive storage and time complexity, and possibly to improve generalization accuracy by avoiding noise and overfitting", [59].

[2]in order of $O(V + E)$, where V is the number of nodes and E is the number of connections (graph edges)

[3]Intuitively by antigens we understand any substance threatening proper functioning of the host organism while antibodies are protein molecules produced to bind antigens. A detailed description of these concepts can be found in [8].

7.4.1 The aiNet Algorithm for Data Clustering

The artificial immune system aiNet [7] mimics the processes of clonal selection, maturation and apoptosis[4] observed in the natural immune system. Its aim is to produce a set of antibodies binding a given set of antigens (i.e., documents). The efficient antibodies form a kind of immune memory capable to bind new antigens sufficiently similar to these from the training set.

Like in SOM, the antigens are repeatedly presented to the memory cells (being matured antibodies) until a termination criterion is satisfied. More precisely, a memory structure consisting of matured antibodies, M, is initiated randomly with a few cells[5]. When an antigen ag_i is presented to the system, its affinity $\mathit{aff}(ag_i, ab_j)$ to all the memory cells is computed. The value of $\mathit{aff}(ag_i, ab_j)$ expresses how strongly the antibody ab_j binds the antigen ag_i. From a practical point of view the $\mathit{aff}(ag_i, ab_j)$ can be treated as the degree of similarity between these two cells[6]. The greater the affinity $\mathit{aff}(ag_i, ab_j)$, the more stimulated ab_j is.

The idea of clonal selection and maturation translates into the next steps (here σ_d, and σ_s are parameters). The cells which are most stimulated by the antigen are subjected to clonal selection (i.e., each cell produces a number of copies proportional to the degree of its stimulation), and each clone is subjected to mutation (the intensity of mutation is inversely proportional to the degree of stimulation of the mother cell). Only the clones cl which can cope successfully with the antigen (i.e., such that $\mathit{aff}(ag_i, ab_j) > \sigma_d$) survive. They are added to a tentative memory, M_t, and the process of clonal suppression starts: an antibody ab_j too similar to another antibody ab_k (i.e., the $\mathit{aff}(ab_j, ab_k) > \sigma_s$)) is removed from M_t. The remaining cells are added to the global memory M.

1. select n most excited antibodies,

[4] Consult [8] for description of these notions.

[5] In section 7.5 another initialization is proposed.

[6] In practical applications this measure can be derived from any metric dissimilarity measure $dist$ as $\mathit{aff}(ag_i, ab_j) = \frac{d_{max} - dist(ag_i, ab_j)}{d_{max}}$, where d_{max} stands for the maximal dissimilarity between two cells

2. clone these cells (the most stimulated cell the more clones it produces),
3. mutate clones (the more stimulated the cell the more careful modification of its components),
4. compute the affinity of ag_i to the mutated clones
5. select $\xi\%$ of highly stimulated mutants and add them to temporary memory structure M_t
6. eliminate those $ab_j \in M_t$ for which $aff(ag_i, ab_j) < \sigma_d$, obtaining a reduction in the size of the M_t
7. calculate the network affinity $aff(ab_j, ab_k)$ for all $ab_j, ab_k \in M_t$
8. (clonal suppression) eliminate too similar antibodies (i.e., if $aff(ab_j, ab_k) > \sigma_s$, one of the respective cells is removed from M_t)
9. add the remaining cells to the global memory M

The n most stimulated antibodies produce a number of clones (proportional to the degree of their stimulation). These clones are subjected to a directed mutation carried out in accordance with the Equation (6.1). However, now the "learning rate" α is not a function of time, but is inversely proportional to $\mathit{aff}(ag_i, ab_j)$. Next, each mutant computes its affinity to the antigen ag_i and $\xi\%$ of highly stimulated mutants are added to a temporary memory structure M_t. The cells in M_t compete for survival, i.e. (a) the cells weakly responding to the antigen, and (b) the cells which are too similar to another, are removed from M_t. Two control parameters, σ_d and σ_s, are used to refine conditions (a) and (b). The remaining cells are added to the global memory.

These steps are repeated until all antigens are presented to the system. Next, the degree of affinity between all pairs $ab_j, ab_k \in M$ is computed and again overly similar (in fact: redundant) cells are removed from the memory. This step represents network suppression of the immune cells. Lastly, $r\%$ (one more parameter) worst individuals in M are replaced by freshly generated cells. This ends one epoch, and the next epoch begins until the termination condition is met.

Step 3 of the algorithm is carried out according to rule resembling Equation (6.1). The only difference is that the "learning rate" α is not a function of time; instead, it is inversely proportional to $aff(ag_i, ab_j)$.

Among all the parameters mentioned above, the crucial one seems to be σ_s, as it critically influences the size of the global memory. Careful examination of the network suppression step allows to observe that when $\sigma_s \to 0$, only clones identical with the presented antigen can survive, and the algorithm reduces to the well-known Leaders algorithm – cf. [31], Ch. 3. Each memory cell can be viewed as an exemplar which summarizes important features of the "bundles" of antigens stimulating.

7.4.2 Identification of Redundant Antibodies

The clonal suppression stage requires $|M_t| \cdot (|M_t| - 1)/2$ calculations of the affinity (in fact: distance) between all the pairs of cells in M_t. To reduce the time complexity of this step, we refer to the agglomerative clustering approach. The crucial concept here is to manage the matrix of distances in a smart way, and to update only those distances which have really changed after merging of two clusters. Among many possible solutions, we have applied the so-called partial similarity matrix and update algorithm presented in [34]. The authors have shown that the expected complexity of a single-step update is of order $O(2c\dot{N} \cdot C \cdot k)$, where N is the number of objects, C is the maximum number of clusters, $k << C$ is the number of maximal column rescanning [7]. This is significantly less than the $O(N^3)$ complexity of a naive approach. Finally, the reduced antibodies are replaced by a single cell being the center of gravity of the set of removed antibodies. Thus, we not only reduce the size of the immune network, but presumably

[7]That means that the number k of columns in the similarity matrix that need to be updated in a single clustering step is significantly lower than the maximum number C of clusters.

compress the information contained in the set of specialized antibodies to the new, universal antibody.

7.4.3 Robust Construction of Mutated Antibodies

In case of high-dimensional data, such as text data represented in a vector space, calculation of the stimulation level is quite costly (proportional to the number of different terms in the dictionary). Thus, the complexity of an immune algorithm might be significantly reduced if we can restrict the number of required expensive recalculations of the stimulation level. The direct, high-dimensional calculations can be replaced by operations on scalar values on the basis of the simple geometrical observation that stimulation of a mutated antibody clone can be expressed in terms of the original antibody stimulation.

Such optimization is based on a generalized Pythagoras theorem: if v_1, v_2, v_3 are vectors forming a triangle ($v_1 + v_2 + v_3 = 0$), then

$$|v_3|^2 = |v_1|^2 + |v_2|^2 - 2|v_1||v_2|cos(v_1, v_2) \tag{7.1}$$

If also $v_1 \perp v_2$, we get the well-known equation: $|v_3|^2 = |v_1|^2 + |v_2|^2$. We can define a mutated clone as: $m = \kappa d + (1 - \kappa)c$, where m is mutated clone, c is cloned antibody, d is antigen (document) and κ is the (random) mutation level.

Taking advantage of Equations (6.1) and (7.1) (where $v_1 := d' = \kappa \cdot d$, $v_2 := c' = (1 - \kappa) \cdot c$, $v_3 := -m$) and having calculated the original antibody stimulation $aff(c, d)$, we can calculate mutated clone stimulation level $aff(m, d)$ as follows: $P = cos(c', d') = cos(c, d) = 1 - aff(c, d)$, $|m|^2 = |d'|^2 + |c'|^2 + 2P|c'||d'| = \kappa^2|d|^2 + (1 - \kappa)^2|c|^2 + 2\kappa(1 - \kappa)P|c||d|$, $s = \kappa|d|^2 + (1 - \kappa)|c||d| = \kappa|d|^2 + (1 - \kappa)P|c||d|$, and finally

$$aff(m, d) = \frac{s}{|m| \cdot |d|} \tag{7.2}$$

Dually, we can find a mutation threshold κ such that the mutated antibody clone stimulation $aff(m, d) < \sigma_d$. More precisely, we are looking for κ_0 such that $aff(m, d) = \sigma_d$, which in turn can be used to create a mutated antibody for a random mutation level $\kappa \in (0, \kappa_0)$. The advantage of such an approach is the reduction in the number of inefficient (too specific) antibodies, which would be created and immediately removed from the clonal memory. Analogously to the previous inference, if we define $p = aff(c,d)$, $x = -p|d| + p^2|c| + \sigma_d^2(p|d| - c)$, $y = |d|^2 - 2p|c||d| + p^2|c|^2 - \sigma_d^2(|d|^2 - |c|^2 + 2p|c||d|)$ and $z = \sigma_d \cdot |d|\sqrt{(p^2-1)\cdot(\sigma_d^2 - 1)}$, then

$$\kappa_0 = \frac{|c| \cdot (x+z)}{y} \tag{7.3}$$

7.4.4 Stabilization via Time-dependent Parameters

A typical problem with the immune paradigm based algorithms is the stabilization of the cardinality of the memory cells set. Majority of algorithms are also quite sensitive to the choice of the input parameters. The detailed analysis presented in [58] shows that in case of aiNet the crucial factor is the choice of the suppression threshold σ_s and the number of antibodies to be cloned m_b. Our experiments showed that the correct choice of the death threshold σ_d can also significantly influence the final model. To stabilize the network, we decided to treat its parameters, like σ_s, σ_d and m_b as functions of time (i.e. $\sigma_s(t)$, $\sigma_d(t)$ and $m_b(t)$). Let ρ be any of these parameters and let ρ_0 be the initial value of ρ and ρ_1 be the final value of ρ. So the function $\rho(t)$ has to be defined as follows: If T is the number of iterations through which we want to proceed, then obviously we have to ensure that $\rho(0) = \rho_0$ and $\rho(T) = \rho_1$.

In particular, in our system, both $\sigma_s(t)$ and $\sigma_d(t)$ are reciprocally increased, while $m_b(t)$ is linearly decreased with time. So to ensure the marginal conditions at $t = 0$ and $t = T$ the time dependence has the form:

$$\sigma_i(t) = \sigma_{i_0} + (\sigma_{i_1} - \sigma_{i_0}) \cdot \frac{t \cdot (T+1)}{T \cdot (t+1)} \tag{7.4}$$

$$m_b(t) = m_0 + \frac{m_1 - m_0}{T} \cdot t \tag{7.5}$$

where i stands for either s or d, $\sigma_{s_0} = 0.05$, $\sigma_{s_1} = 0.25$ for $\sigma_s(t)$; $\sigma_{d_0} = 0.1$, $\sigma_{d_1} = 0.4$ for $\sigma_d(t)$; $m_0 = 3$, $m_1 = 1$ for $m_b(t)$.

7.4.5 Robust Antibody Search in an Immune Network

One of the most computationally demanding part of any AIS algorithm is the search for the best fitted (most stimulated) antibodies. In particular, in application to web documents, where both the text corpus size and the number of idiotypic network cells is huge, the cost of even a single global search phase in the network is prohibitive.

We propose to replace the global search approach with a modified local search (i.e., searching for current iteration's most stimulated antibody through the graph edges of the idiotypic network, starting from the last iteration's most stimulated antibody). The modification relies upon remembering the most stimulated cell for more than one connected component of the idiotypic network, and to conduct in parallel a single local-winner search thread for each component. Obviously, this requires one-for-iteration recalculation of connected components, but this is not very expensive: the complexity of this process is of order $O(V + E)$, where V is the number of cells and E is the number of connections (graph edges).

A special case is the possibility of an antibody removal. When the previous iteration's most stimulated antibody for a particular document (antigen) has been removed from the system's memory, we activate search processes (in parallel threads) from each of its immediate neighbors in the graph.

We have developed another, slightly more complicated, but, as the experiments show, more accurate method. It exploits the well-known Clustering Feature Tree (CF-Tree, [65]) to group similar network cells in dense clusters. Antibody clusters are arranged in a hierarchy and stored in a balanced search tree. Thus, finding the most stimulated (similar) antibody for a document requires $O(log_t V)$ comparisons, where t is the tree branching factor (refer to [65] for details). The amortized tree structure maintenance cost (insertion and removal) is also proportional to $O(log_t V)$.

7.5 Adaptive Visualization of the Model

Despite their many advantages over SOM approach, both GNG and AIS have one serious drawback: high-dimensional networks cannot be easily visualized. However, we can build Kohonen map on the referential vectors of a GNG or AIS network, similarly to the case of single documents, i.e., treating each vector as a centroid representing a cluster of documents.

To obtain the visualization that singles out the main topics in the corpus and reflects the conceptual closeness between topics, the proper initialization of SOM cells is required. We have developed a special initialization method aimed at identifying broad topics in the text corpus. Briefly, in the first step we find the centroids of a few main clusters (via the fast ETC Bayesian tree [37] and SVD eigenvectors decomposition [21]). Then, we select *fixpoint cells*, uniformly spread them on the map surface, and initialize them with the centroid vectors. Finally, we initialize the remaining map cells with *intermediate* topics, calculated as the weighted average of main topics, with the weight proportional to the Euclidean distance from the corresponding fixpoint cells.

After initialization, the map is learned with the standard Kohonen algorithm [43]. Finally, we adopt the so-called *plastic clustering* algorithm [8] to adjust the position of GNG/AIS model nodes on the SOM projection map, so that the distance on the map reflects the similarity of the adjacent nodes as closely as possible. Here the topical initialization of the map is crucial for assuring the stability of the final visualization [39].

The resulting map is a visualization of the GNG/AIS network with the detail level depending on the SOM size (a single SOM cell can gather more than one GNG/AIS node). The user can access the document content via the corresponding GNG/AIS node, which in turn

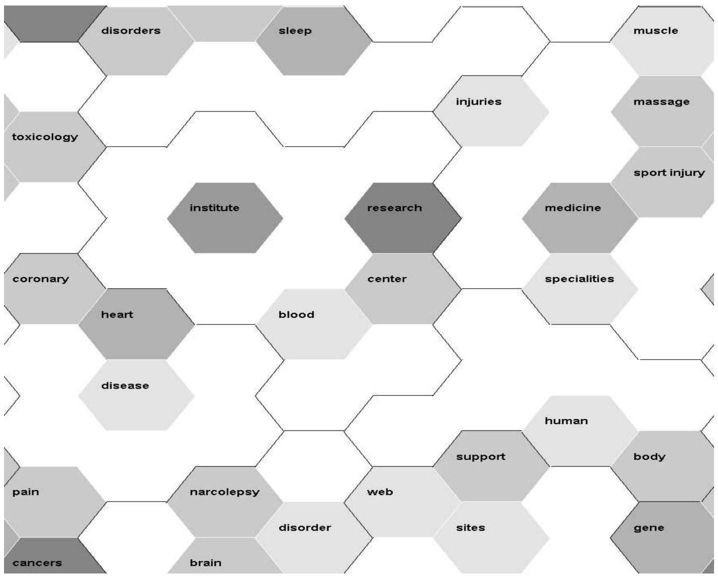


Figure 7.1: Example of a GNG model visualization

can be accessed via a SOM node - the interface here is similar to the hierarchical SOM map case.

An exemplary map can be seen in Figure 7.1. The color brightness is related to the number of documents contained in the cell. Each cell which contains at least one document is labeled with a few descriptive terms (only one is visible here, the rest are available via the BEATCA search engine). The black lines represents borders of topical areas[8]. It is important to stress that this planar representation is in fact a torus surface (which can also be visualized in 3D), so the cells on the map borders are adjacent.

After initialization, the map is learned with the standard Kohonen algorithm [43]. Finally, we adopt the attraction-repelling algorithm [57] to adjust the position of AIS antibodies on the SOM projection map, so that the distance on the map reflects the similarity of the adjacent cells as closely as possible. The topical initialization of the

[8]The clustering of map nodes, based on the combination of Fuzzy C-Means algorithm and the minimal spanning tree technique is described in Section 6.6

map is crucial here to assure the stability of the final visualization [39]. The resulting map visualizes the AIS network with resolution depending on the SOM size (a single SOM cell can gather more than one AIS antibody).

7.6 Contextual Maps

In our work we use the well known approach which consists in representing documents as points in the term vector space. It is a known phenomenon that text documents are not uniformly distributed over that space. The characteristics of frequency distributions of individual terms depend strongly on document location. On the basis of experimental results presented in the previous section, we suggest automatical identification of groups containing similar documents via a preprocessing step in document maps formation. We argue that after splitting documents in such groups, term frequency distributions within each group become much easier to analyze. In particular, it appears to be much easier to select significant and insignificant terms for efficient calculation of similarity measures during the map formation step. Such document clusters are called *contextual* groups. For each contextual group, separate maps are generated. To obtain more informative maps, one needs to balance the size of each cluster (during initial contextual clustering). The number of documents presented on the map cannot be too high, because the number of cells in the graph (and the time required to create a map) would then to grow adequately. On the other hand, a single map model should not hold only a few irrelevant documents.

The constraints on the cluster size are obtained by recurrent divisions and merges of fuzzy document groups, created by a selected algorithm (e.g., EM combined with ETC or Chow-Liu Bayesian net, SVD, Fuzzy C-Means). In the case of Fuzzy-ISODATA algorithm, there is an additional modification in the optimized quality criterion, which penalizes imbalanced splits (in terms of cluster size).

