Evaluating Memory Efficiency and Robustness of Word Embeddings

Johannes Jurgovsky, Michael Granitzer, Christin Seifert

Evaluating Memory Efficiency and Robustness of Word Embeddings

Johannes Jurgovsky, Michael Granitzer, and Christin Seifert

Media Computer Science, Universität Passau, Germany
{Johannes.Jurgovsky,Michael.Granitzer,Christin.Seifert}@uni-passau.de
http://mics.fim.uni-passau.de

Abstract. Skip-Gram word embeddings, estimated from large text corpora, have been shown to improve many NLP tasks through their high-quality features. However, little is known about their robustness against parameter perturbations and about their efficiency in preserving word similarities under memory constraints. In this paper, we investigate three post-processing methods for word embeddings to study their robustness and memory efficiency. We employ a dimensionality-based, a parameter-based and a resolution-based method to obtain parameter-reduced embeddings and we provide a concept that connects the three approaches. We contrast these methods with the relative accuracy loss on six intrinsic evaluation tasks and compare them with regard to the memory efficiency of the reduced embeddings. The evaluation shows that low Bit-resolution embeddings offer great potential for memory savings by alleviating the risk of accuracy loss. The results indicate that post-processed word embeddings could also enhance applications on resource limited devices with valuable word features.

Keywords: natural language processing, word embedding, memory efficiency, robustness, evaluation

1 Introduction

Word embeddings, also referred to as "word vectors" [7], capture syntactic and semantic properties of words solely from raw natural text corpora without human intervention or language dependent preprocessing. In natural language texts, the co-occurrence of words to appear together in the same context depends on the syntactic form and meaning of the individual words. In word embeddings the various nuances of word-to-word relations are distributed across several dimensions in vector space. These vector spaces are high-dimensional to provide enough degrees of freedom for hundreds of thousands of words to allow the relative arrangement of their embeddings reflect as many pairwise relations as possible out of the corpus statistics. However, embeddings carry a lot of information about words, which is hard to understand, interpret and quantify or may even be redundant and non-informative.

The NLP community has been successfully exploiting these embeddings over the last years, e.g. [3, 6]. However, the gain in task-accuracy brings the downside that high-dimensional continuous valued word vectors require a large amount of memory. Moreover, embeddings are trained by a fixed-size network architecture that sweeps through a huge text corpus. Consequently, the total number of parameters in the embedding matrix is implicitly defined a-priori. Further, there is no natural transition to more memory efficient embeddings, by which one could trade accuracy for memory. This is particularly limiting for NLP-applications on resource limited devices where memory is still a scarce resource. An embedding matrix with 150,000 vocabulary words can easily require 60-180 Megabytes of memory, which is rather inconvenient to be transferred to and stored in a browser or mobile application. This restriction gives rise to contemplate different types of post-processing methods in order to derive robust and memory-efficient word vectors from a trained embedding matrix.

In this paper, we investigate three post-processing methods for word vectors trained with the Skip-Gram algorithm that is implemented in the WORD2VEC software toolkit[1][7]. The employed post-processing methods are (i) dimensionality reduction (PCA), (ii) parameter reduction (Pruning) and (iii) Bit-resolution reduction (Truncation). To isolate the effects on embeddings with different sizes, sparsity levels and resolutions, we employ intrinsic evaluation tasks based on word relatedness and abstain from extrinsic classification tasks. Our work makes the following contributions:

- We show through evaluation that Skip-Gram word embeddings are robust against linear dimensionality reduction, pruning and resolution-reduction on all tasks with only moderate loss of $< 10\%$ at a reduction of 40%.
- Our experiments reveal that higher-dimensional embeddings capture a larger fraction of redundant information, which can be exploited in favor of memory savings.
- We propose the resolution-based post-processing method as a means to gradually trade word vector quality for memory.

The remainder of this paper is structured as follows: First we present related work from the domain of language modeling and word representations. Next, we provide a conceptional overview of the post-processing methods used to reduce the amount of parameters in word embeddings. Then, we describe the experimental setup and the results in detail. A final discussion highlights the benefits of the different findings for practical applications.

