Improving text classification with vectors of reduced precision

Krzysztof Wróbel†‡, Maciej Wielgosz†‡, Marcin Pietrórós†‡, Michał Karwatowski†‡, Aleksander Smywiński-Pohl†

∗Jagiellonian University
Kraków, Poland
†AGH University of Science and Technology
Kraków, Poland
‡Academic Computer Centre CYFRONET
Kraków, Poland

Email: {kwrobel, wielgosz, pietron, mkarwat, apohllo}@agh.edu.pl

Abstract—This paper presents the analysis of the impact of a floating-point number precision reduction on the quality of text classification. The precision reduction of the vectors representing the data (e.g. TF-IDF representation in our case) allows for a decrease of computing time and memory footprint on dedicated hardware platforms. The impact of precision reduction on the classification quality was performed on 5 corpora, using 4 different classifiers. Also, dimensionality reduction was taken into account. Results indicate that the precision reduction improves classification accuracy for most cases (up to 25% of error reduction). In general, the reduction from 64 to 4 bits gives the best scores and ensures that the results will not be worse than with the full floating-point representation.

I. INTRODUCTION

NATURAL Language Processing (NLP), as well as Image Processing, is a part of Artificial Intelligence. Despite intensive research and huge recent progress in Deep Learning Techniques, applications of NLP have not reached a level that would allow a construction and a practical implementation of robots and machines operating like humans. Such human-level solutions would allow for seamless and smooth communication between machines and people. The future communication interfaces will allow to convey information directly to the machines processing units using natural language [1][2][3]. This future vision, however, requires a substantial progress in both speech recognition and text processing domains. Applications of those two domains are in an essence very similar and share most of the processing flow. In our research [4] we focus on text processing, but the proposed modules may also be employed in voice processing solutions.

NLP as a research and application field has been developed in a course of last few decades [5][6][7][8][9]. Three different models of the language representation have been established, namely Boolean Model, Vector Space Model (VSM) and Sparse Representation Model [10]. The latter model slowly becomes a standard for applications and systems using Natural Language Processing [11]. This is due to its superior performance, which in turn results from the fact that it mimics the language representation within a human brain [12]. It is worth noting that a language as such belongs to a human cognition domain. It was developed by humans to enable communication and was implemented with biological components in a neural fashion [13]. Therefore, pure ontological models of the language tend to be inferior to the biologically–inspired ones [14].

Representation of knowledge within a human brain is highly distributed, sparse and hierarchical [12][13]. Neural operations of cognition, which also involve language processing, are performed using single bit precision. Every bit of the information carries semantic meaning which reflects relationships between concepts acquired and stored within the brain. Inspired by this we decided to examine to what extent it is possible to implement such a bit processing scheme on a top of currently used models in NLP. We focused on the Vector Space Model as one which is popular and widely used in various applications. However, the research results may also be transferred to the other models since all of them employ vector as a basic representation structure. The vectors are a collection of fixed or floating-point numbers which represent a certain dynamic range of a data representation. It turns out that the dynamic range, at least in the case of floating-point numbers, is too large and can accommodate much more information than necessary. Therefore, we decided to reduce the range to the extent that, on the one hand still preserves a required precision and on the other hand substantially decreases the number of bits. Furthermore, such an approach also has a beneficial effect on semantics of the processed lingual content since it leads to a generalization. This, in turn, has all the positive properties of distributed representations [10]. Consequently, precision reduction of vector representation may be perceived as way of concept generalization or abstraction which is in its essence similar to latent semantic indexing [15] and random projection [16].

Precision reduction approach may not have significant per-
formance impact on standard processors, as they typically operate on fixed data width, usually stored in IEEE–754 floating-point representation. Therefore, reduction to below standard width or, moreover, not byte aligned width, does not introduce notable speedup. The situation improves for single instruction, multiple data (SIMD) processors, like general–purpose computing on graphics processing units (GPGPU), or vector CPUs however data alignment is still required and speedup is only achieved through parallelism and reduction of clock cycles required to process given an amount of data. Real benefits of precision reduction can be observed on fully customizable platforms, such as field-programmable gate arrays (FPGA) [17][18][19]. They are not bound to any specific bitwidth or representation. Data may be stored in any integer bitwidth, which can also differ between consecutive processing stages. Narrower representation requires a less complicated circuit to execute calculations, which improves operating frequency. Switching to fixed point representation further reduces circuit complexity, thus increases operating frequency, which can also vary between processing stages. Data flow architecture can also be designed to process data in a parallel manner. A combination of aforementioned features makes FPGA a very interesting choice as a hardware platform. However, creating efficient design architecture and its implementation are not trivial and generate interesting research task. As authors of this paper already began work on the dedicated hardware platform and presented their initial results in [20], we will not cover this topic. Still much effort needs to be put into FPGA implementation in order to utilize its potential in NLP tasks.

