Study of the Parallel Techniques for Dimensionality Reduction and Its Impact on Performance of the Text Processing Algorithms

Marcin Pietron[1,2], Maciej Wielgosz[1,2], Pawel Russek[1,2] and Kazimierz Wiatr[1,2]
[1]*AGH University of Science and Technology, Mickiewicza 30, 30-059 Cracow, Poland*
[2]*ACK Cyfronet AGH, Nawojki 11, 30-950 Cracow, Poland*

Keywords: Latent Semantic Indexing, Random Projection, Singular Value Decomposition, Vector Space Model, TFIDF.

Abstract: The presented algorithms employ the Vector Space Model (VSM) and its enhancements such as TFIDF (Term Frequency Inverse Document Frequency). Vector space model suffers from curse of dimensionality. Therefore various dimensionality reduction algorithms are utilized. This paper deals with two of the most common ones i.e. Latent Semantic Indexing (LSI) and Random Projection (RP). It turns out that the size of a document corpus has a substantial impact on the processing time. Thus the authors introduce GPU based on acceleration of these techniques. A dedicated test set-up was created and a series of experiments were conducted which revealed important properties of the algorithms and their accuracy. They show that the random projection outperforms LSI in terms of computing speed at the expanse of results quality.

1 INTRODUCTION

With the rapid growth of the Internet and other electronic media, the on-line availability of text information has significantly increased. As a result, the problem of automatic text classification turns out to be very important because text categorization has become one of the key techniques for handling and organizing data in many applications for industry, entertainment, and digital libraries, which require access and execution of text-based queries. For this purpose, it is often necessary to automatically classify all given texts into predefined classes. Text classification can be used for clustering (creating clusters of texts without any external information or database), information retrieval (retrieving a set of documents that are related to the query), information filtering (rejecting irrelevant documents) and information extraction (extracting the fragments of information, e.g. email addresses, phone numbers, etc.). Possible applications include such tasks as: email spam filtering, organization of web-pages into hierarchical structures, product review analysis, text sentiment mining, organization of papers according to subject class, and categorization of newspaper articles into topics. Text classification and categorization is considered to be the most popular and the most often performed operation. It always involves documents comparison as an atom step. This in turn requires building a corpus and its reduction to the computationally comprehensible size.

2 SYSTEM DESCRIPTION

The system consists of the Internet data retrieval module, text extraction module, text preprocessing, and the set of methods based on TFIDF, SVD or Random Projection and similarity text metrics for the text classification and clustering (Jamro et al., 2013)(Pietron et al., 2013)(Wielgosz et al., 2013)(Ko and Seo, 2000). The key for successful implementation of the proposed methods is the construction of the reliable corpus for the area and topic of interest detection. At the moment, the topic of interest detection corpus is being built automatically. The news articles are retrieved from news portal (Interia.pl, 2015). The downloaded articles are already classified by journalists which makes it possible to use them as a reference in selected topic and area of interest. The adopted methods are applied to all texts stored in the repository. The first step involves preprocessing activities which are performed in the following order: removal of all special and redundant characters (e. g. colons, brackets, numbers, etc.), lemmatization, and filtering all stop-words. We use the Vector Space Model which requires conversion of each text into a numerical representation. Thus, the second step consists of symbols (words) converting into numerical values that can

Pietron, M., Wielgosz, M., Russek, P. and Wiatr, K.
Study of the Parallel Techniques for Dimensionality Reduction and Its Impact on Performance of the Text Processing Algorithms.
DOI: 10.5220/0005756903150322
In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016) - Volume 1*, pages 315-322
ISBN: 978-989-758-172-4

be used in the subsequent processing stages. In the next step, a selection of the appropriate classification method is performed. The choice depends on which hypothesis is to be verified. For verification of the first hypothesis unsupervised learning model is adopted and for the second one supervised learning is use. In the case of supervised classification, a model (corpus) is built based on all texts in the repository. For unsupervised procedure, the VSM and TFIDF vectors are mapped to the reduced space, which is built with the SVD and RP algorithms.

Figure 1 shows the generic processing flow of the system.

