Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 373–379

ISBN 978-83-60810-27-9
ISSN 1896-7094

Parallel, Massive Processing in SuperMatrix—a General Tool for Distributional Semantic Analysis of Corpus

Bartosz Broda, Damian Jaworski, Maciej Piasecki
Institute of Informatics, Wrocław University of Technology, Poland
Email: {bartosz.broda, maciej.piasecki}@pwr.wroc.pl

***Abstract*—The paper presents an extended version of the SuperMatrix system — a general tool supporting automatic acquisition of lexical semantic relations from corpora. Extensions focus mainly on parallel processing of massive amounts of data. The construction of the system is discussed. Three distributed parts of the system are presented, i.e., distributed construction of co-incidence matrices from corpora, computation of similarity matrix and parallel solving of synonymy tests. An evaluation of a proposed approach to parallel processing is shown. Parallelization of similarity matrix computation demonstrates almost linear speedup. The smallest improvements were achieved for construction of matrices, as this process is mostly bound by reading huge amounts of data. Also, a few areas in which functionality of SuperMatrix was improved are described.**

I. Introduction

HUGE text corpora are now relatively easy to be collected, e.g., on the basis of the Web. They can be a valuable, direct source of linguistic data, e.g., use examples, but also a basis for the automated extraction of linguistic knowledge. Such applications as automatic induction of morphological descriptions or extraction of syntactic subcategorisation frames are well known. However, recently we can also observe applications of automated extraction methods in the lexical semantics, too. Construction of a large scale language resource describing lexical semantics, e.g. a wordnet[1], is a very laborious process, in which skilled lexicographers must be involved. Such a process can be supported by the automatic extraction of semantic relations among words, or multi-word units in general, as well, as semantic restrictions imposed by particular words on their occurrence contexts. The extracted knowledge can be consulted by linguists during the construction of a resource or can be used for suggesting to the lexicographers some new elements of a semantic lexicon, e.g., lexical semantic relations between lexical units or groups of lexical units (i.e., wordnet synsets) in a wordnet, cf [2].

There are two main paradigms of automatic extraction of instances of lexical semantic relations, e.g., [3]:

- *pattern-based*,
- and *clustering-based*.

The clustering-based paradigm is also called *distributional paradigm*, as it originates directly from the *Distributional Hypothesis* formulated by Harris [4].

[1]By wordnet we mean here an electronic thesaurus of a structure following the main lines of the Princeton WordNet thesaurus [1].

According to the pattern based approaches there are some lexico-syntactic patterns, which combine two words and mark the word pair as an instance of some lexical semantic relation, e.g., hypernymy [5]. So every single occurrence of a precise pattern is treated as evidence for the semantic association of the given words. The pattern-based methods can utilise individual occurrences of word pairs in a corpus, but are error prone in the same time, as they depend on individual occurrences. This drawback can be corrected by taking into account statistics of word co-occurrences with different patterns across the whole corpus, cf Espresso [3] and Estratto [6] methods.

Statistical analysis of co-occurrences can lead to the automatic construction of a Measure of Semantic Relatedness (MSR) (also called Measure of Semantic Similarity) which is modelled as a function:

$$MSR : W \times W \rightarrow R \tag{1}$$

where W is a set of words and R is the set of real numbers.

MSR construction is based on the analysis of the similarity of distributions of some words across different lexico-syntactic or even semantic contexts as evidence for their close semantic relation.

There are plenty of methods of the automatic extraction of MSRs. Most of them start with processing a corpus and constructing a co-incidence matrix describing co-occurrences of LUs (rows) and lexico-syntactic contexts (columns). They differ in three aspects: definitions of contexts, transformations of the raw frequencies and calculation of the final measure value. A context can be a whole document, but much better results can be obtained using a more precise description of the context. So, a word occurrence is mostly described by other words occurring in its context and, possibly, syntactic relations linking them to it, e.g., a particular noun can be characterised by adjectives such that they occur in its context and syntactically modify it. A *feature* is a particular type of co-occurrence describing the context, and a lexico-syntactic feature is a pair: a word and a syntactic relation, e.g., $\langle$*fast*, `modifier`$\rangle$. The initial value of the features are the frequencies of their co-occurrence with the words that are described. However, these raw values are mostly transformed later in order to emphasise the semantic information they express in relation to words being described.