Consequently, the paper addressed three main objectives:

- an examination of the precision reduction impact on the text classification results,
- proposition and practical verification of various popular classification methods with different grade of reduced precision,
- building a framework for precision reduction which is available on–line.

The rest of the paper is organized as follows. Section II introduces vector space model as text documents representation. Section III describe a procedure of precision reduction used in our experiments. Sections IV and V describes SVD and parameters of the employed classifiers, respectively. Experiments are presented in Section VI. Finally, we present our conclusions in Section VII.

II. VECTOR SPACE MODEL

We decided to use Vector Space Model in our experiments to validate an adopted approach to semantic vector reduction. The section that follows may be considered as a short introduction to the model and its aspects that we address in the paper.

The VSM has already been successfully used as a conventional method for the text representation [21]. The documents are represented as vectors in an N–dimensional vector space that is built upon the N different terms that occur in the considered document set (i.e. text corpus). Since the model ignores the order of the words it is usually called a bag-of-words model, to reflect the process of throwing the words form the text to an unstructured bag. The coefficients of the vector are the weights that identify the significance of a particular term in the document. Consequently, a set of documents in this scheme may be presented as the term–document matrix. An example of such a matrix is given in Table I

| Table I | A sample term–document matrix in the VSM |
|---------|-----------------------------------------|
|         | doc0 | doc1 | doc2 |
| term0   | 2    | 3    | 1    |
| term1   | 1    | 3    | 2    |

Fig. 1 presents a simple example of the three different documents mapped to the two–dimensional vector space, which means that they are built from two different words.

The text similarity can be easily calculated by the cosine similarity measure in the VSM. The cosine measure is given by the equation

$$
cosine\ similarity(u, v) = \frac{\sum_{s=0}^{N-1}(u_s \cdot v_s)}{\|u\| \cdot \|v\|}, \quad (1)
$$

where $v = (v_0, \ldots, v_{N-1})$ and $u = (u_0, \ldots, u_{N-1})$ are the vectors representing documents.

The most common method of words weighing in VSM is the computation of the coefficients of so–called Term Frequencies (TF) and Inverted Document Frequencies (IDF). TF is the number of times a word appears in a given document that is normalized i.e. it is divided by the total number of words in
a document under consideration. Respectively, IDF measures how common a word is among all documents in consideration. The more common the 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.

Consequently, TF–IDF is a numerical statistic that indicates how important the word is in characterizing a given document in the context of the whole collection. It is often used as a weighting factor in information retrieval and text mining. The mathematical formula for TF–IDF computation is

$$tf \cdot idf = tf \cdot idf_t.$$ (2)

The $tf$ value is the term $t_i$ frequency in the document $d$, and it is computed as

$$tf = \frac{n_t}{\sum_{i=0}^{k} n_s},$$ (3)

where $n_t$ is the number of occurrences of term $t_i$ in a document $d \in D$. The $idf_t$ value is the inverse document frequency that is given as

$$idf_t = \log \frac{|D|}{|\{d \in D : t_i \in d\}|},$$ (4)

where $|D|$ is the number of documents in the corpus, and $|\{d \in D : t_i \in d\}|$ is the number of documents containing at least one occurrence of the term $t_i$. The TF–IDF 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, which helps control the fact that some words are generally more common than others. Therefore, common words which appear in many documents will be almost ignored. Words that appear frequently in a single document will be scaled up. In this work, the algorithms use the TF–IDF coefficients.

### III. Precision Reduction

Language models are usually very large multidimensional structures composed of vectors. The vectors contain IEEE–754 floating-point numbers which can be either stored in dense or sparse format for a sake of a storage space utilization reduction. In the sparse representation only non–zero vector elements are kept and the rest is skipped. Regardless of the representation, the low–level elements of the vectors are numbers. In order to reduce the representation precision, we apply a three–stage procedure given by Eq. 5 7 9 11 and 13 for the IEEE–754 single precision case. The procedure for IEEE–754 double precision is the same and the equations were provided for a sake of completeness and consistency.