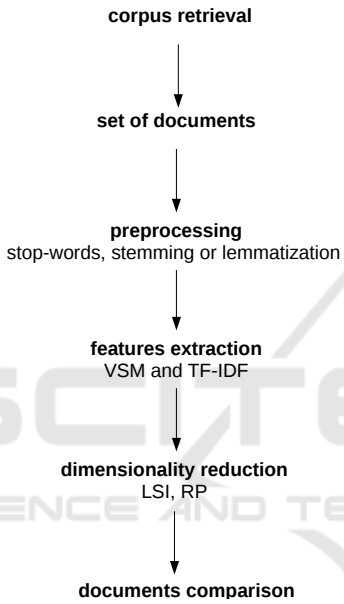

Figure 1: The generic processing flow of the system.

3 TEXT REPRESENTATION

One of the major challenges in the automatic text processing is high dimension of obtained feature vectors. To overcome this problem, both dimensionality and the size of the input text documents can be reduced significantly. The procedure usually consists of methods as removal of all unnecessary characters (e.g. dots, colons etc.), sentence boundary detection, natural language stop-words elimination (Kim et al., 2006) (Lili and Lizhu, 2008), and stemming (Porter, 1980) or lemmatization. Then text documents can be transformed to numerical representation. In our system vectors space model is used.

3.1 Vector Space Model

A text categorization task requires that all symbols (words or n-grams) are converted into a numerical representation, i.e. vector. In the case of this implementation, single words are used as terms. The vector space model has been successfully used as a conventional method for a text representation. This model represents a document as a vector of features (Salton et al., 1975).

A set of documents in this scheme may be presented as a two dimensional term/document matrix. The matrix example is given in table 1. Documents are represented as vectors in an N-dimensional vector space that is built upon all the different terms which occur in the considered text corpus (i.e. document set). Comparison and matching of the texts can be performed by the cosine similarity measure in the VSM. The cosine measure can be calculated according to equation 1.

$$cosine\ similarity(\mathbf{u},\mathbf{v}) = \frac{\sum_{i=0}^{N}(u_i \cdot v_i)}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|} \quad (1)$$

Table 1 and figure 2 present a simple example of three different documents mapped to the two dimensional vector space which means that they are built of two different words.

Table 1: Vector Space Model - sample term/document matrix.

	doc0	doc1	doc2
term0	2	3	1
term1	1	3	2

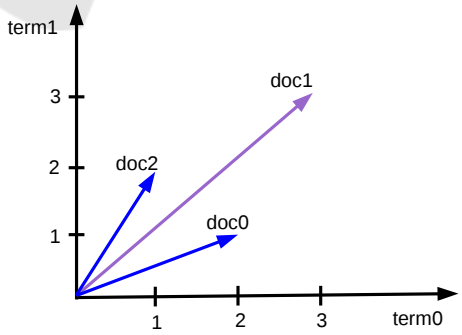

Figure 2: Vector Space Model - sample document visualization.

3.2 TFIDF Representation

The most common algorithm for weighing words in VSM, is the computation of so-called *Term Frequency*

(TF) and *Inverted Document Frequency* (IDF) coefficients. TFIDF is a numerical statistic that is intended to reflect how important is a word to the document in the context of the whole collection. It is often used as a weighting factor in information retrieval and text mining. TF is the number of times a word appears in a given document. This number is normalized, i.e. dividing by the total number of words in a document under consideration. IDF measures how a word is common among all the documents in consideration. The more common a word is, the lower its IDF is. The IDF is computed as the ratio of the total number of documents to the number of documents containing a given word. The TFIDF value increases proportionally to the number of times a word appears in the document, but it is scaled down by the frequency of the word in the corpus. Therefore, common words which appear in many documents, will be almost ignored. Words that appear frequently in a single document will be scaled up. The mathematical formula for TFIDF computation is as follows:

$$(tfidf)_{i,j} = (tf)_{i,j} \times (idf)_{i,j} \tag{2}$$

where $(tf)_{i,j}$ is the term frequency and it is computed as follows:

$$(tf)_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{3}$$

$n_{i,j}$ is the number of occurrences of term t_i in a document d_j. $(idf)_i$ is the inverse document frequency and it is given as:

$$(idf)_i = log\frac{|D|}{|\{d : t_i \in D\}|} \tag{4}$$

$|D|$ is the number of documents in the corpus, $|\{d : t_i \in D\}|$ is the number of documents containing at least one occurrence of the term t_i.

