SCIENTIFIC DOCUMENTS MANAGEMENT SYSTEM
Application of Kohonens Neural Networks with Reinforcement in Keywords Extraction

Błażej Zyglarski
Faculty of Mathematics And Computer Science, Nicolaus Copernicus University, Chopina 12/18, Toruń, Poland

Piotr Bała
Faculty of Mathematics And Computer Science, Nicolaus Copernicus University, Chopina 12/18, Toruń, Poland
ICM, Warsaw University, ul.Pawinskiego 5a, Warsaw, Poland

Keywords: Keywords extraction.

Abstract: Keywords choice is an important issue during the document analysis. We've developed the document management system which performs the keywords-oriented document comparison. In this article we presents our approach to keywords generation, which uses self-organizing Kohonen neural networks with reinforcement. This article shows the relevance of this method to the standard statistical method.

1 INTRODUCTION

This article contains complete description and comparison of two approaches of keywords choosing: the statistical one and the neural network based one. Second one consists of two types: the simple neural network and the neural network with the reinforcement. In both cases we are using as the analysis example the same input string built over a two letters alphabet, which is "aab abb abaa abb bbba abb bbaa baa bbaa bbbb bbba bbbb". The last part of this document shows effectiveness of the reinforced learning and other algorithms. All presented approaches was tested with example set of about 200 documents.

2 THE STATISTICAL APPROACH

The standard approach to selecting keywords is to count their appearance in the input text. As it was described in narimura, most of frequent words are irrelevant. The next step is deleting trash words with use of specific word lists.

In order to speed up the trivial statistical algorithm for keywords generation, weve developed the algorithm which constructs the tree for text extracted from a document. The tree is built letter by letter and each letter is read exactly once, which guarantees that the presented algorithm works in a linear time.

2.1 An Algorithm

Denote the plain text extracted from an i-th document as $T(i)$. Let $\widehat{T}(i)$ be a lowercase text devoid of punctuation. In order to select appropriate keywords, we need to build a tree (denoted as $\tau(\widehat{T}(i))$).

1. Let r be a root of $\tau(\widehat{T}(i))$. Let p be a pointer, pointing at r
$$p = r \tag{1}$$
2. Read a letter a from the source text.
$$a = ReadLetter(\widehat{T}(i)) \tag{2}$$
3. If there exists an edge labeled by a, leaving a vertex pointed by p, and coming into some vertex s then
$$p = s \tag{3}$$
$$count(p) = count(p) + 1 \tag{4}$$
else, create a new vertex s and a new edge labeled by a leaving vertex p and coming into vertex s
$$p = s \tag{5}$$
$$counter(p) = 1 \tag{6}$$

Zyglarski B. and Bała P. (2009).
SCIENTIFIC DOCUMENTS MANAGEMENT SYSTEM - Application of Kohonens Neural Networks with Reinforcement in Keywords Extraction.
In *Proceedings of the International Conference on Knowledge Management and Information Sharing*, pages 55-62
DOI: 10.5220/0002298100550062

4. If a was a white space character then go back with the pointer to the root

$$p = r \tag{7}$$

5. If $\widehat{T}(i)$ is not empty, then go to step 2.

After these steps every leaf l of $\tau(\widehat{T}(i))$ represents some word found in the document and $count(l)$ denotes appearance frequency of this word. In most cases the most frequent words in every text are irrelevant. We need to subtract them from the result. This goal is achieved by creating a tree ρ which contains trash words.

1. For every document i

$$\rho = \rho + \tau(\widehat{T}(i)) \tag{8}$$

Every time a new document is analyzed by the system, ρ is extended. It means, that ρ contains most frequent words all over files which implies that these words are irrelevant (according to different subjects of documents). The system is also learning new unimportant patterns. With increase of analyzed documents, accuracy of choosing trash words improves.

Let $\Theta(\rho)$ be a set of words represented by $\Theta(\rho)$. We should denote as a trash word a word, which frequency is higher than median m of frequencies of words which belongs to $\Theta(\rho)$.

