

SuperMatrix: A General Tool for Lexical Semantic Knowledge Acquisition

Bartosz Broda and Maciej Piasecki

Institute of Informatics, Wrocław University of Technology, Poland
{bartosz.broda, maciej.piasecki}@pwr.wroc.pl

ABSTRACT

The paper presents the SuperMatrix system, which was designed as a general tool supporting automatic acquisition of lexical semantic relations from corpora. The construction of the system is discussed, but also examples of different applications showing the potential of SuperMatrix are given. The core of the system is construction of co-occurrence matrices from corpora written in any natural language as the system works on UTF-8 encoding and possesses modular construction. SuperMatrix follows the general scheme of distributional methods. Many different matrix transformations and similarity computation methods were implemented in the system. As a result the majority of existing Measures of Semantic Relatedness were re-implemented in the system. The system supports also evaluation of the extracted measures by the tests originating from the idea of the WordNet Based Synonymy Test. In the case of Polish, SuperMatrix includes the implementation of the language of lexico-syntactic constraints delivering means for a kind of shallow syntactic processing. SuperMatrix processes also multiword expressions as lexical units being described and elements of the description. Processing can be distributed, as a number of matrix operations were implemented. The system serves huge matrices.

1. Introduction

If a *wordnet*¹ for some language does not exist, then... it should be created as quickly as possible. This point of view is probably shared by the majority of researchers working in the area of Natural Language Processing. The stand of developers and companies is much less clear, but even they would love to have an occasion to criticise an existing wordnet for not solving all the large scale problems. There are two weakest points of the wordnet in general: its construction is very laborious process, in which skilled lexicographers must be involved, and it takes a lot of time to construct a new wordnet, even if we start with translating a wordnet built for another language (i.e., usually the English WordNet). Both problems are strictly correlated. While starting a project on the construction of the Polish Wordnet [2], called *plWordNet* (or *Słowosieć* in Polish), we decided to build it from scratch, in order to construct it as a faithful description of the Polish lexical semantic relations. So we increased the amount of work to be done, but at the same time we did not increase the amount of money

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

assigned to the project (anyway, as the sum was quite moderate, so it was not a big difference). However, from the very beginning we planned to support the work of lexicographers by different types of language tools automatically constructed on the basis of large corpora:

- delivering some means of intelligent semantic browsing across *lexical units*² (henceforth, LUs),
- or even suggesting to the lexicographer some lexical semantic relations between LUs or groups of LUs (e.g., wordnet synsets).

Browsing on the basis of LU meaning relations requires some way of measuring *semantic relatedness* between pairs of LUs. Following Edmonds and Hirst [3] we prefer the term semantic relatedness instead of the widely used term of *semantic similarity*, because the former better expresses the nature of a numerical measure one extracts from corpora. A Measure of Semantic Relatedness (henceforth MSR) is a function which for a given pair of LUs returns some real number expressing how semantically close the elements of the given pair are, regardless of the exact nature or cause of this relation:

$$MSR : L \times L \rightarrow R, \quad (1)$$

where L is the set of lexical units and R is the set of real numbers.

As our objective is to build a set of language tools supporting wordnet construction, we will limit the rest of our considerations only to the automatic extraction of instances of lexical semantic relations from corpora. There are two main paradigms of automatic extraction of instances of lexical semantic relations, e.g., [4]:

- *pattern-based*,
- and *clustering-based*, called also distributional paradigm, as it originates directly from the *Distributional Hypothesis* formulated by Harris [5].

According to the pattern-based approaches, there are some lexico-syntactic patterns which combine two LUs and mark the two LU as an instance of some lexical semantic relation, e.g., hypernymy, see e.g., a seminal work of Hearst [6]. So only one occurrence of a precise pattern can signal the given association of LUs.

The clustering-based approaches assume that the similarity of distributions of some LUs across different lexico-syntactic or even semantic contexts is evidence for their close semantic relation. The stronger the similarity, the closer the LUs are in their meaning. The name of the paradigm emphasises that we are looking for similar distributions of LUs and, in some way, we cluster them into groups of highly semantically related LUs.

There are plenty of methods proposed for the automatic extraction of Measures of Semantic Relatedness. All start with processing a corpus and constructing a coincidence 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. At the beginning of our project project it was completely unclear which known MSRs perform better, and which of

²A lexical unit is a one word or multiword lexeme named in the lexicon by its morphological base form and representing a whole set of one word or multiword forms possessing the same meaning and differing in the values of morphological categories.

them would work for Polish. Polish is not only a language using alphabet extended in comparison to the ASCII code,³ is a language typologically different than English, but also is a language with fewer language tools and resources than English. The third problem was the worst, as the difference between English and Polish in this area is huge. Thus we decided to construct a system capable of utilising existing Polish language tools, Polish corpora and reimplementing, investigating and evaluating as many MSRs as possible.