2 Related Work

In computational linguistics, generating count-based language models has been an active research area since decades. The most common approach involves three parts: Collecting co-occurrence statistics of words from large text corpora, transforming (e.g. tf-idf, Point-wise Mutual Information (PMI)) the counts to derive word association scores and finally applying a dimensionality reduction

[1] `https://code.google.com/p/word2vec/`

method (e.g. PCA, SVD). Dimensionality reduction is used for both smoothing sparsity and reducing the overall amount of parameters in order to obtain a low-dimensional and dense embedding matrix [1]. In this kind of approach, the quality of word vectors depends on the choice of methods used. In contrast, advances in recent years gave rise to new techniques [2–5], that implicitly model word co-occurrences by predicting context words from observed input words. Instead of first collecting co-occurrences of context words and then re-weighting these values with tf-idf or PMI, *predictive models* directly set the word vectors to optimally predict the contexts in which the corresponding words tend to appear. Since similar words appear in similar contexts, the classifier in a predictive model is trained to assign similar vectors to similar words. In an extensive evaluation, Baroni et al. [6] ascertain that embeddings of predictive models are superior to their count-based counterparts on word similarity tasks.

One particularly efficient representative of the family of predictive models is the Skip-Gram method, proposed by Mikolov et al. [7]. It offers the convenient property that the output of the model is a linear function of an input word vector and a context word vector, which not only results in meaningful nearest neighbours but also in informative relative positions of pairs of word embeddings. The intriguing finding is that arithmetic operations on word vectors in embedding space accurately reflect semantic and syntactic operations on words. We chose to use these embeddings in our experiments, since they encode a variety of language related information in both local and global neighborhoods. A thorough explanation of the rationale behind this technique was given by Levy and Goldberg [9, 10].

Evaluations of word embeddings are published whenever new embedding methods are proposed. Besides manually inspecting 2D-projections of word vectors (e.g. t-SNE [11], PCA), it is difficult to associate meaning to individual dimensions. In language modeling, authors have traditionally employed perplexity to evaluate their models. In recent years, the common approach shifted towards testing the embeddings on various word similarity or word analogy test datasets [6, 8]. In this domain, the work of Chen et al. [12] is the closest one to ours. Therein, they include a short section about information reduction capabilities of embeddings with limited experiments on other types of embeddings. We were particularly interested in preserving the linear structure in Word2Vec-embeddings under limited memory conditions. So far, we are not aware of other experiments that explore ways to reduce word vectors in terms of memory.

3 Methodology

The word embeddings we use in our experiments, are obtained from Mikolov's Skip-Gram algorithm [7]. As a recent study [9] revealed, the algorithm factorizes an implicit word-context matrix, whose entries are the Point-wise Mutual Information of word-context pairs shifted by a constant offset. This PMI-matrix $M \in \mathbb{R}^{|V| \times |V|}$ is factorized into a word embedding matrix $W \in \mathbb{R}^{|V| \times d}$ and a context matrix $C \in \mathbb{R}^{d \times |V|}$, where $|V|$ is the number of words in the vocabulary and d is the number of dimensions of each word vector. The context matrix is only required during training and usually discarded afterwards. The result of optimizing the Skip-Gram's objective is that word vectors (rows in W) have high similarity with respect to their cosine-similarity in case the words are syntactically or semantically similar. Besides that, the word vectors are dense and have significantly fewer dimensions than there are context words - columns in M. With sufficiently large d, the PMI-matrix could be perfectly reconstructed from its factors W and C, and thus provide the most accurate information about word co-occurrences in a corpus [6]. However, increasing the dimensionality d of word vectors also increases the amount of memory required to store the embedding matrix W. When using word embeddings in an application, we do not aim for a perfect reconstruction of the PMI-matrix but for reasonably accurate word vectors that reflect word similarities and word relations of language. Therefore, a more memory-efficient, yet accurate version of W would be desirable.