1. For every leaf $l \in \tau(\widehat{T}(i))$ If there exist $z \in \rho$, such as z and l represents the same word and $count(z) > count(m)$ then

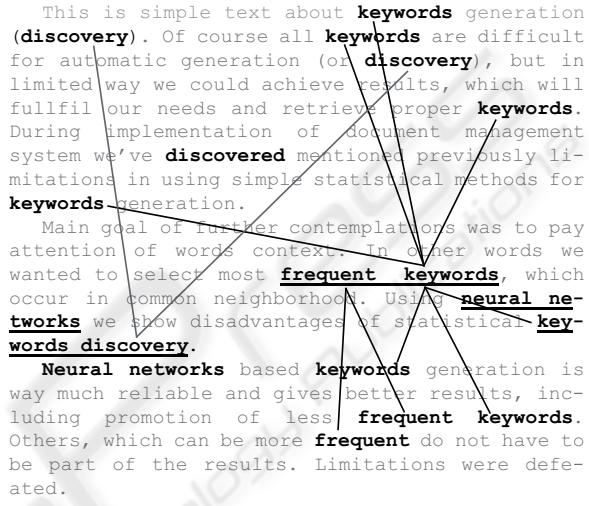
$$count(l) = 0 \tag{9}$$

2.2 Limitations

Main limitation of this type of generating keywords is loosing the context of keywords occurrences. It means that words which are most frequent (which implies that they are appearing together on the result list) could be actually not related. The improvement for this issue is to pay the attention at the context of keywords. It could be achieved by using neural networks for grouping words into locally closest sets. In our approach we can select words which are less frequent, but their placement indicates, that they are important. Finally we can give them a better position at the result list.

3 KOHONEN'S NEURAL NETWORKS APPROACH

During implementation of the document management system (Zyglarski et al., 2008) weve discovered previously mentioned limitations in using simple statistical methods for keywords generation. Main goal of further contemplations was to pay attention of the word context. In other words we wanted to select most frequent words, which occur in common neighborhood. Table 1 presents an example text, with discovered keywords. Presented algorithm selects the most frequent words, which are close enough. It is achieved with a words categorization (Frank et al., 2000).


This is simple text about **keywords** generation (**discovery**). Of course all **keywords** are difficult for automatic generation (or **discovery**), but in limited way we could achieve results, which will fullfil our needs and retrieve proper **keywords**. During implementation of document management system we've **discovered** mentioned previously limitations in using simple statistical methods for **keywords** generation.

Main goal of further contemplations was to pay attention of words context. In other words we wanted to select most **frequent keywords**, which occur in common neighborhood. Using **neural networks** we show disadvantages of statistical **keywords discovery**.

Neural networks based **keywords** generation is way much reliable and gives better results, including promotion of less **frequent keywords**. Others, which can be more **frequent** do not have to be part of the results. Limitations were defeated.

Figure 1: Example keywords selection.

Precise results are shown in the Table 1. Each keyword is presented as a pair (a keyword, a count). Keywords are divided into categories with use of Kohonen neural networks. Each category has assigned rank, which is related to number of gathered keywords and their frequency).

Results were filtered using words from ρ defined in first part of this article.

3.1 An Algorithm

Algorithm of neural networks based keyword discovery consists of 3 parts. At the beginning we have to compute distances between all words. Then we have to discover categories and assign proper words to them. At the end we need to compute the rank od each category.

3.1.1 Counting Distances

Lets $\widehat{T}^f = \widehat{T}(f)$ be a text extracted from a document f and $\widehat{T}^f(i)$ be a word placed at the position i in this text. Lets denote the distance between words A and B

Table 1: Example keywords selection.

Category	Rank	Keyword	Count
1	0,94	keywords	8
		discovery	3
		frequent	3
		simple	2
		text	1
		automatic	1
2	0,58	neural	2
		networks	2
		discovered	1
		neighborhood	1
3	0,50	fulfill	1
4	0,41	statistical	2
		words	2
		contemplations	1
		promotion	1
		disadvantages	1
5	0,36	retrieve	1
		reliable	1

within the text $\widehat{T}^f$ as $\delta^f(A,B)$.

$$\delta^f(A,B) = \quad (10)$$

$$= \min_{i,j\in\{1,..,n\}} \{\|i-j\|; A = \widehat{T}^f(i) \wedge B = \widehat{T}^f(j)\} \quad (11)$$

By the position i we need to understand a number of white characters read so far during reading text. Every sentence delimiter is treated like a certain amount W of white characters in order to avoid combining words from separate sequences (we empirically chose $W = 10$). Weve modified previously mentioned tree generation algorithm. Analogically we are constructing the tree, but every time we reached the leaf (which represents a word) we are updating the distances matrix M (presented on figure 2), which is $n \times n$ upper-triangle matrix and $n \in N$ means number of distinct words read so far.