The goal of the paper is to present the constructed system called *SuperMatrix* and discuss its various successful applications. As we would like to make SuperMatrix free for research uses, we hope that the latter can guide potential users to the areas of its applications. The first experiments done with the help of SuperMatrix were presented in [7], and the general scheme of processing was discussed in [8]. This paper presents the first thorough description of SuperMatrix.

2. Blueprint for the Construction of MRSs

There is a plethora of approaches to extraction of similarity between LUs, e.g., [9–12]. Basically, they all follow similar pattern for construction. This general blueprint as implemented in SuperMatrix is shown in Figure 1. Following the idea of *distributional similarity* and Distributional Hypothesis [5], first, co-occurrence data is collected from text corpora for selected words – “Matrix construction” step in Figure 1.

There are three main approaches to represent a context. One can count words occurring together with the given LU inside a passage of text, e.g., inside a paragraph or document. This approach has been used in the technique called *Latent Semantic*

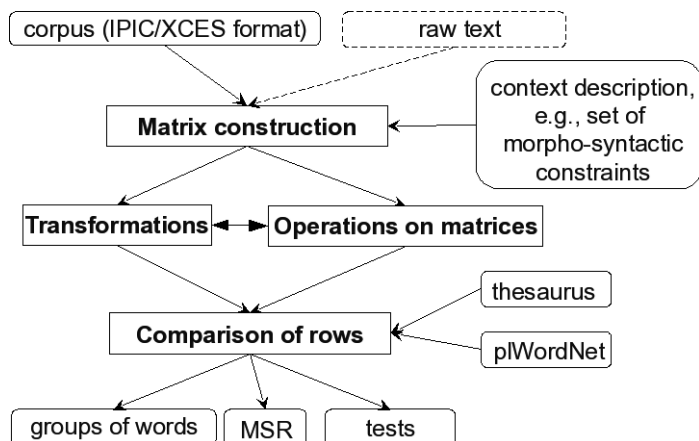


Figure 1. General blueprint for creation of measures of semantic relatedness.

³In 2005, when we started, many similar systems did not process the extended ASCII code, not mentioning UTF-8.

Analysis (LSA) [13]. Another popular method for context representation is counting words co-occurring inside a text window, this approach is called a *Word Space* [14]. In *Hyperspace Analogue to Language* (HAL) [15] smaller weights are assigned to words if they occur further from the centre of the context – the centre is occupied by the LU being described.

Very good results were observed after enriching description of a context with syntactic information, e.g., [8, 9, 12]. This method counts only co-occurrences between an LU in the context centre and selected lexico-syntactic relations in which the LU is involved.

Usually LUs are represented as feature vectors in a high dimensional space. Each feature corresponds to a single context, in which an LU occurs in corpus. It is convenient for further processing to think about the collection of feature vectors in terms of a matrix M (see figure 2), consisting of i rows (words) described by j features (context). The value of $M[n_i, c_j]$ tells how many times the word n_i occurred in the context c_j .

In the last step (“Comparison of rows” in in Figure 1) cooccurrence matrices are used to calculate similarity between words. There are many methods for doing this. One can measure the Euclidean distance between word vectors, calculate cosine between vectors to measure how close they are one to another, etc. [16].

Experiments showed that raw frequency counts of cooccurrences are not very useful from the perspective of lexical semantic knowledge acquisition. First, after the analysis of the collected data it can become apparent that not all the features used are descriptive enough to differentiate between LU meanings. That is why an additional step of *filtering* features is often performed [8, 12]. Second, there is a need to emphasise an inner structure (or a latent structure) of the data before comparing word vectors. There are two main approaches to this problem: based on *transformation* and *weighing*. In the general scheme in Figure 1, the filtering, transformation and weighing are collectively represented by “Transformation” step.

Matrices collected in the first step can be huge. Moreover, the construction of the overall matrix can be based on the combination of several types of lexico-syntactic features. The joint use of lexico-syntactic features and the description based on the text window is also possible. Different types of features can be treated as describing different perspectives on lexical meaning. Thus, a flexible model based on partial matrices and their combination was introduced. Partial matrices based on a selection of features can be independently created, transformed and next combined into one. However, some types of transformation must be applied after the partial matrices has been combined. The flexibility of the processing is signalled in Figure 1 by different possible paths across the steps: “Transformation” and “Operation on matrices”. There are two possible

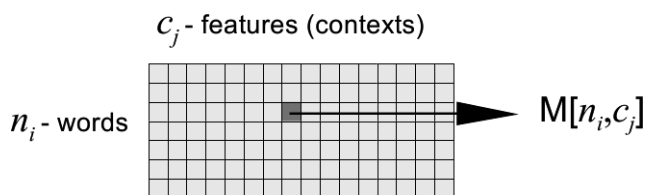


Figure 2. Schematic view of co-occurrence matrix.

ways of combining matrices: summing and joining. Both can be performed along rows and columns. Summing is especially useful in building partial matrices from subcorpora, while joining is especially dedicated to combining partial matrices describing subsets of features or subsets of the lexical unit list.

One of the well known examples of transformation is *Singular Value Decomposition* [17]. It is a method for reducing matrix dimensionality, and was applied in LSA to achieve a form of generalisation from the raw frequency counts.