In the first step, the whole document set is split into a few (2-5) groups. Next, each of these groups is recursively divided until the number of documents inside each group meets the required criteria. After such a process, we obtain a hierarchy represented by a tree of clusters. In the last phase, the groups which are smaller than a predefined constraint are merged with the closest group. The similarity measure is defined as a single-linkage cosine angle between both the clusters centroids.

The crucial phase of contextual document processing is the division of terms space (dictionary) into - possibly overlapping - subspaces. In this case, it is important to calculate the fuzzy membership level, which will represent the importance of a particular word or phrase in different contexts (and implicitly, the ambiguity of its meaning). The estimation of fuzzy within-group membership of the term m_{tG} is calulated as:

$$m_{tG} = \frac{\sum_{d \in G} (f_{td} \cdot m_{dG})}{f_G \cdot \sum_{d \in G} m_{dG}} \tag{7.6}$$

where f_G is the number of documents in the cluster G, m_{dG} is the degree of document d membership in group G, f_{td} is the number of occurrences of term t in document d.

Finally, the vector-space representation of a document is modified to take into account the document context. This representation increases the weights of the terms which are significant for a given contextual group, and decrease the weights of insignificant terms. In the boundary case, insignificant terms are ignored, which leads to reduction in the representation space dimensionality. To estimate the significance of a term in a given context, the following measure is applied:

$$w_{tdG} = f_{td} \cdot m_{tG} \cdot log\left(\frac{f_G}{f_t \cdot m_{tG}}\right) \tag{7.7}$$

where f_{td} is the number of occurrences of term t in document d, m_{tG} is the degree of membership of term t in group G, f_G is the number of documents in group G, f_t is the number of documents containing term t.

The main idea behind the proposed approach is to replace a single model (growing neural gas, an immunological net or hierarchical SOM maps) by a set of independently created contextual models, and to merge them together into a hierarchical model. The training data for each model is a single contextual group. Each document is represented as a standard referential vector in the term-document space. However, the $tfidf$ measure of vector components is replaced by w_{tdG}.

To visually represent the similarity relation between contexts (represented by a set of contextual models), an additional "global" map is required. Such a model becomes a root of contextual maps hierarchy. The main map is created in a manner similar to the previously created maps, with one distinction: an example in training data is a weighted centroid of referential vectors of the corresponding contextual model:

$$\overrightarrow{x_i} = \sum_{c \in M_i} (d_c \cdot \overrightarrow{v_c}) \tag{7.8}$$

Finally, cells and regions in the main map are labeled with keywords selected according to the following contextual term quality measure:

$$Q_{tG} = ln(1 + f_{tG}) \cdot (1 - |EN_{tG} - 0.5|) \tag{7.9}$$

where EN_{tG} denotes normalized entropy of the term frequency within the group.

The learning process of the contextual model is to some extent similar to the classic, non-contextual learning. However, it should be noted that each constituent model (and the corresponding contextual map) can be processed independently. In particular, it can be distributed and calculated in parallel. Also a partial incremental update of such models appears to be much easier to perform in terms of model quality, stability and time complexity. The possibility of incremental learning stems from the fact that the very nature of the learning process is iterative. So if new documents arrive, we can consider the learning process as having been stopped at some stage, and being resumed now with all the documents. We claim that it is not necessary to start the learning process from a scratch either in the case when the new documents "fit" the distribution of the previous ones or when their term distribution is significantly different. This claim is supported by the experimental results presented in Section 8.2.1. In Section 8.2.2, we present some remarks on the scalability issues of the contextual approach.

Chapter 8

Experimental Results

To evaluate the effectiveness of the overall incremental map formation process, we compared it to the map formation "from scratch". In this section, we describe the overall experimental design, the quality measures used and the results obtained.

The architecture of our system supports comparative studies of the clustering methods at the various stages of the process (i.e. initial document grouping, initial topic identification, incremental clustering, model projection and visualization, identification of topical areas in the map and its labeling). In particular, we have conducted series of experiments to compare the quality and stability of the GNG and SOM models for various model initialization methods, winner search methods and learning parameters [40]. In this chapter we only focus on evaluation of the GNG winner search method and the quality of the resulting incremental clustering model with respect to the topic-sensitive learning approach.

8.1 Quality Measures for Document Maps

Various measures of quality have been developed in the literature, covering diverse aspects of the clustering process. The clustering process is frequently referred to as "learning without a teacher", or "unsupervised learning", and is driven by some kind of similarity measure.

The term "unsupervised" does not completely reflect the real nature of learning. In fact, the similarity measure used is not something "natural", but reflects rather the intentions of the teacher. So we can say that clustering is a learning process with a hidden learning criterion. The criterion is intended to reflect some esthetic preferences, like: uniform split into groups (topological continuity) or appropriate split of documents with known a priori categorization. As the criterion is somehow hidden, we need tests to tell if the clustering process really fits the expectations. In particular, we have accommodated for our purposes and investigated the following well known quality measures of clustering [66, 6, 30]:

- **Average Map Quantization**: average cosine distance between each pair of adjacent nodes. The goal is to measure topological continuity of the model (the lower this value, the "smoother" the model):

  $$AvgMapQ = \frac{1}{|N|}\sum_{n \in N}\left(\frac{1}{|E(n)|}\sum_{m \in E(n)} c(n,m)\right) \tag{8.1}$$

  where N is the set of graph nodes, $E(n)$ is the set of nodes adjacent to node n, and $c(n,m)$ is the cosine distance between nodes n and m.

- **Average Document Quantization**: average distance (according to the cosine measure) for the learning set between the document and the node it has been classified into. The goal is to measure the quality of clustering at the level of a single node:

  $$AvgDocQ = \frac{1}{|N|}\sum_{n \in N}\left(\frac{1}{|D(n)|}\sum_{d \in D(n)} c(d,n)\right) \tag{8.2}$$

  where $D(n)$ is the set of documents assigned to node n.

Both measures take values in the interval [0, 1]; the lower values correspond to "smoother" inter-cluster transitions and more "compact" clusters. To some extent, optimization of one of the measures entails increase in the other one. Still, experiments reported in [40] show that the GNG models are much smoother than SOM maps while the clusters are of similar quality.

The two subsequent measures evaluate the agreement between the clustering and the a priori categorization of documents (i.e. a particular newsgroup in case of newsgroups messages).

- **Average Weighted Cluster Purity**: average "category purity" of a node (node weight is equal to its density, i.e. the number of assigned documents):

$$AvgPurity = \frac{1}{|D|} \sum_{n \in N} max_c \left(|D_c(n)| \right) \tag{8.3}$$

where D is the set of all documents in the corpus and $D_c(n)$ is the set of documents from category c assigned to the node n.

- **Normalized Mutual Information**: quotient of the total category and the total cluster entropy by the square root of the product of category and cluster entropies for the individual clusters:

$$NMI = \frac{\sum_{n \in N} \sum_{c \in C} |D_c(n)| \;\; log \left(\frac{|D_c(n)| \;\; |D|}{|D(n)| \;\; |D_c|} \right)}{\sqrt{\left(\sum_{n \in N} |D(n)| \;\; log \left(\frac{|D(n)|}{|D|} \right) \right) \left(\sum_{c \in C} |D_c| \;\; log \left(\frac{|D_c|}{|D|} \right) \right)}} \tag{8.4}$$

where N is the set of graph nodes, D is the set of all documents in the corpus, $D(n)$ is the set of documents assigned to node n, D_c is the set of all documents from category c and $D_c(n)$ is the set of documents from category c assigned to node n.

Again, both measures take values in the interval [0,1]; the higher the value, the better agreement between clusters and a priori categories.

8.2 Growing Neural Gas

8.2.1 Incrementality Study

Model evaluation were executed on 2054 documents downloaded from 5 newsgroups with quite well separated main topics (antiques, computers, hockey, medicine and religion). Each GNG network was trained for 100 iterations with the same set of learning parameters, using previously described winner search method.

In the main case (depicted with the black line), the network was trained on the whole set of documents. This case was the reference one for the quality measures of adaptation, as well as the comparison of the winner search methods.

Figure 8.1 presents the comparison of a standard global winner search method with our own CF-tree based approach. The local search method is not taken into consideration since, as it has already been mentioned that it is completely inappropriate in case of unconnected graphs. Obviously, the tree-based local method is invincible in terms of computation time. The main drawback of the global method is that it is not scalable and depends on the total number of nodes in the GNG model.

At first glance the results seemed surprising. On the one hand, the quality was similar, on the other - global search appeared to be worse of the two! We have investigated it further, and it has turned out to be the aftermath of process divergence during the early iterations of the training process. We shall explain it later, on the example of another experiment.

In the next experiment, in addition to the main reference case, we considered another two cases. During the first 30 iterations, the network was trained on 700 documents only. In one of the cases (light grey line), documents were sampled uniformly from all five groups, and

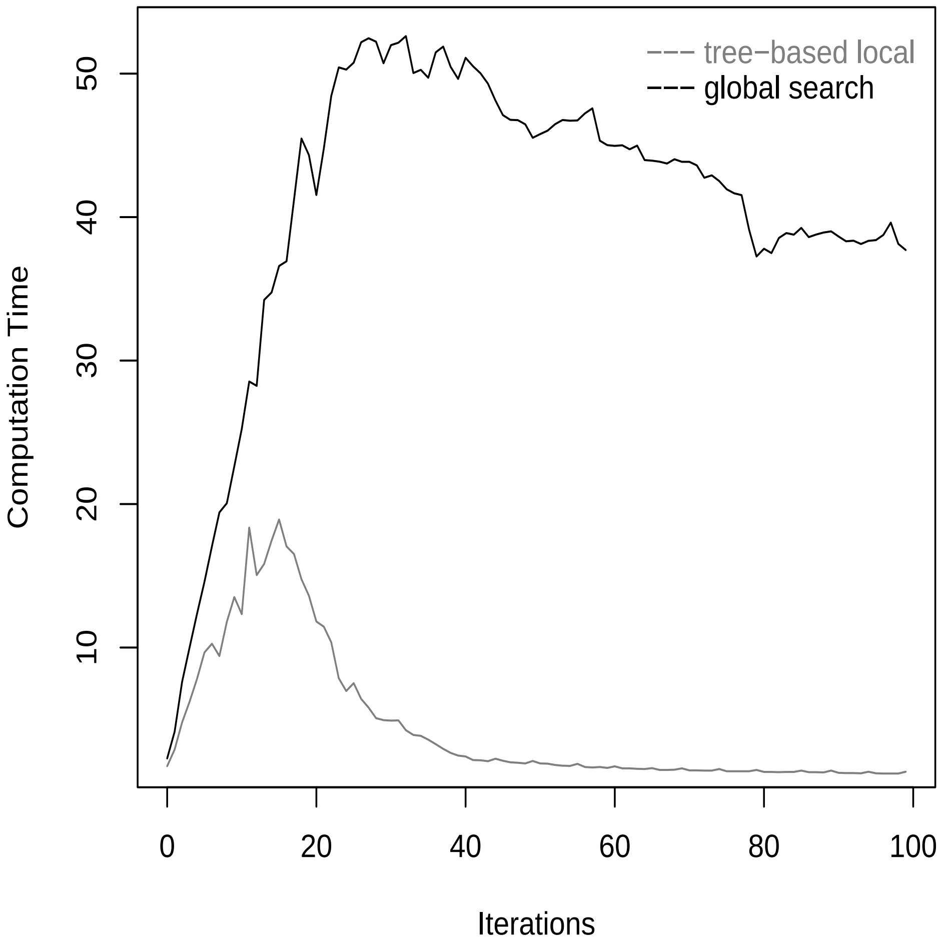


Figure 8.1: Winner search methods - computation time

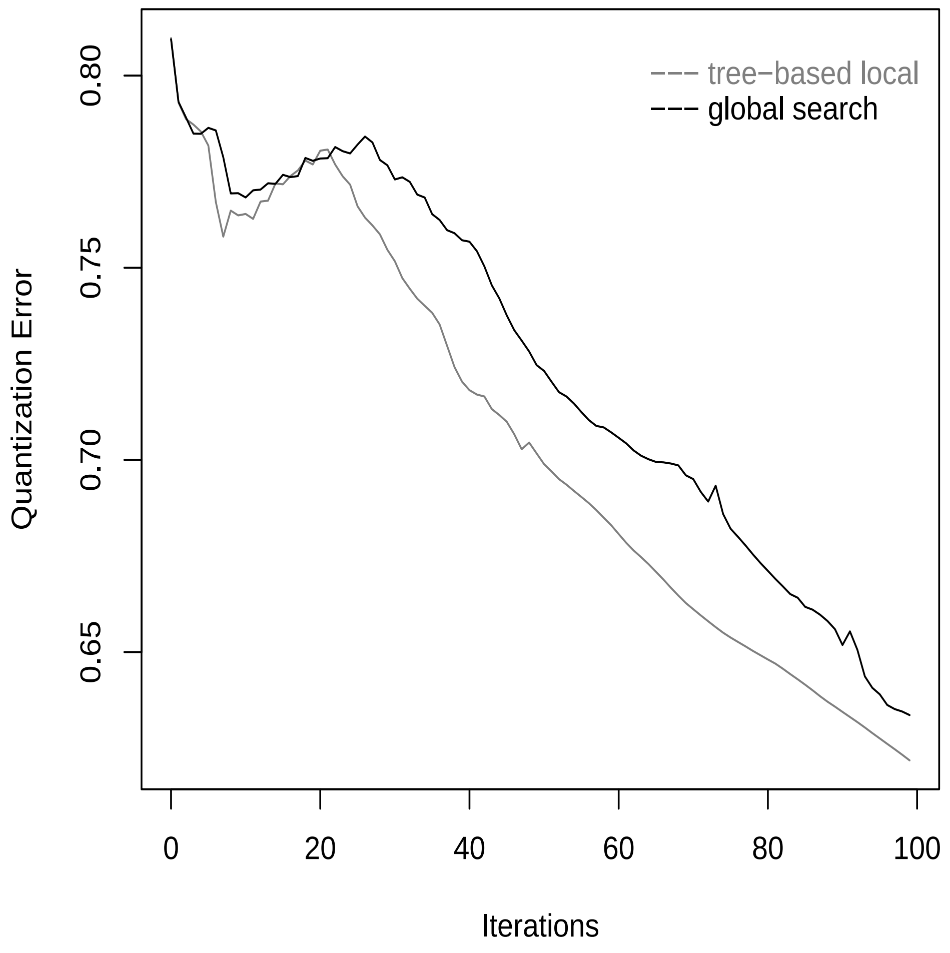


Figure 8.2: Winner search methods - model quality

in the 33rd iteration another 700 uniformly sampled were introduced to training. After the 66th iteration, the model was trained on the whole dataset.

In the last case (dark grey line) initial 700 documents were selected from two groups only. After the 33rd iteration of training, documents from the remaining newsgroups were gradually introduced in the order of their newsgroup membership. It should be noted here that in this case we had an a priori information on the categories of documents. In the general case, we are collecting fuzzy category membership information from a Bayesian Net model.

As expected, in all cases the GNG model adapts quite well to topic drift. In the global and the topic-wise incremental case, the quality of the models were comparable, in terms of the Average Document Quantization measure (see Figure 8.5), Average Weighted Cluster Purity, Average Cluster Entropy and Normalized Mutual Information (for the final values, see Table 8.1). Also the subjective criteria, such as visualization of both models and the identification of topical areas in the SOM projection map, were similar.