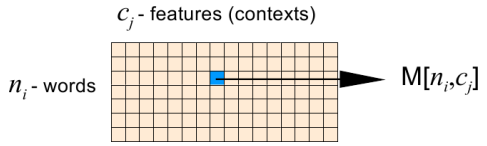

Fig. 1. Schematic view of co-occurrence matrix.

Several algorithms for collecting corpus frequencies, transforming feature value and calculating semantic relatedness of words on the basis of their feature vectors were implemented in the *SuperMatrix* system [7] – a general tool for the extraction of lexical semantic knowledge on the basis of distributional analysis of corpora. The increasing size of corpora processed by SuperMatrix and the resulting huge amount of the extracted data resulted in very long processing time in practical applications of SuperMatrix. From the very beginning a typical practice was to divide manually huge processing tasks into several parts processed independently: several copies of SuperMatrix were run in parallel, on different computers, and next the partial results were merged using manually controlled functions of SuperMatrix. The general requirement for the computing time is unavoidable, if one wants to work with huge corpora, but the total time of task completion can be reduced by extending SuperMatrix with elements of the parallel architecture.

The goal of the paper is to present a version of SuperMatrix, which is extended with elements of the parallel processing architecture, that is aimed towards better support for efficient distributional analysis of massive corpora. Moreover, several extensions and improvements concerning extraction and clustering algorithms of SuperMatrix are also presented.

In the following sections, first we will briefly describe the SuperMatrix system as a tool supporting semantic analysis, i.e., its basic algorithms and functionality. Next we will identify these algorithms that can be successfully transformed into their parallel versions. We will present the distributed, parallel architecture of the new version of SuperMatrix. Finally, we will report on the experiments performed, practical applications and plans for further extensions.

II. SuperMatrix

SuperMatrix is a collection of libraries for programmers and end-user tools for creating, storing and manipulating co-incidence matrices that describe distributional patterns of words. A co-incidence matrix is a matrix, in which rows correspond to words being described and columns to features, see Fig. 1. Initial values of cells are set to frequencies with which features occurred in the contexts of the subsequent words. The system consists of the four main modules,[2] namely:

- *Matrices* – a library for storing and accessing matrices. SuperMatrix supports a few formats of storing matrices, but the most important one are sparse matrices.
- *Comparator* – a library enabling computation of similarity between rows of matrices (i.e., between one word and multi-word lexical units) using different MSRs. Comparator was also designed in a modular manner. The computation of a similarity matrix was decomposed into a few steps, i.e., global filtering of features, local selection of features, weighting of features and comparing feature vectors. For every stage there are several algorithms implemented in SuperMatrix.
- *Matrix tools*, including (but not limited to) tools for: constructing co-incidence matrices (collecting raw frequencies and transforming feature values), MSRs, joining matrices and analysing matrix content, e.g., manually browsing selected rows and columns from any matrix – including transformed and weighted matrices.
- Clustering – a package consisting of several clustering algorithms, e.g., implementation of Clustering by Committee [8] (modified and extended [9]), RObust Clustering using linKs [10] and Growing Hierarchical Self-Organising Maps [11], and also an interface to the CLUTO package [12].

[2]We present a brief description of a system, for detailed discussion see [7].

III. Distributed SuperMatrix

There are two main problems of processing huge corpora: long processing time and huge amount of memory required. Fortunately, corpus processing has in general a parallel nature and the majority of matrix operations can be easily decomposed into operations performed on matrix parts. In the following subsection we will discuss parallel algorithms introduced into extended SuperMatrix, and in the next subsection we will present the contemporary, improved functionality of the system.

A. Parallel Algorithms

We have identified three main areas of the SuperMatrix functionality, in which parallel algorithms can introduce improvements:

- co-incidence matrix construction,
- similarity matrix calculation,
- MSR evaluation by WordNet-based Synonymy Test (WBST)[3] [16].

Corpus processing is the fourth possible area, which is suitable for parallel processing, but SuperMatrix is mostly applied to text previously tagged morphosyntactically and text processing is reduced to the application of morphosyntactic constraints or reading annotation – both these elements are included in the matrix construction.