We reduce precision of each vector element given by Eq. 5 and 6

1. It should be noted, however that OKAPI BM25 model [22] that is similar to TF–IDF works better in the context of information retrieval and is assumed a standard implemented in the major full text search engines such as SOLR and ElasticSearch.

$$S_{single} : \{\pm 2^{-126} \ldots (2 - 2^{-23}) \times 2^{127}\} \times n$$ (5)

$$S_{double} : \{\pm 2^{-1022} \ldots (2 - 2^{-52}) \times 2^{1023}\} \times n$$ (6)

where $S$ and $n$ is a vector of IEEE–754 floating–point numbers and its dimension, respectively.

$$S_{norm–single} : [0, \frac{1}{2^{32}}, \ldots, 1 - \frac{1}{2^{32}}, 1] \times n$$ (7)

$$S_{norm–double} : [0, \frac{1}{2^{64}}, \ldots, 1 - \frac{1}{2^{64}}, 1] \times n$$ (8)

In the first step a vector of floating–point numbers: Eq. 5 and 6 are mapped to the fixed–point representation: Eq. 7 and 8 by means of projection operation: Eq. 9 and 10

$$S_{single} \xrightarrow{\pi_{mapping}} S_{norm–single}$$ (9)

$$S_{double} \xrightarrow{\pi_{mapping}} S_{norm–double}$$ (10)

In the second step the vector elements represented as fixed–point numbers: Eq. 11 and 12 which may be regarded as a projection expressed by Eq. 13 and 14

$$S_{single–reduced} : [0, \frac{1}{2^{32} - r}, \ldots, 1 - \frac{1}{2^{32} - r}, 1] \times n$$ (11)

$$S_{double–reduced} : [0, \frac{1}{2^{64} - r}, \ldots, 1 - \frac{1}{2^{64} - r}, 1] \times n$$ (12)

$$S_{norm–single} \xrightarrow{\pi_{reduction}} S_{single–reduced}$$ (13)

$$S_{norm–double} \xrightarrow{\pi_{reduction}} S_{double–reduced}$$ (14)

Generated TF–IDF coefficients are in IEEE–754 double floating–point format and their values span between 0 and 1. Therefore to map these values to desired fixed precision is enough to multiply them by the maximal value possible to encode with that precision.

1. $max\_value = 2^{bit\_width} - 1$
2. for $tf \_idf \ in \ database:$
3) \( \text{norm}_tfidf = \text{ceil}(tfidf \times \text{max_value}) \)

Back normalization to floating-point format is performed accordingly, only the value needs to be divided by maximal value.

1) \( \text{max_value} = 2^{\text{bitwidth}} - 1 \)
2) for \( \text{norm_value} \) in results:
3) \( \text{value} = \text{norm_value}/\text{max_value} \)

The reduction parameter \( r \) strongly affects performance results since it directly decides about a number of bits which are left for the vector elements representation. It is worth noting that it is possible to employ global dimensionality reduction techniques such as SVD along with the methods proposed in this paper. In this work, we consider the order of these operations (precision reduction before or after SVD) for the sake of the best final results.

IV. SINGULAR VALUE DECOMPOSITION

Singular value decomposition (SVD) is used frequently with the term–document matrix representation. SVD allows for dimensionality reduction of the sparse data to a low-rank dense matrix. It also addresses the problem of polysemy in the context of text documents, since words that share meaning are mapped to vectors occupying narrow region in the reduced vector space.

SVD is explained with the following formula:

\[
\mathbf{M} = \mathbf{U} \Sigma \mathbf{V}^\top,
\]

where \( \mathbf{M} \) is the sparse term-document matrix, \( m \) is the number of terms, \( n \) is the number of documents, \( r = \min(m, n) \). In such case \( \mathbf{U} \) and \( \mathbf{V} \) are orthogonal matrices and \( \Sigma \) is a diagonal matrix.