Table 8.1: Final values of model quality measures

	Cluster Purity	*Cluster Entropy*	*NMI*
non-incremental	0.91387	0.00116	0.60560
topic-wise incremental	0.91825	0.00111	0.61336
massive addition	0.85596	0.00186	0.55306

The results were noticeably worse for massive addition of documents, even though all the covered topics were present in the training from the very beginning and should have occupied their own, specialized areas in the model. However, as can be noticed on the same graph, a complex mixture of topics can pose a serious drawback, especially in the first training iterations. In the global reference case, the attempt to cover all topics at once leads the learning process to a

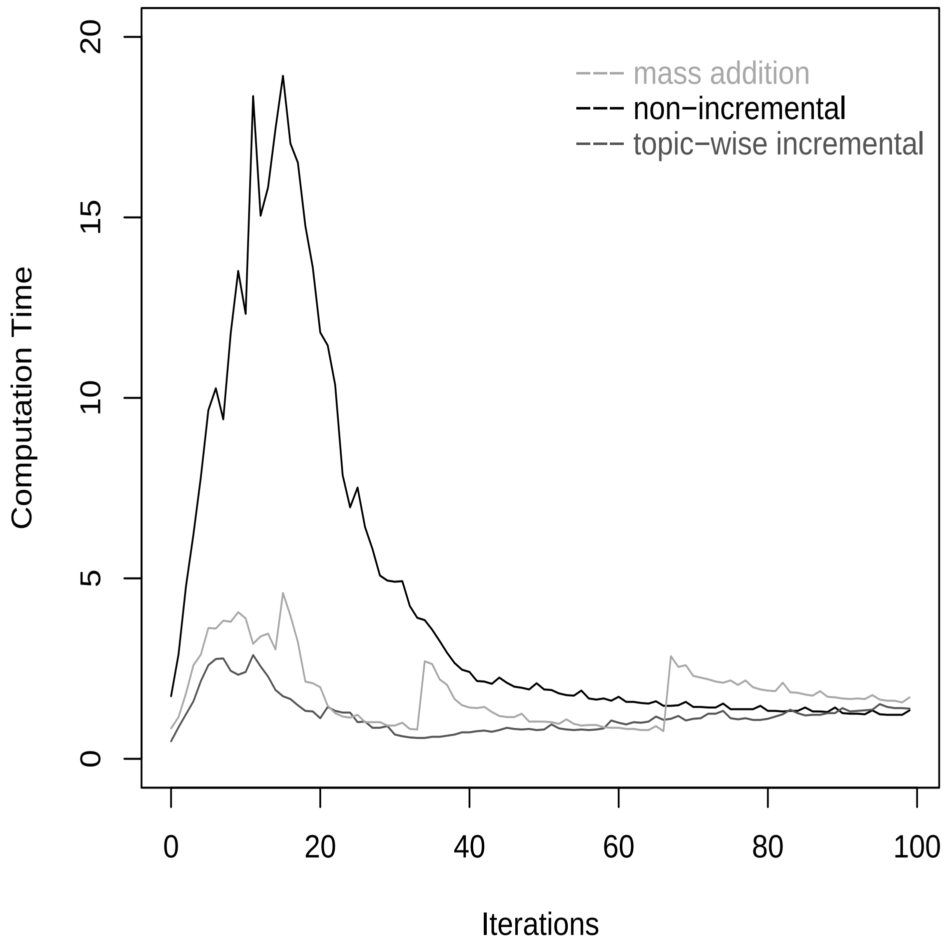


Figure 8.3: Computational complexity - execution time of a single iteration

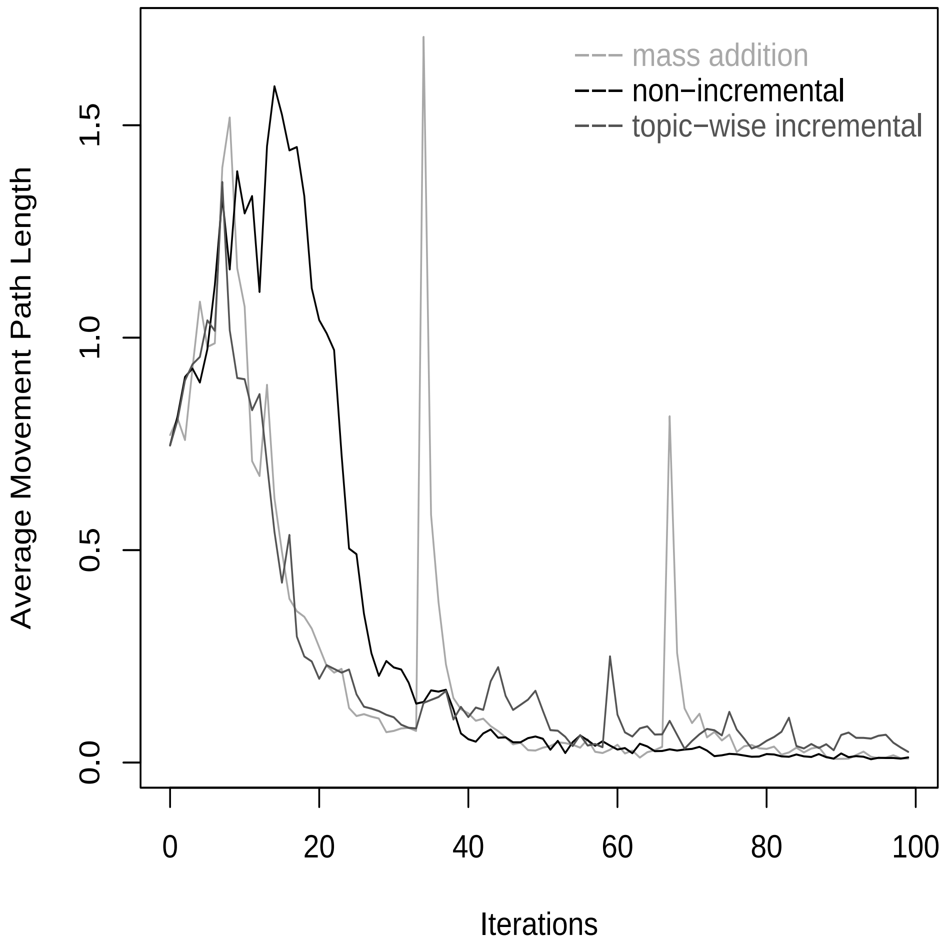


Figure 8.4: Computational complexity - average path length for a document

local minimum and to subsequent divergence (what, in fact, is quite time-consuming, as one can notice in Figure 8.3).

As we have previously noticed, the above-mentioned difficulties apply also to the case of global winner search (Figure 8.2). As a matter of fact, the quality of the final models in case of using the incremental approach is almost the same for global search and CF-tree based search (Cluster Purity: 0.92232 versus 0.91825, Normalized Mutual Information: 0.61923 versus 0.61336, Average Document Quantization: 0.64012 versus 0.64211).

Figure 8.4 presents the average number of GNG graph edges traversed by a document during a single training iteration. It can be seen that massive addition causes temporal instability of the model. Also, the above mentioned attempts to cover all topics at once in case of a global model caused much slower stabilization of the model and extremely high complexity of computations (Figure 8.3). The last reason for such slow computations is the representation of GNG model nodes. The referential vector in such a node is represented as a balanced red-black tree of term weights. If a single node tries to occupy too big portion of a document-term space, too many terms appear in such a tree and it becomes less sparse and - simply - bigger. On the other hand, better separation of terms which are likely to appear in various newsgroups and increasing "crispness" of topical areas during model training leads to highly efficient computations and better models, both in terms of the previously mentioned measures and a subjective human reception of the results of search queries.

The last figure, 8.6, compares the change in the value of the Average Map Quantization measure, reflecting "smoothness" of the model (i.e. continuous shift between related topics). In all three cases the results are almost identical. It should be noted that the extremely low initial value of the Average Map Quantization is the result of model initialization via broad topics method [39], shortly described in Section 7.5.

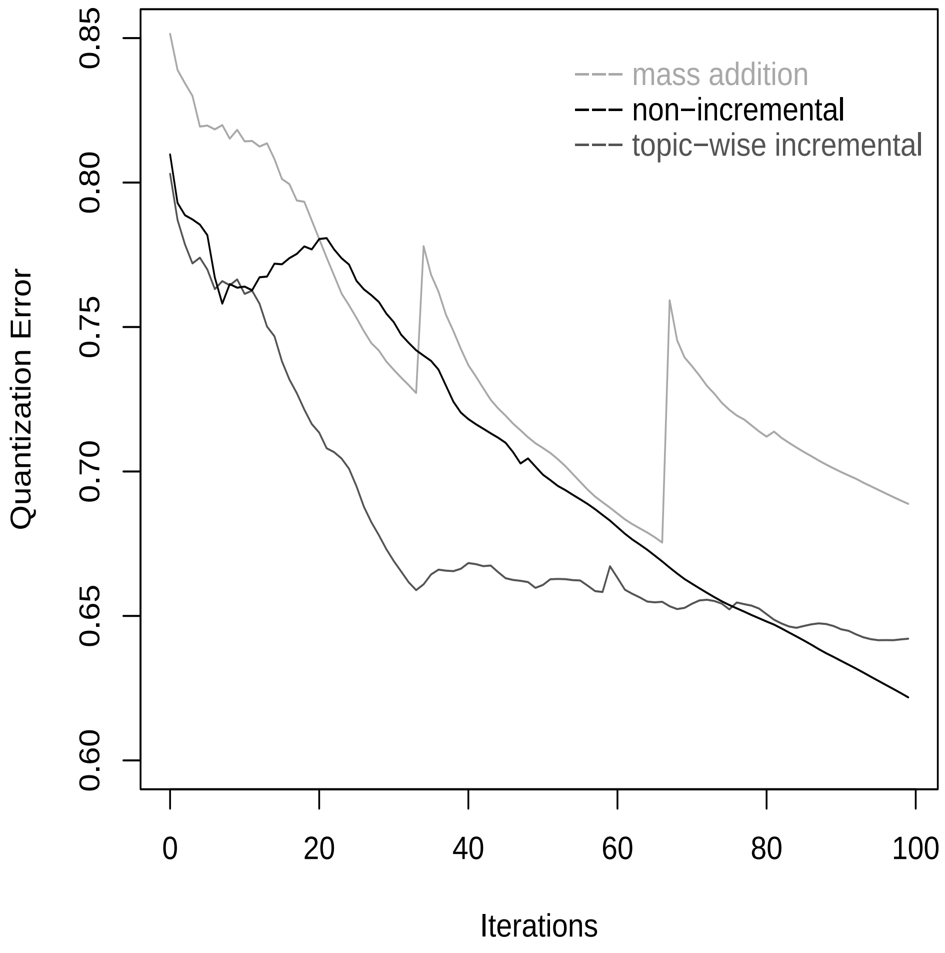


Figure 8.5: Model quality - Average Document Quantization

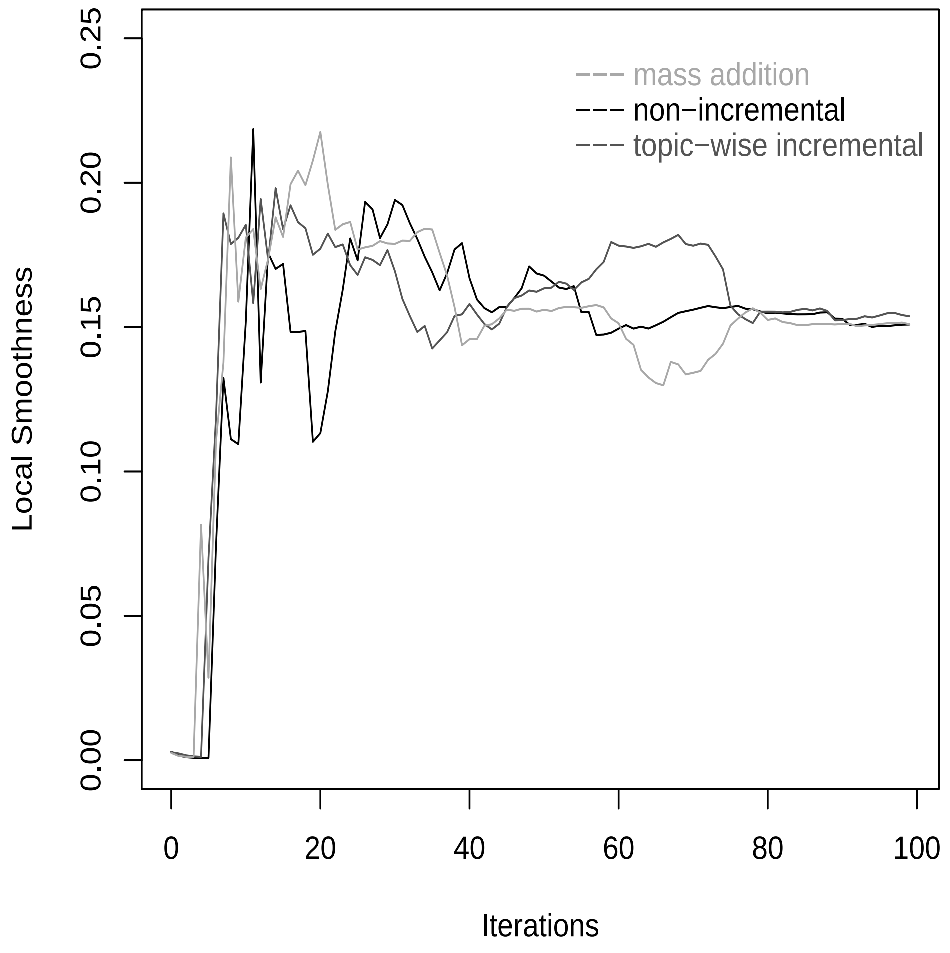


Figure 8.6: Model quality - Average Map Quantization

8.2.2 Scalability Issues

To evaluate scalability of the proposed contextual approach (both in terms of space and time complexity), we built a model for a collection of million documents crawled by our topic-sensitive crawler starting from several Internet news sites (`cnn`, `reuters`, `bbc`).

The resulting model consisted of 412 contextual maps, which means that the average density of a single map was about 2500 documents. The experimental results shown in this section are presented in a series of box-and-whisker graphs, which allows us to present a distribution of a given evaluation measure (e.g. time, model smoothness or quantization error) over all 412 models, measured after each iteration of the learning process (horizontal axis). The horizontal line represents the median value, the area inside the box represents 25% - 75% quantiles, the whiskers represent extreme values and each dot represents outlier values.

Starting with an initial document clustering/context initialization via hierarchical Fuzzy-ISODATA (see Section 7.6), followed by GNG model learning (see Section 6.5) and GNG-SOM projection (see Section 7.5), the whole cycle of map creation process took 2 days. It is an impressing result, taking into account that Kohonen and his co-workers reported processing times in order of weeks [44]. It should also be noted that the model was built on a single personal computer[1]. As it has been stated before, a contextual model construction can be easily distributed and parallelized, which would lead to even shorter execution times.

The first observation is the complexity of a single iteration of GNG model learning (Figure 8.7), which is almost constant, regardless of the increasing size of the model graph. It confirms the observations from Section 8 concerning the efficiency of the tree-based winner search methods. One can also observe the positive impact of the homogeneous distribution of term frequencies in the documents grouped to in a single map cell. Such homogeneity is - to some extent - acquired by the initial split of a document collection into contexts. Another way of

[1]Pentium IV HT 3.2 GHz, 1 GB RAM

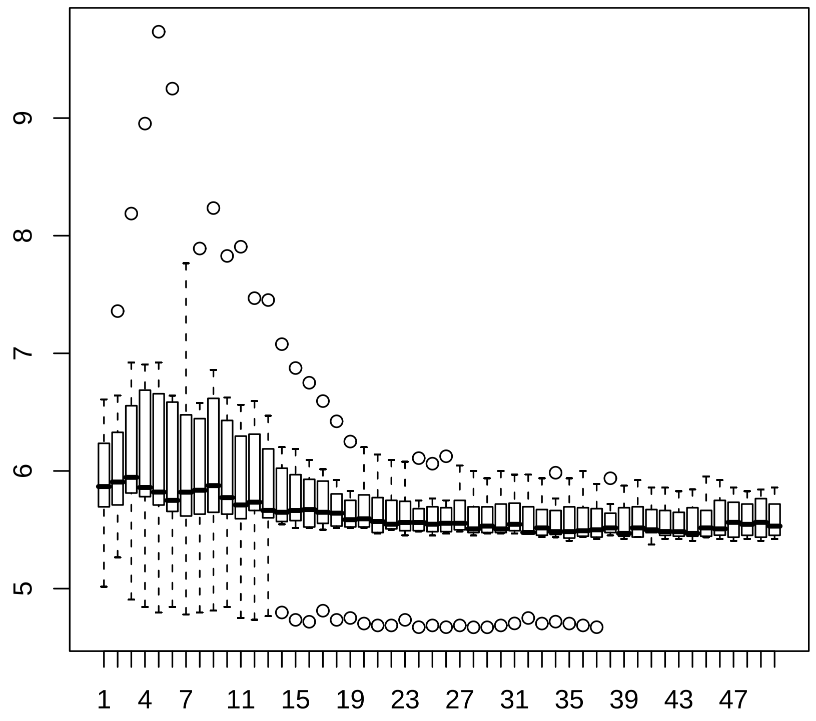


Figure 8.7: Computation complexity of the contextual model - execution time for a single iteration

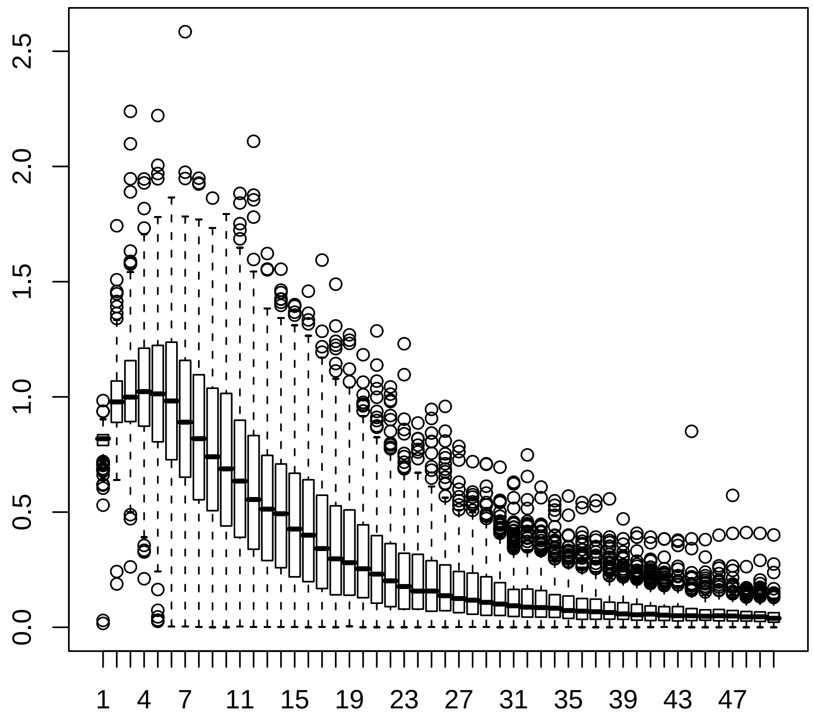


Figure 8.8: Computation complexity of the contextual model- average path length of a document

the processing time reduction can be the contextual reduction of vector representation dimensionality, described in the previous section.