A natural way of task distribution among processing nodes during matrix construction seemed to be assignment of the subsequent parts of the corpus to the different nodes. However, this solution encounters several problems. First, the applied binary corpus format of Poliqarp is best suited for sequential corpus processing in our wrapping library. Secondly, definition of the matrix columns in the form of lexicalised morphosyntactic constraints can consume huge amount of operating

[3]The task in WBST test is to determine which word among a few choices given is synonymous to the given question word. There is a long tradition of evaluation of MSRs using synonymy tests, e.g., [13]–[15].

memory, as the constraints are kept in the compiled form due to efficiency reasons. Thus, a large constraint-based matrix of around 500 thousands of columns is too large to be kept in the memory[4], but we face the necessity to build such a matrix as a tool in the analysis of a huge Polish corpus of 930 million of words. Construction of a matrix including features based on the morpho-syntactic constraints requires huge amounts of memory, but it is also the most valuable type of a matrix in terms of the accuracy of the results generated and possible applications of SuperMatrix.

Instead of distributing data across processing nodes, we distribute processing. Each processing node is assigned a matrix with a subset of the original set of columns. The sub-matrices (column slices) are defined by the specification of the features sent by the central managing node. All nodes are run on the corpus in parallel. A corpus is stored on the shared resource, as this is the usual approach in computing clusters. Also, a corpus can be huge so keeping it copied in multiply places is not practical. Nevertheless, this is a potential bottleneck of the system. However, as our experiments with corpus copies placed on the processing nodes showed, the data transfer influences the system efficiency to a minor extent. At the end, the data stored in the sub-matrices are read sequentially by the central node and merged into the resulting matrix of possible huge dimensions, but typically very sparse. Constraints are kept in memory only for processing in the nodes, columns in the resulting matrix are described by labels.

The big advantage of the above model is the ability to construct word descriptions unlimited in the number of features used distributed across processing nodes. Scheduling is also simple in the case of a limited set of nodes – the central manager can gradually sent the subsequent sub-matrices to nodes which just finished their task.

In the case of the similarity matrix calculation, the smallest unit of processing which can be distributed independently of specifics of measure of semantic relatedness is the calculation of the similarity of two rows. As a typical task is computation of the k most semantically related words to the given one, the central manager assigns identifiers of rows to be processed to the processing nodes. The central node collects the computed values or ranking lists for the subsequent words.

In the parallel implementation of WBST-based evaluation of MSR, all processing nodes store the full list of question-answer (QA) pairs, the similarity matrix and the central manager assigns subsequent <test, MSR, matrix> tuples to free nodes. Testing results (the number of correct and incorrect answers, as well, as omitted QA pairs when one of the words was not described by a given matrix) are sent to the central node which calculates the final aggregated test statistics.

B. Improved Functionality

The functionality of SuperMatrix has been improved in a few areas. Aside from bugfixes and general code clean-up we have added a few new algorithms for construction of MSR or extended existing ones. We have added simple quantization of weighted values to mutual information based measures (Mutual Information of Lin's [17] measure and Pointwise Mutual Information [8]). Several different methods of weighting inside RWF function have been implemented using contingency table based definitions proposed by [18]. We have added RWF based on: mutual information, log likelihood and χ^2 measures of associations. Rank weight functions were also extended with a possibility of building non-linear rankings of features (e.g., exponential).

Treating co-occurrence data gathered using different lexico-semantic constraints (or different grammatical relations acquired from parser) in an uniform way may be sometimes inappropriate. Thus, we have added simple *ensemble-like* MSRs that works on different parts of matrices independently. This is one of the ongoing area of research within our team.

As it was mentioned above, SuperMatrix does data clustering via either using built-in algorithms (e.g., GHSOM or CBC) or interfacing with CLUTO. Both of these ways expose pretty advanced clustering algorithms for the user. However, there was no easy way to use some very simple algorithms. Thus we have implemented two baseline clustering algorithms: k-means and k-medoids. The latter uses classical *partition around medoids* approach. The implementation of those algorithms is a little bit different then classical ones, i.e., we perform clustering on the basis of similarity (any weighting scheme in SuperMatrix can be used, but similarity is restricted to cosine in current version). In terms of clustering we have improved our integration with CLUTO, i.e., for internal algorithms we use output format of CLUTO and we added tools for evaluation of clustering based on this format. We have implemented only a few evaluation measures, namely: purity, Rand Index, normalised mutual information and precision, recall and f-measure [19]. Having this tight integration with CLUTO enables us to use also SenseCluster [20] method of evaluation, which uses a little bit different definition for computing precision and recall.

We also work on the web service base access to SuperMatrix and part of the targeted functionality is already available[5]. This work is done as a part of the CLARIN project. Currently we have most of the technical bits of SOA-based communication with SuperMatrix worked out and a user can query a few pre-build matrices for similarity between words.

