KEYWORDS EXTRACTION
Selecting Keywords in Natural Language Texts with Markov Chains and Neural Networks

Błażej Zyglarski and Piotr Bała
Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Chopina St. 13/18, Torun, Poland

Keywords: Keywords extraction, Text mining, Neural networks, Kohonen, Markov chains.

Abstract: In this paper we show our approach to keywords extraction by natural language processing. We present revised and extended version of previously shown document analysis method, based on Khonen Neural Networks with Reinforcement, which uses data from the large document repository to check and improve results. We describe new improvements, which we've achieved with preprocessing set of words and creating initial ranking using Markov Chains. Our method shows, that keywords can be selected from the text with great accuracy. In this paper we present evaluation and comparison of both methods and example results of keywords selection upon random documents.

1 INTRODUCTION

Large increase of available data can be noticed nowadays. Modern search engines use simple language querying and return results containing query words. Most of the results are unfortunately inadequate to the question. It shows that very important modern computer science problems is automatic analysis of data, which is able to pinpoint most important parts of data. Problem is easier, if the data is structurised and the structure is known (e.g. specific databases). Challenge is to find the structure in "free language". We have developed algorithms, which can analyse text with use of structual distance between words and Kohonen Neural Networks with Reinforcement based on previously computed distances between documents in the large repository (see (Zyglarski and Bała, 2009)). It means that effective keywords selection needs a large knowledge about background information such as other documents. This approach is similar to human learning, which can be more effective, if a person knows more. In this article we introduce extended version of this algorithm, imporooved by text preprocessing and computing initial words weights (instead of using 1 as startup weights) with use of Markov Chains built on idea of PageRank.

2 NEURAL NETWORKS CATEGORIZATION ALGORITHM

Our idea of keywords selection used a distance between words defineded as:

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

$$\delta^f(A,B) = \quad (1)$$

$$= \min_{i,j\in\{1,..,n\}} \{\|i-j\|; A = \widehat{T}^f(i) \wedge B = \widehat{T}^f(j)\} \quad (2)$$

By the position i we need to understand a number of additional characters (such as space, dots and commas) 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. After tests we choose differend W for different additional characters, as is shown in table 1

The knowledge of distances between words in document was used to categorize them with use of the self-organizing Kohonen Neural Network (figure 1) (Kohonen, 1998).

The categorization procedure consisted of 5 steps:

1. Create rectangular $m \times m$ network, where $m = \lfloor \sqrt[4]{n} \rfloor$, where n is number of all words Presented

Zyglarski B. and Bała P..
KEYWORDS EXTRACTION - Selecting Keywords in Natural Language Texts with Markov Chains and Neural Networks.
DOI: 10.5220/0003088003150321
In *Proceedings of the International Conference on Knowledge Management and Information Sharing* (KMIS-2010), pages 315-321
ISBN: 978-989-8425-30-0

Table 1: Weights of additional characters.

Character	Weight	Reason
":",	2	weak relation
";"	10	very weak relation
".!?"	30	no relation
" " and others	1	strong relation

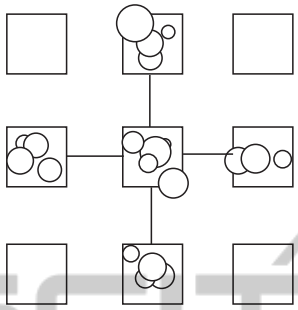

Figure 1: The scheme of the neural network for words categorization with selected element and its neighbors.

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 $p \in \{1,2,...,n\}$.
3. For each word $k \in \{1,2,...,n\}$ choose closest prototype $\widehat{T}^f_\omega(x,y)$ in network and add it to list $\beta^f_\omega(x,y)$.
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:

$$\Sigma_{B\in\beta}\delta^{f2}(A,B) \tag{3}$$

Set $\widehat{T}^f_\omega(x,y) = A$

5. Update word weights, by checking if other documents containing actually best (highest positions in result list) words are in fact related to analysed one:

   For random number of actually best words:

   - check random number of documents containing this word,
   - if more than 66% is close enough to analyzed one, increase weight of tested keyword,
   - if less than 33% is close enough to analyzed one, decrease weight of tested keyword;

6. Repeat, 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 divided the set of all words found in document into separate subsets, which contained only locally close words. In other words it groupped words into related sets. During it's workflow weight of each keyword candidate was updated, by checking if this word can be found in similar documents. Similarity of documents was checked with use of three kinds of distances:

Let's $S_x = \sum_{i=1..n} x_i$ *a* $S_y = \sum_{i=1..n} y_i$. *Let's define distance between documents* $P_1(x_1,x_2,...,x_n)$ *and* $P_2(y_1,y_2,...,y_n)$ *as*

$$d_s((x_1,x_2,...,x_n),(y_1,y_2,...,y_n)) = \tag{4}$$

$$|\frac{x_1}{S_x}-\frac{y_1}{S_y}|+|\frac{x_2}{S_x}-\frac{y_2}{S_y}|+...+|\frac{x_n}{S_x}-\frac{y_n}{S_y}| \tag{5}$$

where x_i, y_i *mean number of occurences of word i id document* P_1 *and* P_2.

Let's define distance between documents $P_1(x_1,x_2,...,x_n)$ *i* $P_2(y_1,y_2,...,y_n)$ *as*

$$d_n((x_1,x_2,...,x_n),(y_1,y_2,...,y_n)) = \tag{6}$$

$$|x_1-y_1|+|x_2-y_2|+...+|x_n-y_n| \tag{7}$$

where x_i, y_i *mean number of occurences of n-gram i ind document* P_1 *and* P_2.

Let's define Kolmogorov distance between documents P_1 *i* P_2 *as*

$$d_k(P_1,P_2) = \tag{8}$$

$$\left|\frac{K(P_1|P_2*)-K(P_2|P_1*)}{K(P_1,P_2)}\right| \tag{9}$$

where $K(-|-)$ *is Conditional Kolmogorov Complexity.*

Document X is close to document Y if X is close to Y in the meaning of d_s, d_n and d_k.

Effectivenes of this method was quite good. Tests show that in some examples correctness of results can achieve even 90%, but in the most cases it is about 50%-70%. Results are shown on figure 2.

3 WEIGHTS PREPROCESSING

One of the assumption of this algorithm was equal startup weights (= 1) of all words. We show, that preprocessing this weight can signifitiantly improove this results. This task can be performed with use of Google PageRank idea ((Avrachenkov and Litvak, 2004)).

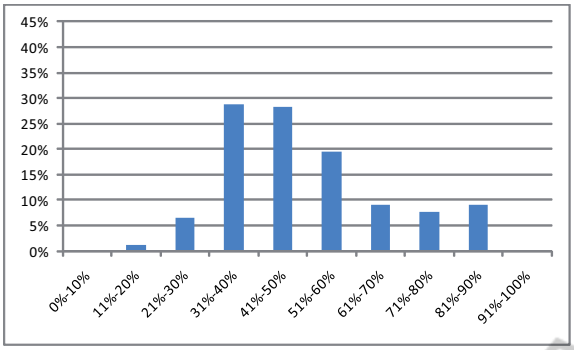

Figure 2: Effects of neural network method with reinforcement. X axis shows effectiveness, Y axis shows number of documents with processed with this effectiveness.

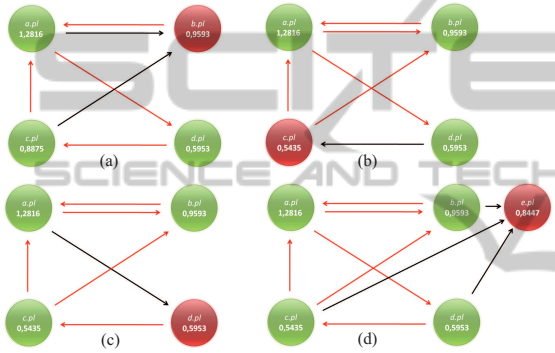

Figure 3: Example PageRank computation.

3.1 PageRank

As states in (Avrachenkov and Litvak, 2004) PageRank bases on knowledge about connections between websites. Simply the site is as important, as important are sites linking to it.

PageRank for every site depends on number of hyperlinks from other sites to considered one. Any hyperlink is interpreted as a vote.