4 DATA DIMENSIONALITY REDUCTION

The data dimensionality reduction process is crucial in many data and text mining algorithms and systems. It can tremendously reduce efficiency of data analysis. The most popular methods are: SVD and Random Projection.

4.1 Latent Semantic Indexing

LSI is a popular method of data dimensionality reduction. It is based on SVD (Singular Value Decomposition (Abidin et al., 2010). The singular value decomposition is a factorization of a real or complex matrix. It expresses a matrix A of $M \times N$ size as a product of three matrices, which is given by equation 5. Matrix A may contain word/document co-occurrences in a case of VSM or TFIDF coefficients.

$$\mathrm{A} = U \times \Sigma \times V^T \tag{5}$$

where: U, V are orthogonal matrices and Σ is a diagonal matrix of singular values (figure 3).

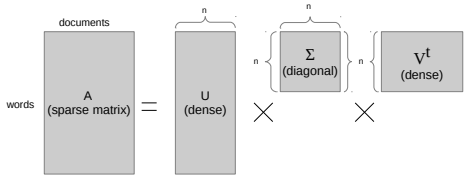

Figure 3: SVD factorization.

SVD transformation is considered to be the best possible approximation of the matrix A. Components (Eq. 5) in the diagonal matrix Σ are located in a decreasing order which is a very convenient feature from data processing perspective since it allows for the elimination of the least important ones (Abidin et al., 2010). The process of dimensionality reduction as applied in SVD makes similar components more similar and different components even more different ones. However, the challenge is the choice of the right number of singular values to be used for matrix reduction. More values do not always mean the better. The SVD factorization is strictly related to Principal Component Analysis (PCA) algorithm with finding iteratively components with maximal variance.

The SVD algorithm is available in many numerical libraries for CPU (e.g. LAPACK) as well as GPU hardware accelerators (e.g. CUDA Toolkit 7.0). SVD and PCA can be computed in three different ways.

- compute SVD and extract components with maximal variance
- compute SVD approximation (e.g. Quic-SVD)
- find iteratively components which carry most information about input data variance and correlation

The first method is accurate but time consuming, the second one has lower computational complexity with worse accuracy, the third one can be flexible i.e. it can be parameterized in terms of a number of computing orthogonal components and accuracy. In this work NIPALS algorithm based on nonlinear regression to find an orthogonal base was employed. It is an iterative algorithm and the sequence of iterations must be preserved. Each iteration of the main loop is a good candidate for hardware parallelization. It is worth noting that there are several linear algebra operations (e.g. vector and matrices multiplication) inside iteration. In our implementation CUBLAS library is used for parallelization of these sections of the code.

4.2 Random Projection

Random projection is a powerful method for dimensionality reduction (Keogh and Pazzani, 2000). In random projection, the original d-dimensional data is projected to a k-dimensional (k << d) subspace through the origin, using a random k × d matrix R whose columns have unit lengths. Using matrix notation where X is the original set of N d-dimensional observations, the algorithm is a projection of the data onto lower k-dimensional subspace. The idea of random mapping arises from the Johnson-Lindenstrauss lemma (Dasgupta and Gupta, 1999)(Dasgupta, 2000), if points in a vector space are projected onto a randomly selected subspace of suitable high dimension, then the distances between the points are approximately preserved (Bingham et al., 2001). Random projection has low computational complexity. It consists of forming the random matrix R and projecting the d×N data matrix X into k dimensions with complexity O(dkN) and if the data matrix X is sparse with about c nonzero entries per column, the complexity is of order O(ckN). Our implementation of Random Projection is based on (Bingham et al., 2001).

5 DIMENSIONALITY REDUCTION AND COSINE METRIC IMPLEMENTATIONS IN GPU

This section describes implementations of both mentioned dimensionality reduction algorithm and cosine metric for sparse and dense vectors.