We have added an interface to work with Doug Rohde's SVDLIBC library[6] — a cleaned up version of Berry's SVDPACKC [21].

Also a few technical improvements have been made in SuperMatrix. The performance of (sequential) parts of MSR computation have been greatly improved (by a factor of 2-10 depending on complexity of MSR, the more complex the higher the speedup) by allowing user to weight whole matrix, before computations (which can be undesirable in a few cases,

[4]On the processing node with 4 GB of RAM only construction of a matrix with 76,000 columns could be started, but was not completed.

[5]http://nlp.pwr.wroc.pl/clarin/ws/supermatrix/

[6]http://tedlab.mit.edu/ dr/SVDLIBC/

e.g., some MSRs use unweighted data in the later stages of processing, or raw frequencies are required in different part of user's application). We have improved a parser of MSR request in a way that every part of MSR can be now specified via a character string without resorting to assembling MSR inside the code. In previous version of SuperMatrix only predefined combinations of parameters were available in this way.

IV. Experiments

All experiments were aimed at comparing the processing time of the single-node version of SuperMatrix with the new parallel version run in the distributed network of processing nodes. For the needs of semi-automated development of plWordNet (a wordnet for Polish) [2] a corpus of the size 900 million words[7] is processed, around 350,000 lexicalised constraints (based on adjectives, nouns and verbs) were identified as delivering statistically meaningful information and the description is built for around 25,000 nouns (some of them are multiword expressions)[8]. Concerning these numbers, performing the repeated tests on the corpus of this size would consume the power of our computing cluster to the very large extent, while the cluster is being extensively used in everyday practice of the works on preparing semantic resources for plWordNet. Also, it is not possible to fit a matrix of this size in the memory which is needed for comparison with performance of a run on a single node – only up to 40–60 thousands of column constraints, depending on the complexity of the constraints, can be stored in 4GB RAM. Thus all experiments were performed on smaller matrices or on partially completed tasks.

All experiments were performed on the computing cluster consisting of 6 computing nodes. Each node is equipped with two dual core processors Intel(R) Xeon(R) CPU 5160 3GHz. 4 processes can be run in parallel on each node that gives maximally 24 virtual processing nodes. Three of the nodes have 5 GB RAM, three 6 GB RAM – this is a strong limitation for SuperMatrix, concerning a single processing node. The nodes are connected by the InfiniBand network of 96Gbit/s bandwidth. The cluster runs under CentOs 5.3. All parallel algorithms of the extended SuperMatrix were written in C++ and the inter-process communication using Message Passing Interface (MPI) library.

Aside from time of execution we mention also *speedup* $S(p) = \frac{T_s}{T_p(p)}$, where p is a number of processors, Ts is a time of execution of best sequential algorithm solving the problem and $T_p(p)$ is time of execution of parallel algorithm using p processors. In practice $T_s = T_p(1)$ for convenience [24].

[7]A joint corpus consisting of 10 smaller, including the IPI PAN Corpus [22], the corpus of *Rzeczpospolita* [23] and several corpora built on the basis of the Web content.

[8]The set includes words already included in plWordNet and new ones, as the aim of the process is to extract from the corpus information concerning semantic associations among words and next use this information in generating suggestions concerning the placement of the new words in the plWordNet structure.

TABLE I
Matrix construction on the different number of processing nodes.

No of nodes	1	2	4	6	8	10	12	14
Time (min)	90.5	71.5	64.25	62	61.75	54.5	51.5	53
Speedup		1.27	1.41	1.46	1.47	1.66	1.76	1.71

A. Co-incidence Matrix Construction

A matrix was constructed on the basis of the IPI PAN Corpus (around 250 millions of words). Description of 25,000 nouns was based on constraints based only on adjectives that resulted in around 35,000 lexical constraints (columns). A constraint of this type recognises occurrence of the given adjective in text as modifying the given noun which is being described.

During the first experiment constraints were uniformly assigned to the processing nodes. The results for different number of nodes used are presented in Tab. I. The processing time decreases with the increasing number of nodes. However, the process saturates with 10 processing nodes, and starting with 12 nodes the speed of processing starts to increase. It is caused by the communication overhead: adding new nodes brings a relatively small increase of processing speed but increases the complexity of communication. The reason for this is that a significant part of the process has a sequential character, i.e., mainly reading the corpus.