Assuming that $\{X_1, X_2, ..., X_n\}$ contain hyperlinks pointing on page A, and $\|X_i\|$ means number of all hyperlinks on page X_i, $\|X_i\|_A$ means number of hyperlinks pointing from X_i to page A and PR_{X_i} is PageRank of X_i page, and N is number of all websites in the internet, then

$$PR_A = \frac{1-d}{N} + d(PR_{X_1}\frac{\|X_1\|_A}{\|X_1\|} + ... + PR_{X_n}\frac{\|X_n\|_A}{\|X_n\|})$$

PageRank algorithm constructs matrix $P = [p_{ij}]_{i,j\in N}$, where p_{ij} is a probability of moving from page i to page j. Probability p_{ij} equals quotient of number of hyperlinks on page i pointing on page j and number of all hyperlinks on page i, or 0 if there are no hyperlinks from i and j. Then transition matrix $P' = cP + (1-c)(1/n)E$ is defined, where E has all elements equal to 1, and c is probability of concious click on specific hyperlink. $1-c$ means random enter to some page.

Algorithm assumes, that $P(X_o = s_i) = p_i(0) = \frac{1}{n}$ dla $i = 1, ..., n$.

X_k distribution is computed with:

$$[p_1(k-1), p_2(k-1), ..., p_n(k-1)]$$

$$\begin{bmatrix} p'_{11} & p'_{12} & \cdots & p'_{1n} \\ p'_{21} & p'_{22} & \cdots & p'_{2n} \\ \vdots & \vdots & & \vdots \\ p'_{n1} & p'_{n2} & \cdots & p'_{nn} \end{bmatrix}$$

$$= [p_1(k), p_2(k), ..., p_n(k)]$$

Sequence $X_0, X_1, ..., X_n, ...$ fulfill assumption for being Markov Chain and P' is a transition matrix and (X_n) converges to stationary distribution π.

$$\pi(P') = \pi$$

$$\pi 1 = 1$$

3.2 Markov Chains

As states in (Bremaud, 2001) and (Haggstrom, 2002):

- a sequence of random variables $(X_n)_{n=0,...}$ with values in countable set S (state space) is called *Markov Chain* when $\forall_{n\in N}$ and every sequence $s_0, s_1, ..., s_n \in S$

$$P(X_n = s_n | X_{n-1} = s_{n-1}, ..., X_0 = s_0) = \quad (10)$$
$$P(X_n = s_n | X_{n-1} = s_{n-1}) \quad (11)$$

  if only $P(X_{n-1} = s_{n-1}, ..., X_0 = s_0) > 0$, it means that state of chaing in time n (X_n) depends only on state in time $n-1$ (X_{n-1}).

- *Markov property*: Matrix $P = [p_{ij}]_{i,j\in S}$ is called transition matrix on S, if all it's elements are positive, and sum of elements in every row equals 1.

$$p_{ij} \geq 0, \sum_{k\in S} p_{ik} = 1 \quad (12)$$

- Random variable X_0 is called initial state and it's probability distribution $v(i) = P(X_0 = i)$, is called initial distribution.
- Distribution of the Markov Chain depends on initial distribution and transition matrix.

We've decided to use these Markov Chains to create a startup ranking of words inside each document. This ranking is used as input data for main Kohonen Neural Network algorithm. We've called it WordRank.

Table 2: Example keywords.

Example words connections
an ontology, **ontology driven**, **driven similarity**, **similarity algorithm**, algorithm tech, tech report, report kmi, **kmi maria**, **maria vargas**, **vargas vera**, vera and, and enrico, **enrico motta**, motta an, an ontology, **ontology driven**, driven similarity, **driven similarity**, similarity algorithm, knowledge media, media institute, **institute kmi**, kmi the, the open, open university, university walton, **walton hall**, **hall milton**, **milton keynes**,**keynes mk**, **mk aa**, **aa united**, **aa united**, **united kingdom**, kingdom m, m.vargas, vargas vera , vera open, open.ac, ac.uk, **uk abstract**, abstract.this, this paper, paper presents, presents our, our similarity, similarity algorithm, algorithm between, between relations, relations in, in a, a user, user query, query written, written in, in fol, fol first, first order, order logic, logic and, and ontological, **ontological relations**, relations.our, our similarity, similarity algorithm, algorithm takes, takes two, two graphs, graphs and, and produces, produces a, a mapping, mapping between, between elements, elements of, of the, the two, two graphs, graphs i.e, i.e.graphs, **graphs associated**, graphs associated, associated to, to the, the query, query a, a subsection, subsection of, of ontology, **ontology relevant**

3.3 WordRank

While parsing simple text document, one have to find relations between words. Our idea of Word Rank assumes using of simple natural connections between words, based on their position upon the text. Simply we can consider two words as connected, if they are neighbours. Additionally, according to assumptions presented in previous papers, initial statistical analysis of the texts in repository was performed and unimportant words were chosen[1]. They should not be considered during connections analysis and selection connected words. Example set of connected words is presented in table 2. All words, which are not ommited, are potential keywords.

