doi:10.5128/ERYa9.10

EESTI RAKENDUSLINGVISTIKA ÜHINGU AASTARAAMAT 9, 149–160

# HOW TO CREATE ORDER IN LARGE CLOSED SUBSETS OF WORDNET-TYPE DICTIONARIES

**Ahti Lohk, Ottokar Tilk, Leo Võhandu**

**Abstract.** This article presents a new two-step method to handle and study large closed subsets of WordNet-type dictionaries with the goal of finding possible structural inconsistencies. The notion of closed subset is explained using a WordNet tree. A novel and very fast method to order large relational systems is described and compared with some other fast methods. All the presented methods have been tested using Estonian[1] and Princeton WordNet[2] largest closed sets.

**Keywords:** thesaurus, closed set, seriation, Power Iteration Clustering (PIC), reducing number of crossings, WordNet

## 1. Introduction

There are more than 60 WordNets in the world[3]. The main idea and basic design of all these lexical resources came from Princeton WordNet (more in Miller et al. 1990). Each WordNet is structured along the same lines: synonyms (sharing the same meaning) are grouped into synonym sets (synsets). Synsets are connected to each other by semantic relations, like hyperonymy (IS-A) and meronymy (IS-PART-OF). In this article only hyperonymy-hyponymy relations are considered as objects of analysis. Of course, it is easy to extend the analysis over different word classes and different semantic relations.

WordNet has been used for a number of different purposes in information systems, including word sense disambiguation (Li et al. 1995), information retrieval (Rila et al. 1998), automatic text classification and structuring (Morato et al. 2004), automatic text summarization, natural language generation (Jing et al. 1998), machine translation (Khan et al. 2009) and even language teaching applications (Morato et al. 2004). A description of the Estonian WordNet and its properties has been given by Orav et al. (2011).

In applications where WordNet usage is considerable, the quality of the result depends on the quality of the WordNet used. Our analysis shows clearly that many

[1] Estonian WordNet: http://www.cl.ut.ee/ressursid/teksaurus/test/estwn.cgi.et (08.01.2013).
[2] Princeton WordNet: http://wordnet.princeton.edu/ (08.01.2013).
[3] The Global WordNet Association: http://www.globalwordnet.org/gwa/wordnet_table.html (08.01.2013).

WordNet-type dictionaries have a large closed subset (Table 1) caused by such semantic relations where one synset has connections to more than one supersynset. Liu et al. analyse mistakes in WordNet structures that arise particularly in cases where a synset has more than one sypersynsets (Liu et al. 2004). Richens extends the ideas of Liu et al. and presents a list of anomalies in the WordNet verb hierarchy and methods for finding them (Richens 2008)[4]. Vider (2001) proposes that in the best case every synset has only one supersynset. Closed subsets with more than one supersynset refer to possible causes of errors (Richens 2008, Lohk et al. 2012a). We present a convenient tool to study the possible structural inconsistencies of such large separated subsets.

For every synset in WordNet we have a matrix representation, as in Figure 3, upper level on the right. As we have to deal with very big matrices one needs a well ordered final representation of such a matrix to understand its hidden structure. Our goal is to reorder that matrix into the form of Figure 3, lower level on the right. That representation corresponds to the so-called Multidimensional Scale representation in psycholinguistics.

In the next section we explain the content of a closed set (Lohk et al. 2012b).

## 2. Closed sets

The synsets of WordNet-type dictionaries have as semantic connections hierarchy creating ones (*has_hypernym*, *has_meronym*, etc.) as well nonhierarchical ones (*near_synonym*, *be_in_state*, etc.). Using hierarchical connections makes WordNet to be a set of trees, whereby part of those trees are threaded (That is a fact from authors' analysis). The vertices of trees are synsets and edges are always some semantic connections.

Such tree has always a notion (synset) on the highest level (so-called root vertice) and other vertices on different levels. In given context we call root vertice also a root synset.