5.1 GPGPU Hardware Platform

GPGPU is constructed as group of multiprocessors with multiple cores each. The cores share an Instruction Unit with other cores in a multiprocessor. Multiprocessors have dedicated memory chips which are much faster than global memory, shared for all multiprocessors. These memories are: read-only constant/texture memory and shared memory. The GPGPU cards are constructed as massive parallel devices, enabling thousands of parallel threads to run which are grouped in blocks with shared memory. This technology provides three key mechanisms to parallelize programs: thread group hierarchy, shared memories, and barrier synchronization. These mechanisms provide fine-grained parallelism nested within coarse-grained task parallelism. Creating the optimized code is not trivial and thorough knowledge of GPGPUs architecture is necessary to do it effectively. The main aspects to consider are the usage of the memories, efficient division of code into parallel threads and thread communications. Another important thing is to optimize synchronization and the communication of the threads. The synchronization of the threads between blocks is much slower than in a block. If it is not necessary it should be avoided, if necessary, it should be solved by the sequential running of multiple kernels.

5.2 PCA NIPALS and Random Projection implementation

The PCA/SVD Nipals algorithm is based on non-linear regression. SVD factorization can also be associated with another matrix transformation - Principal Component Analysis. The left singular vectors of input matrix X multiplied by the corresponding singular value equal single score vector in PCA method. The non-linear iterative partial least squares (NIPALS) algorithm calculates score vectors (T) and load vectors (P) from input data (X). The outer product of these vectors can then be subtracted from input data (X) leaving the residual matrix (R). Residual matrix can be then used to calculate subsequent principal components. The NIPALS algorithm consists of sequential iterations responsible for computing single orthogonal component. The outer iterations find iteratively projections of input data to the principal components which inherit the maximum possible variance from the input (using non- linear regression). Each iteration consists of few matrix-vector operations (multiplication). These operations are main parts of the algorithm which can be parallelized. Our parallel GPU implementation is based on running highly optimized matrix-vector instruction from CuBLAS library. The core of the algorithm is as follows:

```
// PCA model: X = TP + R
// input: X, MxN matrix (data)
// M = number of rows in X
// N = number of columns in X
// K = number of components (K<=N)
// output: T, MxK scores matrix
// output: P, NxK loads matrix
// output: R, MxN residual matrix

for(k=0; k<K; k++)
{
   cublasScopy (M, &dR[k*M], 1, &dT[k*M], 1);

   a = 0.0;
   for(j=0; j<J; j++)
   {
      cublasSgemv('T', M, N, 1.0, dR, M, \
      &dT[k*M], 1, 0.0, &dP[k*N], 1);
```

```
      if(k>0)
      {
        cublasSgemv('T', N, k, 1.0, dP, N, \
        &dP[k*N], 1, 0.0, dU, 1);
        cublasSgemv('N', N, k, -1.0, dP, N, \
        dU, 1, 1.0, &dP[k*N], 1);
      }
      cublasSscal(N, 1.0/cublasSnrm2(N, \
      &dP[k*N], 1), &dP[k*N], 1);
      cublasSgemv ('N', M, N, 1.0, dR, M, \
      &dP[k*N], 1, 0.0, &dT[k*M], 1);
      if(k>0)
      {
        cublasSgemv('T', M, k, 1.0, dT, M, \
        &dT[k*M], 1, 0.0, dU, 1);
        cublasSgemv('N', M, k, -1.0, dT, M, \
        dU, 1, 1.0, &dT[k*M], 1);
      }
      L[k] = cublasSnrm2(M, &dT[k*M], 1);
      cublasSscal(M, 1.0/L[k], &dT[k*M], 1);
      if (fabs(a - L[k]) < er*L[k])
        break;
      a = L[k];
    }
    cublasSger(M, N, - L[k], &dT[k*M], 1,\
    &dP[k*N], 1, dR, M);
}
```

In case of large data matrices, or matrices with high degree of column collinearity, NIPALS suffers from loss of orthogonality (machine precision limitations accumulated in each iteration step). A Gram-Schmidt (Andrecut, 2009) re-orthogonalization algorithm is applied to both the scores and the loadings at each iteration step to eliminate this problem.