Lowering rank to \( k \) is done by taking \( k \) of the largest values (components) along the diagonal of \( \Sigma \) and truncating \( \mathbf{U} \) and \( \mathbf{V} \).

\[
\mathbf{M}_k = \mathbf{U}_k \Sigma_k \mathbf{V}_k^\top
\]

This approach is named as Latent Semantic Analysis (LSA) or Latent Semantic Indexing (LSI) [23]. A simple interpretation of LSI is that SVD combines terms into more general concepts (synonyms or topics) and documents are represented by a weighted set of the topics.

V. CLASSIFICATION

In order to evaluate the influence of the precision reduction on the robustness of VSM model we employed them in the problem of multi-class (single-label) text classification. We have chosen \( k \)-nearest neighbors algorithm (KNN), linear regression (LR) and support vector machines (SVM) as the tested classifiers.

KNN was used with cosine similarity metric and the number of neighbors \( k \in \{1, 5\} \). The algorithm does not require training, but the testing phase involves calculating similarity with every document. It also needs to store all the documents from the training corpus. As such it is not well suited for large corpora, which are much more popular in the recent years.

LR and SVM are similar algorithms. In LR we applied L2 regularization. SVM was trained with hinge loss and linear kernel. Both execute iterative training and do not store documents for testing.

For macro-averaged objective the weights associated with classes were adjusted inversely proportional to class frequencies in the input data

\[
w_c = \frac{\sum n_i}{n_c}
\]

where \( w_c \) is a weight associated with class \( c \) and \( n_i \) is a number of samples in class \( i \).

VI. EXPERIMENTS AND THE DISCUSSION

A. Experimental Setup

4 modules were developed in order to execute experiments:

1) TF–IDF: Term frequency-inverse document frequency was calculated on training data without setting any limit on the number of words.

2) Precision Reduction (b): Precision reduction was performed on VSM representation of documents as described in [III] where \( b \) is the precision in bits.

3) SVD (k): Singular value decomposition was used to reduce the dimensionality of data, where \( k \) is the number of components.

4) Classification: 4 classifiers were used: \( k \)-nearest neighbors algorithm with cosine similarity metric for \( k \in \{1, 5\} \), logistic regression and support vector machines with linear kernel.
5 variants of experiments were performed:

- TF–IDF and Classification,
- TF–IDF, Precision reduction (b) and Classification,
- TF–IDF, Precision reduction (b), SVD (k) and Classification,
- TF–IDF, SVD (k) and Classification,
- TF–IDF, SVD (k), Precision reduction (b) and Classification,

where $b \in \{16, 8, 7, 6, 5, 4, 3, 2, 1\}$ and $k \in \{100, 200, 300, 400, 500, 1000\}$.

All results were obtained by taking an average of 5–fold cross–validation scores. Each datasets was randomly shuffled, partitioned into 5 subsets. The process of evaluation was repeated 5 times, with one subset used exactly once as testing data and the rest 4 as training data.

All experiments were performed in Python using scikit–learn [24] library with default parameters. Calculations were performed on 64–bit floating point type with 4 cores of Intel Xeon E5–2680v3.

Framework performing precision reduction is available at: https://github.com/kwrobel-nlp/precision-reduction. It determines what is the best number of bits for classification of specified corpus. Datasets used in this work are shared for reproducibility of results.

B. Datasets

Experiments were performed on multi–class (single–label) datasets. 5 datasets are publicly available:

- **webkb** - webpages collected from computer science departments,
- **r8** - Reuters articles with single label from R10 subcollection of Reuters-21578,
- **r52** - Reuters articles with single label from R90 subcollection of Reuters-21578,
- **20ng** - newsgroup messages,
- **cade** - webpages extracted from the CADÊ Web Directory.

All of them are pre–processed by [25]:

- all letters turned to lowercase,
- one and two letters long words removed,
- stopwords removed,
- all words stemmed.

Multi–label datasets were transformed to single–label by removing samples with more than one class.

Table II shows summary of corpora’s main features. Corpora webkb, r8, r52 and 20ng are in English. cade is in Brazilian-Portuguese. cade is the largest dataset in terms of the number of documents, vocabulary and average length of documents. 20ng is the most balanced (0.1 relative standard deviation), others are very skewed.