Figure 8.8, presents the dynamics of the learning process. The average path length of a document is the number of shifts over graph edges when documents are moved to a new, optimal location. It can be seen that model stabilizes quite fast; actually, most models converged to final state in less than 30 iterations. The fast convergence is mainly due to the topical initialization. It should also be noted here that the proper topical initialization can be obtained for well-defined topics, which is the case in contextual maps.

Figure 8.9 presents the quality of the contextual models. The final values of the average document quantization (Figure 8.9) and the map quantization (Figure 8.10) are low, which means that the resulting maps are both "smooth" in terms of local similarity of adjacent cells and accurately represent documents grouped in a single node. Moreover, such low values of the document quantization measure have been obtained for a moderate size of GNG models (the majority of the models consisted of only 20-25 nodes - due to their fast convergence - and represented about 2500 documents each).

8.3 Artificial Immune Systems

8.3.1 Immune Network Quality

Beside the clustering structure represented by cells, an idiotypic network should also be treated as a meta-clustering model. Similarity between the individual clusters is expressed by graph edges, linking the referential vectors in the antibodies. Thus, the quality of the structure of the edges needs to be evaluated.

There is a number of ways for evaluating an idiotypic model structure. In this section we present the one which we have found to provide the clearest interpretation. This approach is based on the analysis of the edge lengths in the minimal spanning tree (MST) constructed over the set of antibodies in each iteration of the learning process.

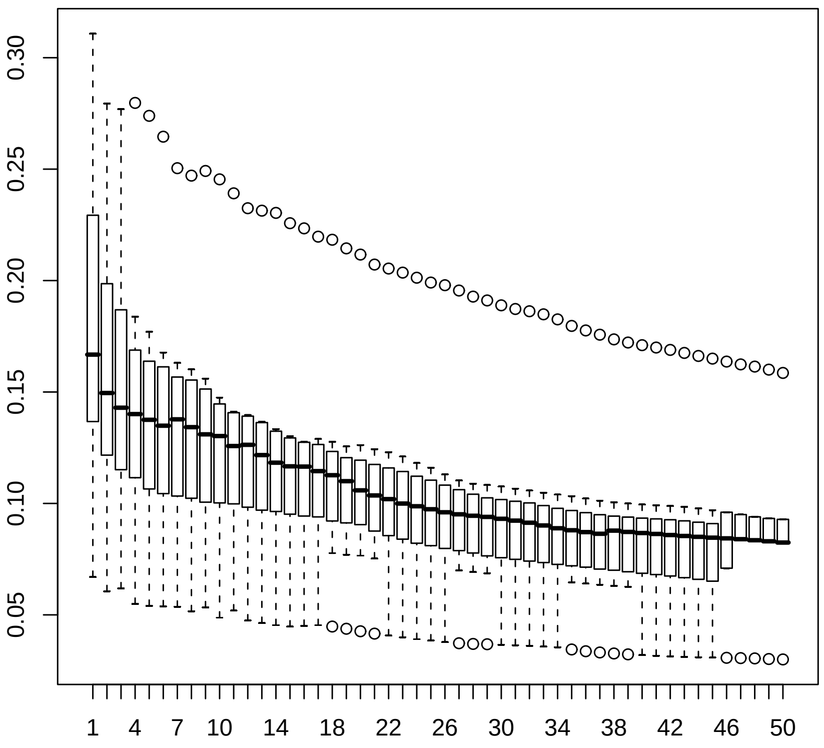


Figure 8.9: Contextual model quality - Average Document Quantization

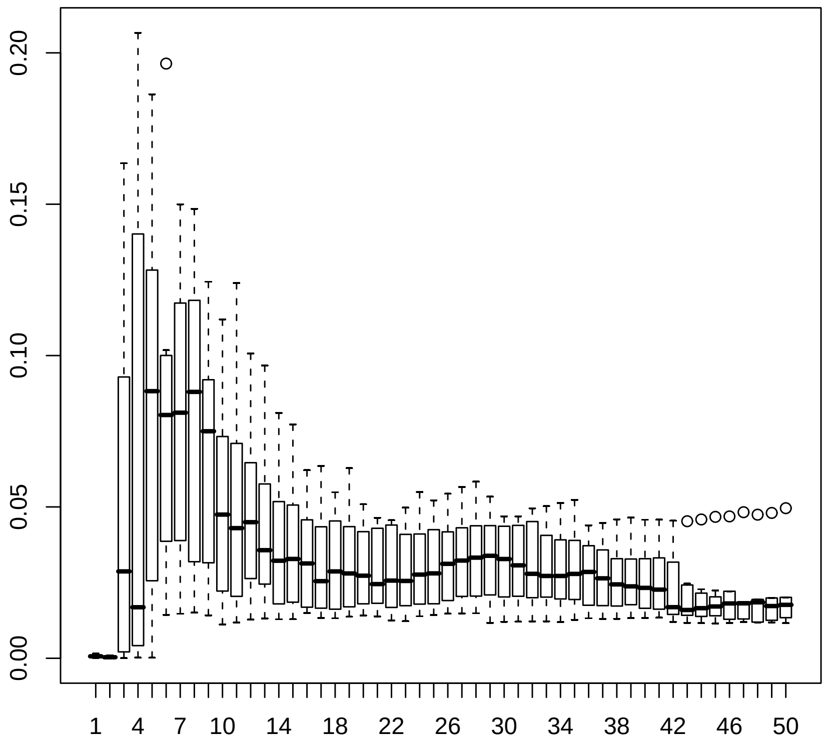


Figure 8.10: Contextual model quality - Average Map Quantization

8.3.2 Experimental Settings

The architecture of the BEATCA system supports comparative studies of clustering methods at the various stages of the process (i.e., initial document grouping, initial topic identification, incremental clustering, graph model projection to a 2D map and visualization, identification of topical areas on the map and their labeling). In particular, we conducted series of experiments to compare the quality and stability of the AIS, GNG and SOM models for various model initialization methods, cell/antibody search methods and learning parameters [42, 41]. In this section we only focus on evaluation and comparison of the immune models.

This study required manually labeled documents, so the experiments were executed on the widely used 20 Newsgroups document collection[2] of approximately 20 thousands newsgroup messages, partitioned into 20 different newsgroups (about 1000 messages each). As a data preprocessing step in the BEATCA system, entropy-based dimensionality reduction techniques were applied [39], so the training data dimensionality (the number of distinct terms used) was 4419.

Each immune model was trained for 100 iterations, using the previously described algorithms and methods: contexts extraction (Section 7.6), agglomerative identification of redundant antibodies (Section 7.4.2), robust construction of mutated antibodies (Section 7.4.3), time-dependent parameters (Section 7.4.4) and CF-tree based antibody search method (Section 7.4.5).

8.3.3 Time-dependent Parameters Impact

In the first two series of experiments, we compared models built using time-dependent parameters $\sigma_s(t)$ and $\sigma_d(t)$ with constant, a priori defined values of σ_s and σ_d. As a reference case we took a model where $\sigma_s(t)$ variedd from the initial value of 0.05 up to 0.25, and $\sigma_d(t)$ from 0.1 up to 0.4 (cf. Equation (7.4)).

First, we compared the reference model and the four models with a constant σ_d. The parameter σ_s changed identically as in the reference

[2]`http://people.csail.mit.edu/jrennie/20Newsgroups/`

model. The values of σ_d varied from the initial value in the reference model (i.e., 0.1) up to the final value of 0.4 by step 0.1. The results[3] are presented in Figures 8.11-8.14.

Figure 8.11 presents the variance of the edge length in the minimal spanning tree built over the set of antibodies in the immune memory in $i - th$ iteration of the learning process. At first glance, one can notice te instability of this measure for high values of σ_d. Comparing the stable values, we notice that the variance for the reference network has the highest value. It means that the idiotypic network contains both short edges, connecting clusters of more similar antibodies, and longer edges, linking more distant antibodies, probably stimulated by different subsets of documents (antigens). Such a meta-clustering structure is obviously desirable and preferred over networks with equidistant antibodies (and, thus, low edge length variance).

Comparing the network sizes, Figure 8.12, and the quantization error, Figure 8.13, one can notice that for the highest values of σ_d, the set of antibodies reduces to just a few entities on the other hand - for the lowest values almost all antibodies (both universal and overspecialized ones) are retained in the system memory. It is not surprising that the quantization error for a huge network (e.g. $\sigma_d = 0.1$) is much lower than for smaller nets. Still, the time-dependent $\sigma_d(t)$ gives a similarly low quantization error for a moderate network size. Also, both measures stabilize quickly during the learning process. The learning time, Figure 8.14, is – to some extent – a function of network size. Thus, for the reference model, it is not only low but very stable over all iterations.

In the next experiment – dually – we compare the reference model and another five models with constant σ_s (and varying σ_d). Analogically to the first case, the values of σ_s varied from the starting value (0.05) up to the final value in the reference model (0.25) by step 0.05. The results are presented in Figures 8.15-8.18. We restrict the discussion of the results to the conclusion that also in this case a time-dependent parameter $\sigma_s(t)$ had a strong, positive influence on

[3]This and the following figures show the average values of the respective measures over 20 contextual networks