The same method can be adapted for Random Projection algorithm which is based on single matrix-matrix multiplication. In this case two operands are sparse matrices. The complexity when using Achlioptas (Achlioptas, 2001) matrix is: O(ckN/3). The implementation is based on cuBlas matrix multiplication functionality. The efficiency is compared with MKL BLAS matrix operations.

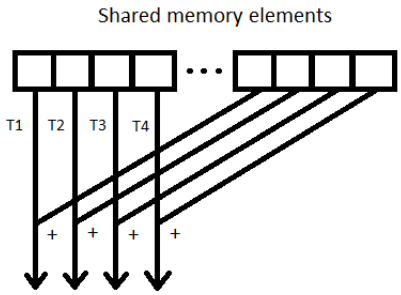

Figure 4: Reduction process on shared memory (Tn - block thread).

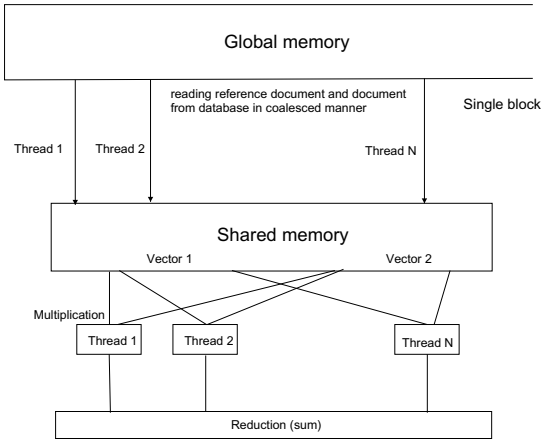

Figure 5: Cosine metric architecture on sparse data.

5.3 Cosine Metric Implementation

One of the algorithm in our classification module is cosine metric computation. Each document is compared with each other by cosine metric computation. The cosine metric was implemented in two formats: original uncompressed (e.g. for reduced data) and compressed for input data (not reduced, sparse data). Uncompressed cosine metric is implemented in such a way that each thread is responsible for reading from global memory and multiplication of single elements from two multiplied vectors (figure 5). The sum of the results of these multiplications is computed by well known GPU reduction operation (scheme on figure 4). In our implementation each block is responsible for single cosine metric computation between two vectors.

In case of compressed format the additional step is done to above algorithm. The binary search is run by each thread to find if read element from first vector exists in the second one (figure 6). If it is true multiplication of their values is run. The final step is a reduction. The compressed vectors are written to memory in such manner that keys and values are stored seperately to avoid bank conflicts. The drawback of this algorithm is that random bank conflicts could happen while executing binary search procedure.

6 EXPERIMENTS

In order to verify the adopted hypothesis several experiments were conducted. All subsequent experiments use the same repository which contains 4200 texts divided into five categories: business, culture, automotive, science and sport. The texts were downloaded from Polish language news website.

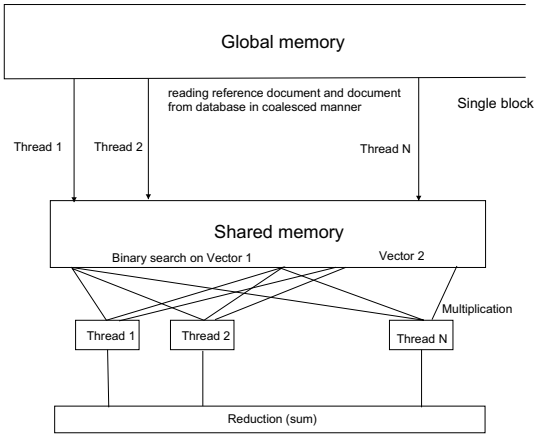

Figure 6: Cosine metric architecture on dense data.

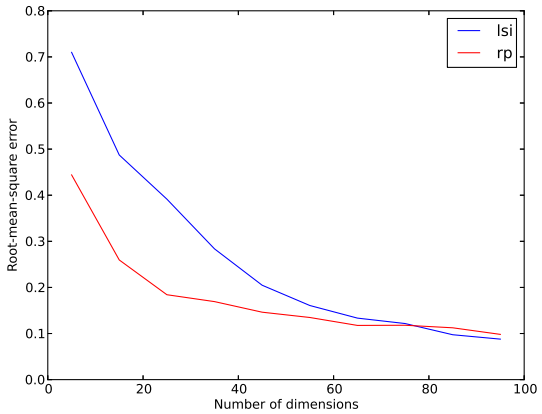

Figure 7: Root mean square error of the dimensionality reduction performed by Latent Semantic Indexing and Random Projection (reduction from 5 to 100).