In the second experiment, we compared the behaviour of the system in the environment of the limited memory capacity. The maximal number of the constraints stored was limited to 7800 only. A sequential non-parallel version was run 4 times and was compared with the parallel version run on the 4 nodes simultaneously. The results are, respectively: 169.5 minutes and 64.25 minutes. The total cost of processing 257 min. (4 times 64.25 min.) is much higher in the case of the parallel version, however we get the result 2.64 times faster.

B. Similarity Matrix Calculation

Experiments in this subsection were based on the co-incidence matrix of 25,000 rows (describing nouns) and 240,000 columns representing lexicalised constraints. Cosine measure was applied as row similarity measure, so no transformation took place during the similarity calculation.

In the first experiment, computations was repeated for several different numbers of words, in each case the goal was to find $k = 20$ most semantically related words to the given one. Experiment was repeated for the different numbers of processing nodes used. As complete would require a lot of processing time, the final results were estimated on the basis of the first hour of processing in each case. The results are presented in Tab. II.

The achieved speedup is almost linear – with the increasing number of nodes the speedup of the processing is increasing (see Fig. 2). The sequential part is performed on each node and encompasses reading the co-incidence matrix (from the shared hard drive) and calculating the similarity function. So

TABLE II
TIME (MIN) REQUIRED FOR THE SIMILARITY MATRIX CALCULATION ON THE DIFFERENT NUMBER OF PROCESSING NODE – "ND." AND FOR DIFFERENT SIZE OF THE PROBLEM.

	No of words						
Nd.	200	500	1000	2000	5000	10000	25000
1	107	267	476	1120.5	2719	4486.5	14760
2	57.25	138	302	553.75	1323.5	3102	6402
4	34.25	74	154.5	306.5	727	1488.25	3682
6	23	45	93.5	180	453.5	911	2318
8	20	37.5	78.75	152.5	387	759.5	1987
10	16.75	28.5	60	115.25	288	581	1408.5
12	13.75	23	47.5	88	225.5	445	1104.5
14	13.25	19.5	40.25	75	186.5	373.25	927
20	12.5	16.75	33.25	70.25	157	307	781

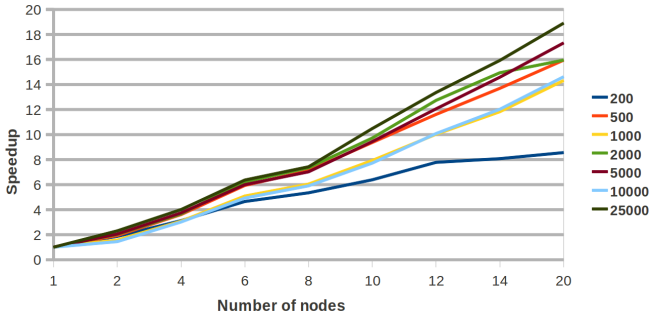

Fig. 2. Speedup of similarity matrix calculation in relation for different size of the problem

the parallel part of the process increases with the increasing number of words.

A typical task is calculation of a list of the k most semantically related words to each described word. Such a list is used next, e.g., as a knowledge source in a *WordNet Weaver* system that supports extension of a Polish wordnet in a semiautomatic way.

During the second experiment, we tested the influence on the speedup of the number of k most semantically related words calculated for each word and sent to the central manager. Our aim was to analyse consequence of the expected increasing communication complexity with the increasing length of lists returned to the central manager. The results for the two values of k: 20 – a typical one, and 25,000 – the full list, are presented in Tab. III. The lists were computed for a sample of 500 words (nouns).

In the case of 1 and 2 nodes, the processing time was estimated on the basis of part of the process. If size of the results had had an influence on the communication, the time required for computing a list for $k = 25000$ would have been larger than for $k20$, but it is much smaller in fact! It is caused by the lack of sorting in the case the whole list of 25,000 similarity values is returned. According to the assumed algorithm, if k is smaller than the number of words a list must be sorted and only the k top are returned.

Both experiments showed that with the increasing complexity of the problem, but also with increasing number of processing time the system expresses very positive efficiency.

TABLE IV
TIME (MIN) USED REQUIRED FOR PERFORMING WBST TESTS FOR DIFFERENT CO-INCIDENCE MATRICES AND MEASURES OF SEMANTIC RELATEDNESS. THE VALUE OF SPEEDUP IS GIVEN IN THE BRACKETS.