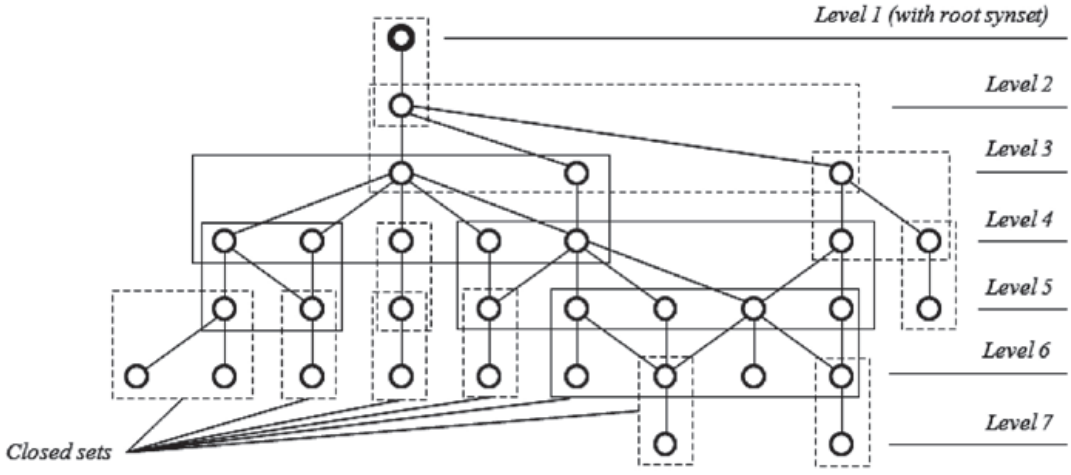



**Figure 1.** Natural tree of the WordNet with closed sets



[4] Results tables relevant to 'Anomalies in the WordNet Verb Hierarchy', paper delivered to COLING 2008, Manchester, UK, August 2008: http://www.rockhouse.me.uk/Linguistics/ (08.01.2013).

We have an invented example of such a WordNet tree in Figure 1. The synsets of the given tree (vertices) can be divided into seven levels. On the first level is the most general semantic synset – the root synset, and on the last levels (level 6 and level 7) synsets with a possibly concrete meaning. For example, based on semantic connection *has_hyponym* Princeton WordNet (version 3.0) has 346 root synsets (= trees) and Estonian WordNet (version 64) 204 synsets.

In order to understand closed subsets (Lohk et al. 2012b) in Figure 1 we have to consider only any two neighbouring levels. Let us take for example levels 3 and 4. If we separate those levels with their vertices, one can see that the connections between vertices create two closed sets of vertices. To recognise possible errors it is important to study such sets, where subsynset has a semantic connection with at least two different supersynsets. Such sets are presented in Figure 1 with thick lines and there are four of them. (The number of all closed subsets in Figure1 is 15). For example, Estonian WordNet (version 64) has as a maximal closed set with dimensions 4,945 x 457. In the language of Figure 1, this closed set has 4,945 vertices in the lower level and 457 vertices in the upper level.

The following table presents an overview of maximal closed sets in the WordNet-type dictionaries that we have analysed to date.

**Table 1.** Dimensions of the maximal closed set in a WordNet

| Seq. No. | Name and version of the WordNet | Number of the synsets | Dimensions of the maximal closed set |
|---|---|---|---|
| 1 | Polish WordNet 1.7 | 105 074 | 28 279 x 3 595 |
| 2 | Cornetto, 1.3 | 70 492 | 10 418 x 556 |
| 3 | Estonian WordNet, 64 | 54 078 | 4 945 x 457 |
| 4 | Princeton WordNet, 3.0 | 117 659 | 1 333 x 167 |
| 5 | Finnish WordNet, 1.1.2 | 117 659 | 1 248 x 165 |
| 6 | Catalan WordNet, 3.0 | 99 253 | 1 007 x 91 |
| 7 | Slovenian WordNet, 3.0 | 42 919 | 248 x 3 |

The number of closed subsets separated using the semantic relation *has_hyperonym* for all those WordNets remains between 4000 and 20 000.

A very suitable algorithm to separate closed subsets is given by Flannery et al. (2009). An example of a closed subset with real data is presented in Figure 2.

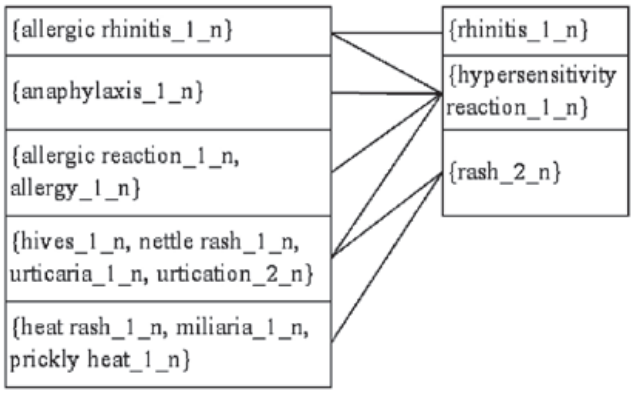



**Figure 2.** Real example of a closed subset (Princeton WordNet, version 3.0), rotated 90 degrees

The next section is dedicated to the study of such maximal closed sets.

## 3. Improving identification of mistakes by reducing the number of crossings

To visually identify possible mistakes in the connections between the synsets of a closed subset of a WordNet it is necessary to visualize the connections as clearly and with as little clutter as possible. One way to achieve this goal is to reduce the number of crossings in the graph representation of the WordNet by reordering vertices as shown in Figure 3.