ξ^f	*1*	*2*	...	*k*	...	*n*
1	0			*		
2		0		*		
...			...	*		
k				0*	*	*
...					...	
n						0

Figure 2: Distances matrix.

Generated tree (presented on figure 3) is used for checking previous appearance of actually read word, assigning the word identifier (denoted as $\xi^f(j) = \xi(\widehat{T}^f(j))$,$\xi^f(j) \in \mathbf{N}$) and to decide whether update existing matrix elements or increase matrix's dimension by adding new row and new column. Figure 2 marks with dashed borders fields updateable in the first case. For updating existing matrix elements and counting new ones we also use the array with positions of last occurrences of all read words. Lets denote this array as $\lambda(\xi^f(j))$

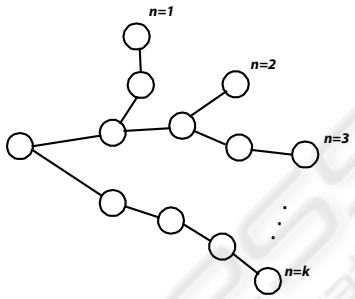


Figure 3: Word tree.

We need to consider two cases:

1. Lets assume that $j-1$ words was read from input text and a word with position j is read for the first time.

$$\forall_{i<j} \widehat{T}^f(j) \neq \widehat{T}^f(i) \quad (12)$$

$$\lambda(\xi^f(j)) = j \quad (13)$$

$$\forall_{k\in\{1,...,\xi^f j\}} \mathbf{M}[k, \xi^f(j)] = |j - \lambda(k)| \quad (14)$$

This case is shown on figures 4 and 5, which contains a visualization of the state of am algorithm after reading three first words from example string "aab abb abaa — abb bbba abb bbaa baa bbaa bbbb bbba bbbb". At the figure 5 there is also presented the table of last occurrences.

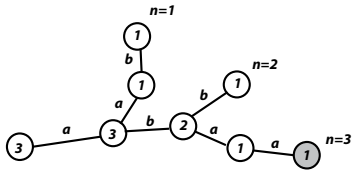


Figure 4: The word tree generated after reading first three words.

	$\lambda(\xi^f)$	1	2	3
	ξ^f	*1*	*2*	*3*
aab	*1*	0	1	2
abb	*2*		0	1
abaa	*3*			0

Figure 5: The distance matrix generated after reading first three words.

2. Lets assume that $j-1$ words was read from input text and a word with position j was read ealier and already has given a worde identifier.

$$\exists_{i<j}\widehat{T}^f(j)=\widehat{T}^f(i) \quad (15)$$

We have to update the last occurrences table by

$$\lambda(\xi^f(j))=j \quad (16)$$

and update appropriate row and column in existing matrix

$$\forall_{k\in\{1,\ldots,\xi^f(j)-1\}}M[k,\xi^f(j)]=\Delta(k,\xi^f(j)) \quad (17)$$

$$\forall_{k\in\{\xi^f(j)+1,\ldots,\widehat{\xi}^f\}}M[\xi^f(j),k]=\Delta(\xi^f(j),k) \quad (18)$$

where

$$\widehat{\xi}^f=\max_{i=\{1,\ldots,j\}}\{\xi^f(i)\} \quad (19)$$

$$\Delta(k,l)=min(|\lambda(k)-\lambda(l)|,M[k,l]) \quad (20)$$

This case is shown on figures 6 and 7, which contains visualization of the state of the algorithm after reading four first words from an example string "aab abb abaa abb — bbba abb bbaa baa bbaa bbbb bbba bbbb". In this case a word "abb" was read twice, so we need to update λ array and the actual distance matrix M.

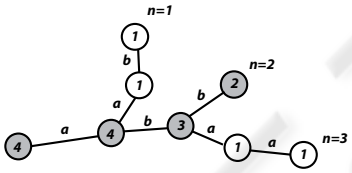


Figure 6: The word tree generated after reading first four words.

$\lambda(\xi^f)$	1	**4**	3
ξ^f	*1*	*2*	*3*
aab *1*	0	1	**2**
abb *2*		0	**1**
abaa *3*			0

Figure 7: The distance matrix generated after reading first four words.