MSRs	5		10	
Matrices	1	2	1	2
Nodes				
1	21	51	66.5	157.5
2	13 (1.61)	26 (1.96)	35 (1.9)	82.5 (1.91)
3	10.2 (2.05)	22 (2.32)	28 (2.37)	65 (2.42)
4	10 (2.1)	19.5 (2.61)	24.2 (2.74)	55.5 (2.84)
5	6 (3.5)	15 (3.4)	18 (3.69)	42.2 (3.73)
8	6 (3.5)	11.2 (4.53)	16.2 (4.09)	28.5 (5.53)
10	6 (3.5)	8 (6.35)	16.2 (4.09)	22.5 (7)

These results have a very practical meaning, as row similarity matrices are comptuted very often in practice, and in the case of a non-parallel version of SuperMatrix their calculation was a serious bottleneck for more complex process of the extraction of lexico-semantic relations.

C. Measure of Semantic Relatedness Evaluation

For the needs of the experiment a WBST test was randomly generated from plWordNet. The test consisted of around 15,000 of question-answers (QA) pairs. Two large co-incidence matrices were collected. For the experiment we selected also 10 MSRs based on different algorithms of different complexity. All MSRs were calculated during experiment for the pairs of words from the test. Matrices were stored on the processing nodes.

Two versions of the WBST-based evaluation were compared. The old one, which is parallel (thread-based parallelism) but not distributed. It can assign different <test, MSR, matrix> tuples to different cores of the single machine. The second one of extended SuperMatrix is capable of distributing subsets of <test, MSR, matrix> tuples across processing nodes.

The parallel version was tested for 5 and 10 different MSRs to be tested and 1 and 2 co-incidence matrices on up to 10 processing nodes. The old non-distributed version was tested for the same combinations of MSRs and matrices but only up to 4 cores – the maximal number in the PC used for the test. The results of the experiment are presented in Tab. IV.

We expected that the WBST performance would be increasing with the increasing number of processing but the speed will not rise linearly in relation to the size of the problem. In our expectations we took into account different complexity of the used MSR algorithm and different properties of the two co-incidence matrices applied. WBST parts, i.e., subsets of QA pairs are different from the computation point of view – they differ only in their size. In order to solve one QA pair, the system has to compare four row vectors. The achieved results are generally compatible with our initial expectations. The only difference is that in the case of two matrices processed the speedup grows almost linearly, while in the case of one matrix saturates around 5 nodes.

TABLE III
TIME (MIN) REQUIRED FOR COMPUTING LIST OF THE k MOST SEMANTICALLY RELATED WORDS TO THE GIVEN ONE.

	Number of Nodes								
	1	2	4	6	8	10	12	14	20
k=20	267	138	74	45	37.5	28.5	23	19.5	16.75
k=25000	215.5	117.25	75.5	45.25	37.75	29.25	23.25	19.75	16.75

V. RELATED WORKS

There are a few systems that are similar in functionality to SuperMatrix, but process data in a sequential manner. Reimplementation of LSA [13] is possible with the combination of *MC Toolkit* [25] and SVDPACKC [21]. There are a few problem with this approach, e.g., MC Toolkit supported only ASCII encoding and could only create a words by documents matrix. *Infomap NLP* package [26] supports similar functionality to MC Toolkit with SVD, but it was especially created for the extraction of word meanings from free text corpora. Infomap NLP package has been abandoned in favour of a new system called *Semantic Vectors* (SV) [27]. Main differences with Infomap NLP are: usage of random projection instead of SVD for dimensionality reduction and the implementation, which was written completely in Java, using Apache Lucene as a document-indexing engine.

Distribution of computation in natural language processing has a long tradition in domain of Information Retrieval (IR). However, in IR the main approaches focuses on reducing volumes of fetching, sending and processing of (inverted) index, cf [28], [29].

Pantel et al. present work similar to subset of ours — scaling similarity calculation to very large datasets [30]. However, the focus in [30] was put on distribution of only the similarity function, like cosine or Dice. In our approach we parallelized all the computations (matrix construction, feature selection and filtering, weighting and similarity calculation). Pantel et al. use a approach based on MapReduce framework which is popular in many application in NLP, e.g., clustering [31].

VI. CONCLUSIONS

SuperMatrix appeared to be a very valuable system during the semi-automated construction of plWordNet, see the detailed report on techniques applied in [2]. However, the growing size of both: corpus and plWordNet, caused that the further usage of a single-machine version of SuperMatrix started to be difficult. Work on transforming the system into a system based on the parallel architecture have been initiated.