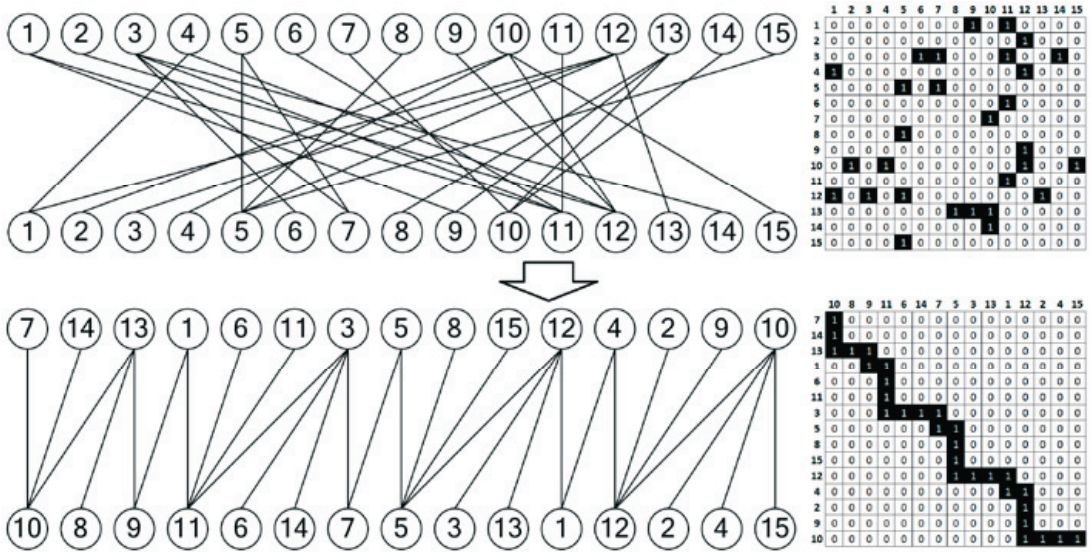



**Figure 3.** Graph (its corresponding adjacency matrix plotted on the right) with different permutations of vertices, illustrating how a good reordering can reduce the number of crossings in the bipartite graph (from 206 to 0 in this example)

There are many algorithms for this task, with different approaches – such as genetic algorithms (Mäkinen, Sieranta 1994), heuristic algorithms (e.g. barycenter (Sugiyama et al. 1981), median (Eades, Wormald 1994)) and for small graphs even exact methods (Jünger, Mutzel 1997). In the same paper in which Jünger and Mutzel introduced their exact method, they also compared different heuristic algorithms on larger graphs for which the exact method is not viable. They concluded that the iterated barycenter method was clearly the best choice for both its speed and solution quality.

### 3.1. The two-step method for reducing the number of crossings

In this work we introduce a novel technique which outperforms other widely used methods including barycenter heuristic. Our method consists of two steps:

1. Power iteration seriation;
2. Median heuristic.

First let us focus on the second step – the median heuristic by Eades and Wormald (1994). The median heuristic is a well known method for crossing minimization. To

reduce crossings, this method finds the median of the positions of adjacent vertices for every vertex and then ranks the vertices in a layer according to these values. Technically it is very similar to the barycenter heuristic (Sugiyama et al. 1981), the only difference being that the latter uses mean values of the positions of adjacent vertices instead of median values. This method is often applied iteratively, fixing one layer and reordering the other in turns, until there is no change in the order of vertices. The final outcome of the median (and also barycenter) heuristic depends on the initial state of the graph. To gain better results one can restart the algorithm a number of times with different random initial orderings and choose the best result but, as Jünger and Mutzel (1997) concluded, for bigger graphs the results improve only slightly. The purpose of the first step of our method is not to rely on random ordering, but to preprocess the graph with the aim of providing as good a starting point for the median heuristic as possible.

The first step of our method is a custom modification of a very fast (approximately linear to the input size), effective and simple clustering algorithm, Power Iteration Clustering (PIC) by Lin and Cohen (2010). The name of their method comes from the power iteration eigenvalue algorithm on which it is based.

The power iteration algorithm is used to find the dominant eigenvalue $\lambda_1$ (assuming there is one i.e. $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$) and eigenvector $\boldsymbol{v_1}$ of a matrix A. The algorithm takes the steps described in Figure 4. After a sufficient amount of iterations $\boldsymbol{b_t}$ converges to $\boldsymbol{v_1}$ of A.

***Procedure Power Iteration***$(A)$:

1. Choose random nonzero vector $\boldsymbol{b}_0 \neq \boldsymbol{0}$; $t = 0$;
2. Calculate $\boldsymbol{b}_{t+1} = \frac{A\boldsymbol{b}_t}{\|A\boldsymbol{b}_t\|}$;
3. $t = t + 1$ and repeat step 2.