Our procedure takes following steps, shown also on figure 4.

1. Mark all punctuation marks and all unimportant words as division elements. Mark all other words as potential keywords. Lets V be a set of all potential keywords.

[1]Word is unimportant if it is appearing often in all analyzed documents.

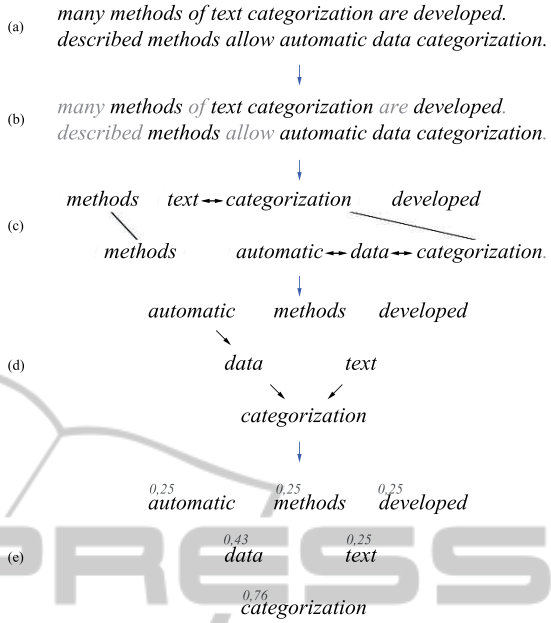

Figure 4: Example graph of connected words

2. Set as connected every two neighbor words, which are not marked as division elements. Consider each connection as bidirectional.
3. Build directed graph $G = (V,E)$.
4. Label every connection with weight from domain $[0,1]$. ($E = VxVx[0,1]$) Let's x and y be two connected words. According to carried out tests, weight of connection between word and its successor should equal 1 and weight of connection between word and its predecessor should equal 0.3.

$$E = E \cup \{(x,y,1)\} \cup \{(y,x,0.3)\} \qquad (13)$$

5. For each word in graph compute it's ranking $\omega() : V \rightarrow [0,2]$.

Now main algorithm of categorization presented in (Zyglarski and Bała, 2009) can categorize those words and prepare final keywords lists.

3.4 Main Part of the Algorithm

Last step of algorithm (*For each word in graph compute it's ranking* $\omega() : V \rightarrow [0,2]$.) is the most important. It is based on the idea of Google PageRank ((Page et al., 1999)), where importance of each website depends on the number of hyperlinks, linking to this website. Similarly in our algorithm, weight of each word depends on weight and number of it's neighbors. By the word we need to understand abstract class of the word (not connected with its position in the text).

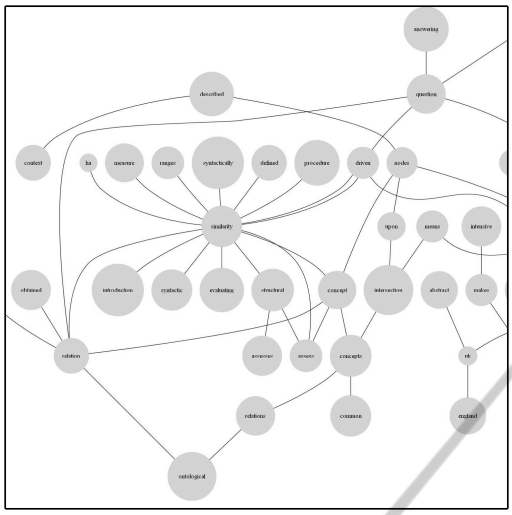

Figure 5: Example workflow for preprocessing weights.

Weight $\omega(y)$ of word y in time t is calculated with

$$\forall_y \omega_t(y) = \frac{p}{n} + (1-p)\frac{\sum_{v \in V} \omega(v,y) * \omega_{t-1}(v)}{n_v} \quad (14)$$

where n is number of words in the document, $\omega(v,w)$ is the weight of the edge from v to w, and n_v is a sum of weights of all edges starting in v, and $\omega_0(v) = 1$.

p is a probability of fact, that analysed word is keyword, while $p-1$ means that being keyword depends on neighbor words.