In SuperMatrix, the basic data structure is a matrix, the majority of operations have a sequential character in relation to processing rows or columns of the matrix. Thus, the main concern was in which areas the effort on introducing parallel and distributed processing will be most effective in terms of the reduced time of processing or the increased scale of problems processed. The latter was even a more severe limitation than the former.

Three main areas were identified for transformation into parallel computation, namely: co-incidence matrix construction, similarity matrix calculation and MSR evaluation by WBST. The first two are the most costly processes in terms of computation complexity and both were bottlenecks for larger problems: the former because of the memory usage, and the latter due to time complexity. Less attention was given so far to the feature-oriented transformations of the matrix, as they are applied to the already collected frequencies and sparse matrices. Matrix-oriented transformations, like SVD, form a separate class of problems, SuperMatrix depends in this area on the third-party solutions, so this important issue was left for further development. Corpus preprocessing, e.g., by a morpho-syntactic tagger, is also an area suitable for parallel processing. However there is an ongoing work on some solutions in this phase in another project, cf. [32], so it was not solved here – SuperMatrix is able to co-operate with some language tools, e.g., the TaKIPI tagger, but from the point of view of parallel SuperMatrix architecture we assume that the input corpus is already pre-processed. Moreover, the morpho-syntactic constraints, which are heavily applied during matrix construction are in fact a language tool similar to a chunker (a kind of a parser).

All introduced solutions appeared to be effective in performed tests and in practice. In the case of the matrix construction the decrease of processing time saturates with the increasing number of processors, but even more important profit is that parallel architecture enables construction of matrices of practically unlimited size in a convenient way, i.e., without human guiding the system through the process. This is important feature, because the construction of large amount of submatrices independently and summing them in correct order is long, laborious and error prone process when done manually. The use of 500,000 features and even more is justified as distribution of words in the natural language is very sparse and different subsets of lexicon can be described by different subsets of lexico-syntactic features, e.g., by different specific adjectives. The present version of SuperMatrix is prepared for this.

The applied technical solution based on the MPI, makes possible to use the system not only on a specialised computing cluster by also on a network of computers or even utilise multi-core architecture of a modern PC.

We plan to make SuperMatrix more flexible with respect to the application to different languages and different corpus annotation schemes. The present version works for Polish and English, but some elements are tightened to some specific language tools.

Possible application areas for SuperMatrix include extraction of MSRs from corpora and semantic correction of handwritten text. This system can be also used to perform unsupervised word sense disambiguation, named entity disam-

biguation, sentiment analysis, document indexing, clustering and retrieval and search engine construction. Last but not least, we suspect that SuperMatrix can be used outside the domain of natural language processing, i.e., everywhere were an object can be represented as a sparse feature vector (especially in high dimensional space).

ACKNOWLEDGMENT

Work financed by Innovative Economy Programme project POIG.01.01.02-14-013/09. Work supported by the DCS-Lab, which is operated by the Department of Distributed Computer Systems (DDCS) at the Institute of Informatics, Wroclaw University of Technology.

REFERENCES

[1] C. Fellbaum, Ed., *WordNet — An Electronic Lexical Database*. The MIT Press, 1998.

[2] M. Piasecki, S. Szpakowicz, and B. Broda, *A Wordnet from the Ground Up*. Oficyna wydawnicza Politechniki Wroclawskiej, 2009.

[3] P. Pantel and M. Pennacchiotti, "Espresso: Leveraging generic patterns for automatically harvesting semantic relations." ACL, 2006, pp. 113–120. [Online]. Available: http://www.aclweb.org/anthology/P/P06/P06-1015

[4] Z. S. Harris, *Mathematical Structures of Language*. New York: Interscience Publishers, 1968.

[5] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora." in *Proceeedings of COLING-92*. Nantes, France: The Association for Computer Linguistics, 1992, pp. 539–545.

[6] R. Kurc and M. Piasecki, "Automatic Acquisition of Wordnet Relations by the Morpho-Syntactic Patterns Extracted from the Corpora in Polish," in *Proceedings of the International Multiconference on Computer Science and Information Technology – Third International Symposium Advances in Artificial Intelligence and Applications*, 2008, pp. 181–188.

[7] B. Broda and M. Piasecki, "SuperMatrix: a General tool for lexical semantic knowledge acquisition," *Speech and Language Technology*, vol. 11, pp. 239–254, 2008.

[8] P. Pantel, "Clustering by committee," Ph.D. dissertation, Edmonton, Alta., Canada, Canada, 2003, adviser-Dekang Lin.