3.1 Memory Reduction with Post-Processing

More formally, we want to have a mapping τ from the full embedding matrix W to $\hat{W} = \tau(W)$, where $\hat{W}$ can be stored more efficiently while at the same time its word vectors are similarly accurate as the original vectors in W. For the vectors in $\hat{W}$ to have an accuracy loss as low as possible, word vectors in W must be robust against the mapping function τ. We consider W *robust* against the transformation τ, if the loss of $\tau(W)$ is small compared to W across very different evaluation tasks. A memory reduction through τ can be induced by reducing the number of dimensions, the amount of effective parameters or the parameters' Bit-resolution. Accordingly, we employed three orthogonal post-processing methods that can be categorized into *dimensionality-based*, *parameter-based* and *resolution-based* approaches:

- *Dimensionality-based:* Methods in this category can be described by the mapping $\tau_{dim} : \mathbb{R}^{|V| \times d} \rightarrow \mathbb{R}^{|V| \times \hat{d}}$, where $\hat{d} < d$. Fewer dimensions directly relate to less required memory. The dimensionality-based approaches provide a transformation that projects embeddings onto a lower-dimensional subspace while preserving the dominant properties of words. Both linear (e.g. PCA) and non-linear dimensionality (e.g. multilayer Autoencoder) reduction methods are applicable, as long as the inverse τ^{-1} of the transformation is available. In both variants, the computational overhead for computing τ and the memory overhead for storing the inverse τ^{-1} have to be considered. For linear transformations there is no memory overhead since the transformation can once be applied to the embedding matrix and subsequent methods can use the transformed embeddings alike. If there is reason to assume that the word vectors lie on a nonlinear manifold, nonlinear dimensionality reduction techniques can find a mapping to the components of the potentially low-dimensional manifold. Both the computational overhead for estimating the nonlinear components and the memory overhead for storing the inverse of the mapping alongside with the transformed embeddings is high.

- *Parameter-based:* Whereas dimensionality-based methods change the bases of the embedding space, parameter-based methods leave the structure untouched but directly modify individual parameter values in the embedding matrix: $\tau_{par} : \mathbb{R}^{|V| \times d} \rightarrow \mathbb{R}^{|V| \times d}$, where τ_{par} is supposed to map most values to zero and leave only few non-zero elements in the output matrix. For instance, a simple pruning strategy can be used. Then, the output of τ_{par} is a sparse matrix that can be stored more efficiently.
- *Resolution-based:* $\tau_{res} : \mathbb{R}^{|V| \times d} \rightarrow \{0, \cdots, r-1\}^{|V| \times d}$, where $r \in \mathbb{N}^{+}$ is the resolution of the coordinate axes. With discrete coordinates, values can be stored at lower Bit-precision. Resolution-based methods discretize the coordinate axes into distinct intervals and thus reduce the resolution of the word vectors. For instance, the Bit-Truncation method subdivides the embedding space into regions of equal size.

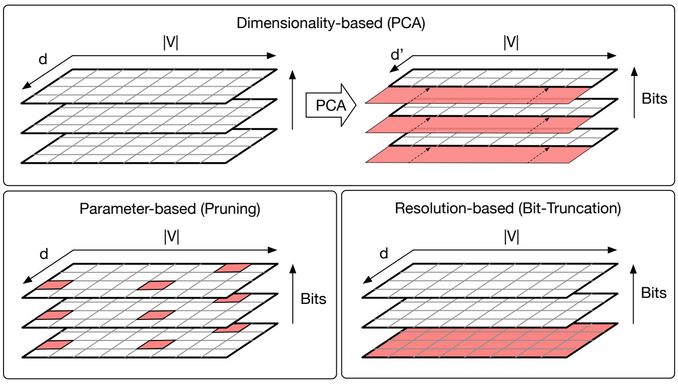

Dimensionality-based (PCA)
d
|V|
PCA
d'
|V|
Bits
Parameter-based (Pruning)
d
|V|
Bits
Resolution-based (Bit-Truncation)
d
|V|
Bits

Fig. 1: Three methods for post-processing a word embedding matrix: PCA-Reduction (*top*), Pruning across all Bit-planes (*left*) and Truncation of the least significant Bits (*right*).