When comparing nouns one can quickly arrive at the conclusion that almost every noun can be modified by “*liczny*” (*numerous*), but “*bezołowiowy*” (*unleaded*) will be a feature of a few very specific LUs. This is where the weighing is helpful. The basic idea of weighing is to assign greater weights to features that are more descriptive than the others.

The above steps are required to create an MSR. But rarely is the construction of the MSR the main aim of one’s work. Usually it is only a means for achieving some other goal. For example one can cluster LUs in order to semi-automatically extend lexicons [18] or use similarity for the correction of medical handwritten documents [19].

Last but not least, there remains an issue of comparing different MSRs. There are three main approaches to the evaluation of MSRs [20, 21]: mathematical analysis of their formal properties, application specific evaluation and comparison with human judgement. For example, Landauer and Dumais used the third approach for the evaluation of LSA. They used a synonymy part of the *Test of English as a Foreign Language* (TOEFL) to test the ability to automatically differentiate between synonymous and non-synonymous LUs. Because TOEFL is limited in the number of questions it includes and is available only for English, a *WordNet Based Synonymy Test* (WBST) was proposed to generate “a large set of questions identical in format to those in the TOEFL” [22]. An analysis of WBST-based approach to evaluation of MSRs is presented in [21].

An instance of the WBST test is built thus: first, a pair of LUs: $\langle q, s \rangle$, is chosen from a wordnet (*WordNet 2.0* in [22]), where q is a word in question, and s is a randomly chosen synonym of q ; next, three other words that are not in synsets of q and s are randomly drawn from the wordnet – they are a detractor set D . The task for MSR is to point which word from the set $A = D \cup \{s\}$ is a synonym to q . For example for the word *administracja* (*administration*) A consists of: *poddasze* (*attic*), *repatriacja* (*repatriation*), *zarząd* (*board, management*) and *zwolennik* (*follower, zealot*). The word *zarząd* is the correct answer.

3. SuperMatrix

SuperMatrix is a collective name for a set of libraries for programmers and end-user tools for creation, storing and manipulation of co-occurrence matrices describing distributional patterns of LUs.

Overall, the implementation and design has been dictated by requirements placed upon SuperMatrix. Above all the system should be *extensible* and *flexible*. This property is necessary for experimenting with different methods of MSR extraction. *Efficient* processing is also crucial, as statistical methods tend to yield better results

with the increasing amount of data [23]. Thus, the software has been written in C++, with additions of Python bindings and helper scripts.

Because we are working in heterogeneous environment we wanted the system to be as *portable* as possible. This could also lead to lessen effort in embedding parts of the system in end-user applications. After conducting a few experiments we realized that the ability to quickly test new algorithms would be very convenient, so we added *fast prototyping* to the requirement list.

SuperMatrix consists of several modules, namely:

- Matrices – a library for storing matrices.
- Comparator – a library enabling computation of similarity between rows of matrices (i.e., between LUs) using different MSRs.
- Set of tools, including (but not limited to) tools for the creation of matrices: LUs by features, tools for evaluating of MSRs, tools for joining different matrices and analysis of the matrix content, e.g., manually browsing selected rows and columns from any matrix, including transformed and weighed matrices.
- Clustering – package consisting of several clustering algorithms. SuperMatrix can interact with CLUTO [24] and perform Clustering by Committee (CBC) [18], RO-bust Clustering using linkS (ROCK) [25] and Growing Hierarchical Self-Organising Maps (GHSOM) [26]. We reimplemented ROCK and GHSOM with a little modification, CBC was reimplemented, as well as significantly extended [27].
- Set of helper scripts and SWIG⁴ wrappers for main classes of the Matrices and Comparator libraries.

For portability reasons we wanted to avoid external dependencies as much as possible. The only required components are open-source and cross-platform. CMake⁵ is used for our build system. SuperMatrix is also heavily dependant on availability of Boost libraries.⁶ Other used software packages are not so crucial for SuperMatrix, e.g., SWIG is needed only for generating Python wrappers, CLUTO for clustering. For the construction of matrices from Polish corpora we also use parts of the TaKIPI [28] engine – an open-source morpho-syntactic tagger for Polish.⁷ We have tested SuperMatrix under different flavours of Linux as well as MicrosoftWindows.

This system has been under active development for almost two and a half years now. At the time of writing it consists of almost 24 thousands lines of C++ code and almost 3.5 thousands of lines of code written in Python.

3.1. Matrices

The most fundamental question for a software toolkit performing heavy computation on matrices is: how to represent a matrix object in computer memory? There are many options for doing this, i.e., one can store it in a dense or sparse format, for a sparse format there exist many possible representations. Not wanting to be bound to only one implementation we have defined set of operations that a matrix has to be able to perform and tested several different approaches.

⁴Simplified Wrapper and Interface Generator, <http://www.swig.org/>.

⁵<http://www.cmake.org/>.

⁶<http://www.boost.org/>.