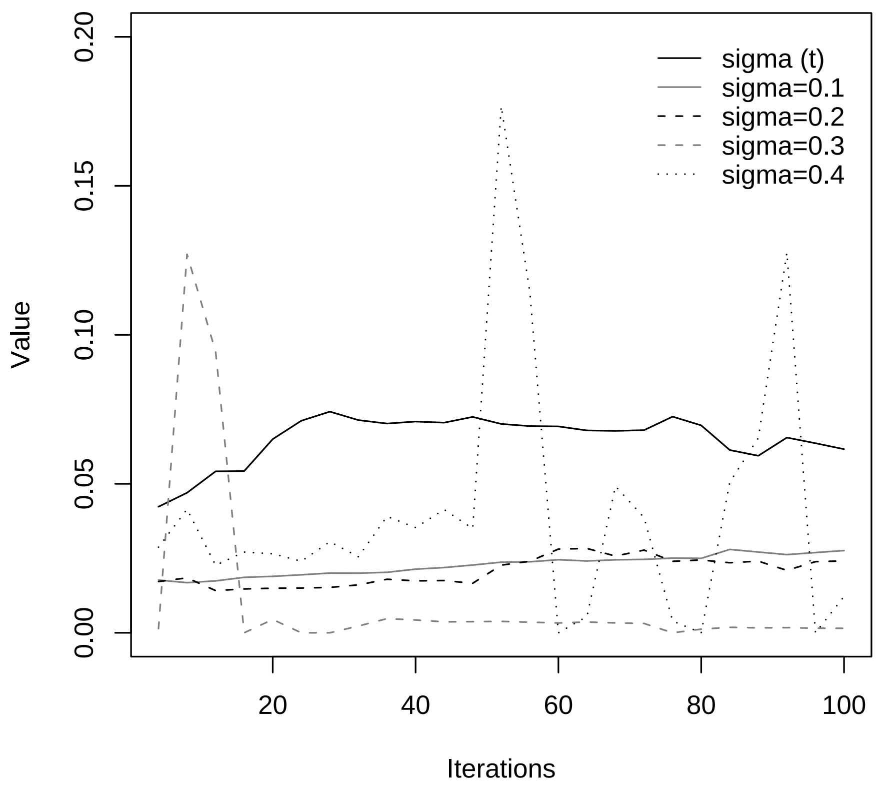


Figure 8.11: Time-dependent σ_d - edge length variance

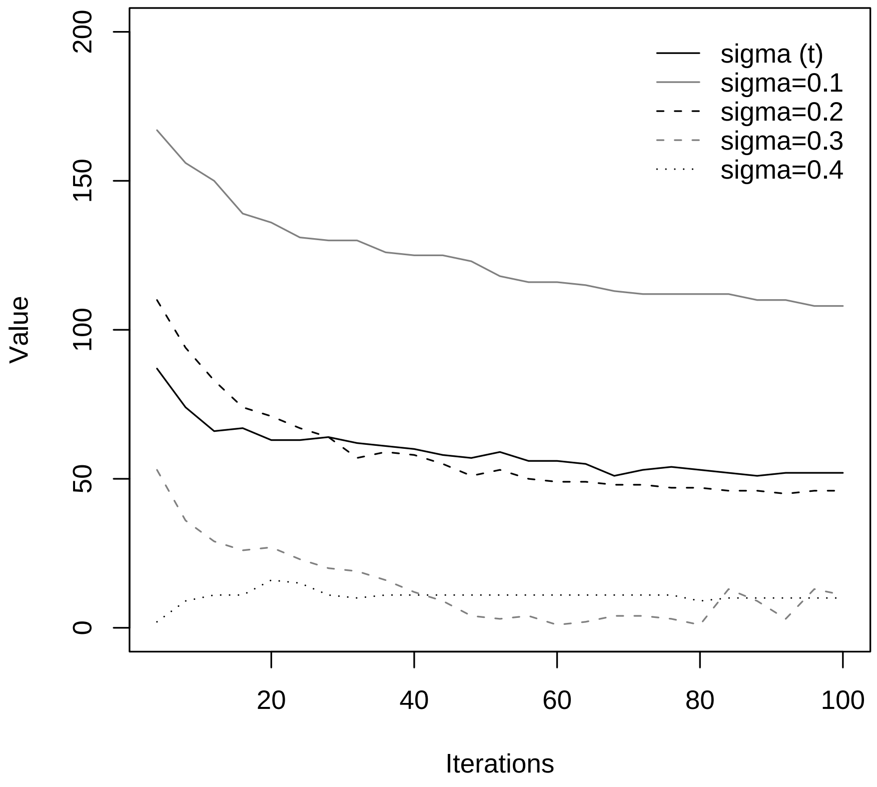


Figure 8.12: Time-dependent σ_d - net size

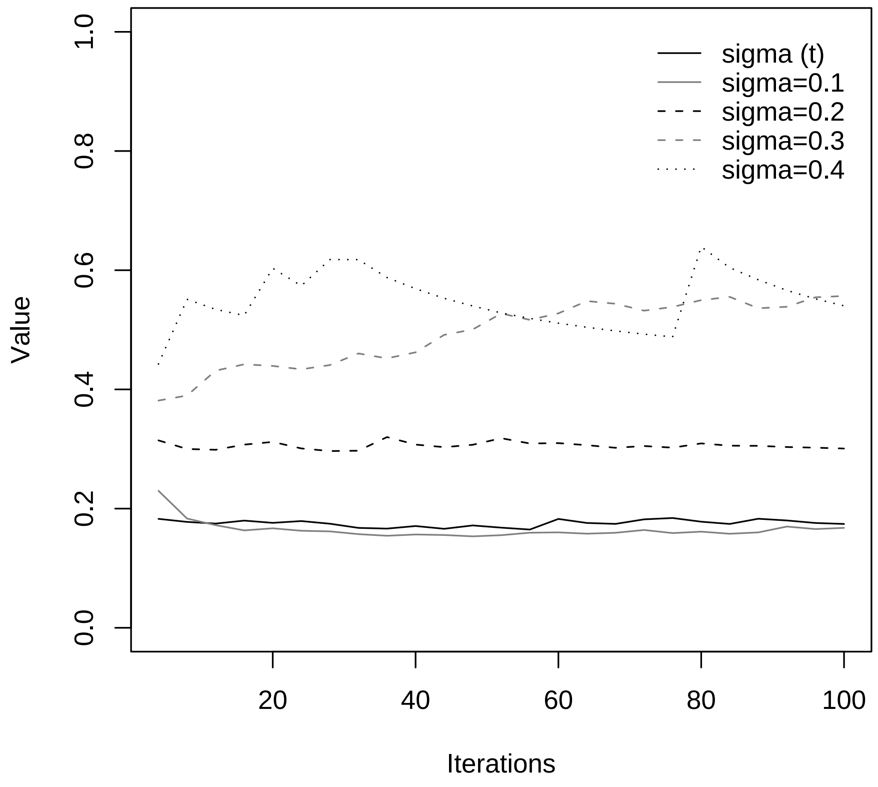


Figure 8.13: Time-dependent σ_d - quantization error

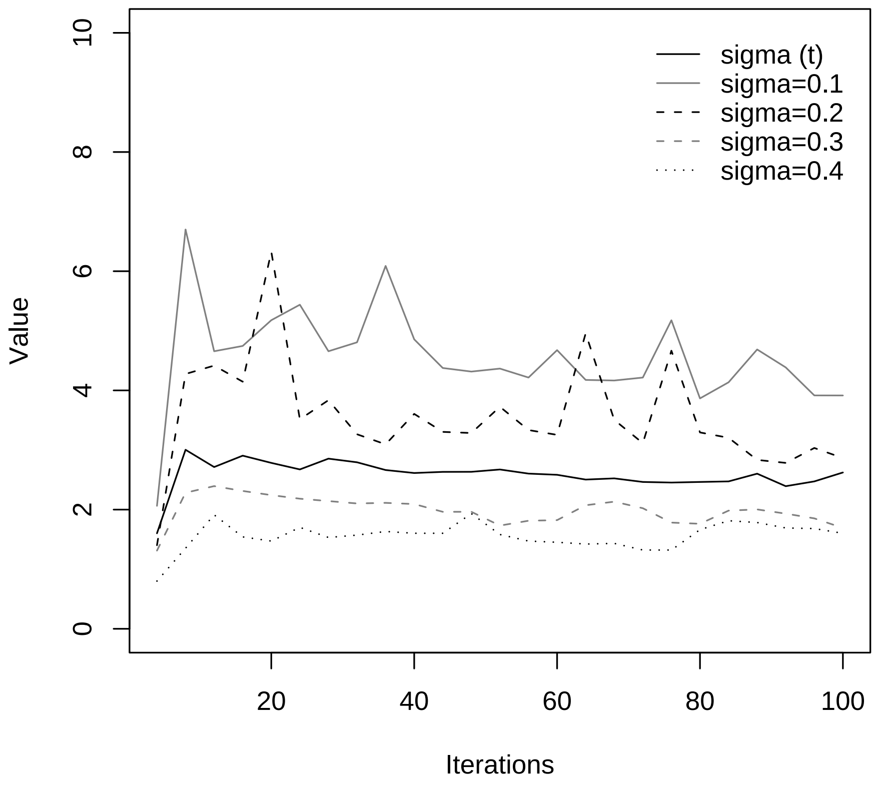


Figure 8.14: Time-dependent σ_d - learning time

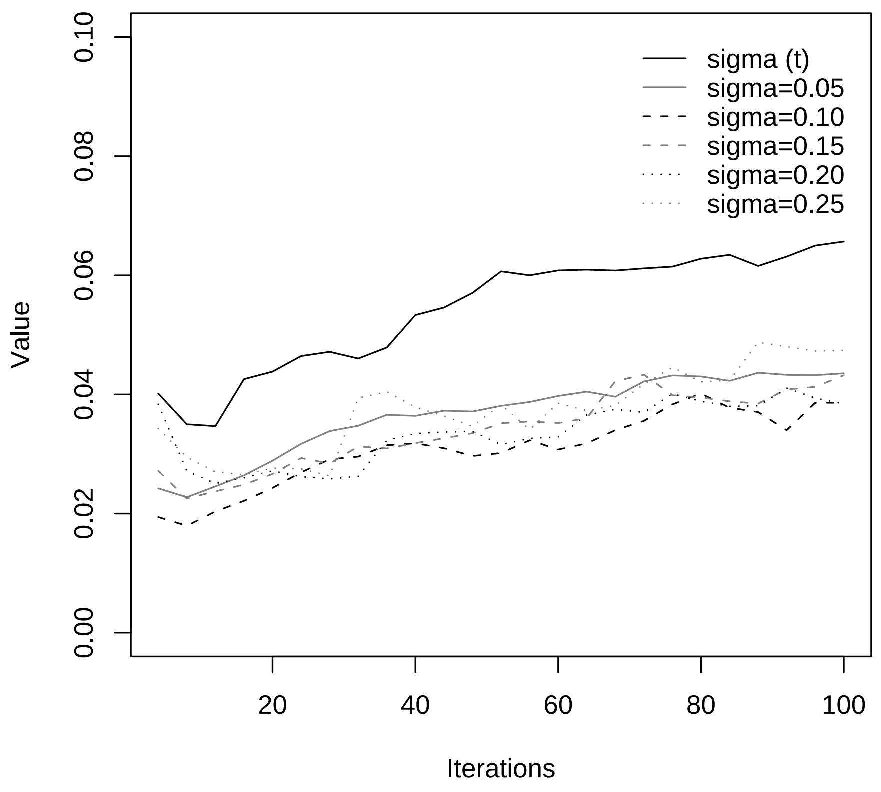


Figure 8.15: Time-dependent σ_s - edge length variance

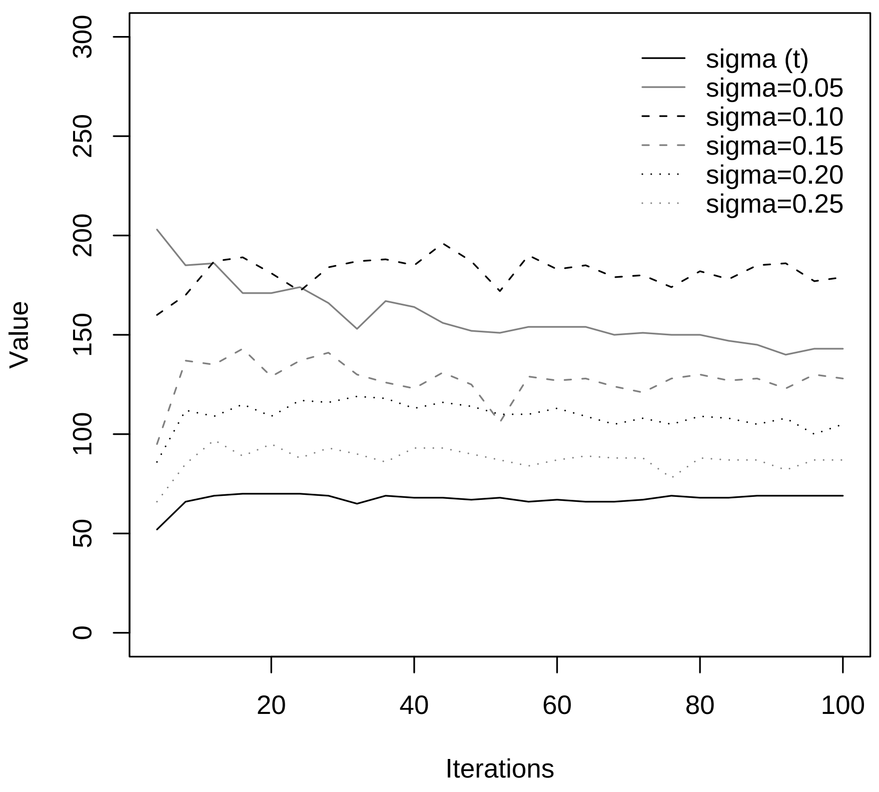


Figure 8.16: Time-dependent σ_s - net size

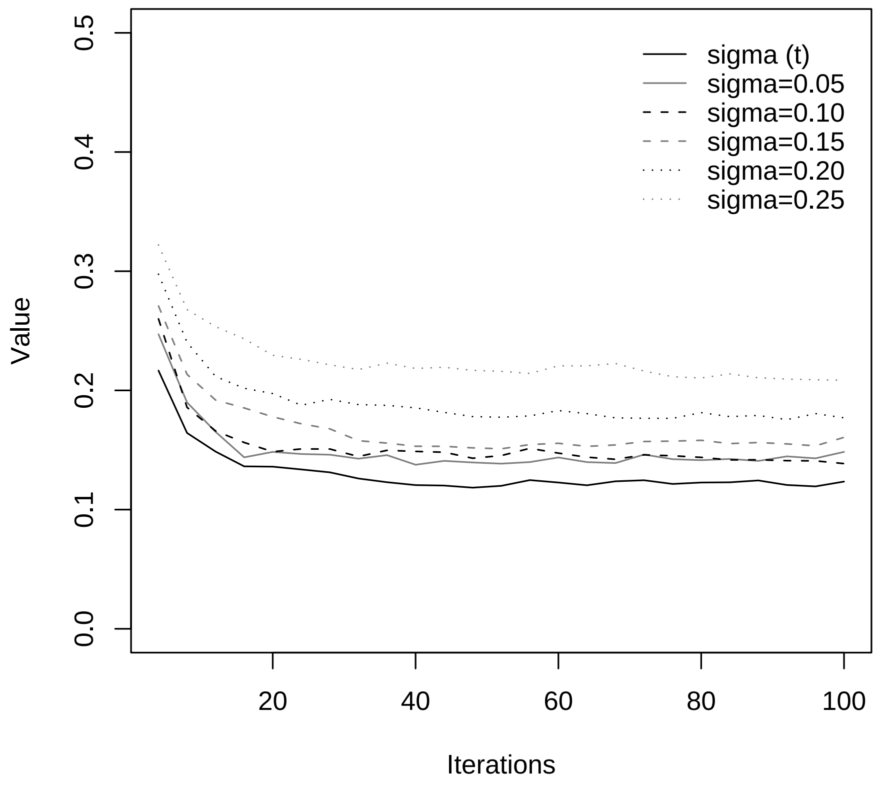


Figure 8.17: Time-dependent σ_s - quantization error

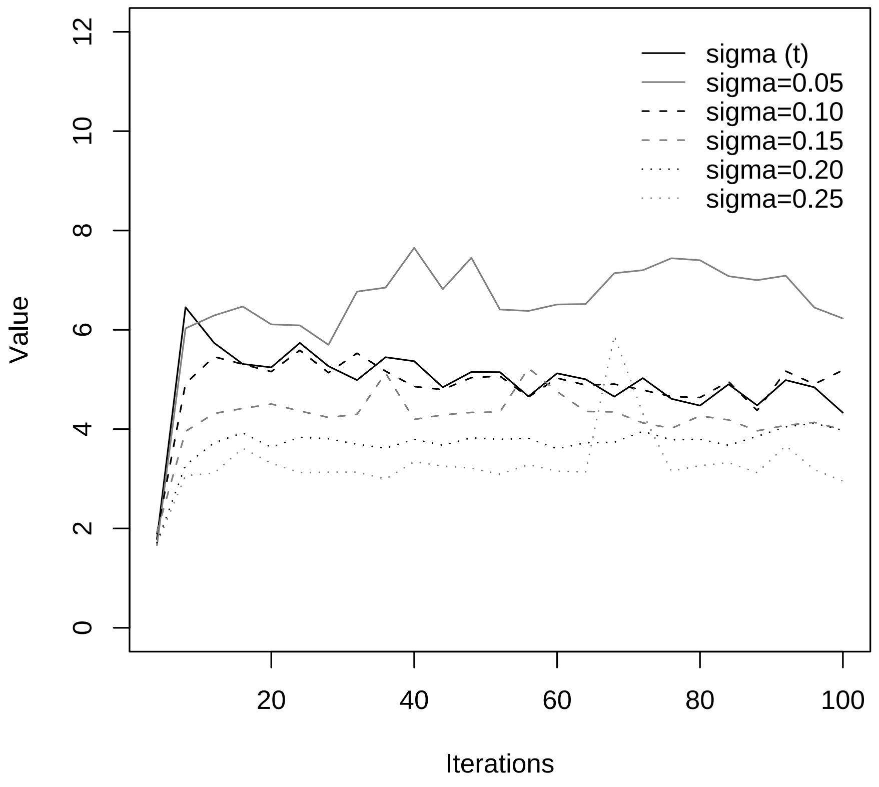


Figure 8.18: Time-dependent σ_s - learning time

the resulting immune model.

A weakness of the approach seems to be the difficulty in selecting the appropriate values of the parameters for a given dataset. We investigated independently changes to the values of both parameters, but it turn out that they should be changed "consistently"; that is, the antibodies should not be removed too quickly, nor aggregated too quickly. However, once found, there is a justified hope that for an incrementally growing collection of documents the parameters do not need to be sought anew, but rather gradually adapted.

8.3.4 Scalability and Comparison with Global Models

Comparing hierarchical and contextual models described in Section 7.6, with a "flat", global model the most noticeable difference is the learning time[4]. The total time for 20 contextual networks amounted to about 10 minutes, against over 50 minutes for a hierarchical network and almost 20 hours (*sic*!) for a global network. Another disadvantage of the global model is the high variance of the learning time at a single iteration as well as the size of the network. The learning time varied from 150 seconds to 1500 seconds (10 times more!) and the final network consisted of 1927 antibodies (two times more than for the contextual model). It should also be noted that in our experimental setting, each model (both the local and the global one) was trained for 100 iterations, but it can be seen (e.g. Figure 8.22) that the local model stabilizes much faster. Recalling that each local network in the hierarchy can be processed independently and in parallel, it makes the contextual approach robust and scalable[5] alternative to the global immune model.

One of the reasons for such differences in the learning time is the representation of antibodies in the immune model. The referential

[4]By learning time we understand the time needed to create an immune memory consisting of the set of antibodies representing the set of antigens (documents).

[5]Especially with respect to a growing dimensionality of data, which - empirically - seems to be the most difficult problem for the immune-based approach

vector in an antibody is represented as a balanced red-black tree of term weights. If a single cell tries to occupy a "too big" portion of a document-term vector space (i.e. if it covers documents belonging to different topics), many terms which rarely co-occur in a single document have to be represented by a single red-black tree. Thus, the tree becomes less sparse and – simply – bigger. On the other hand, better separation of terms which are likely to appear in various topics and increasing "crispness" of topical areas during model training leads to faster convergence and better models, in terms of the previously defined quality measures. While the quantization error is similar for the global and the contextual model (0.149 versus 0.145, respectively), both the supervised measures – showing correspondence between documents labels (categories) and clustering structure – favor contextual model. The final value of the Normalized Mutual Information was 0.605 for the global model and 0.855 for the contextual model, while Average Weighted Cluster Purity was 0.71 versus 0.882, respectively.

One can also observe the positive impact of homogeneity of the distribution of term frequencies in documents grouped to a single antibody. Such homogeneity is - to some extent - acquired by initial split of a document collection into contexts. Another cause of the learning time reduction is the contextual reduction of vector representation dimensionality, described in the section 7.6.

It can be seen that model stabilizes quite fast; actually, most models converged to final state in less than 20 iterations. The fast convergence is mainly due to topical initialization. It should also be noted here that the proper topical initialization can be obtained for well-defined topics, which is the case in contextual model.

We have also carried out experiments comparing presented immune approach with SOM models: the flat (i.e. standard, global Kohonen's map) and our own variant of the contextual approach - the hierarchy of contextual maps (C-SOM). To compare the immune network structure, with the static grid of the SOM model, we have built a minimal spanning tree on the SOM grid. The results are summarized in Figure 8.19. Again, the global model turned out to be of a lower quality than both the contextual SOM and the contextual AIS model.

Similarly to the global immune model, also in this case the learning time (over 2 hours) was significantly higher than for the contextual models. Surprisingly, the average edge in the contextual SOM model was much longer than in case of the contextual immune network and the standard SOM, which may be the result of the limitations of the rigid model topology (2D grid). We defer the discussion of the edge length distribution (Figure 8.20) to Section 8.3.6.

8.3.5 Contextual versus Hierarchical Model

The next series of experiments compared the contextual model with the hierarchical one. Figures 8.22 and 8.23 presents the network sizes and convergence (wrt Average Document Quantization measure) of the contextual model (represented by the black line) and hierarchical model (the grey line).

Although convergence to a stable state is fast in both cases and the quantization errors are similar, it should be noted that the error occurs for a noticeably smaller network in the contextual case (and in a shorter time too, as mentioned in the previous section).

However, the most significant difference is the generalization capability of both models. For this experiment, we partitioned each context (group of documents) into the training and test subsets (in 10:1 proportion). The training documents were used during the learning process only, while the quantization error was computed for both subsets. The results are shown in Figure 8.23 – the respective learning data sets are depicted with black lines, while the test data sets with grey lines. Nevertheless, the quantization errors for the learning document sets are similar, the difference lies in the test sets and the hierarchical network is clearly *overfitted*. Again, there is no room to go into a detailed study here, but it can be shown that this undesirable behavior is the result of the noised information brought by the additional terms, which finally appears to be not meaningful in the particular context (and thus is disregarded in contextual weights w_{dtG}).