While $t \to \infty$, $\omega_t(y) \to \omega(t)$. While repeating iteratively above equation, we can evaluate $\omega(t)$.

3.5 Theoretical Basis

This idea uses Markov Chains for creating WordRank.

Every step of above iteration can be represented in matrix form.

Let's $P \in M(n \times n) = [p_{ij}]_{i,j=1,\ldots,n}$ be a square matrix, and n means number of words in the document. Let's $p_{ij} = \frac{\omega(i,j)}{\sum_{k=1..n} \omega(i,k)}$, assuming, that $p_{ik} = 0$ if word i wasnt neighbor of j, and vice versa. Let's W_i be a vector of weights of all words in text in time i. Let's $W_0 = \overline{1}$.

Now we can define transition matrix $P' = [p'_{ij}] \in M(n \times n)$

$$P' = p\frac{1}{n}\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix} + (1-p)P$$

P' fullfill conditions of Markov property.

$$\sum_{j=1..n} p_{ij} = \sum_{j=1..n} \frac{\omega(i,j)}{\sum_{k=1..n}\omega(i,k)} \quad (15)$$

$$= \frac{\sum_{j=1..n}\omega(i,j)}{\sum_{k=1..n}\omega(i,k)} = 1 \quad (16)$$

$$\sum_{j=1..n} p'_{ij} = \sum_{j=1..n} p\frac{1}{n} + (1-p)p_{ij} \quad (17)$$

$$= \sum_{j=1..n} p\frac{1}{n} + \sum_{j=1..n}(1-p)\frac{\omega(i,j)}{\sum_{k=1..n}\omega(i,k)} \quad (18)$$

$$= p\sum_{j=1..n}\frac{1}{n} + (1-p)\sum_{j=1..n}\frac{\omega(i,j)}{\sum_{k=1..n}\omega(i,k)} \quad (19)$$

$$= p + (1-p)\sum_{j=1..n} p_{ij} = p + 1 - p = 1 \quad (20)$$

Let's $W_i = P'W_{i=1}$. Let's $f : R^n \times M_{n\times n}(R) \to R^n$. Let's $(Z_k) = P'\forall_{k\in N}$ and $W_0 = E$. Then chain $W_{i+1} = f(W_i, Z_{i+1})$ defines a Markov chain.

Proof is obvious.

Initial state of the chain is:

$$W_0 = \begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}$$

Then, each step of the algorithm is described by:

$$W_i = P'W_{i-1}$$

and each state:

$$W_i = \begin{bmatrix}\omega_i(v_1)\\\omega_i(v_2)\\\vdots\\\omega_i(v_n)\end{bmatrix}$$

converges to stationary distribution W', which fullfills following:

$$W' = P'W'$$

Proof of Convergence. We need to show, that W' exists and

$$W' = lim_{n=0..\infty} W_n$$

This proof needs a fundamental theorem of Markov chains, prooved in (Grinstead and Snell, 1997).

Markov chain is called regular if all elements of some power of transition matrix are ≥ 0.

Theorem. *Let's P be the transition matrix of regular Markov chain. Sequence of P^i converges with $i \to \infty$ to matrix M, which all rows are equal (denoded as m). Additionally m is a probability vector, which means that it's elements are ≥ 0 and their sum equals* 1.

Matrix P' defines regular Markov chain.

$$W_i = P'W_{i-1} = P'^iW_0$$

Chain W_i fullfills assumption of definition. Additioinally P'^n converges to M. Any row m from matrix M fullfils $m = P'm$. Let's denote $W' = m$.

Theorem. *Let's P be the transition matrix of aregular Markov chain and $W = lim_{n\to\infty}P^n$. Let's w be a row from W and c be a colum equal to* 1. *Then*

1. *$wP = w$, and every vector v such as $vP = v$ equals w multiplied by a scalar,*
2. *$Pc = c$ and every column x such as $Px = x$ is a column c multiplied by a scalar.*

First proof of this theorem was described by Doeblin in (Doeblin, 1933).

This theorem proofs, that $P'^nW_0 \to W'$ with $n \to \infty$.