⁷Available for download at <http://plwordnet.pwr.wroc.pl/g419/tagger/> and <http://nlp.ipipan.waw.pl/TaKIPI/>.

A dense format was used for small matrices. *Compressed Column Storage* (CCS) [17] is not very convenient for a matrix whose content is being changed constantly. During first experiments [7] we created implementation for storing matrix in database. We tested three possible representations (storing columns, rows or non-zero cells of a matrix), but performance overhead was too large for practical usage. So we have removed support for the storage in database.

It appeared that the most powerful (in terms of flexibility and efficiency) representation is the one using map collection of the standard C++ library. Thanks to guaranteed $O(\log(n))$ complexity we have achieved both flexible and efficient implementation of a matrix. We have encapsulated functionality of a word feature vector using this representation in `CompressedVector` class. Matrices using this representation were called `CCSMatrix` (for storing matrix column-wise) and `CRSMatrix` (for storing in row fashion).

To improve performance we cache some information together with vectors of features. Most importantly we keep entropy of a vector and sum of vector cells. A `SuperMatrix` class is a composition of such cached data (`Feature` class) and `CRSMatrix` (`CCSMatrix`).

Classes in the namespace `Matrices::IO` are capable of saving matrices in a few popular formats. Native `SuperMatrix` format is based on sparse format used by CLUTO [24]. This makes interaction with CLUTO easier. Matrix can also be exported to CCS or CRS format, which enable interaction with such tools like Infomap NLP [29], Sense-Clustres [11] or SVDPACK [17].

In the current representation we have been able to perform efficient computation⁸ using matrices for 13 thousands words described by more then 270 thousands of features created on the basis of a corpus consisting of more then 450 million words.⁹

3.2. Comparator

Comparator library is used for calculating similarity values between rows of a matrix. It is extensible library used for constructing and testing different approaches to the computation of MSRs.

Atypical framework for processing is presented in Figure 3. After the creation of a matrix some global filtering of columns is performed. It follows intuition that some features are not good discriminators for a matrix row. `SuperMatrix` can do filtering using three methods:

- Using a stoplist. This step is performed by most of the algorithms, even if it is not explicitly mentioned. Stoplist consists of functional words (like conjunction or prepositions) and words that are not good meaning bearers. Usually this step is performed before the creation of a matrix, but we allow to do it afterwards.
- Filtering using the *minimal global term frequency* of a word. Most statistically based methods do not cope well with events of extremely low frequencies, so those events are usually treated as outliers and are removed from dataset.

⁸On a contemporary PC, i.e., 1,5 GHz processor with 1 GB of RAM.

⁹254 million words from IPI Pan Corpus [30] 100 million from Rzeczpospolita [31] (Polish newspaper), and 100 from the Polish edition of Wikipedia [32].

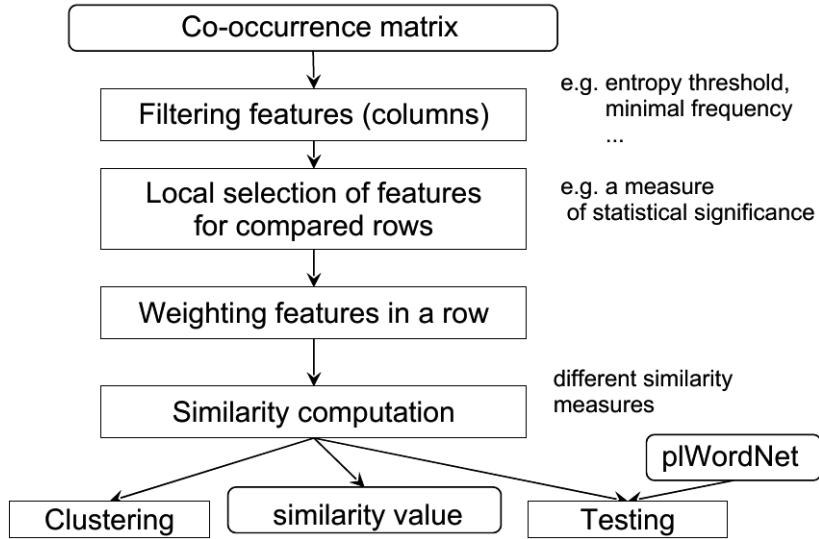


Figure 3. Framework for the computation of MSR.

- Filtering using entropy of a column as a measure of noise it introduces. We used Shannon's entropy:

$$Entropy_w = - \sum_i p_{w,i} \cdot \log p_{w,i} \quad (2)$$

where $p_{w,i}$ is probability of occurrence of the i 'th feature with the word w . Entropy is maximised for the events of maximal uncertainty – here for the features that do not differentiate good between rows of a matrix.

We observed that some features are globally good discriminators, but their values for certain LUs can be caused by some accidental frequencies (e.g., an error of morpho-syntactic disambiguation, sentence boundary detection or a simple spelling mistake). That is why we have isolated yet another step in the process, namely *local feature selection*. It is implicitly present in co-occurrence retrieval models [12] (CRM). The feature sequences selected from the rows for both LUs may have to be padded (usually with zeroes), if the similarity measure requires equal-size vectors.