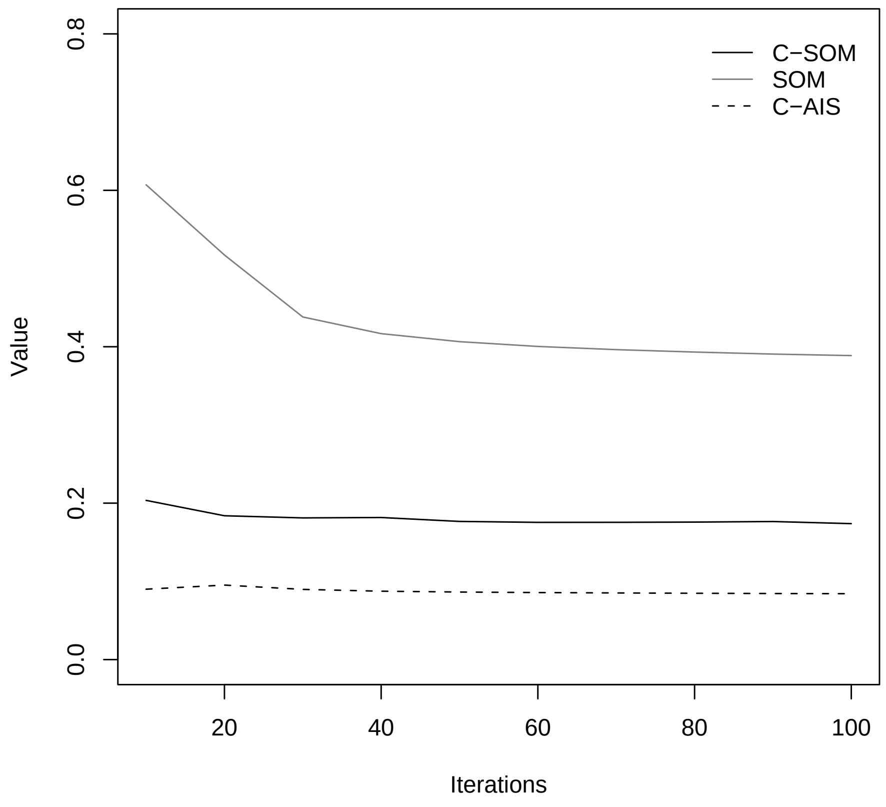


Figure 8.19: Immune model vs. SOM - quantization error

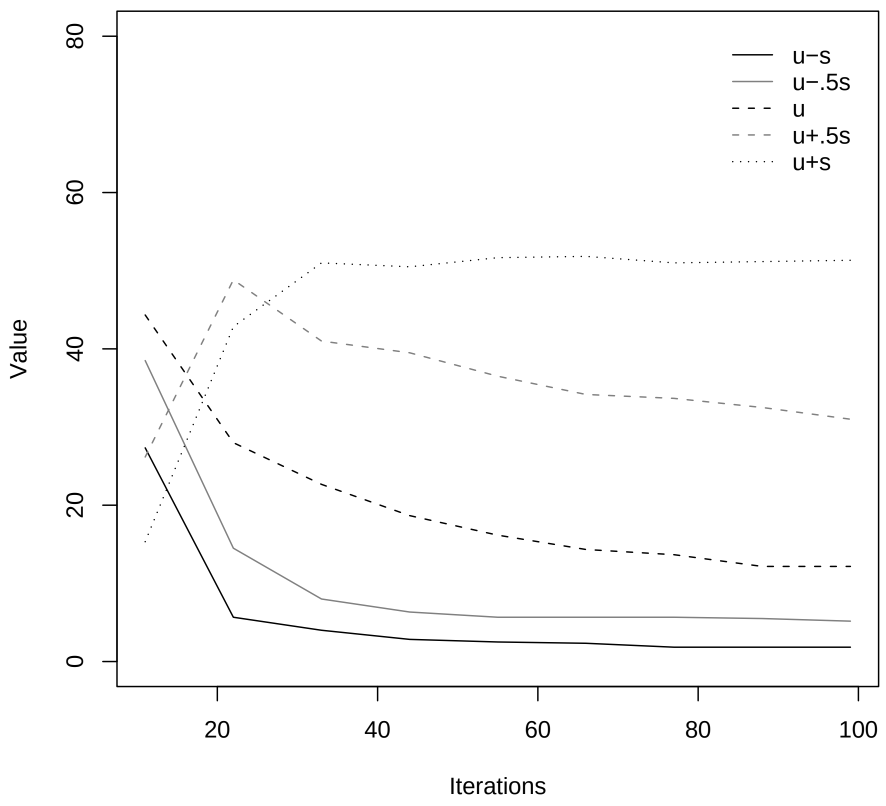


Figure 8.20: Immune model vs. SOM - SOM(MST) edge length distribution

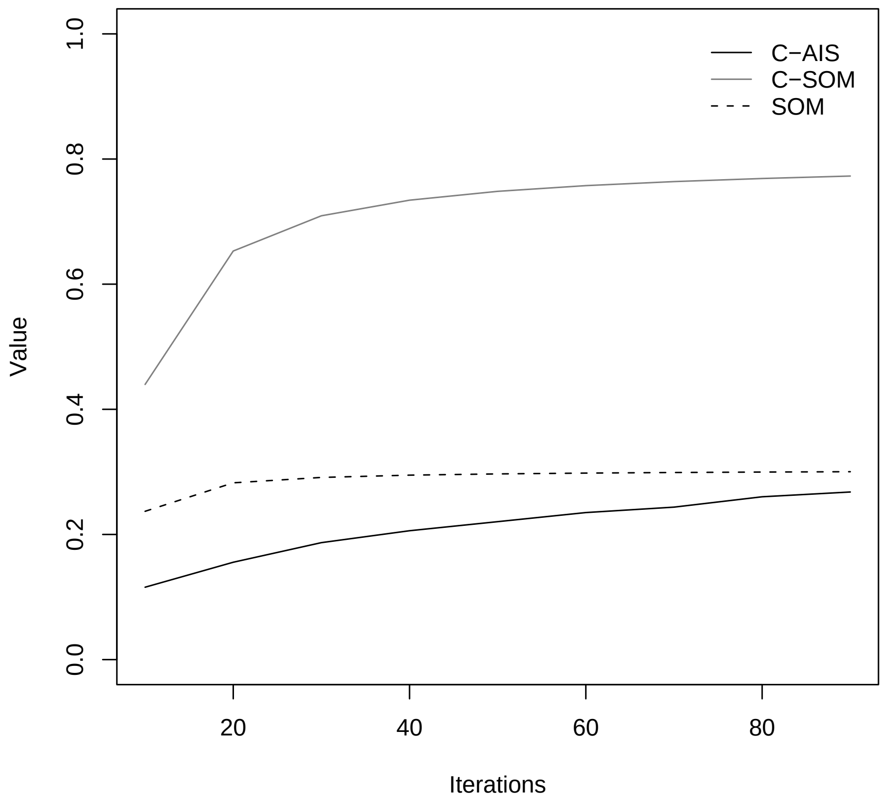


Figure 8.21: Immune model vs. SOM - average edge length

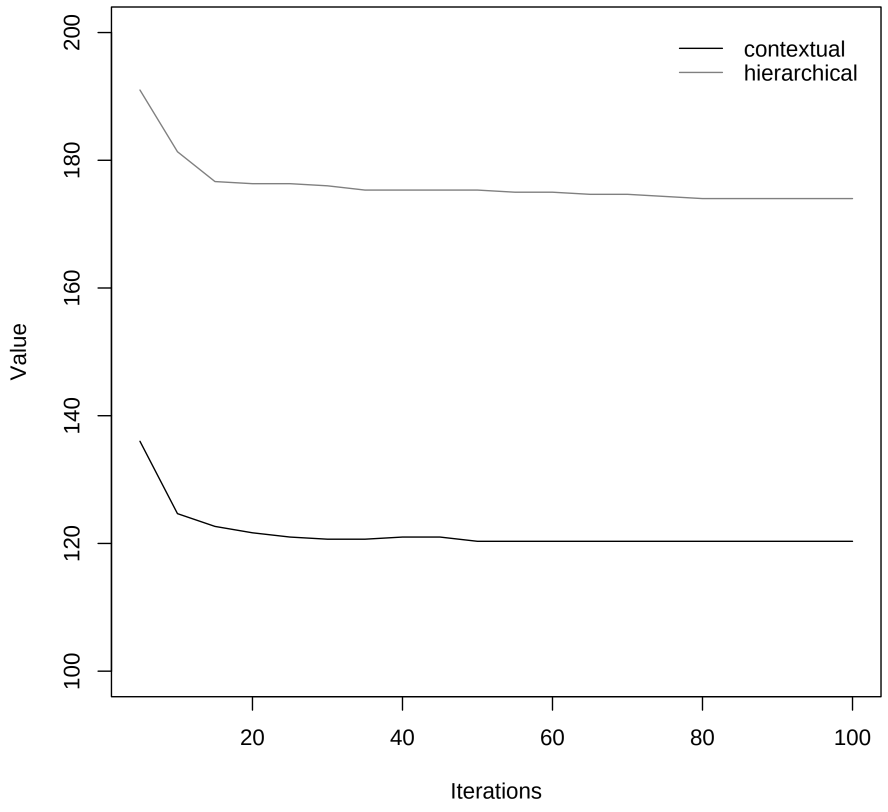


Figure 8.22: Contextual vs. hierarchical model - network size

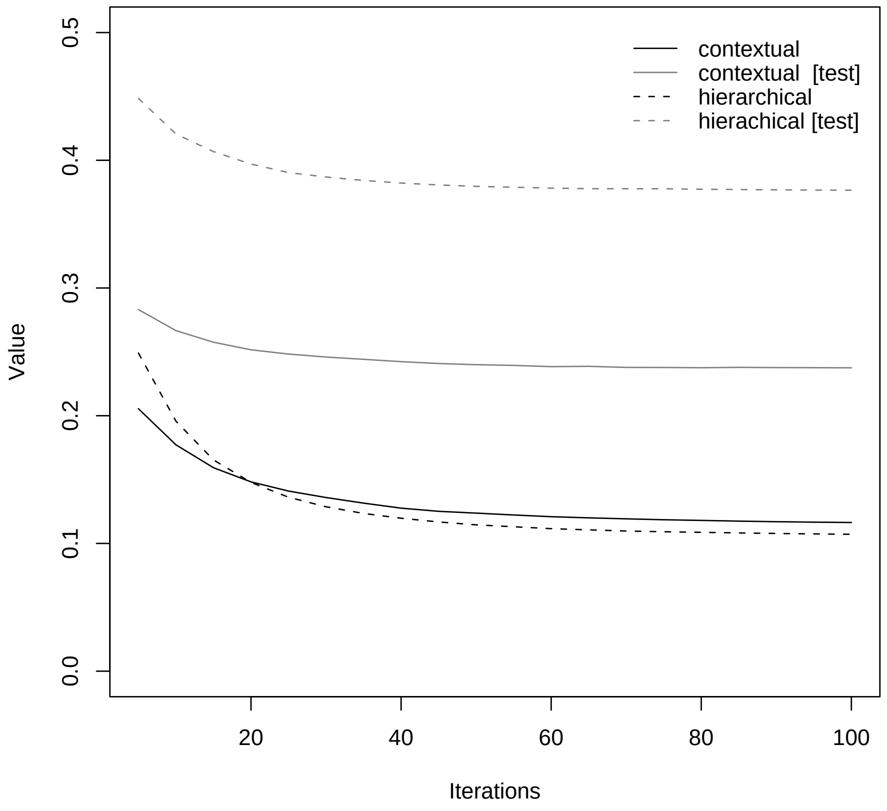


Figure 8.23: Contextual vs. hierarchical model - quantization error

8.3.6 Immune Network Structure Investigation

To compare the robustness of different variants of immune-based models, more profound investigation of the graph structure quality has been carried out. In each learning iteration, for each of the immune networks: contextual [Figure 8.25], hierarchical [Figure 8.26], global [Figure 8.27] and MST built on SOM grid [Figure 8.21], the distributions of the edge lengths have been computed. Next, the average length u and the standard deviation of the length s have been calculated and edges have been classified into one of the five categories, depending on their length. The first category consists of the edges no longer that $u-s$, i.e. the shortest ones. The second category contains edges with lengths in the interval $(u-s, u-0.5s]$, the next - "average length" edges between $u-0.5s$ and $u+0.5s$. The last two categories contained longer edges: 4^th - edged shorter that $u+s$ and the last one - longer than $u+s$.

Additionally, in Figure 8.24, we can see the average length of the edges for the hierarchical and contextual immune networks (dashed and solid black lines, respectively) and complete graphs antibodies of both the models (cliques - depicted with grey lines). Actually, in both cases a clustering structure has emerged, and the average length of the edge in the immune network is much lower than in the complete graph. However, the average length for the contextual network is lower, whereas the variance of this length is higher which signifies a more explicit clustering structure.

There are quite a few differences in the edge length distribution. One can notice than in all models, the number of the shortest edges diminishes with time. This is consistent with the intention of a gradual elimination of the redundant antibodies from the model. However, such an elimination is much slower in case of the global model, which is another reason of slow convergence and high learning time. Also in case of the SOM model, which has a static topology and where no removal of inefficient cells is possible, we can see that the model slowly reduces the number of redundancies, represented by too similar referential vectors.