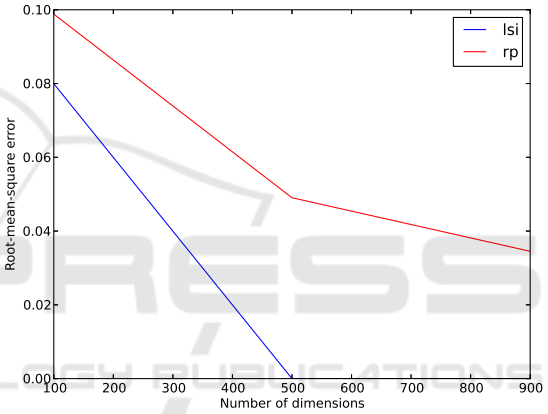

Figure 8: Root mean square error of the dimensionality reduction performed by Latent Semantic Indexing and Random Projection (reduction from 100 to 1000).

6.1 Experimental Setup

In order to facilitate the design process, the dedicated open-source vector space modeling library - Gensim (Rehurek and Sojka, 2010) was used. The library contains a set of text processing procedures such as TFIDF, LSA (Latent Semantic Analysis) which uses SVD, which were optimized by employing NumPy, and SciPy libraries.

6.1.1 Efficiency of Data Dimensionality Reduction

In order to examine properties of Random Projection and Latent Semantic Indexing an experiment was designed and performed. It involved comparing all the reduced vectors against not unreduced ones which resulted in 363×(363 - 1) = 131406 comparisons. Consequently, the mean square error was computed for all the comparisons for each iteration (dimensionality reduced step) (figures 7 and 8). The experiment was conducted for both random projection and latent semantic analysis with the corpus of 363 documents (12394 unique tokens and 63994 corpus positions).

The experiments revealed the statistical properties of the random projection (Bingham et al., 2001) based on Achlioptas approach. Consequently, Random Projection is superior to Latent Semantic Indexing for a low number of dimensions but as the number of dimensions grows LSI supersedes RP. This is the case because LSI (SVD) reduction model is created from the data which means that it reflects the profile of the documents being processed.

The results of SVD/PCA NIPALS algorithm implementation (see pseudocode) is described in table 2. It shows scalability of the algorithm on single core CPU and GPGPU. The GPGPU is about five times faster than CPU with SIMD instruction implementation. The table 3 presents Random Projection algorithm run by CUBLAS matrix-matrix operation and MKL BLAS on CPU. Tables 4 and 5 describe speed-up gained by cosine metric GPGPU implementations. The version with uncompressed format (fromat used for comparison after reduction) is about 15 times faster in GPGPU then its counterpart in CPU. It is worth noting that compressed format should be used. Therefore table 6 presents results of computing compressed cosine metric of 360 documents corpus in original and not in reduced form. The tests where run on Intel Xeon E5645 2.4GHz CPU single core and NVIDA Tesla m2090. The input matrices were defined by single precision floating point format. All results presented in tables are average values collected in five measure probes (standard deviation less than 10% of average values).

Table 2: Results (in miliseconds) of SVD/PCA algorithm implementation in CPU (1 core) and GPU (corpus of 360 documents with 29000 term space).

Reduction size	GPGPU	CPU
10	33	80
20	77	305
30	107	420
40	161	624
60	219	1050
100	369	1789

Table 3: Results (in miliseconds) of RP algorithm implementation in CPU and GPU (corpus of 360 documents with 29000 term space).

Reduction size	GPGPU	CPU (1 core)
10	1	2.5
20	2	5
30	2.7	6
40	3	9
60	4	13
100	9	20

Table 4: Results (in miliseconds) of cosine metric computation (compressed format) in GPU and CPU for different size of corpuses (with constant size of each document equals 512).