Z-Score (variant of t-score) can be used as a measure of association between an LU and a feature. If the observed frequency of occurrence of some feature with some LU is significantly greater than expected, then this feature is a good discriminator for this LU.

In the following step we performed weighing of feature vectors of LUs. The aim of this stage is to emphasise the important data in matrix. Several weighing schemes were implemented in SuperMatrix:

- Term Frequency – Inverse Document Frequency is a popular method for decreasing weights of the very frequent words used in Information Retrieval (IR) for assigning lower scores to words occurring in all documents:

$$tf \cdot idf_{t,d} = tf_{t,d} \cdot \log \frac{N}{df_t}, \quad (3)$$

where $tf_{t,d}$ is a number of occurrences of the word t in the document d , df_t is the number of documents containing the word t , N is number of documents. This weighing scheme has been used mainly for the processing of document-by-word matrices.

- In LSA [13] before reduction of the dimension, a matrix has been weighed in two-step process. First the cells of a matrix were scaled logarithmically: for each i, j : $M[w_i, c_j] = \ln(M[w_i, c_j] + 1)$ and divided by entropy of a row of a matrix. We use *logent* as a name for this weighing scheme.
- Z-Score (or t-score) can be used not only for local selection of features, but also as a weighing function, e.g., [8, 33].
- Another popular family of weighing schemes is based on *Mutual Information* (MI). In [9] some formal introduction to MI in the context of extraction of MSRs is presented. In *The Sketch Engine* [34] this measure was used for the generation of a distributional thesauri. In [18] a variant of this measure called *Pointwise Mutual Information* was used (extended with a *discounting factor*).
- Some measures of similarity operate in probability space. There are a few methods for transition from frequencies to probabilities. We used for this purpose *Maximum Likelihood Estimation*.
- To reimplement best faring MSRs from [12] we added weighing schemes based on CRM_{MI} and CRM_{dt} .
- During experiments with different MSRs we noticed [8] that feature values in the matrix depend too directly on frequencies. However no corpus is perfectly balanced, and any weighing function alone does not solve the problem. We need some generalisation from the raw frequencies. Applying SVD to very sparse matrices does not help [7]. We assumed that the similarity of two types of objects depends more on which significant features characterise them than on the exact numerical values of those features' "strength". So, we developed *Rank Weight Function* (RWF) – a weighing scheme that builds relative ranking of importance of features from raw frequencies.¹⁰ Experiments showed that for Polish the MSR based on RWF produces better results in the WBST test [35]. SuperMatrix supports methods enabling transition into the rank space.

After weighing of a matrix one can perform similarity computation. We implemented several similarity functions, e.g., the commonly used, geometry inspired *cosine function*, SIM_{IRAD} – function using divergence of the two probability distributions, SIM_{CRM} for the reimplement of the cooccurrence retrieval models (CRM), Lin's measure based on information theory, etc. Surprisingly, cosine performed very well in comparison to other functions.

On the figure 3 one step is not shown: transformation of a matrix. Transformations are usually computationally intensive, so most of the time they are performed independently of typical process shown described earlier. We use a few methods of transformation in current version of SuperMatrix, namely:

- Singular value decomposition using SVDPACKC [17].

¹⁰For detailed description of RWF see [8, 35].

- Transformation of a LUs-by-feature matrix into the similarity matrix. This transformation is useful for example during exporting data into CLUTO for clustering.
- Relative Frequency Focus – a transformation required for reimplementing of the approach presented in [36].

As a final note on Comparison module we want to emphasise that the framework presented on Figure 3 is not fixed in SuperMatrix. Our framework seems to encompass many, if not all, methods of MSR construction. For instance, to reimplement CRM, one needs the identity function for global selection, some weigh function analysed in CRM for transformation, local selection by the condition $M[w_i, c_j] > 0$ (applied after transformation) and the CRM F-score as the similarity measure.

3.3. Tools

This section selectively describes tools available in Super-Matrix package for usage with little to no coding at all.

1) *Architect*: Collective name for applications used to create different kinds of co-occurrence matrices. Using tools in this category we can create:

- A document-by-word matrix (for document clustering or Information Retrieval).
- A window matrix, i.e., a matrix in which context describes co-occurrence of words in text window of fixed sized (e.g., 5 words to the left and 5 words to the right from the target word).
- A HAL-like matrix – a matrix created in a similar manner to window matrix, but higher scores are assigned to words occurring closer in the text window.
- A sentence matrix – a window matrix with non-fixed size of a text window, in which only co-occurrences in the same sentence are counted.
- A matrix describing LUs by co-occurrence of those LUs in syntactic relations.

Because there is no robust parser available for Polish, for the creation of a matrix describing co-occurrences of LUs in syntactic relation we used a simplified approach based on defining morpho-syntactic constraints. Those constraints are expressed in JOSKIPI – a specialised language developed for TaKIPI [28] – a Polish morpho-syntactic tagger.