In this work, we select one method of each category. In particular, we explore the robustness and memory efficiency of embeddings after applying *PCA*-reduction, *Pruning* and *Bit-Truncation.* In the following section we describe the selected methods along with the rationale for the selection (see Figure 1).

3.2 Post-Processing Methods

We employ the following post-processing methods:

Linear Transformation Dimensionality-based approaches assume that points are not uniformly scattered across the embedding space but exhibit certain directions of dominant variations. If there is some kind of structure in the data, it should be possible to exploit it by means of representing the same data with fewer dimensions. If the discarded dimensions only accounted for redundant information, we would obtain basis vectors that describe the word embeddings equally well but with fewer parameters. Since our evaluation tasks rely on vector arithmetic and cosine similarities, we do not use nonlinear dimensionality reduction methods as these operations would be meaningless on the transformed embeddings $\hat{L}$ produced by a nonlinear mapping. Therefore, we used the PCA-solution as a linear transformation to obtain lower dimensional embeddings.

Pruning With *Pruning* we refer to a parameter-based method that discards a subset of the values in the embedding matrix by setting them to zero. With $\lambda \in [0, 1]$ we denote the *Pruning level.* Our naive pruning strategy is agnostic to word vectors since it determines a global threshold value p_λ from the whole matrix in such a way that $\lambda * 100\%$ of the matrix's values are greater than the threshold. All values w_{ij} below that threshold $|w_{ij}| < |p_\lambda|$ are set to zero. As a result of the pruning operation, we obtain a sparser embedding matrix with a degree of sparsity equivalent to $1-\lambda$. Sparse matrices can be compressed more easily and thus require less memory than dense matrices. The rationale for using Pruning as reduction strategy arose from the observation that on normalized word vectors, pruning gradually projects points onto their closest coordinate axis. As we increase the pruning level, more points have coordinates that are aligned with the coordinate axes. Due to the normalization, this alignment gradually affects some but not all dimensions of individual word vectors. We hypothesize that up to a certain pruning level, the inaccuracy induced by Pruning has no qualitative effect on word vector arithmetic and word similarity computations.

Bit-Truncation Besides a plain reduction of parameters by means of projection on fewer principle components, we explored a rather memory-focused approach that leaves the embedding dimensions untouched but migrates continuous word embeddings to discrete ones. The motivation is that in distributed embeddings the factors on all dimensions partially contribute to the meaning of a word. Thus, there should exist some degree of contribution which makes the meaning shift from one notion to another whereas for smaller contributions, the meaning is unaffected. We can exploit this relationship between the proximity of the embeddings' values and their similarities in meaning for purposes of memory efficiency by imposing resolution constraints on the value range along each coordinate. The Skip-Gram algorithm is defined on continuous valued word vectors which assumes each dimension to be real-valued. Figuratively, continuous embeddings allow for arbitrary positioning of a word's embedding in embedding space up to the precision of the datatype used. With *Bit-Truncation* we rasterize the embedding space uniformly by subdividing the range of values on each coordinate axis into distinct groups. Thus, all the factors of a distributed embedding still contribute to the meaning but only up to some pre-defined precision.

For the Bit-Truncation method, we adopt the approach described in [12] with slight adaptions. To reduce the resolution of the real numbers that make up the embedding matrix, first we shift the values to the positive range. Then we re-scale the values to the interval $[0, 1]$ and multiply them by 2^B, where B is the number of Bits we want to retain. Finally we cast the values to a 32-Bit Integer datatype. After casting to Integer, each coordinate axis has a resolution of $r = 2^B$ non-overlapping equally-spaced intervals. Consequently, the number of distinguishable regions in embedding space $R_\top = r^d$ is exponential in the number of dimensions d.