Bold-faced fields indicates values, that could be evaluated each time using λ array and dont have to be memorized. According to that observation we can omit the distance table, using instead of it only specific lists (Figure 10), containing these elements, which cannot be evaluated with λ array. Final results of our example (after reading all words from string "aab abb abaa abb bbba abb bbaa baa bbaa bbbb bbba bbbb —") are shown on figures 8 and 9.

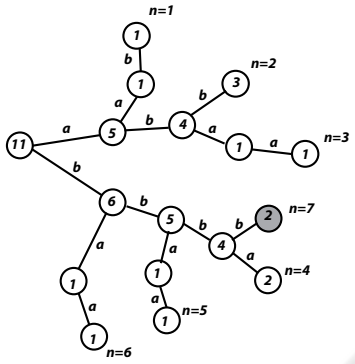


Figure 8: The word tree generated after reading all words.

$\lambda(\xi^f)$	1	6	3	11	9	8	12
ξ^f	*1*	*2*	*3*	*4*	*5*	*6*	*7*
aab *1*	0	1	**2**	4	6	**7**	9
abb *2*		0	1	1	1	**2**	4
abaa *3*			0	2	4	**5**	7
bbba *4*				0	**2**	**3**	**1**
bbaa *5*					0	**1**	1
baa *6*						0	2
bbbb *7*							0

Figure 9: The distance matrix generated after reading all words.

Presented algorithm guarantees that its result matrix is a matrix of shortest distances between words.

$$\delta^f(A,B)=M[\xi(A),\xi(B)] \quad (21)$$

3.1.2 Generating Categories

The knowledge of distances between words in document is used to categorize them with use of the self-organizing Kohonen Neural Network (Figure 11) (Kohonen, 1998).

This procedure takes 5 steps:

1. Create rectangular $m\times m$ network, where $m=\lfloor\sqrt[4]{\widehat{\xi}^f}\rfloor$. Presented algorithm can distinguish maximally m^2 categories. Every node (denoted as $\omega_{x,y}$) is connected with four neighbors and contains a prototype word (denoted as $\widehat{T}^f_\omega(x,y)$) and a set (denoted as $\beta^f_\omega(x,y)$) of locally close words.
2. For each node choose random prototype of the category $p\in\{1,2,\ldots,\widehat{\xi}^f\}$.
3. For each word $k\in\{1,2,\ldots,\widehat{\xi}^f\}$ choose closest prototype $\widehat{T}^f_\omega(x,y)$ in network and add it to list $\beta^f_\omega(x,y)$.

$\lambda(\xi^f)$	$\lambda(1)$	$\lambda(2)$	...	$\lambda(\widehat{\xi}^f)$
ξ^f	*1*	*2*	...	$\widehat{\xi}^f$
	$\delta^f(1,u_{1,1})$	$\delta^f(1,u_{2,1})$	...	$\delta^f(1,u_{\widehat{\xi}^f,1})$
	$\delta^f(1,u_{1,2})$	$\delta^f(1,u_{2,2})$	...	$\delta^f(1,u_{\widehat{\xi}^f,2})$
	...	...	...	...
	$\delta^f(1,u_{1,k_1})$	$\delta^f(1,u_{2,k_2})$	...	$\delta^f(1,u_{\widehat{\xi}^f,k_{\widehat{\xi}^f}})$

Figure 10: The scheme of the distance table.

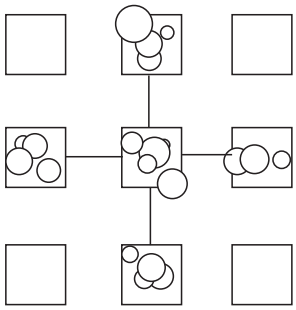


Figure 11: The scheme of the neural network for words categorization with selected element and its neighbors.

4. For each network node $\omega_{x,y}$ compute a generalized median for words from $\beta^f_\omega(x,y)$ and neighbors lists (denoted as β). A generalized median is defined as an element A which minimizes a function:
$$\Sigma_{B\in\beta}\delta^{f2}(A,B) \tag{22}$$
Set $\widehat{T}^f_\omega(x,y) = A$

5. Repeat step 4 until the network is stable. Stability of the network is achieved, when in two following iterations all word lists are unchanged (without paying attention to iternal lists sturcture and their position in nodes).

Such algorithm divides the set of all words found in document into separate subsets, which contains only locally close words. In other words it groups words into related sets (see Figure 12).