On the extreme end, the distribution of the dynamics of the longest

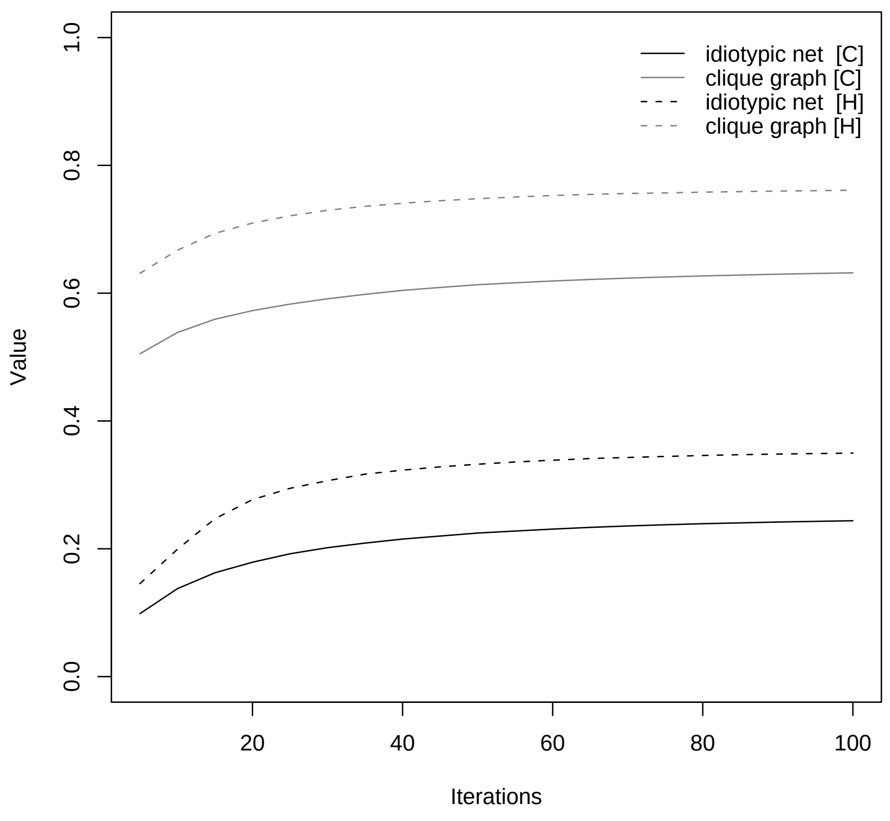


Figure 8.24: Edge length distribution - complete graph

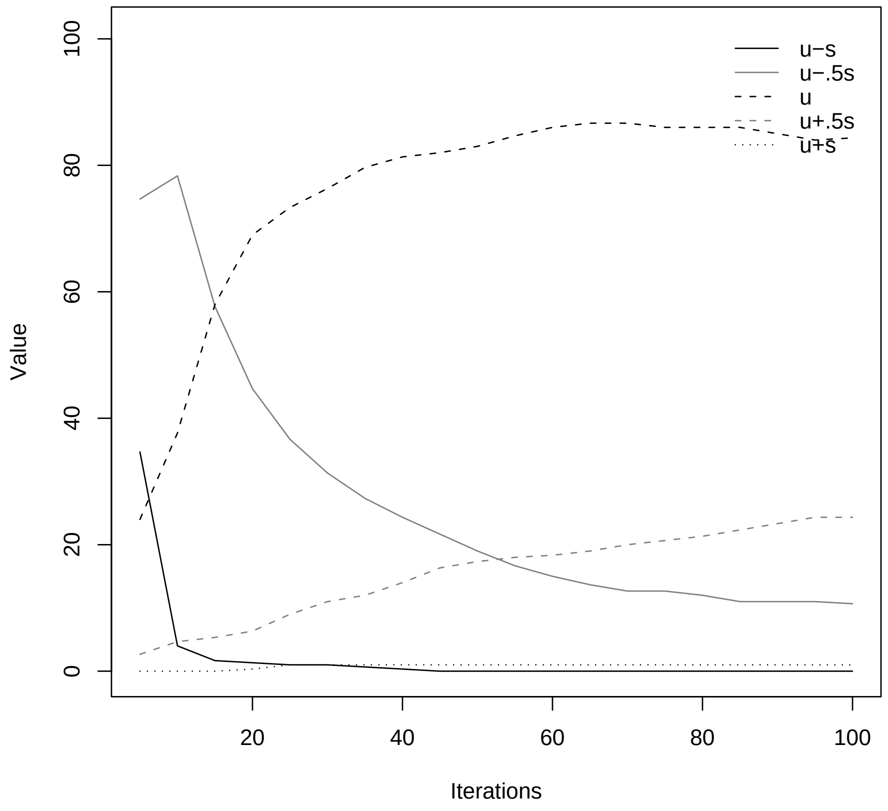


Figure 8.25: Edge length distribution - contextual net

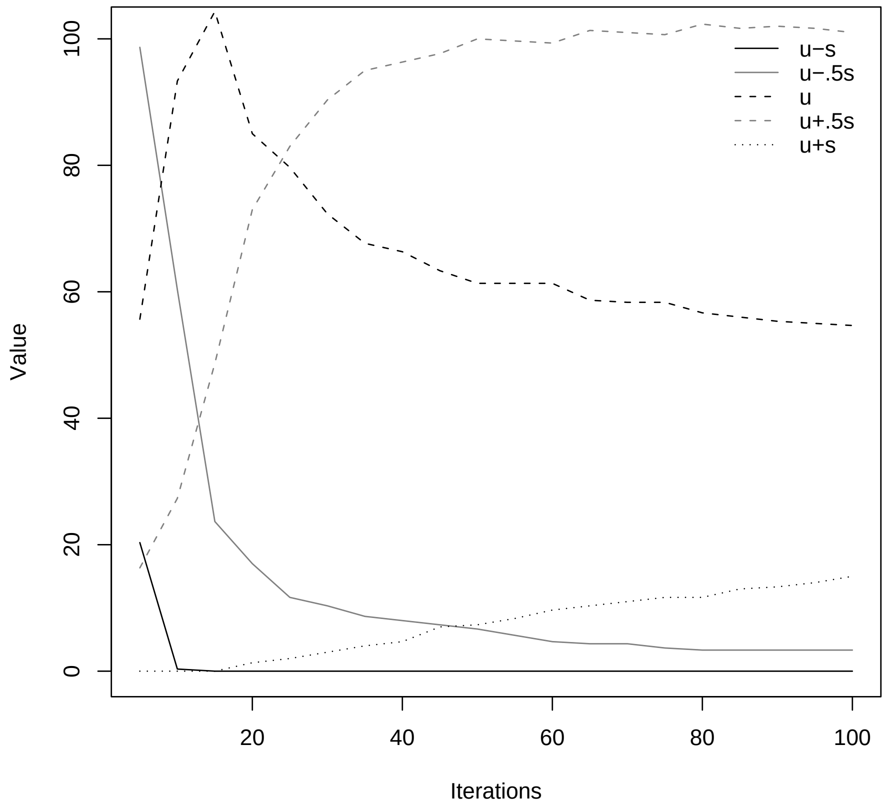


Figure 8.26: Edge length distribution - hierarchical net

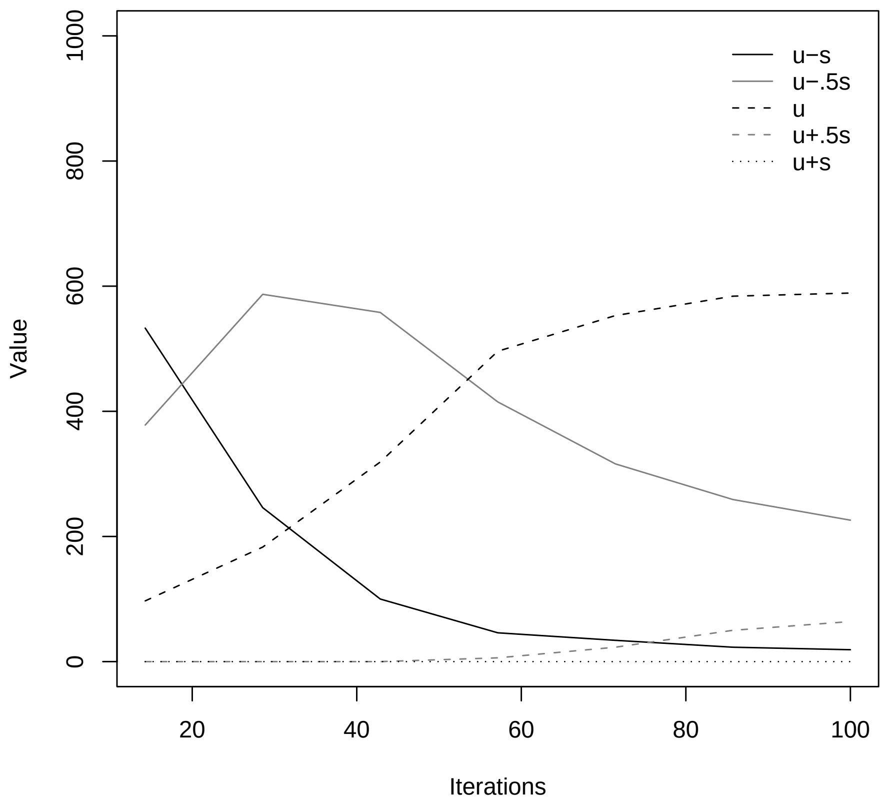


Figure 8.27: Edge length distribution - global net

edges is similar for the contextual and the global models, but different in case of the hierarchical model. In particular, a hierarchical idiotypic network contains many more very long edges. Recalling that the variance of the edge lengths has been low for this model and the average length has been high, we can conclude that hierarchical model is generally more discontinuous. The same is true for the SOM model, which is another indication of the imperfection of the static grid topology.

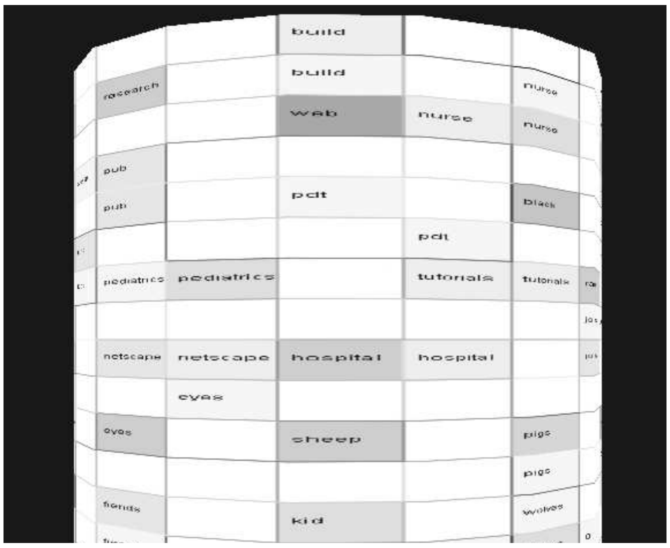
Chapter 9

User Interface

The presentation of document maps in our system is similar to that one used in the WebSOM project, but it is enriched with our own modifications.

There are 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 presented in 2D. The well-known problem of SOMs in an Euclidean space is a limited number of cell neighbors (3,4 and 6 – for triangular, quadratic and hexagonal cells, respectively). The SOM in hyperbolic spaces (HSOM) [50] is free from such limitation (see Figures 9.4(a) and 9.4(b) for exemplary tessellations). The only requirement is that $(p-2)\cdot(q-2) > 4$, where p is the number of vertices in the polygon and q is the number of neighbors of each cell. In such a space, the document map is presented as part of a hyperbolic plane, and then its projection into the unit circle is computed (the type of projection determines the type of model, e.g.: a Poincaré or a Klein one [51]). What makes HSOM the favored approach is the exponential growth of the number of cell neighbors with the growing distance from the center of the circle. This not only influences presentation (the so-called "fisheye effect"), but also gives a 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



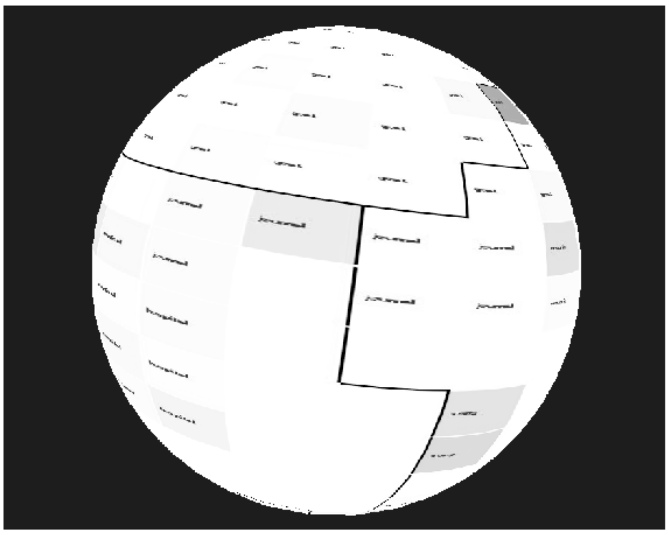


Figure 9.2: 3D map – sphere

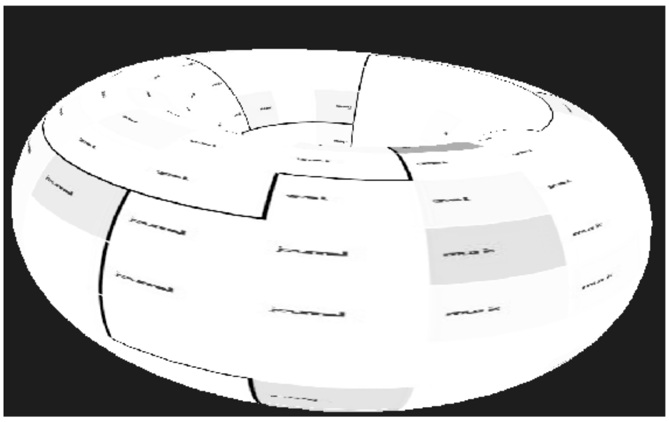


Figure 9.3: 3D map – torus

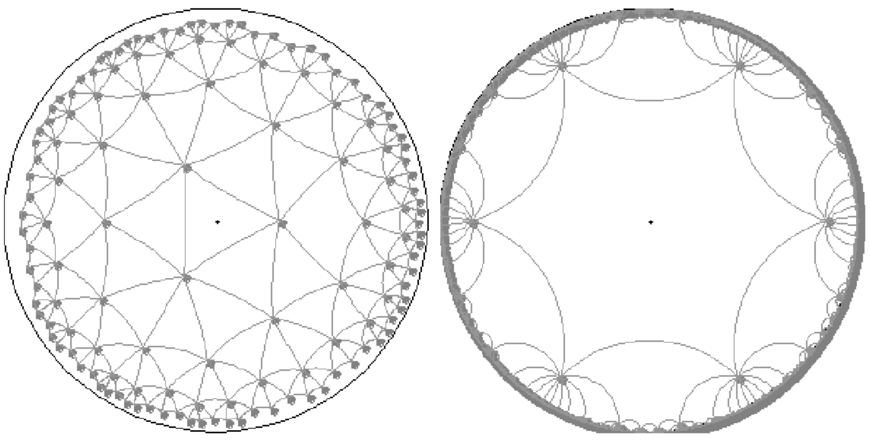


Figure 9.4: Regular tessellations of the hyperbolic space: (a) triangular (b) hexagonal

by a 2D map (see Fig. 7.1), following WebSOM's paradigm [46], or in 3D, by cylinder(see Fig. 9.1), sphere (see Fig. 9.2) or a torus (see Fig. 9.3). Our basic concern was with boundary regions, where within the original WebSOM approach the phenomenon of a too few neighbors occurred. As suggested by the WebSOM authors and others, we extended the map so that the left and the right boundaries were connected, and did the same with the top and the bottom ones. This junction is not visible in case of a planar map, but it is visible with the rotating cylinder in the horizontal direction. Rotating sphere and torus representations make the vertical junction visible too. Of course the change in the intended presentation influences the clustering is implemented in the WebSOM algorithm. Clustering of map regions is also affected by this concept. In case of the spherical representation, the distance measures in non-boundary regions are also affected.

Our further extension concerns use of a multi-map approach (maps generated with various clustering algorithms, and hence explicating different aspects of the data). When presenting the results of a query, the most appropriate map is selected out of those available, so that the documents obtained in response to a query form "well-formed" clusters in such a map.

The interface has the form usual for a search engine. When the user inputs his query, it is processed via a searcher on a system of inverse lists. In the indexing process, typical inverse indexing lists with Huffman encoding are created. Then the search is carried out for terms starting with the one having the shortest list of relevant documents.

Chapter 10

Related Works

Nowadays we have a rapid growth in the amount of written information. Therefore we need a means of reducing the flow of information by concentrating on the major topics in the document flow, including the one on the World Wide Web. Grouping documents based on similar contents may be helpful in this context as it provides the user with meaningful classes or clusters. Document clustering and classification techniques help significantly in organizing documents in this way.

For years now, therefore, we have observed a growing research effort around the so-called "Web Mining".

Web mining is generally understood as an application area of data mining techniques aimed at extracting of information from the World Wide Web. The field is oriented towards improvement of site design and site structure, generation of dynamic recommendations, and improving marketing. Inside the web mining research, there exist essentially three main streams: web-content mining (around which the research presented in this paper concentrates), web-(link)structure mining and web usage mining.

Web content mining concentrates around the discovery of useful information from the data contained in Web documents, including text, images, audio, and video content. The main difficulty is that on the one hand the Web content can be truly understood only by humans, and on the other hand the size of this content is immense.

Two groups of methodologies are applicable here: Information retrieval, oriented towards unstructured (free text) data, and data based techniques, concerned primarily with structured and semi-structured Web data.

The information retrieval methods treat the documents as "bags of words", while the database oriented methods exploit the semi-structured nature of HTML documents.

The database oriented methods attempt to find schemes (including type hierarchies) for semistructured data, to identify frequent substructures etc., by applying e.g. association rules or attribute oriented induction.

The information retrieval methods seek to find keywords and keyphrases, to discover grammatical rules, collocations, to predict word relations, to classify and/or cluster text/hypertext documents or named entities, to devise information extraction models (e.g. for event detection and tracking), to learn ontologies, and to find patterns.

The methods used here include decision trees, kNN, rule learning methods, inductive logic programming, reinforcement learning, support vector machines, self-organizing maps, but also Naive Bayes, Bayesian networks, logistic regression and other statistical methods.

Recently, the importance of visual presentation of Web mining results has been increasingly appreciated. For recent results concerning visual Web-structure mining, the reader may refer to [63], while with respect to visual Web-usage mining he may consult e.g. [9].

As far as visual presentation of Web-content mining is concerned, there have been several important projects in the recent years. The early project SPIRE (Spatial Paradigm for Information Retrieval & Exploration) [60] represented documents as "stars in the sky", where the distance reflected word similarity and word pattern matching (with severe limitations in the document set related to screen resolution). The *Themescape* project (later also known as *Aureka!*), presented a document collection as a typical topological map, where "hills" corresponded to frequent terms ("themes")[1]. While *Themescape* was "flat" (with presented through colors), the competing *SpaceCast*[2] project in-

[1]http://www.micropatent.com/static/index.htm

[2]http://www.geog.ucsb.edu/ sara/html/research/spacecast/spacecast.html

sists on an intrinsic 3D representation. Similar in spirit is the patented *VxInsight*[3] project. Another interesting example is the project *Island of Music*[4], working with "music documents" (MP3 tunes), relying on similarity of compositions based on some psychoacoustic approximations.

A prominent position among the techniques of Visual Web-content mining is occupied by the WebSOM by Kohonnen and co-workers [43]. However, the overwhelming majority of the existing document clustering and classification approaches rely on the assumption that the particular structure of the currently available static document collection will not change in the future. This seems to be highly unrealistic, because the interests of both the information consumer and of the information producers change over time.

A recent study described in [33] demonstrated deficiencies of various approaches to document organization under non-stationary environment conditions of growing document quantity. The mentioned paper pointed among others to weaknesses of the original SOM approach (which is to some extent adaptive itself) and proposed a novel dynamic self-organizing neural model, the so-called Dynamic Adaptive Self-Organising Hybrid (DASH) model. This model is based on an adaptive hierarchical document organization, supported by human-created concept-organization hints available in terms of WordNet.

Other strategies, like that of [55, 23], attempt to capture the movement of topics, dynamically enlarge the document map (by adding new cells, not necessarily in a rectangular map).

In this book we took a different perspective claiming that the adaptive and incremental nature of a document-map-based search engine cannot be confined to the map creation stage alone, and in fact involves all the preceding stages of the whole document analysis process.

In particular, already the crawler may have some impact on the incrementality of the document clustering. As we have seen from our experiments, topic-wise addition is easier to accept by the incremental processes, hence topic-sensitive crawling contributes to stable map

[3]http://www.cs.sandia.gov/projects/VxInsight.html
[4]http://www.oefai.at/ elias/music/index.html

creation. Our indexer has also been designed in an incremental manner. A special focus, however, has been the stage of document map creation. We followed a carefully balanced way between hierarchical and flat map design, between map rigidness and clustering structure flexibility. Hierarchy helped us to go around complexity of document-by-document comparisons. The flexible GNG and AIS helped to get rid of the rigidity of WebSOM like flat map. But we recognized the importance of the WebSOM rectangular map idea, so that finally the GNG/AIS structure was projected to the WebSOM map.

Last not least we want to point at the fact that our contextual maps concept gives rise in a natural way to emergence of abstract topical concepts which are in other approaches bound to some predefined structures (like Wordnet).

Chapter 11

Concluding Remarks

As indicated e.g. in [33], most document clustering methods, including the original WebSOM, suffer from the inability to accommodate streams of new documents, especially those involving a drift, or even radical change of topic.

Though one could imagine that such an accommodation could be achieved by "brute force" (learning from scratch whenever new documents arrive), there exists a fundamental technical obstacle to such a procedure: the processing time required. However, the problem is deeper and has a "second bottom": as the clustering methods like those of WebSOM contain elements of randomness, any re-clustering of the same document collection may lead to a radical change in the view of the documents.

The results of this research are concerned with both aspects of adaptive clustering of documents.

First of all, the whole process of document map formation has been designed in an incremental way, which includes crawling, indexing and all the stages of map formation (document grouping, document group to WebSOM mapping and map region identification). Accordingle, the Bayesian Network driven crawler is capable of collecting documents around an increasing number of distinct topics. The incremental structures of the indexer adapt to the changing dictionary. The query answering interface, in particular its query extension capability based on the Bayesian network and GNG – derived dynamic automated thesaurus, also adapts to the growing document collection. Though the proper clustering algorithms used in our system, like GNG [26, 27, 28], SOM [43, 22, 44], or fuzzy-k-means [4, 29], are by their nature adaptive, nonetheless their tuning and modification was not a trivial task, especially with respect to our goal of achieving a quality of the incremental map comparable to that of the non-incremental one.