[9] B. Broda and M. Piasecki, "Experiments in documents clustering for the automatic acquisition of lexical semantic networks for polish," in *Proceedings of the 16th International Conference Intelligent Information Systems*, 2008, to appear.

[10] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," *Information Systems*, vol. 25, no. 5, pp. 345–366, 2000. [Online]. Available: citeseer.ist.psu.edu/guha00rock.html

[11] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing maps: exploratory analysis of high-dimensional data," 2002.

[12] G. Karypis, "CLUTO - a clustering toolkit," Tech. Rep. #02-017, nov 2003.

[13] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition," *Psychological Review*, vol. 104, no. 2, pp. 211–240, 1997.

[14] P. D. Turney, "Mining the Web for synonyms: PMI-IR versus LSA on TOEFL," in *Proceedings of the Twelfth European Conference on Machine Learning*. Berlin: Springer-Verlag, 2001, pp. 491–502.

[15] M. Piasecki, S. Szpakowicz, and B. Broda, "Extended similarity test for the evaluation of semantic similarity functions," in *Proceedings of the 3rd Language and Technology Conference, October 5–7, 2007, Poznań, Poland*, Z. Vetulani, Ed. Poznań: Wydawnictwo Poznańskie Sp. z o.o., 2007, pp. 104–108.

[16] D. Freitag, M. Blume, J. Byrnes, E. Chow, S. Kapadia, R. Rohwer, and Z. Wang, "New experiments in distributional representations of synonymy." in *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 25–32.

[17] D. Lin, "Automatic retrieval and clustering of similar words," in *COLING 1998*. ACL, 1998, pp. 768–774. [Online]. Available: http://acl.ldc.upenn.edu/P/P98/P98-2127.pdf

[18] S. Evert, "The statistics of word cooccurrences: word pairs and collocations," *Unpublished doctoral dissertation, Institut f "ur maschinelle Sprachverarbeitung, Universit "at Stuttgart*, 2004.

[19] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[20] A. Purandare and T. Pedersen, "Senseclusters - finding clusters that represent word senses," in *HLT-NAACL 2004: Demonstration Papers*, D. M. Susan Dumais and S. Roukos, Eds. Boston, Massachusetts, USA: Association for Computational Linguistics, May 2 - May 7 2004, pp. 26–29.

[21] M. Berry, "Large scale singular value computations." *International Journal of Supercomputer Applications*, vol. 6, no. 1, pp. 13–49, 1992.

[22] A. Przepiórkowski, *The IPI PAN Corpus: Preliminary version*. Warsaw: Institute of Computer Science, Polish Academy of Sciences, 2004.

[23] "Korpus rzeczpospolitej," [on-line] http://www.cs.put.poznan.pl/dweiss/rzeczpospolita.

[24] B. Wilkinson and M. Allen, *Parallel programming: techniques and applications using networked workstations and parallel computers*. Prentice Hall, 1999.

[25] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1, pp. 143–175, Jan 2001.

[26] D. Widdows, "Unsupervised methods for developing taxonomies by combining syntactic and statistical information," in *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 197–204.

[27] D. Widdows and K. Ferraro, "Semantic vectors: a scalable open source package and online technology management application," in *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, E. L. R. A. (ELRA), Ed., Marrakech, Morocco, may 2008.

[28] A. Moffat, W. Webber, J. Zobel, and R. Baeza-Yates, "A pipelined architecture for distributed text query evaluation," *Information Retrieval*, vol. 10, no. 3, pp. 205–231, 2007.

[29] F. Silvestri, S. Orlando, and R. Perego, "WINGS: a parallel indexer for Web contents," *Computational Science-ICCS 2004*, pp. 263–270, 2004.

[30] P. Pantel, E. Crestan, A. Borkovsky, A. Popescu, and V. Vyas, "Web-scale distributional similarity and entity set expansion," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*. Association for Computational Linguistics, 2009, pp. 938–947.

[31] J. Uszkoreit and T. Brants, "Distributed word clustering for large scale class-based language modeling in machine translation," *Proceedings of ACL-08: HLT*, pp. 755–762, 2008.

[32] B. Broda, M. Marcińczuk, and M. Piasecki, "Building a node of the accessible language technology infrastructure," in *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis, M. Rosner, and D. Tapias, Eds. Valletta, Malta: European Language Resources Association (ELRA), may 2010.