It is worth noting that Architect can create matrix for LUs as well as for multiword expressions (MWE). We require only a limited description of syntactic dependencies between constituents of MWEs. In SuperMatrix a module for the automatic extraction of those dependencies is available (for the description of this method see [37]).

As the amount of textual data is increasing, processing of a raw text (or annotated text in XML format) is becoming performance bottleneck. To speed up the process of matrix construction SuperMatrix can read binary format generated by PoliQarp [38].

2) *WBST Tester*: A tool performing evaluation of MSRs by the application of the *WordNet Based Synonymy Tests* (WBST). With addition of a few Python scripts we can generate WBST, test MSRs and perform test on statistical difference of the results produced by different MSRs.

3) *Summator*: A tool for joining different matrices. We observed that combining several matrices created with usage of different morpho-syntactic constraints resulted in better MSRs [21].

4) *VectorExtractor*: A tool for supporting manual analysis of a matrix content, i.e., manual browsing of parts (rows and columns) of huge matrices.

5) *Relations*: SuperMatrix includes also a set of tools for preparing training data used during training classifiers processing pairs of LU and assigning them to different types of wordnet relations, see [39]. Features extracted on the basis of raw matrices or transformed matrices are next stored in the ARFF format which is supported by many Machine Learning systems e.g., Weka system [40] used in [39].

6) *Simbuilder*: In many applications it is easier first to transform the matrix describing LUs by row vectors of features into the square matrix of LU similarity, e.g., the generation of the list of the most similar LUs to the given one applied in the initial phase of CBC [18], or calculation of weights in the RFF MSR [36].

This last task has complexity of $O(n^2)$, where n is dependent on the number of LUs in the matrix. For large matrices the expected time of transformation to the domain of similarity is barely acceptable. For solving this problem, one can apply several approaches, e.g., based on some heuristics that are introduced to increase the speed of performed computations, e.g., [33, 34]. Mostly, precision becomes a little decreased, but the time of processing is reduced a lot.

>From the point of view of the extraction of MSR for the needs of the construction of a lexical semantic network, precision is most important, so its decrease cannot be accepted. Fortunately, the calculation of the LU similarity matrix can be easily distributed. We constructed a tool that can divide the whole task into several computers or processing nodes in a cluster of computers. Communication between processes is based on the Message Passing Interface.

4. Usage examples

The primary application of SuperMatrix is construction of tools for semi-automatic extraction of instances of lexical semantic relations used next in extending plWordNet [41]. Mostly the system is used for the construction of different MSRs [7, 8, 21, 35], but also was applied to clustering text documents [42].

SuperMatrix was applied to the construction of MSRs for: Polish nouns [7, 8, 21], verbs and adjectives [35]. The system was also used in the development of the Rank Weight Function [8], which was next implemented in it. SuperMatrix was used for preparing training data for the construction of classifiers of types of lexical semantic relations [39], e.g., hypernymy, meronymy etc.

An interesting application was the support for the construction of a corpus on the basis of documents from the web. SuperMatrix was applied to construct a tool discovering duplicates of documents in a semi-automatic way. For the documents downloaded from the web¹¹ a document-by-word matrix was built. Next the matrix was transformed to the similarity matrix. On the basis of the similarity matrix pairs of documents whose similarity was above some defined threshold were stored in a file sorted in the descending order of their similarity. The high similarity of documents was a precise signal of duplication, so it was enough to manually check some of the top documents in order to remove duplicates.

¹¹First the downloaded documents were filtered according to the presence of too many words not recognised during the morphological analysis.

Finally, SuperMatrix was utilised in the project whose goal was to develop an OCR of handwritten medical documents. A language model based on the distributional semantic similarity of words was built on the basis of SuperMatrix [19]. The model was next used for the correction of recognition done on the graphical level. A sequence of token positions was delivered to the system. Each token position was assigned a list of potential recognitions for this position. The semantic language model built on the basis of the domain corpus was used in the algorithm called SemWnd, which tried to find a sequence of potential recognitions maximising the semantic consistency of the sequence.

5. Existing systems

There exists a few software solutions that can satisfy some of design requirements stated in Section 1. Reimplementation of LSA for Polish [43] was performed using the combination of *MC Toolkit* [44] and *SVDPACKC* [17]. We have stumbled upon a few problems with that combination. Most important, MC Toolkit supported only ASCII encoding and could only create a word-by-document matrix. Also SVD approach is computationally expensive, so we were able to reduce dimensions of a matrix describing only four thousands nouns appearing in about 180 thousands short documents [7]. *Infomap NLP* package [29] supports similar functionality to MC Toolkit with SVD, but it was especially created for the extraction of word meanings from free text corpora. One noticeable improvement over MC Toolkit is ability to operate in Word Space. It suffered from similar limitations because of using combination of MC Toolkit with *SVDPACKC*. Additionally we had problems with building and installing this system.