4 Experimental Setup

In all experiments we used word vectors estimated with the Skip-Gram method of the WORD2VEC-toolkit from a text corpus containing one billion words. The corpus was collected from the latest snapshot of English Wikipedia articles[2]. After removing words that appeared less than 100 times, the vocabulary contained 148,958 words, both uppercase and lowercase. We used a symmetric window covering $k = 9$ context words and chose the negative-sampling approximation to estimate the error from $neg = 20$ noise words. With this setup, we computed word vectors of several sizes in the range $d \in [50,\ 100,\ 150,\ 300,\ 500]$. After training, all vectors are normalized to unit length. To evaluate the robustness and efficiency of word vectors after applying post-processing, we compare PCA-reduction, Pruning and Bit-Truncation on three types of intrinsic evaluation tasks: word relatedness, word analogy and linguistic properties of words. In each of these tasks, we use two different datasets.

Word Relatedness: The *WordSim353*(WS353)[13] and *MEN*[14] datasets are used to evaluate pairwise word relatedness. Both consist of pairs of English words, each of which has been assigned a relatedness score by human evaluators. The *WordSim353* dataset contains 353 word pairs with scores averaged over judgments of at least 13 subjects. For the *MEN* dataset, a single annotator ranked each of the 3000 word-pairs relative to each of 50 randomly sampled word-pairs. The evaluation metric is the correlation (Spearman's ρ) between the human ratings and the cosine-similarities of word vectors.

Word Analogy: The word analogy task is more sensitive to changes of the global structure in embedding space. It is formulated as a list of questions of the form "a is to $\hat{a}$ as b is to $\hat{b}$", where $\hat{b}$ is hidden and has to be guessed from the vocabulary. The dataset we use here was proposed by Mikolov et al. [8] and consists of 19544 questions of this kind. About half of them are morpho-syntactical (wa-syn) (*loud* is to *louder* as *tall* is to *taller*) and the other half semantic (wa-sem) questions (*Cairo* is to *Egypt* as *Stockholm* is to *Sweden*). It is assumed that the answer to a question can be retrieved by exploiting the relationship $a \to \hat{a}$ and applying it to b. Since WORD2VEC-embeddings exhibit a linear structure in embedding space, word relations are consistently reflected in sums and differences of their vectors. Thus, the answer to an analogy question is given by the target word w_t whose embedding $\boldsymbol{w}_t$ is closest to $\boldsymbol{w}_q = \hat{\boldsymbol{a}} - \boldsymbol{a} + \boldsymbol{b}$

[2] https://dumps.wikimedia.org/enwiki/20150112/

with respect to the cosine-similarity. The evaluation metric is the percentage of questions that have been answered with the expected word.

Linguistic Properties: Schnabel et al. [15] showed that results from intrinsic evaluations are not always consistent with results on extrinsic evaluations. Therefore, we include the recently proposed QVEC-evaluation[3][16] as additional task. This evaluation uses two dictionaries of words, annotated with linguistic properties: a syntactic (QVEC-syn) dictionary (e.g., `ptb.nns`, `ptb.dt`) and a semantic (QVEC-sem) dictionary (e.g., `verb.motion`, `noun.position`). The proposed evaluation method assigns to each embedding dimension the linguistic property that has highest correlation across all mutual words. The authors showed that the sum over all correlation values can be used as an evaluation measure for word embeddings. Moreover, they showed that this score has high correlation with the accuracy the same embeddings achieve on real-world classification tasks.