**Figure 4.** Description of power iteration eigenvalue algorithm

The PIC algorithm applies power iteration to a row-normalised (all elements in a row sum up to 1) similarity matrix W. Since the dominant eigenvector of W is a constant vector, it is useless for clustering (that's also the reason for additional constraint: $\boldsymbol{b}_0 \neq c\boldsymbol{1}$, i.e. initial vector must not be a constant vector). Therefore, to turn the power iteration eigenvalue algorithm into a clustering algorithm, Lin and Cohen augmented it with a stopping criterion which stops the process before converging to the constant dominant eigenvector. As a result we get a vector $\boldsymbol{b_t}$ (PIC-vector) which is an eigenvalue-weighted linear combination of all the eigenvectors of W and turns out to be a good clustering indicator. The main procedure of PIC is described in Figure 5, where $\boldsymbol{\epsilon}$ is a small number (e.g. $10^{-5}$) used as a parameter for stopping criterion and $\boldsymbol{d}_t$ is a vector describing the changes (compared to previous iteration) in the values of the elements of vector $\boldsymbol{b}_t$. The algorithm is stopped when for two consecutive iterations $\boldsymbol{d}_t$ has remained almost constant i.e. none of the absolute differences of changes are larger than $\boldsymbol{\epsilon}$. Lin and Cohen used *k-means* on the PIC-vector to obtain the final result in the form of clusters.

**Procedure $PIC(W, \epsilon)$:**

1. Choose random vector $\boldsymbol{b}_0 \neq \mathbf{0} \wedge \boldsymbol{b}_0 \neq c\mathbf{1}$; $\boldsymbol{d}_0 = \mathbf{1}$
2. Calculate $\boldsymbol{b}_{t+1} = \frac{W\boldsymbol{b}_t}{\|W\boldsymbol{b}_t\|_1}$; $\boldsymbol{d}_{t+1} = |\boldsymbol{b}_{t+1} - \boldsymbol{b}_t|$;
3. If $\|\boldsymbol{d}_{t+1} - \boldsymbol{d}_t\|_\infty > \epsilon$, then $t = t + 1$ and repeat step 2; otherwise output $\boldsymbol{b}_t$.

**Figure 5.** Description of the main subroutine of PIC algorithm

Our own work has shown that PIC-vector can also be successfully used for seriation. To do that, we first calculate two PIC-vectors – one for rows and the other for columns. Then we reorder the rows and columns of the matrix according to the ascending or descending order of the values in the corresponding PIC-vector. The exact procedure is shown in Figure 6 where $W_r$ and $W_c$ are normalised similarity matrices, $\boldsymbol{b}_r$ and $\boldsymbol{b}_c$ PIC-vectors and $\boldsymbol{l}_r$ and $\boldsymbol{l}_c$ labels for reordering. Lower indices *r* and *c* denote rows and columns respectively.

**Procedure $PISeriation(W_r, W_c, \epsilon)$:**

| | | |
|---|---|---|
| 1. | $\boldsymbol{b}_r = PIC(W_r, \epsilon)$ | Get PIC-vector for rows... |
| 2. | $\boldsymbol{b}_c = PIC(W_c, \epsilon)$ | ...and columns. |
| 3. | $\boldsymbol{l}_r = sort(\boldsymbol{b}_r)$ | Sort both vectors and get labels |
| 4. | $\boldsymbol{l}_c = sort(\boldsymbol{b}_c)$ | indicating the original positions. |
| 5. | $A = A[\boldsymbol{l}_r, \boldsymbol{l}_c]$ | Reorder rows and columns of A |

**Figure 6.** Description of seriation procedure using PIC algorithm

For the first step of the crossing minimisation method we use the power iteration seriation with a very simple symmetric similarity function where the similarity $s(x_i,x_j) = s(x_j,x_i)$ between two vertices $x_i$ and $x_j$ from the same layer is equal to the number of their common neighbours in the opposite layer: $s(x_i,x_j) = s(x_j,x_i) = |n(x_i) \cap n(x_j)|$ (where n(x) denotes the set of neighbours of x). If we represent the bipartite graph as an adjacency matrix A and nth row of A as A(n), then we can rewrite the function as follows:

$$s(x_i, x_j) = s(x_j, x_i) = \begin{cases} A(i) \cdot A(j), & \text{if } x_i \text{ and } x_j \text{ are upper layer vertices} \\ A^T(i) \cdot A^T(j), & \text{if } x_i \text{ and } x_j \text{ are lower layer vertices} \end{cases}$$

The similarity matrix of upper layer vertices $S_r$ (or row similarity matrix of A) where element $S_r(i,j) = s(x_i,x_j)$ can then be calculated as $S_r = AA^T$ and the similarity matrix of lower layer vertices (or column similarity matrix of A) can be calculated as $S_c = A^TA$. Both matrices have to be normalised before using power iteration seriation on them.