Recently, Infomap NLP package has been abandoned in favour of a new system called *Semantic Vectors* (SV) [10]. Main differences in respect to 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. Although Semantic Vectors looks interesting, it currently does not support many MRSs. Also, it does not support other methods for the description of context than co-occurrences in document or text window. First public release of SV happened in October, 2007, when SuperMatrix had most of its functionality already implemented.

Natural Language Toolkit (NLTK) [45] is a popular software for performing fundamental natural language processing task. It does not fully support statistical lexical semantic knowledge acquisition and does not support any Polish corpora.

SenseClusters (SC) [11] is the most relevant and similar package to SuperMatrix. It is a collection of Perl modules and programs for the clustering of similar words (and contexts) based on distributional similarity. It supports a couple of MSRs, creation of Word Space matrices and interacts very well with *SVDPACKC* for dimensionality reduction and CLUTO for clustering. Unfortunately, it does not support Unicode, and cannot use morpho-syntactic constraints or a similar mechanism for the definition of features during matrix construction.

Sketch Engine (SE) [34] provides support for distributional thesauri out of text corpora. At the time of design phase of SuperMatrix SE did not provide full support

for multiword expressions. For computation it used a modification of Lin's measure and currently it uses modification of the Dice coefficient for performance reasons. Being a commercial, closed-source application, we are not aware of possibilities of experimenting with self-madeMSRs inside it. Keeping data on an external server would be inconvenient for our needs too.

6. Conclusions

We have presented SuperMatrix – a general tool for the acquisition of lexical semantic knowledge from text corpora. At the end of the project for construction of Polish wordnet we would like to release the version 1.0 for public usage under a free research licence. Commercial licenses are already available.

Possible application areas for SuperMatrix include already mentioned 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 disambiguation, sentiment analysis, document indexing, clustering and retrieval and search engine construction. One can even use SuperMatrix for the extraction of lexical semantic relation in a way following the pattern-based paradigm, e.g., lexico-syntactic patterns expressed in JOSKIPI are used as features describing matrix columns.

We suspect that our system can be also used outside the domain of natural language processing, i.e., everywhere where an object can be represented as a feature vector (especially in high dimensional space).

We plan to use and extend SuperMatrix in upcoming research projects. Because of the highly parallel nature of processing we will extend tools in a way enabling flexible computation in distributed environment via Message Passing Interface (MPI).

We also want to create tools for the creation of matrices for other languages. As a primary goal we will focus on English, with possible addition of other languages. Also we will extend Comparator module with additional weighing schemes and similarity functions.

Acknowledgment. Work financed by the Polish Ministry of Education and Science, project No. 3 T11C 018 29.

BIBLIOGRAPHY

- [1] C. Fellbaum, Ed., *WordNet – An Electronic Lexical Database*. The MIT Press, 1998.
- [2] M. Derwojedowa, M. Piasecki, S. Szpakowicz, and M. Zawisławska, "Polish WordNet on a shoestring," in *Proceedings of Biannual Conference of the Society for Computational Linguistics and Language Technology, Tübingen, April 11–13 2007*. Universität Tübingen, 2007, pp. 169–178.
- [3] P. Edmonds and G. Hirst, "Near-synonymy and lexical choice," *Computational Linguistics*, no. 28(2), pp. 105–144, 2002. [Online]. Available: <http://ftp.cs.toronto.edu/pub/gh/Edmonds+Hirst-2002.pdf>.
- [4] 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>.

- [5] Z. S. Harris, *Mathematical Structures of Language*. New York: Interscience Publishers, 1968.
- [6] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora." in *Proceedings of COLING-92*. Nantes, France: The Association for Computer Linguistics, 1992, pp. 539–545.
- [7] M. Piasecki and B. Broda, "Semantic similarity measure of Polish nouns based on linguistic features," in *Business Information Systems 10th International Conference, BIS 2007, Poznan, Poland, April 25–27, 2007, Proceedings*, ser. LNCS, W. Abramowicz, Ed., vol. 4439. Springer, 2007, pp. 381–390.
- [8] M. Piasecki, S. Szpakowicz, and B. Broda, "Automatic selection of heterogeneous syntactic features in semantic similarity of Polish nouns," in *Proc. Text, Speech and Dialog 2007 Conference*, ser. LNAI, vol. 4629. Springer, 2007, pp. 99–106.
- [9] 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>
- [10] 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.
- [11] 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.
- [12] J. Weeds and D. Weir, "Co-occurrence retrieval: A flexible framework for lexical distributional similarity," *Computational Linguistics*, vol. 31, no. 4, pp. 439–475, 2005.
- [13] T. Landauer and S. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge," *Psychological review*, vol. 104, no. 2, pp. 211–240, 1997.
- [14] H. Schütze, "Word space," in *Advances in Neural Information Processing Systems 5*, S. Hanson, J. Cowan, and C. Giles, Eds. Morgan Kaufmann Publishers, 1993. [Online]. Available: citeseer.ist.psu.edu/schutze93word.html.
- [15] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, pp. 203–208, 1996.
- [16] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. The MIT Press, 2001.
- [17] M. Berry, "Large scale singular value computations." *International Journal of Supercomputer Applications*, vol. 6, no. 1, pp. 13–49, 1992.
- [18] P. Pantel, "Clustering by committee," Ph.D. dissertation, Edmonton, Alta, Canada, Canada, 2003, adviser-Dekang Lin.
- [19] B. Broda and M. Piasecki, "Correction of Medical Handwriting OCR Based on Semantic Similarity?" *LNCS*, vol. 4881, p. 437, 2007.
- [20] T. Zesch and I. Gurevych, "Automatically creating datasets for measures of semantic relatedness," in *Proceedings of the Workshop on Linguistic Distances*. Sydney, Australia: Association for Computational Linguistics, July 2006, pp. 16–24. [Online]. Available: <http://www.aclweb.org/anthology/W/W06/W06-1104>.
- [21] 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.
- [22] 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.