Convergence speed of $P'^nW_0 \to W'$ is geomethrical, so we can assume that W' equals W_k for such k, that $|W_k - W_{k-1}| < \overline{\varepsilon}$ for some small $\varepsilon > 0$

it means

$$|W_k - W_{k-1}| = \begin{bmatrix} |\omega_k(v_1) - \omega_{k-1}(v_1)| \\ |\omega_k(v_2) - \omega_{k-1}(v_2)| \\ \vdots \\ |\omega_k(v_n) - \omega_{k-1}(v_2)| \end{bmatrix}$$

and

$$\forall_{s=1..n} |\omega_k(v_s) - \omega_{k-1}(v_s)| < \varepsilon$$

4 RESULTS DISCUSSION

Usage of this preprocessing of words weights significiantly improoves efficiency of categorization algorithm.

Table 3 shows example of keywords generated for random document with and without usage of preprocessing. Comparison of both methods tested on the set of 1000 documents is presented on figure 7.

All results were checked empirically by reading tested documents.

As it is shown on illustration 7 new algorithm can produce more accurate results than the previous one. In particular cases correctness of of keywords chose can achieve even 100%, in most cases results are 80% correct, which is great result.

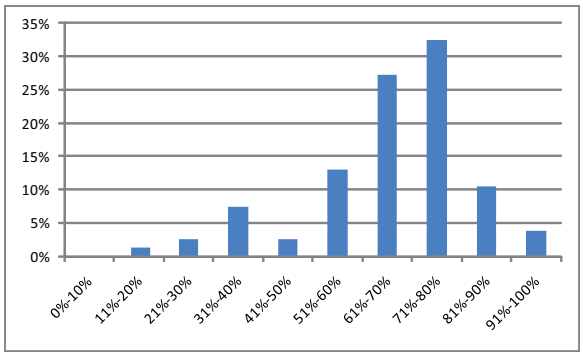

Figure 6: Effects of neural network method with reinforcement and preprocessing of the weights. X axis shows effectiveness, Y axis shows number of documents with processed with this effectiveness.

Table 3: Example of keywords selection with and without preprocessing.

without preprocessing	with preprocessing
component , **analyses**, deep , configuration , interface , described, ne , **generated** , qa , **linguistic** , names , standard , item , **xsl** , items , **extraction** , **grammar** , hpsg , np , id , named , efficiency , evaluation , **lexical** , attributes , **semantics** , elements , **german** , infl , main	**parsing**, **german**, **sentence**, **annotation**, id, **architecture**, **xsl**, **grammar**, **parser**, **rmrs**, **linguistic**, component, pos, shallow, nlp, whiteboard, deep, **string**, hybrid, en, **structures**, thesis

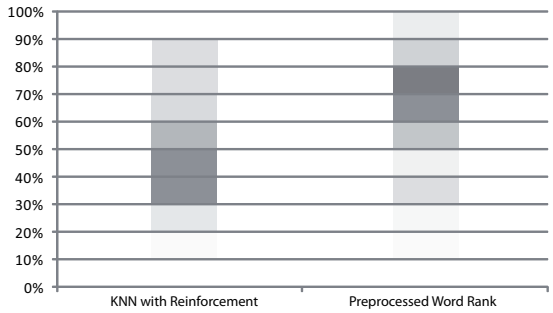

Figure 7: Results of both algorithms.

5 FURTHER RESEARCH

Further research should touch the problem of more accurate words distance measure. It should be considered to use some grammar relations and more sophisticated way to chose initial distances of words in

graph. Other area of interests is building semantics ((Shadbolt et al., 2006)) with use of generated information.

REFERENCES

Avrachenkov, K. and Litvak, N. (2004). Decomposition of the Google PageRank and Optimal Linking Strategy. Research Report RR-5101, INRIA.

Avrachenkov, K. and Litvak, N. (2004). Decomposition of the google pagerank and optimal linking strategy.

Bremaud, P. (2001). *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag New York Inc., corrected edition.

Doeblin, W. (1933). Exposé de la théorie des chains simples constantes de markov a un nombre fini d'états. *Rev Math Union Interbalkanique*, 2:77–105.

Grinstead, C. M. and Snell, J. L. (1997). *Introduction to Probability*. American Mathematical Society, 2 revised edition.

Haggstrom, O. (2002). Finite markov chains and algorithmic applications.

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

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Shadbolt, N., Berners-Lee, T., and Hall, W. (2006). The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101.

Zyglarski, B. and Bała, P. (2009). Scientific documents management system. applikaction of kohonen neural networks with reinforcement in keywords extraction. In *In Proceedings of IC3K 2009*, pages 55–62. INSTICC.