Since the elements of PIC-vector corresponding to similar objects (vertices in our case) tend to obtain similar values, the positions of vertices after seriation also tend to correlate with the number of common neighbours. As a result subsets of vertices with many common neighbours clump together after processing with power iteration seriation in the first step of our method. This kind of approach alone does not always provide very good results in terms of the number of crossings. For example, it is possible that one layer has to be reversed, because power iteration seriation can produce results where the band of ones in the adjacency matrix runs from top-right

to bottom-left (Figure 7b) instead of top-left to bottom-right (Figure 7a), which is not good from the crossing number perspective. In some conditions (when the graph consists of more than one connected component or even when there are multiple components which are weakly connected to each other; with 'noise'; suboptimal $\epsilon$, etc.) it is possible that some subset of similar vertices will be positioned too far away from their common neighbours (Figure 7c). Additionally, there is a risk that some subsets of vertices within layers could be in reverse order (Figure 7d). Even worse, very often multiple problems come up simultaneously.

All the problems mentioned above are solved by applying a median heuristic to the result of power iteration seriation. The median heuristic is not just compensating for the weaknesses of power iteration seriation, but the output of the latter is also a very good initial permutation for the former, enabling it to achieve much better results than some random permutation would. For example: on the inital graph from Figure 1, the iterative median heuristic could only reduce the number of crossings to 27 (Figure 8), while power iteration seriation in conjunction with median heuristic reduced the number of crossings to 0 (Barycenter heuristic reduced the number of crossings to 28).

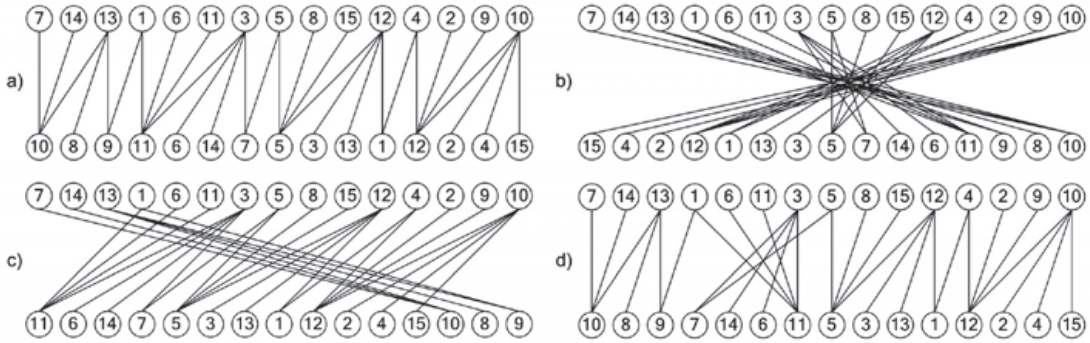



**Figure 7.** Example graph from Figure 1 illustrating some problems with power iteration seriation: a) one of the optimal permutations of vertices; b) lower layer in reverse order; c) subset of similar vertices in one layer are too far from their common neighbours; d) Subset of vertices in one layer is in reverse order

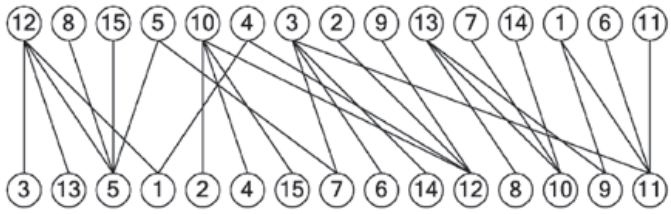


**Figure 8.** Result of iterative median heuristic on the initial graph from Figure 3

This kind of two-step method not only provides a much smaller number of crossings but may also provide these results while being faster than the iterative barycenter or median algorithm. This is possible because after preprocessing the graph with power iteration seriation, only one iteration of the median heuristic is sufficient to produce a superior result than the iterative barycenter or median method alone. If time is not crucial, then additional iterations of the median heuristic may be applied to polish the result further. Some additional improvement can also be found by trying different values for PIC's stopping criterion parameter $\epsilon$ ($10^{-5}$ – $10^{-7}$ divided by

number of rows in similarity matrix was usually optimal for us). Next we will give some examples how this method performed on the real WordNet graphs.

### 3.2. Experiments on WordNet graphs

In this section we give an overview of our tests on the largest closed sets of synsets from Estonian and Princeton WordNets. The largest closed set from Estonian WordNet can be represented as a 4,945 by 457 matrix (see Table 1). In the case of the Princeton WordNet the matrix size is 1 333 x 167 (Table 1).