Second, special algorithms for topical map initialization as well as for identification of document collection topics, based on GNG, SVD and/or Bayesian networks, lead to stabilization of the overall map. At the same time GNG detects the topic drift so that it may be appropriately visualized, due to plastic clustering approach, as the new emerging map regions. We should stress at this point that map stabilization does not preclude obtaining different views of the same document collection. Our system permits maintaining several maps of the same document collection, obtained via different initializations of the map, and, which is more important, automatically tells the user which of the maps is the most appropriate for viewing the results of his actual query.

The most important contribution of this report demonstrating, that the whole incremental machinery not only works, but that it works well. For the quality measures we have investigated, we have found that our incremental architecture compares well to non-incremental map learning both under the scenario of "massive addition" of new documents (many new documents, not differing in their topical structure, presented in large portions) and the scenario of "topic-wise-increment" of the collection (small document groups added, but with newly emerging topics). The latter seemed to be the most tough learning process within incremental learning, but apparently the application of GNG prior to WebSOM allowed for cleaner separation of new topics from those already discovered, so that the quality (e.g. in terms of cluster purity and entropy) was higher under incremental learning than under non-incremental learning.

The experimental results indicate that the real hard task for the incremental map creation process is a learning scenario where the documents with new topical elements are presented in large portions. But also in this case the results have proved satisfactory.

A separate issue is the learning speed in the context of crisp and fuzzy learning models. Apparently, separable and topically "clean" models allow for faster learning, as the referential vectors of SOM are smaller (i.e., contain fewer non-zero components).

From the point of view of incremental learning under SOM, a crucial factor for the processing time is the global winner search for assignment of documents to neurons. We were capable to elaborate a very effective method of mixing the global winner search with local one, which does not deteriorate the overall quality of the final map and at the same time comes close to the speed of local search.

Our future research will concentrate on exploring further adaptive methods, like artificial immunological systems for more reliable extraction of context-dependent thesauri and adaptive parameter tuning.

Before closing this chapter, let us draw the attention of the Reader of this book to an important contribution to the generally discussed issues of *S*emantic Web. As a well-known definition states: "the Semantic Web is an extension of the current web in which information is given well-defined meaning (semantics), better enabling computers and people to work in cooperation."[1]. In spite of considerable effort everyone is in the meanwhile aware that except for narrow, restricted domains the currently dominating approach of service definition via input / output data types is insufficient for usable service description, so informal human readable ones are indispensable. But the current state of the development of information technology does not allow the computer to grasp a meaning out of the human readable information present on the Web, just because this meaning is not applicable to computer "awareness". This deficiency can be bridged, however, if a web-application is not designed to act upon the human-readable information itself, but rather is designed to preprocess the information for recipient human being in an intelligent way so that the human can be supported in its decision making in particular through context-aware presentation of information [2]. Apparently, document maps seem to

[1]http://www.w3.org/RDF/Metalog/docs/sw-easy

[2]http://reasoningweb.org/2006/Objectives.html

be the most appealing for humans with respect to provision of context awareness. Hence they would be of vital importance in the coming era of semi-automatic application synthesis out of available elementary web services. Note that document clustering in the sense of a map requires a considerable amount of reasoning (primarily statistical inference) so that a map-based search engine can be truly called an "intelligent service".

Another aspect of the technologies we talked about is the personalization of information systems. At the moment, a map-based search engine can be truly used by a single user, who can tune it to his needs. The crawler, driven by user query words, adjusts the content of the document base to the user profile. As a side effect, the derived Bayesian network classifier can be viewed as a reflection of the user profile in the context of the Internet at large. This profile serves as a query expansion support for the user queries. In our search engine, the same document set, on user request, can be described by different maps, as a result of application of different clustering methods, or different clustering parameter sets. When the user issues a query, the most appropriate map for presentation of retrieved documents will be suggested to the user. Over time, the selected maps may constitute a description of the personal profile of the user.

Acknowledgements

This research has been partially supported by the KBN research grant 4 T11C 026 25 "Maps and intelligent navigation in WWW using Bayesian networks and artificial immune systems", and by the European Commission and by the Swiss Federal Office for Education and Science with the 6th Framework Programme project REWERSE no. 506779 (cf. `http://rewerse.net`)

The authors would like to thank Dawid Weiss, who provided Polish and English "stop words" lists, and the Polish stemmer. Finally, the authors would like to thank to Beata Konikowska for her advice in preparation of English version of the manuscript.

Bibliography

[1] C.C. Aggarwal, F. Al-Garawi, P.S. Yu, Intelligent crawling on the World Wide Web with arbitrary predicates. In Proc. 10th International World Wide Web Conference, 2001, pp. 96-105, 2001.

[2] A. Baraldi and P. Blonda, A survey of fuzzy clustering algorithms for pattern recognition, IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 29 (1999) 786-801,

[3] M.W. Berry, Z. Drmac, E.R. IJessup, Matrices, vector spaces and information retrieval, SIAM Review, 41 (1999) 335-362 1999

[4] J.C.Bezdek, S.K. Pal, Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data, IEEE, New York, 1992

[5] E. Bingham, H. Mannila, Random projection in dimensionality reduction: applications to image and text data, In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. San Francisco, California, 2001, pp. 245 - 250

[6] C. Boulis, M. Ostendorf, Combining multiple clustering systems, In: Proceedings of 8th European conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2004), LNAI 3202, Springer-Verlag, 2004, pp. 63-74

[7] L.N. de Castro, F.J. von Zuben, An evolutionary immune network for data clustering, SBRN'2000, IEEE Computer Society Press, 2000, pp. 84-89

[8] L.N. de Castro, J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach. Springer, 2002

[9] J. Chen, L. Sun, O.R. Zaiane, R. Goebel, Visualizing and discovering Web navigational patterns, `webdb2004.cs.columbia.edu/papers/1-3.pdf`

[10] C.K. Chow, C.N. Liu, Approximating discrete probability distributions with dependence trees, IEEE Transactions on IT, IT-14 (1968) 462-467

[11] C.K. Chow, T.J. Wagner, Consistency of an estimate of tree-dependent probability distribution, IEEE Transactions on Information Theory, IT-19 (1973) 369-371

[12] K. Ciesielski, M. Draminski, M. Kłopotek, M. Kujawiak, S. Wierzchoń, Architecture for graphical maps of Web contents. In: O.Hryniewicz, J. Kacprzyk, J.Koronacki, S.Wierzchoń (eds.) Issues in Intelligent Systems Paradigms, Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2005, pp.27-38

[13] K. Ciesielski, M. Dramiński, M.A. Kłopotek, M. Kujawiak, S. Wierzchoń, Mapping document collections in non-standard geometries. In: B. De Beats, R. De Caluwe, G. de Tre, J. Fodor, J. Kacprzyk, S. Zadrożny (eds) Current Issues in Data and Knowledge Engineering. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2004, pp.122-132.

[14] K. Ciesielski, M. Dramiński, M.A. Kłopotek, M. Kujawiak, S.T. Wierzchoń, Clustering medical and biomedical texts – document map based approach. Proc. Sztuczna Inteligencja w Inżynierii Biomedycznej SIIB'04, Kraków, 2004, ISBN-83-919051-5-2

[15] K. Ciesielski, M. Dramiński, M.A. Kłopotek, M. Kujawiak, S.T. Wierzchoń, On some clustering algorithms for document maps creation, In: Proceedings of the Intelligent Information Processing and Web Mining (IIS:IIPWM-2005), Springer, 2005, pp. 259-266

[16] K. Ciesielski, M. Dramiński, M.A. Kłopotek, D.Czerski, S.T. Wierzchoń, Adaptive document maps In: In: Proceedings of the Intelligent Information Processing and Web Mining (IIS:IIPWM-2006), Springer, 2006, pp. 109-120

[17] D. Cohn, H. Chang, Learning to probabilistically identify authoritative documents, In: Proceedings of the 17th International Conference on Machine Learning, Stanford CA, 2000, `http://www.cs.cmu.edu/~cohn/papers/phits.ps.gz`.

[18] D. Cohn, T. Hofmann, The missing link – a probabilistic model of document content and hypertext connectivity, In: T.K.Leen, T.G.Dietterich and V.Tresp (eds), Advances in Neural Information Processing Systems, Vol. 10, 2001, `http://citeseer.nj.nec.com/cohn01missing.html`

[19] G.F. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Machine Learning, 9 (1992) 309-347.

[20] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, 1990

[21] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman, Indexing by Latent Semantic Analysis, Journal of the American Society of Information Science, 41 (1990) 391-407, `citeseer.nj.nec.com/deerwester90indexing.html`

[22] M. Dittenbach, D. Merkl, A. Rauber, The growing hierarchical self-organizing map, In: Proc. International Joint Conf Neural Networks (IJCNN00), Como, Italy, 2000

[23] M. Dittenbach, A. Rauber, D. Merkl, Uncovering hierarchical structure in data using the Growing Hierarchical Self-Organizing Map. Neurocomputing 48 (2002) 199-216.

[24] D. Dubois, H. Prade, Fuzzy Sets and Systems. Theory and Applications, Academic Press, 1980

[25] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine Learning, 29 (1997) 131-163.

[26] B. Fritzke, A growing neural gas network learns topologies, In: G. Tesauro, D.S. Touretzky, and T.K. Leen (Eds.) Advances in Neural Information Processing Systems 7, MIT Press Cambridge, MA, 1995, pp. 625-632.

[27] B. Fritzke, Some competitive learning methods, draft available from `http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper`

[28] B. Fritzke, A self-organizing network that can follow non-stationary distributions, In: Proceeding of the International Conference on Artificial Neural Networks '97, Springer, 1997, pp.613-618

[29] J.J. deGruijter, A.B. McBratney, A modified fuzzy k means for predictive classification. In: Bock,H.H.(ed) Classification and Related Methods of Data Analysis. Elsevier Science, Amsterdam, 1988, pp. 97-104.

[30] M. Halkidi, Y. Batistakis, M. Vazirgiannis: On clustering validation techniques. Journal of Intelligent Information Systems, 17 (2001) 107-145

[31] J.A. Hartigan, Clustering Algorithms. John Wiley & Sons, New York 1975.

[32] T. Hoffmann, Probabilistic Latent Semantic Analysis, In: Proceedings of the 15th Conference on Uncertainty in AI, 1999

[33] C. Hung, S. Wermter, A constructive and hierarchical self-organising model in a non-stationary environment, In: Proc. of the International Joint Conference in Neural Networks, 2005

[34] S.Y. Jung, K. Taek-Soo, An Incremental similarity computation method in agglomerative hierarchical clustering, Journal of Fuzzy Logic and Intelligent System, December, 2001

[35] S. Kaski, Dimensionality reduction by random mapping: fast similarity computation for clustering, In: Proceedings of International Joint Conference of Neural Networks, vol.1, 1998

[36] M.A. Kłopotek, S.T. Wierzchoń, M. Michalewicz, M. Bednarczyk, W. Pawlowski, A. Wasowski, Bayesian network mining system, In: Proc. X International Symposium on Intelligent Information Systems, Springer 2001, pp. 97-110

[37] M.A. Kłopotek, A new Bayesian tree learning method with reduced time and space complexity. Fundamenta Informaticae, 49 (2002) 349-367

[38] M.A. Kłopotek, Intelligent information retrieval on the Web. In: P.S. Szczepaniak, J. Segovia, J. Kacprzyk, L.A. Zadeh, (eds.), Intelligent Exploration of the Web, Springer, 2003, pp. 57-73 3-7908-1529-2,

[39] M.A. Kłopotek, M. Dramiński, K. Ciesielski, M. Kujawiak, S.T. Wierzchoń, Mining document maps, In Proceedings of Statistical Approaches to Web Mining Workshop (SAWM) at PKDD'04, M. Gori, M. Celi, M. Nanni eds., Pisa, 2004, pp.87-98

[40] M.A. Kłopotek, S.T. Wierzchoń, K. Ciesielski, M. Dramiński, D. Czerski, M. Kujawiak, Understanding nature of map representation of document collections – map quality measurements, In: Proc. Int.Conf. Artificial Intelligence Siedlce, September 2005. pp. 85-92.

[41] M.Kłopotek, S.Wierzchoń, K.Ciesielski, M.Dramiński, D.Czerski, Coexistence of Crisp and Fuzzy Concepts in Document Maps, In: W. Duch, J. Kacprzyk, (eds), Proc. Int. Conf. ICAINN'2005, LNCS 3697, Springer 2005, part II pp. 859

[42] M.A. Kłopotek, S.T. Wierzchoń, K. Ciesielski, M. Dramiński, D. Czerski, Conceptual maps and intelligent navigation in document space (in Polish), to appear in: Akademicka Oficyna Wydawnicza EXIT Publishing, Warszawa, 2006

[43] T. Kohonen, Self-Organizing Maps, Springer Series in Information Sciences, vol. 30, Springer, Berlin, Heidelberg, New York, 2001

[44] T. Kohonen, S. Kaski, P. Somervuo, K. Lagus, M. Oja, V. Paatero, Self-organization of very large document collections, Helsinki University of Technology technical report, 2003, `http://www.cis.hut.fi/research/reports/biennial02-03`

[45] P. Koikkalainen, E. Oja, Self-organizing hierarchical feature maps, In: Proc. International Joint Conference on Neural Networks, San Diego, CA ,1990

[46] K. Lagus, Text Mining with WebSOM, PhD Thesis, Helsinki University of Technology, 2000

[47] B. C., K. Livesay, and K. Lund. Explorations in context space: Words, sentences, discourse. Discourse Processes, 25 (1998) 211-257

[48] H.S. Loos, B. Frizke, DemoGNG v.1.5, 1998, available at `http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/NG_2.html`

[49] J.B. Lovins, Development of a stemming algorithm. Mechanical Translation and Computational Linguistics, 11 (1968) 22-31

[50] J.Ontrup, H. Ritter, Hyperbolic self-organizing maps for semantic navigation, in: Proceedings of NIPS-2001

[51] J. Ontrup, H. Ritter, Text Categorization and Semantic Browsing with Self-Organizng maps on non-euclidean Spaces, PKDD-01 preprint, 2001

[52] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo CA, 1988.

[53] M.F. Porter, An algorithm for suffix stripping, 1980, Program, 14(3) :130-137. It has since been reprinted in Sparck Jones,

Karen, and Peter Willet, 1997, Readings in Information Retrieval, San Francisco: Morgan Kaufmann, ISBN 1-55860-454-4, `http://www.tartarus.org/~martin/PorterStemmer`

[54] W.M. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical recipes: The art of scientific computing. New York, NY: Cambridge University Press, 1986

[55] A. Rauber, Cluster Visualization in Unsupervised Neural Networks. Diplomarbeit, Technische Universität Wien, Austria, 1996

[56] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction and Search, Lecture Notes in Statistics 81, Springer-Verlag, 1993.

[57] J. Timmis, aiVIS: Artificial immune network visualization, In: Proceedings of EuroGraphics UK 2001 Conference, Univeristy College London 2001, pp.61-69

[58] S.T. Wierzchoń, Artificial immune systems. Theory and applications (in Polish), Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2001

[59] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning, 38 (2000) 257-286

[60] J.A. Wise, J. Thomas, J., K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: Spatial analysis and interaction with information from text documents, In IEEE Information Visualization. 1995. p. 51-58.

[61] R.R. Yager, D.P. Filev, Approximate clustering via mountain method, IEEE Trans. on Systems, Man and Cybernetics, 24 (1994) 1279-1284

[62] Y. Yang, X. Liu, A re-examination of text categorization methods, In: 22nd Annual International SIGIR, Berkley, 1999, pp. 42-49

[63] A.H. Youssefi, D.J. Duke, M.J. Zaki, Visual Web Mining, `http://www2004.org/proceedings/docs/2p394.pdf`, WWW2004, May 17–22, 2004, New York, NY USA.

[64] C.T. Zahn, Graph-theoretical methods for detecting and describing Gestalt clusters, IEEE Transactions on Computers, vol. C-20, no. 1, 1971

[65] T. Zhang, R. Ramakrishan, M. Livny, BIRCH: Efficient data clustering method for large databases, in: Proceedings of ACM SIGMOD International Conference on Data Management, 1997

[66] Y. Zhao, G. Karypis, Criterion functions for document clustering: Experiments and analysis, available at `http://www-users.cs.umn.edu/~karypis/publications/ir.html`

[67] English stop words, `http://www.engin.umich.edu/caen/wls/software/oracle/text.901/a90121/astopsu2.htm`

[68] German stop words, `http://www.engin.umich.edu/caen/wls/software/oracle/text.901/a90121/astopsu7.htm`

[69] An implementation of Finite State Automatons in Java, Dawid Weiss, `http://www.cs.put.poznan.pl/dweiss/xml/projects/fsa/index.xml?lang=en`

[70] An implementation of Finite State Automatons in Java, Jan Daciuk, `http://www.eti.pg.gda.pl/~jandac/fsa.html`

[71] Lovins stemmer, developed by Eibe Frank, `http://www.cs.waikato.ac.nz/~eibe/stemmers`

[72] A Fast and Simple Stemming Algorithm for German Words, Jörg Caumanns, developed by Gerhard Schwarz, `http://lucene.apache.org/`

[73] Syskill and Webert Web Page Ratings, `http://kdd.ics.uci.edu/databases/SyskillWebert/SyskillWebert.html`

Index

List of Figures