Size of corpus	GPGPU	CPU (1 core)
360	55	346
500	117	512
1000	451	1876
2000	1797	7551
10000	44631	197695

Table 5: Results (in miliseconds) of cosine metric computation (original uncompressed format) in GPU and CPU for different size of corpuses (with constant size of each document equals 512).

Size of corpus	GPGPU	CPU (1 core)
360	7	126
500	11	236
1000	40	705
2000	152	2477
10000	3639	65046

7 CONCLUSIONS AND FUTURE WORK

Our work shows that dimensionality reduction in text mining can significantly improve computational complexity of algorithms by maintaining accuracy level. Additionally presented methods are scalable and can be speeded up on parallel hardware platforms like GPGPU. The Random Projection algorithm outperforms effectiveness of PCA/SVD by preserving quite good accuracy. The Random projection is a good candidate for data reduction in real time text processing systems. Future work will concentrate on implementing and speeding up other similarity metrics and adjusting them to Random Projection algorithm.

Table 6: Time of execution of compressed cosine metric of 29000 element vectors in corpus of 360 documents (ms).

max number of non zero elements	CPU time
2000	1060
4000	2120
6000	3200
8000	4220
10000	5259

ACKNOWLEDGEMENTS

This research is supported by the European Regional Development Program no. POIG.02.03.00-12-137/13 PL-Grid Core.

REFERENCES

Abidin, T., Yusuf, B., and Umran, M. (2010). Singular value decomposition for dimensionality reduction in unsupervised text learning problems. In *Education Technology and Computer (ICETC), 2010 2nd International Conference on*, volume 4, pages 422–426.

Achlioptas, D. (2001). Database friendly random projections. *ACM Symposium on the Principles of Database Systems*, pages 274–281.

Andrecut, M. (2009). Parallel gpu implementation of iterative pca algorithms. *Journal of Computational Biology*, 11:15931599.

Bingham, Ella, Mannila, and Heikki (2001). Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 245–250, New York, NY, USA. ACM.

Dasgupta, S. (2000). Experiments with random projection. *Uncertainty in Artifficial Intelligence*.

Dasgupta, S. and Gupta, J. (1999). An elementary proof of the johnson-lindenstrauss lemma. *International Computer Science Institute, Technical Report TR-99-006, Berkeley, California, USA*.

Interia.pl (2015). http://interia.pl.

Jamro, E., Wielgosz, M., Russek, P., Pietron, M., Zurek, D., Janiszewski, M., and Wiatr, K. (2013). Implementation of algorithms for fast text search and files comparison. In *Proceedings of the High Performance Computer Users Conference KU KDM 2013*, pages 83–84. Academic Computer Centre Cyfronet AGH, Academic Computer Centre Cyfronet AGH.

Keogh, E. and Pazzani, M. (2000). A simple dimensionality reduction technique for fast similarity search in large time series databases. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Kim, S., Han, K., Rim, H., and Myaeng, S. H. (2006). Some effective techniques for naive bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1457–1466.

Ko, Y. and Seo, J. (2000). Automatic text categorization by unsupervised learning. In *Proceedings of the 18th international conference on computational linguistics*, pages 453–459.

Lili, H. and Lizhu, H. (2008). Automatic identification of stopwords in chinese text classification. In *Proceedings of the IEEE international conference on Computer Science and Software Engineering*, pages 718–722.

Pietron, M., Zurek, D., Russek, P., Wielgosz, M., Jamro, E., and Wiatr, K. (2013). Accelerating aggregation and complex sql queries on a gpu. In *Proceedings of the High Performance Computer Users Conference KU KDM 2013*, pages 21–22, Krakw. Academic Computer Centre Cyfronet AGH.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Rehurek, R. and Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Wielgosz, M., Koryciak, S., Janiszewski, M., Pietroń, M., Russek, P., Jamro, E., Dabrowska-Boruch, A., and Wiatr, K. (2013). Parallel mpi implementation of n-gram algorithm for document comparison. *ACACES 2013 : the 9th international summer school on Advanced Computer Architecture and Compilation for High-performance and Embedded Systems*, pages 217–220.