5 Results

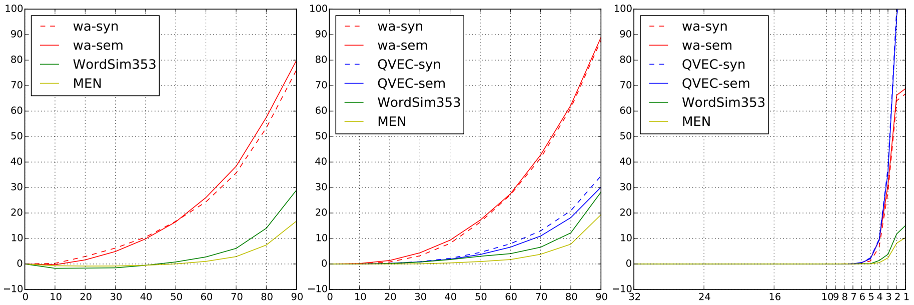
Since we evaluate the robustness of word embeddings against post-processing, we report the *relative loss* induced by applying a post-processing method. The loss is measured as the difference between the score of original embeddings and the score of post-processed embeddings. In case of the word relatedness task, the score is the Spearman correlation. On the word analogy task, the score is given as accuracy. And on the linguistic properties task, the score is the output of the QVEC evaluation method. We divide the loss by the score of the original embeddings to obtain a relative loss that is comparable across tasks.

5.1 Robustness of Word Vectors

In Figure 2 we report the mean relative loss, averaged over the five word vector sizes on all datasets. The percentage of reduction refers to the fraction of principle components with lowest eigenvalues that were discarded after applying PCA and to the fraction of parameters that were set to zero after pruning, respectively.

The word embeddings show a similar trend for all three post-processing methods. A small relative reduction results in a small loss, whereas a large reduction leads to a large loss. For all methods, the loss increases exponentially with the percentage of reduction. On the word analogy datasets, the loss is consistently higher than on the word relatedness and QVEC datasets. In particular, the relative loss on word relatedness datasets is predominantly unaffected (relative loss $< 10\%$) by post-processing up to a reduction threshold of 40%. Compared to the naive Pruning approach, PCA-transformed embeddings suffer lower loss on all tasks. Actually, on WordSim353 and MEN, PCA-reduced embeddings exhibit slight negative loss ($< 3\%$). Bit-Truncation produces no loss on any dataset until the Bit-resolution of the parameters is lower than 8-Bit. To summarize,

[3] `https://github.com/ytsvetko/qvec`



(a) PCA (b) Pruning (c) Bit-Truncation

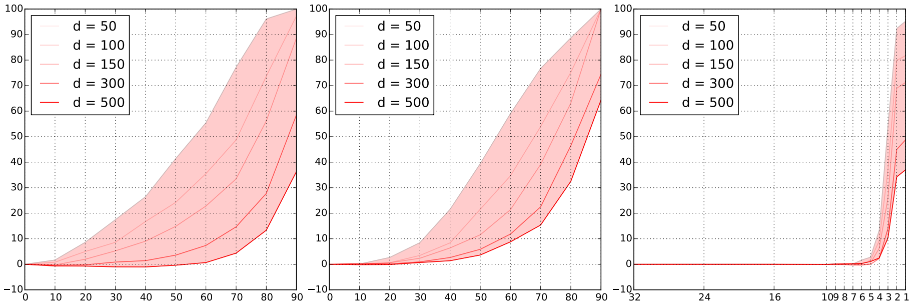
Fig. 2: Mean relative loss of embeddings after (a) PCA: percentage of removed dimensions, (b) Pruning: percentage of removed parameters and (c) Bit-Truncation: remaining Bits. Scores on the QVEC datasets are not shown for PCA since they are not comparable across different word vector sizes.

the Skip-Gram word embeddings are most robust against post-processing with resolution-based Bit-Truncation and the dimensionality-based PCA-reduction.