C. Quality Measure

The macro–averaged F1 score is used as a quality evaluation of the experiments' results presented in this paper. The precision and recall for corresponding classes are calculated as follows:

$$Precision(i) = \frac{tp_i}{tp_i + fp_i}, \quad (18)$$

$$Recall(i) = \frac{tp_i}{tp_i + fn_i}, \quad (19)$$

where $tp_i$ is the number of items of class $i$ that were classified as members of class $i$, $fp_i$ is the number of items of class other than $i$ that were wrongly classified as members of class $i$ and $fn_i$ is the number of items of class $i$ wrongly classified as members of class other than $i$. The class’ F1 score is given by the following formula:

$$F1(i) = 2 \cdot \frac{Precision(i) \cdot Recall(i)}{Precision(i) + Recall(i)}. \quad (20)$$

The overall quality of the classification can be obtained by taking the unweighted average F1 scores for each class. It is given by the equation:

$$F1 = \frac{1}{c} \sum_i F1(i), \quad (21)$$

| Dataset   | webkb | r8   | r52  | 20ng | cade |
|-----------|-------|------|------|------|------|
| Classes   | 4     | 8    | 32   | 20   | 12   |
| Documents | 4199  | 764  | 9100 | 18821| 40983|
| Vocabulary| 77/10 | 17/38| 19/24| 70/213| 193/597|
| Average number of words in document | 909 | 390 | 418 | 851 | 913 |
| Smallest class | 504 | 51 | 3 | 628 | 625 |
| Largest class | 104 | 392.5 | 392.5 | 995 | 844.9 |
| Average size of classes | 1049 | 959 | 175 | 914 | 341.5 |
| Standard deviation of sizes of classes | 408 | 1309 | 613 | 94 | 245.1 |
| Relative standard deviation of sizes of classes | 0.39 | 1.36 | 3.31 | 0.10 | 0.72 |
where $c$ is the number of all classes. The F1 score value ranges from 0 to 1, with a higher value indicating a higher classification quality.

The error is defined as:

$$\text{Error} = 1 - F1.$$  \hspace{1cm} (22)

The error reduction is defined as:

$$\text{ErrorReduction} = \frac{(Error_{\text{ref}} - Error_{\text{new}})}{Error_{\text{ref}}},$$ \hspace{1cm} (23)

where $Error_{\text{ref}}$ is a reference value of error and $Error_{\text{new}}$ is the new value of error.

To compare the results with other studies, micro–averaged accuracy is used. Micro–averaging does not take imbalance of classes into account.

$$\text{Accuracy} = \frac{\sum tp_i}{n},$$ \hspace{1cm} (24)

where $n$ is a number of all training samples.

D. Results

Error values on the corpora for each classifier in function of precision bits are shown in Fig. 2. For every dataset logistic regression and SVM obtain smaller error than KNNs. LR and SVM are more powerful because they model inputs (i.e. terms) in relation to classes. Precision reduction with KNNs improves results on webkb, r8 and 20ng datasets. KNN 5 scores higher than KNN 1 on webkb and cade.

Fig. 3 shows averaged error reduction among the corpora for the classifiers. For SVM the precision reduction is the least beneficial. For other classifiers macro–averaged errors decrease with the reduction of precision down to 3 bits. However, micro–averaged errors are the smallest for the precision of 1-3 bits. Four times reduction of precision from 64 bits to 16 bits does not change the classification results.

Fig. 4 shows averaged error reduction measure among the corpora for the classifiers with a precision reduction after SVD. The results indicate that introducing the precision reduction after SVD generates more errors in every case.

Fig. 5 presents F1 measure for 3 variants: TF–IDF, TF–IDF with the best precision reduction and TF–IDF with the best SVM. Precision reduction gives better or similar results as applying SVD except for KNNs on r8. K–nearest neighbors algorithm with precision reduction gives similar results as raw logistic regression on r8, r52, and 20ng datasets. In the raw form SVM has the best results for the English datasets.

Fig. 6 presents comparison of F1 score on variant TF–IDF with SVM with and without precision reduction before SVD. Precision reduction before SVD has always positive impact, especially seen on webkb dataset.

Table III shows overall macro–averaged F1 scores for every classifier on each corpus. The best results are obtained by logistic regression and SVM. Classification of cade is the most difficult task, the best classifier has only 55% of F1 measure.

Table IV shows overall micro–averaged accuracy in 5-fold cross-validation scheme for each corpus and each classifier compared to another system.

Table VII presents execution time of training and testing. SVM is the most time-consuming phase in training in comparison to classification. However, it can reduce time of testing. Time of testing using KNNs is higher than other classifiers, because it is proportional to number of documents. Time of precision reduction is negligible.