We ran our tests on a PC with 6 GB of RAM and an Intel® Core™ i7-870 Processor and compared three different methods: iterative barycenter, iterative median and our two-step method. In the two-step method we used only one iteration of median heuristic and for PIC's stopping criterion parameter $\epsilon$ we chose $10^{-5}$ divided by number of rows in similarity matrix. All 3 methods were run on the same initial permutation of vertices. The results are shown in Table 2.

**Table 2.** Results of three methods compared with a random permutation on the largest closed sets of two WordNets

| | Estonian WordNet (v 64) | | Princeton WordNet (v 3.0) | |
|---|---|---|---|---|
| | **Time (s)** | **Crossings** | **Time (s)** | **Crossings** |
| Initial | – | 2 349 957 | – | 265 940 |
| Median | 4.1 | 904 629 | 0.9 | 36 862 |
| Barycenter | 16.9 | 308 444 | 1.5 | 22 927 |
| 2-step method | 0.4 | 84 884 | 0.1 | 5 484 |

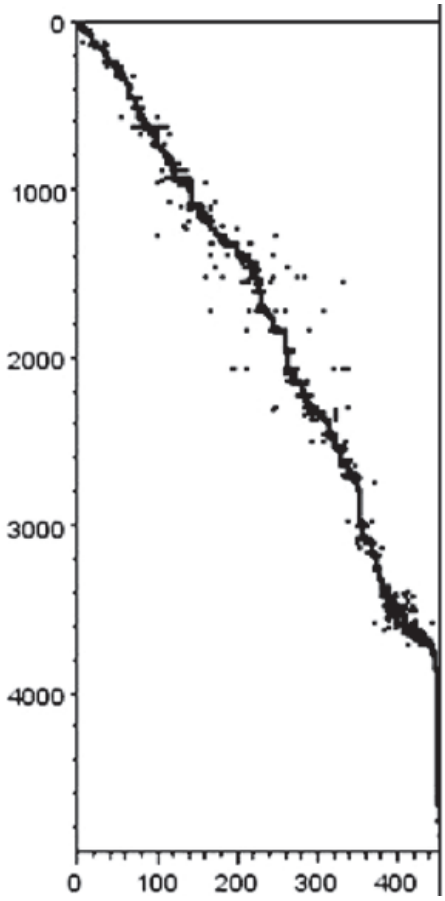
The two-step method turned out to be roughly 9–42 times faster than compared methods while producing more than 3–10 times fewer crossings. From these results we can conclude that our two-step method is the best choice for minimising the number of crossings in WordNet graphs.

Some possible ways of using the Minimal Crossing method to detect inconsistencies in WordNet structures is given by the author and others (Lohk et al. 2012a, 2012b). The detailed handling of those and others inconsistencies would require a separate article.

## 4. Looking at the results

As a result of using our two-step method we did get an ordered matrix (Figure 9a).

By converting this matrix with the labels of synsets into a MS Excel worksheet we have the possibility of studying the large closed subset more methodically. To make it easier to understand the result it is useful to freeze the headings of rows and columns. That makes it possible to move around in the table so that the synsets on both levels are always visible. To find possible errors one has to study such places in that table where conceptual synsets in rows and columns are conspicuously different. Usually such an occasion happens when one concept has several parents. The decision about a possible error will be naturally made by the lexicographer.




a)

| | {ehituskeraamika_1_n} | {katusekivi_1_n} | {läbiviik_1_n} | {taivas_1_n} | {ujuvmärk_1_n} | {ujuk_2_n} |
|---|---|---|---|---|---|---|
| {reimar_1_n} | 0 | 0 | 0 | 1 | 1 | |
| {tooder_1_n} | 0 | 0 | 0 | 1 | 1 | |
| {keraamiline katusekivi_1_n} | 1 | 1 | 0 | 0 | 0 | |
| {savikatusekivi_1_n} | 1 | 1 | 0 | 0 | 0 | |
| {savikivi_1_n} | 1 | 1 | 0 | 0 | 0 | |
| {savist katusekivi_1_n} | 1 | 1 | 0 | 0 | 0 | |
| {betoonkatusekivi_1_n} | 0 | 1 | 0 | 0 | 0 | |
| {harjakivi_1_n} | 0 | 1 | 0 | 0 | 0 | |
| {imbrex_1_n} | 0 | 1 | 0 | 0 | 0 | |
| {valtskivi_1_n} | 0 | 1 | 0 | 0 | 0 | |
| {antenni läbiviik_1_n} | 0 | 1 | 1 | 0 | 0 | |
| {antennikivi_1_n} | 0 | 1 | 1 | 0 | 0 | |
| {tuulutuskivi_1_n} | 0 | 1 | 1 | 0 | 0 | |
| {ventilatsiooni tuulutuskivi_1_n} | 0 | 1 | 1 | 0 | 0 | |
| {poi_1_n} | 0 | 0 | 0 | 0 | 1 | |
| {kork_2_n} | 0 | 0 | 0 | 0 | 0 | |
| {logilaud_1_n} | 0 | 0 | 0 | 0 | 0 | |
| {paravan_1_n} | 0 | 0 | 0 | 0 | 0 | |
| {pontoon_1_n} | 0 | 0 | 0 | 0 | 0 | |
| {ujuk_1_n} | 0 | 0 | 0 | 0 | 0 | |

b)