3.1.3 Selecting Keywords

After execution of described procedure all words from the analyzed document are divided into categories. We need to choose most important categories and then select most frequent words among them. Each category contains list of words, which are locally close. As a category rank we could take

$$\rho_{x,y} = \frac{\Sigma_{w\in\beta^f_\omega(x,y)} count(w)}{|\beta^f_\omega(x,y)|} \tag{23}$$

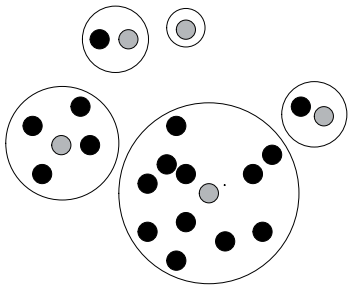


Figure 12: The two dimensional example of Kohonen network grouping locally closest words (small circles) into different categories.

Now ordering categories descending by $\rho_{x,y}$ we can select from each a number of most frequent words. This method allows selecting words, which appearance isnt most frequent in the complete scope of a document, but is frequent enough in some sub-scope. The sample results are shown in table 1. The table presents first eighteen results presented method.

4 THE NEURAL NETWORK REINFORCEMENT

At this stage of work we've decided to extend an idea of presented algorithm with use of a reinforcement for improving it's accuracy. The reinforcement is performed with use of presented in (Zyglarski et al., 2008) system, in which we have created huge repository of documents. Articles used for testing algorithms were also put into this repository. It means that all of them were compared using statistics of words, statistics of n-grams (Cavnar and Trenkle, 1994) (in this particular case 3-grams) and Kolmogorov Complexity (Fortnow, 2004). Three kinds of distances between documents are computed.

Table 2: The comparison of results for the example text.

Statistical		Kohonen		Reinforced	
words	40	*words*	40	*keyword*	25
keywords	25	*keywords*	25	*distance*	7
text	22	*neural*	13	*neural*	13
neural	13	*distances*	3	*statistical*	10
abb	13	*text*	22	*context*	4
$\widehat{T}^f$	11	simple	5	matrix	11
matrix	11	elements	2	denoted	4
tree	9	reading	8	*vertex*	3
frequent	9	extracted	3	web	2
$\widehat{T}$	9	$\widehat{T}^f$	11	*relevant*	2
bbbb	8	matrix	11	string	3
networks	8	*distance*	7	*extracted*	3
extracted	3	*discovery*	5	*networks*	8

4.1 An Idea of the Reinforcement

Main idea of the reinforcement (Sutton, 1996) is to modify a behavior of the neural network depending of a weight of a keywords candidate. At the beginning we need to initiate the weight attribute (with values from interval [0,1]) of every word from a document. Each word, which can be found in previously mentioned trash word set has weight equal to 0, rest of them have weights equal to 1. The neural network algorithm is modified in such way, that words with smaller weight are pushed away from words with greater weight. It means that they are pushed aside the main categories. Of course they will have also small influence on category rank. Moreover we need to add parent iteration, which will modify weights of words and repeat neural network steps until proper words will be selected. After performance of word categorization a set of proposed keywords is generated. At this stage we need to check every keyword for it's accuracy. This is performed by checking a number (in our tests it was 10) of articles (containing tested keyword) randomly selected from repository and comparing normalized distances between them and the analyzed document. If these documents are relatively close (in the terms of counted distances) to initial one, a keyword is prized with increasing it's weight. If distances are relatively far, weight is decreased. In other words, if an selected keyword is good, a network is rewarded. With this improvement, algorithm continues with steps of neural network learning.

4.2 The Results Propriety

Methodology of creating repository, which is described in (Zyglarski et al., 2008) guarantees, that collected documents has various subjects. They are gathered with using of most frequent words appeared in each document. Additional variety is an result of the collection which initiates repository - containing articles from various areas of interests. It means that there is a big chance, for articles containing tested keyword to be really connected with the same subject. If a keyword candidate isn't really a keyword, these documents will probably differ from tested one and network will not be reinforced.

5 THE COMPARISON OF RESULTS

Presented method gives better results than the simple statistical method. In table 2 we show keywords found over this article, chosen with using all three methods (with italic font there are marked actual (subjectively selected by authors) keywords). It's clear that Kohonen Networks related methods gives better results than statistical method and also reinforcement has very good influence on final results.