VII. Conclusions and Future Work

The conducted experiments show that it is beneficial to perform precision reduction on the term–document representations. However, it is unclear what number of bits gives the best results for the specific corpus. For some corpora, a precision reduction to 1 bit is possible without loss of accuracy. On the other hand it is safe to reduce the number of bits from 64 to 4, which usually improves the quality of the obtained results and never leads to their degradation. As such, precision reduction seems the be very promising result, especially combined with FPGA implementation, which should lead to significant computation speed-up and memory footprint reduction.

The precision reduction is also a good alternative to di-
dimensionality reduction by SVD. It can lead to better accuracy and does not required time-consuming matrix manipulation. This feature is specially important for scenarios with very large vocabularies and document data sets. If SVD is still considered, the precision reduction should be applied before SVD, not in opposite order.

It should be also observed that focusing on micro–averaged objective allows for stronger reduction than in macro-averaged measures.

Nowadays neural networks are one of the most popular machine learning tools used to solve NLP problems. Our further research will be focused on testing precision reduction on distributional representations, which are typically used as inputs to neural networks. It is not uncommon that neural networks have millions of parameters – the reduction of precision of the vector weights is an interesting direction of research, which will be pursued in our future work.

REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798–1828, aug 2013.
[2] J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85–117, 2015.
[3] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, “Ask me anything: Dynamic memory networks for natural language processing,” 2015.
[4] M. Karwatowski, P. Russe, M. Wielgosz, S. Koryciak, and K. Wiatr, “Energy efficient calculations of text similarity measure on FPGA-accelerated computing platforms,” Lecture Notes in Computer Science, pp. 31 – 40, 2015.
[5] C. D. Manning and H. Schütze, Foundations of Statistical Natural Language Processing. Cambridge, MA, USA: MIT Press, 1999.
[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” J. Mach. Learn. Res., vol. 12, pp. 2493–2537, Nov. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2078186
[7] N. FitzGerald, O. Täckström, K. Ganchev, and D. Das, “Semantic role labeling with neural network factors,” in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisboa, Portugal: Association for Computational Linguistics, September 2015.
[8] K. M. Hermann, D. Das, J. Weston, and K. Ganchev, “Semantic frame
TABLE V
TIMES OF TRAINING AND TESTING FOR THE CLASSIFIERS ON THE CORPORAS WITHOUT OR WITH SVD WITH DIFFERENT NUMBER OF COMPONENTS.

| Classifier | Variant | Training (seconds) | Testing (seconds) |
|------------|---------|--------------------|------------------|
|            |         | webkb  | r8   | r52  | ng20 | cade | webkb  | r8   | r52  | ng20 | cade |
| KNN 1      | no SVD  | 1.44   | 0.44 | 0.48 | 7.22 | 24.86 | 0.30 | 0.30 | 0.66 | 11.78 |
|            | SVD(100)| 10.48  | 10.32| 10.23| 17.23| 55.77 | 0.54 | 0.38 | 0.43 | 1.54 | 9.05 |
|            | SVD(500)| 51.12  | 55.34| 55.38| 78.15| 204.10| 0.57 | 0.48 | 0.60 | 2.23 | 12.96 |
|            | SVD(1000)| 94.20  | 101.02| 103.53| 146.47| 382.13| 0.67 | 0.68 | 0.88 | 3.17 | 18.45 |
| KNN 5      | no SVD  | 1.44   | 0.44 | 0.48 | 7.22 | 24.86 | 0.52 | 0.57 | 0.76 | 3.99 | 13.27 |
|            | SVD(100)| 10.48  | 10.32| 10.23| 17.23| 55.77 | 0.59 | 0.46 | 0.54 | 1.97 | 10.83 |
|            | SVD(500)| 51.13  | 55.34| 55.38| 78.15| 204.11| 0.65 | 0.55 | 0.60 | 2.65 | 14.80 |
|            | SVD(1000)| 94.22  | 101.02| 103.53| 146.47| 382.13| 0.74 | 0.74 | 0.88 | 3.54 | 20.32 |
| Logistic Regression | no SVD  | 1.56   | 0.79 | 3.03 | 11.14| 45.39 | 0.35 | 0.11 | 0.12 | 0.79 | 5.39 |
|            | SVD(100)| 10.57  | 10.73| 13.18| 22.41| 64.44 | 0.40 | 0.14 | 0.15 | 0.83 | 5.50 |
|            | SVD(500)| 51.69  | 57.76| 72.39| 95.28| 237.10| 0.39 | 0.16 | 0.18 | 1.00 | 5.90 |
|            | SVD(1000)| 95.31  | 105.60| 135.25| 177.72| 450.09| 0.43 | 0.20 | 0.24 | 1.21 | 6.40 |
| SVM        | no SVD  | 1.64   | 0.72 | 1.38 | 6.78 | 38.90 | 0.35 | 0.11 | 0.12 | 0.79 | 5.39 |
|            | SVD(100)| 10.58  | 10.49| 11.30| 18.61| 60.38 | 0.40 | 0.14 | 0.15 | 0.83 | 5.50 |
|            | SVD(500)| 51.66  | 56.33| 59.63| 84.52| 227.25| 0.39 | 0.16 | 0.18 | 1.00 | 5.90 |
|            | SVD(1000)| 95.37  | 103.28| 112.41| 159.62| 437.49| 0.43 | 0.20 | 0.24 | 1.21 | 6.40 |