- [23] J. R. Curran and M. Moens, "Scaling context space," in *ACL*, 2002, pp. 231–238.
- [24] G. Karypis, "CLUTO – a clustering toolkit," Tech. Rep. #02-017, nov 2003.
- [25] 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.
- [26] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing maps: exploratory analysis of high-dimensional data," 2002.
- [27] B. Broda, M. Piasecki, and S. Szpakowicz, "Sense-based clustering of polish nouns in the extraction of semantic relatedness," in *Proceedings of the International Multiconference on Computer Science and Information Technology – 2nd International Symposium Advances in Artificial Intelligence and Applications (AIAA'08)*, 2008.
- [28] M. Piasecki, "Polish tagger TaKIPI: Rule based construction and optimisation," *Task Quarterly*, vol. 11, no. 1–2, pp. 151–167, 2007.
- [29] 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.
- [30] A. Przepiórkowski, *The IPI PAN Corpus: Preliminary version*. Warsaw: Institute of Computer Science, Polish Academy of Sciences, 2004.
- [31] "Korpus Rzeczpospolitej," [on-line] <http://www.cs.put.poznan.pl/dweiss/rzeczpospolita>.
- [32] "Wikipedia," [on-line] <http://pl.wikipedia.org>.
- [33] J. Curran, "From Distributional to Semantic Similarity," Ph.D. dissertation, Ph.D. thesis, University of Edinburgh, 2004.
- [34] A. Kilgarriff, P. Rychly, P. Smrz, and D. Tugwell, "The Sketch Engine," *Information Technology*, vol. 105, p. 116, 2004.
- [35] B. Broda, M. Derwojedowa, M. Piasecki, and S. Szpakowicz, "Corpusbased semantic relatedness for the construction of polish wordnet," in *Proceedings of the 6th Language Resources and Evaluation Conference (LREC'08)*, 2008, to appear.
- [36] M. Geffet and I. Dagan, "Vector quality and distributional similarity," in *Proceedings of the 20th international conference on Computational Linguistics, COLING2004*, 2004, pp. 247–254.
- [37] M. D. Bartosz Broda and M. Piasecki, "Recognition of structured collocations in an inactive language," in *Proceedings of the International Multiconference on Computer Science and Information Technology*, 2007.
- [38] D. Janus and A. Przepiórkowski, "Poliqarp 1.0: Some technical aspects of a linguistic search engine for large corpora," *The proceedings of Practical Applications of Linguistic Corpora*, 2005.
- [39] M. Piasecki, S. Szpakowicz, M. nczuk, and B. Marci Broda, "Classification-based filtering of semantic relatedness in hypernymy extraction," in *Proceedings of the GoTAL 2008*, ser. LNCS, vol. 5221. Springer, 2008, pp. 393–404.
- [40] Weka, "Weka 3: Data Mining Software in Java," 2008, <http://www.cs.waikato.ac.nz/ml/weka/>.
- [41] M. Derwojedowa, M. Piasecki, S. Szpakowicz, M. Zawisławska, and B. Broda, "Words, concepts and relations in the construction of Polish WordNet," in *Proceedings of the Global WordNet Conference, Seged, Hungary January 22–25 2008*, A. Tanács, D. Csendes, V. Vincze, C. Fellbaum, and P. Vossen, Eds. University of Szeged, 2008, pp. 162–177.
- [42] 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.

- [43] M. Piasecki, "LSA based extraction of semantic similarity for Polish," A. Zgrzywa, Ed. Oficyna Wydawnicza Politechniki Wrocławskiej, 2006, pp. 99–107.
- [44] 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.
- [45] S. Bird, "NLTK: The Natural Language Toolkit," in *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*. Sydney, Australia: Association for Computational Linguistics, July 2006, pp. 69–72.
- [46] A. Zgrzywa, Ed., *Proceedings of Multimedia and Network Information Systems*. Oficyna Wydawnicza Politechniki Wrocławskiej, 2006.
- [47] J. Waliński, K. Kredens, and S. Goźdz-Roszkowski, Eds., *The proceedings of Practical Applications in Language and Computers PALC 2005*. Frankfurt am Main: Peter Lang, 2007.