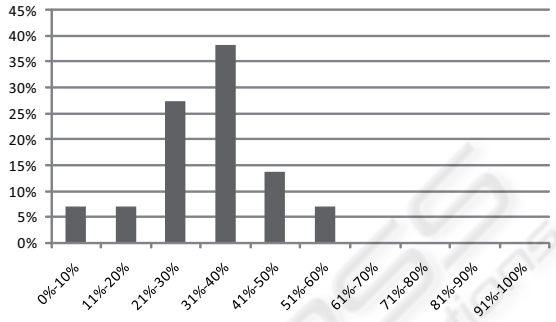


Figure 13: Effects of statistical method. X axis shows effectiveness, Y axis shows number of documents with processed with this effectiveness.

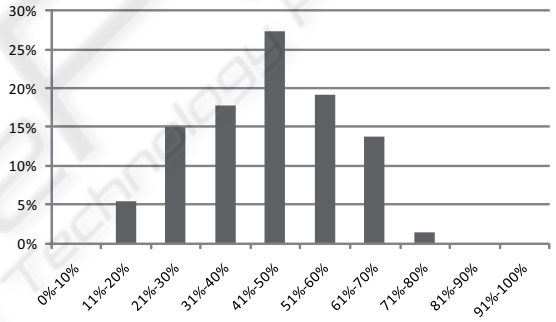


Figure 14: Effects of neural network method. X axis shows effectiveness, Y axis shows number of documents with processed with this effectiveness.

In our tests we've used about 200 various articles. In most cases results given by our approach was more accurate than other approaches. The accuracy was checked manually and is subjective. Finally, according to executed tests, statistical methods gave very poor results (see Figure 13). In the best case list of proposed keywords achieved 65% accuracy. In the worst case it was about 5%. At the figure 13 there are presented accuracies of results from tested articles (for example: in 66% of articles accuracy of keywords was at level between 20% and 40%). Better results archived with Kohonen Networks without the reinforcement are presented at the figure 15). In the best case, list of proposed keywords achieved almost 80% accuracy. 10%-40% accuracy was in this case very less often.

The best result were generated with using Reinforced Kohonen Networks, where best results reached

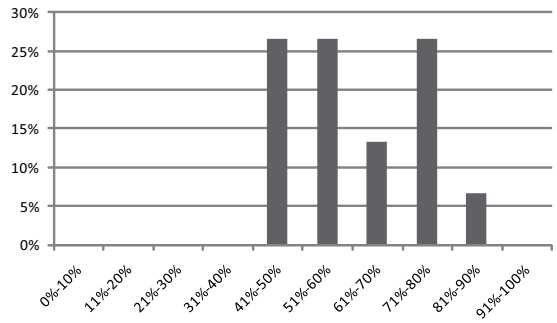


Figure 15: Effects of neural network method with reinforcement. X axis shows effectiveness, Y axis shows number of documents with processed with this effectiveness.

level of 90% accuracy and results between 10%-40% were completely eliminated. Comparison of effectiveness of all presented methods is shown on fig. 16.

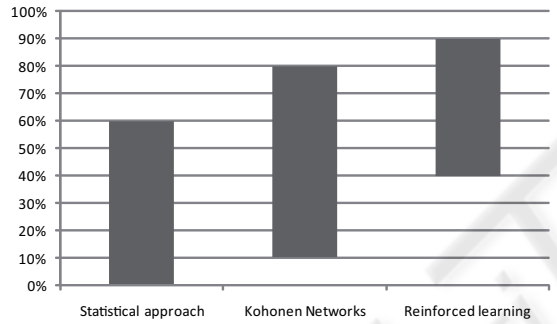


Figure 16: Comparison of presented methods.

6 DOCUMENT MANAGEMENT SYSTEM IMPLEMENTATION

The Web Services based Scientific Article Manager (Zyglarski et al., 2008) has been designed to provide the knowledge management for the users using journal articles and internet documents as main sources of information. The journal articles are usually retrieved in PDF format, however the content can be processed and analyzed automatically (excluding relatively small number of documents which are stored as direct scans). Figure 17 presents the architecture of learning part of the system. This article shows details of the gray shaded part of the figure.