**Figure 9.** The biggest closed subset of Estonia WordNet: a) closed subset after ordering; b) closed subset for the investigation, converted for Excel

## 5. Conclusion

Wordnet as a lexical and semantical database is widely used in different language technology applications. Therefore it is important to ensure the quality of any Wordnet used. Previous study has shown that WordNet with its hierarchical structures consists of many relations which quite easily cause errors in the Wordnet structure (Lohk et al. 2012). In this paper we propose a formal way to detect and study possible inconsistencies using closed subsets. The notion of a closed subset has been explained using the WordNet tree. Separated closed subsets are represented as matrices and a new and fast two-step method reorders such sparse relational systems into an easily visible and understandable view. Our method has been compared with other fast reordering methods and tested on Estonian and Princeton WordNets. As a final suggestion we transform the subsets with correct syntactic labels into an Excel spreadsheet to enable convenient study of places where the structural connections of concepts (synonym synsets) are suspicious.

## References


Eades, Peter; Wormald, Nicholas C. 1994. Edge crossings in drawings of bipartite graphs. – Algorithmica, 379–403.

Flannery, P. B.; Press, H. W.; Teukolsky, A. S.; Vetterling, T. W. 2009. Numerical Recipes in C. The Art of Scientific Computing. South Asia: Cambridge University Press India.

Jing, H. 1998. Usage of WordNet in natural language generation. – Proceedings of the Workshop Usage of WordNet in Natural Language Processing Systems: COLING-ACL 1998; August 16, Montreal, Quebec, Canada, 128–134.

Jünger, Michael; Mutzel, Petra 1997. 2-Layer Straightline Crossing Minimization: Performance of exact and heuristic algorithms. – Journal of Graph Algorithms and Applications, 1–25.

Li, X.; Szpakowicz, S.; Matwin, S. 1995. A WordNet-based algorithm for word sense disambiguation. – Proceedings of IJCAI 1995. Morgan Kaufmann Publishers, 1368–1374.

Lin, Frank; Cohen, William W. 2010. Power iteration clustering. – Proceeding of the 27th International Conference on Machine Learning, June 21-24, 2010, Haifa, Israel. Omnipress, 655–662.

Liu, Y.; Jiangsheng, Y.; Zhengshan, W.; Shiwen, Y. 2004. Two kinds of hypernymy faults in Word-Net: the cases of ring and isolator. – Petr Sojka, Karel Pala, Pavel Smrz, Christine Fellbaum, Piek Vossen (Eds.). Proceedings of the Second Global WordNet Conference. Brno, Czech Republic, 20-23 January 2004. Masaryk University, 347–351.

Lohk, Ahti; Võhandu, Leo 2012. Eesti Wordnet'i struktuuri analüüsist. – Eesti Rakenduslingvistika Ühingu aastaraamat, 8, 139–151. http://dx.doi.org/10.5128/ERYa8.09

Lohk, Ahti; Vare, Kadri; Võhandu, Leo 2012a. First steps in checking and comparing Princeton WordNet and Estonian WordNet. – Miriam Butt et al. (Eds.). Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH. April 23-24 2012, Avignon, France. Association for Computational Linguistics, 25–29.

Lohk, Ahti; Vare, Kadri; Võhandu, Leo 2012b. Visual Study of Estonian WordNet using Bipartite Graphs and Minimal Crossing algorithm. – Proceedings of 6th International Global WordNet Conference, Matsue, Japan, 2012, 167–173.

Miller, G.; Beckwith, R.; Fellbaum, C.; Gross, D.; Miller, K. 1990. Introduction to WordNet: An on-line lexical database. – International Journal of Lexicography 3, 235–312.

Morato, J.; Marzal, M. Á.; Lloréns, J.; Moreiro, J. 2004. WordNet applications. – Petr Sojka, Karel Pala, Pavel Smrz, Christine Fellbaum, Piek Vossen (Eds.). Proceedings of the Second Global WordNet Conference. Brno, Czech Republic, 20-23 January 2004, 270–278.