Fig. 3. Average and standard deviation of error reduction among the corpora for the classifiers in function of precision bits.

Fig. 4. Average and standard deviation of error reduction among the corpora for the classifiers in function of precision bits for the variant with a precision reduction after SVD.

Fig. 5. F1 of the classifiers on the corpora with only TF-IDF, TF-IDF with the best precision reduction and TF-IDF with the best SVD.

identification with distributed word representations,” in Proceedings of ACL. Association for Computational Linguistics, June 2014.

[9] S. Petrov, D. Das, and R. McDonald, “A universal part-of-speech tagset,” in Proc. of LREC, 2012.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation
Fig. 6. F1 of the classifiers on the corpora with TF-IDF with the best precision reduction and SVD compared to TF-IDF with SVD.

of word representations in vector space.” CoRR, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[11] T. Mikolov, W. Yih, and G. Zweig, “Linguistic regularities in continuous space word representations,” in Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013. Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA, 2013, pp. 746–751. [Online]. Available: [http://aclweb.org/anthology/N/N13/N13-1090.pdf)

[12] J. Hawkins and S. Blakeslee, On Intelligence. Times Books, 2004. [Online]. Available: http://www.worldcat.org/isbn/0805074562

[13] V. Mountcastle, “The columnar organization of the neocortex,” Brain, vol. 120, no. 4, pp. 701–722, apr 1997. [Online]. Available: [http://www.brain.oupjournals.org/cgi/doi/10.1093/brain/120.4.701

[14] J. Hawkins and D. George, “Hierarchical temporal memory: Concepts, theory and terminology,” Numenta, Tech. Rep., 2006.

[15] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, vol. 41, no. 6, pp. 391–407, 1990.

[16] E. Bingham and H. Mannila, “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, ser. KDD ’01. New York, NY, USA: ACM, 2001, pp. 245–250. [Online]. Available: [http://doi.acm.org/10.1145/502512.502546

[17] M. Wielgosz, M. Panggabean, and L. A. Rønningen, “Fpga architecture for kriging image interpolation,” International Journal of Advanced Computer Science and Applications (IJACSA), vol. 4, no. 12, pp. 193–201, 2013.

[18] M. Wielgosz, M. Panggabean, J. Wang, and L. A. Rønningen, “An FPGA-based platform for a network architecture with delay guarantee,” Journal of Circuits, Systems and Computers, vol. 22, no. 6, pp. 1350045–1—1350045–20, 2013.

[19] M. Wielgosz, E. Janro, D. Zurek, and K. Wiatr, FPGA Implementation of the Selected Parts of the Fast Image Segmentation. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 203–216.

[20] M. Karwatowski, M. Wielgosz, M. Pietroń, and K. Wiatr, “Comparison of semantic vectors with reduced precision using the cosine similarity measure.” 2017.

[21] G. Salton, A. Wong, and C.-S. Yang, “A vector space model for automatic indexing,” Communications of the ACM, vol. 18, no. 11, pp. 613–620, 1975.

[22] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gafford et al., “Okapi at trec-3,” Nist Special Publication Sp, vol. 109, p. 109, 1995.

[23] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” Journal of the American society for information science, vol. 41, no. 6, p. 391, 1990.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vander-