During selection of documents connected to examined article, there are used three different methods of comparison, which are finally combined. It guarantees accuracy of obtained results, which pay attention to content and structure of documents:

1. Statistical based, performed by checking apperance of all words in two documents. Each word is treated as a dimension and a taxicab metric is used for counting distances betweeen documents. In this case all punctuation and white characters are excluded.
2. N-grams based, performed by checking apperance of all n-grams in two documents. Each n-gram is treated as a dimension, analogicaly to previous method.
3. Kolmogorov distance based. Kolmogorov complexity of text X (denoted as $K(X)$) is length of the shortest compressed binary file $X*$, from which original text can be reconstructed. Formally Kolmogorov Complexity $K(x)$ is defined as length of the smallest program running on Turing Machine, which returns word x . In our case Turing Machine is substituted by compression program and compressed file responds to the Turing program.

Presented system is implemented with Java language and provides data with use of the webservices technology. We've also implemented three kinds of user interfaces:

1. Servlet based;
2. Portlet based (used with Liferay portal);
3. .Net Windows client (used with Windows explorer, in future intended to work seamlessly within Windows explorer)

The users are about to share, and access each other documents, with apropriate permission subsystem.

7 CONCLUSIONS

Presented approach gives in most cases very good results of the context aware keywords recognition. Efectiveness is related with size of the repository of available documents, which are used for reinforcement of the neural network. It means, they are more effective while working with large data collections. Moreover continuously working kohonen network can improve the keywords recognition every time new documents are added to the repository. If neural network is stable, any change of keywords weights implies reorganisation of this network, which (according to size of performed changes - only weights in proposed keywords are changed) is in most cases quite fast. Most of the processing time is spent to check accuracy of proposed keywords. Fortunately, results of this check are used also for categorization of documents collected in our Document Management System.

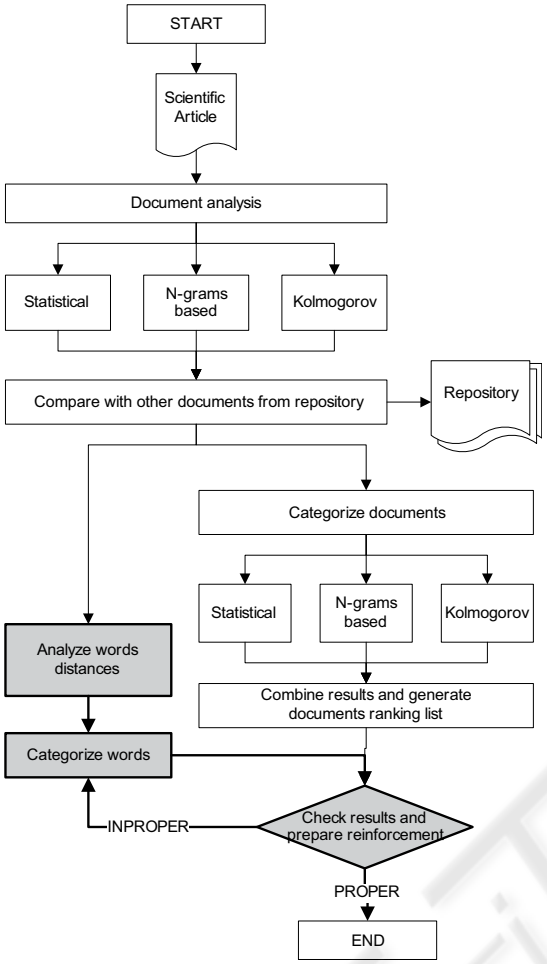


Figure 17: The Document Management System implementation.

REFERENCES

Cavnar, W. B. and Trenkle, J. M. (1994). N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

Fortnow, L. (2004). Kolmogorov complexity and computational complexity.

Frank, E., Chui, C., and Witten, I. H. (2000). Text categorization using compression models. In *In Proceedings of DCC-00, IEEE Data Compression Conference, Snowbird, US*, pages 200–209. IEEE Computer Society Press.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21(1-3):1–6.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, volume 8, pages 1038–1044.

Zyglarski, B., Baa, P., and Schreiber, T. (2008). Web services based scientific article manager. In *Information Systems Architecture and Technology, Web Information Systems: Models, Concepts and Challenges*, pages 205–215. Wrocaw University of Technology.

APPENDIX

Scientific work cofunded from resources of European Social Fund and Budget within ZPORR (Zintegrowany Program Operacyjny Rozwoju Regionalnego), Operation 2.6 ”Regional Innovation Strategies and knowledge transfer” of Kujawsko-Pomorskie voivodship project ”Scholarship for PhD students 2008/2009 - ZPORR”.