Mäkinen, Erkki; Sieranta, Mika 1994. Genetic algorithms for drawing bipartite graphs. – International Journal of Computer Mathematics, 53 (3-4), 157–166. http://dx.doi.org/10.1080/00207169408804322

Orav, Heili; Kerner, Kadri; Parm, Sirli 2011. Eesti Wordnet'i hetkeseisust. – Keel ja Kirjandus, 2, 96–106.

Richens, Tom 2008. Anomalies in the WordNET verb hierarchy. – Proceedings of the 22nd International Conference on Computational Linguistics: COLING-ACL 2008, August, Manchester, UK, 729–736.

Rila, M.; Tokunaga, T.; Tanaka, H. 1998, The use of WordNet in information retrieval. – Proceedings of the Workshop Usage of WordNet in Natural Language Processing Systems: COLING-ACL 1998, August 16, Montreal, Quebec, Canada, 31–37.

Salam, Khan Md Anwarus; Khan, Mumit; Nishino, Tetsuro 2009. Example based English-Bengali machine translation using WordNet. – Proceedings of the Triangle Symposium on Advanced ICT 2009 (TriSAI 2009), October 28-30, 2009. Tokyo, Japan.

Sugiyama, Kozo; Tagawa, Shojiro; Toda, Mitsuhiko 1981. Methods for Visual Understanding of Hierarchical System Structures. – IEEE Transactions on Systems, Man and Cybernetics, 11 (2), 109–125. http://dx.doi.org/10.1109/TSMC.1981.4308636

Vider, Kadri 2001. Eesti keele tesaurus – teooria ja tegelikkus. – Margit Langemets (Toim.). Leksikograafiaseminar "Sõna tänapäeva maailmas" / Leksikografinen seminaari "Sanat nykymaailmassa". Ettekannete kogumik. Eesti Keele Instituudi toimetised 9. Tallinn: Eesti Keele Sihtasutus, 134–156.

## Web References

The Global WordNet Association. http://www.globalwordnet.org/gwa/wordnet_table.html (08.01.2013).

Results tables relevant to "Anomalies in the WordNet Verb Hierarchy" paper delivered to COLING 2008. Manchester August 2008. http://www.rockhouse.me.uk/Linguistics/ (08.01.2013).

Princeton WordNet (version 3.0, 3.1). http://wordnet.princeton.edu/ (08.01.2013).

Estonian WordNet (version 65). http://www.cl.ut.ee/ressursid/teksaurus/test/estwn.cgi.et (08.01.2013).

**Ahti Lohk** (Tallinn University of Technology), main research interests are in the field of data analysis.
ahti.lohk@ttu.ee

**Ottokar Tilk** (Tallinn University of Technology), main research interests are data analysis and machine learning algorithms.
ottokar.tilk@ttu.ee

**Leo Võhandu** (Tallinn University of Technology), main research interests are in the field of data analysis.
leo.vohandu@ttu.ee

# KUIDAS LUUA KORDA WORDNET'I TÜÜPI SÕNARAAMATUTE SUURTES KINNISTES ALAMHULKADES

**Ahti Lohk, Ottokar Tilk, Leo Võhandu**
Tallinna Tehnikaülikool

WordNet kui leksikaalsemantiline andmebaas leiab laialdast kasutust keeletehnoloogia rakendustes, mistõttu on ilmne, et tulemuse kvaliteet sõltub paljuski *wordnet*'i enda kvaliteedist. Varasemad uurimused on näidanud, et *wordnet*'i hierarhiat tekitavates puudes esineb seoseid, mis põhjustavad tema struktuuris vigu (Lohk, Võhandu 2012). Ühe võimalusena pakutakse artiklis taolisi kõrvalekaldeid uurida ja avastada kinniste alamhulkade kaudu, mida esitatakse maatriksina ja millele rakendatakse autorite pakutud uudset kahesammulist meetodit. Kinniseid alamhulki selgitati tehislikult koostatud *wordnet*'i puu alusel. Pakutud kahesammulist meetodit, mis sobib suurte relatsiooniliste süsteemide korrastamiseks, kõrvutati teiste kiirete varasemate meetoditega (raskuskeskme meetod ja mediaanmeetod). Jõuti järeldusele, et kahesammuline meetod pakub tulemuseks nii paremat ristumiste arvu kui ka kiiremat algoritmi kui varasemad meetodid. Meetodit testiti Eesti ja Princetoni *wordnet*'idel. Maatriksina saadud tulemusi soovitati koos sünohulkade nimedega konverteerida tabelarvutusprogrammi, liikuda mööda korrastatud maatriksil olevat lairiba ning uurida ridades ja veergudes olevaid sünohulkade neid kohti, kus mõisted silmatorkavalt erinevad.

**Võtmesõnad:** tesaurus, suletud hulgad, järjestamine, klasterdamine iteratiivse astendamisega, ristumiste arvu vähendamine, WordNet