5.2 Memory Efficiency

The percentage of reduction achieved by PCA is directly proportional to memory savings induced by the smaller number of dimensions. There, the sweet spot is task-dependent and the relative reduction can be rather high before word vector quality suffers a loss. In contrast, the number of pruned values is not directly proportional to memory savings, since the coding of sparse matrices requires additional memory. For instance, the row compressed storage method [17] has, without further assumptions about the shape of the matrix, a memory complexity of $\mathcal{O}(3k)$, where k is the number of non-zero elements in the sparse matrix. Thus, the pruning method would only start to pay off in terms of memory consumption above a pruning level of $\frac{2}{3}$, which would result in serious quality-loss. Finally, post-processing the embedding matrix with Bit-Truncation does not cause any loss on any of the evaluated datasets up to 75% reduction (24Bit). For resolutions below $r = 2^8$, all evaluated datasets respond to the low-precision embeddings with abruptly increasing loss.
Figure 3 shows that higher-dimensional embeddings ($d = 500$) can be reduced more aggressively than lower-dimensional ones before reaching the same level of relative loss. Since a similar behavior holds on all tasks (not shown due to space constraints), the observation is two-fold: First, it suggests that, the higher-dimensional the embedding space is, the more non-zero parameters there are and the higher their resolution is, the more redundant is the information that is captured in the embeddings. Secondly, the consistency across dimensionality-based and parameter-based methods indicates that neither the number of dimensions

or parameters nor the continuous values alone but the number of distinguishable regions in embedding space is crucial for accurate word embeddings.



(a) PCA (b) Pruning (c) Truncation

Fig. 3: Relative loss of embeddings on the syntactic word analogy dataset (wa-syn) after PCA (a), Pruning (b) and Bit-Truncation (c).

With a sufficiently large Bit-resolution the accuracy of all embedding sizes approximates the same accuracy level as with continous values. Thus, we can confirm the finding in [12] also for Skip-Gram embeddings: The same accuracy can be achieved with discretized values at sufficiently large resolutions. Additionally, we state that this observation not only holds for cosine-similarity on word relatedness tasks but also for vector arithmetic on the word analogy task and for QVEC on the linguistic properties task.
To summarize, Skip-Gram word embeddings can be stored more efficiently using a post-processing method that reduces the number of distinguishable regions $R_{\top} = (2^B)^d$ in embedding space. Pruning does so by producing increasingly large zero-valued regions around each coordinate axis ($2^B - const.$). PCA does so by mapping the word vectors into an embedding space with fewer dimensions $\hat{d} < d$. And Bit-Truncation directly lowers the resolution of each coordinate by constraining the Bit-resolution $\hat{B} < B$.

6 Discussion

Memory efficiency: If an application can take a loss in word vector accuracy in favor of memory or transmission times, Skip-Gram embeddings can be reduced with all three evaluated methods. Thereby, pruning is the least efficient method as the overhead introduced by sparse coding could only be compensated by pruning levels above $\frac{2}{3}$. Such an aggressive pruning strategy would result in an average accuracy loss of more than 30%. In contrast, the linear dimensionality reduction technique worked well on our tasks and it allows for a consistent transition from higher to lower dimensional embeddings. The accuracy of PCA-reduced embeddings even improves over equivalently large original non-reduced embeddings on word relatedness and word analogy tasks (see evaluation data online[4]). Thus, in terms of memory, it can be more efficient to first train high-dimensional embeddings and afterwards reduce them with PCA to the desired size. The resolution-based approach provides the greatest potential for memory savings. With only 8-Bit precision per value, there is no loss on any of the tasks. A straight-forward implementation can thus fit the whole embedding matrix in only 25% of memory.

Resolution and Semantic Transition: Another observation is depicted in Table 1. On the word analogy dataset, the transition from lower to higher-precision values not only yields increasingly better average accuracy but also corresponds to a semantic transition from lower to higher relatedness. Even if the embeddings' values have only 3-Bit precision, the retrieved answer words are not totally wrong but still in some kind related to the expected answer word. It seems that some notions of meaning are encoded on a finer scale in embedding space and that these require more Bits to remain distinguishable.

Table 1: Answer words for several country-currency analogy questions from word embeddings at different resolutions. Finally, all answer words are currencies.

Questions	**Expected**	**3-Bit**	**4-Bit**	**5-Bit**	**6-Bit**	**7-Bit**
Europe euro : Japan ______?	yen	Nagasaki	Taiwan	yen	yen	yen
Europe euro : Korea ______?	won	Kim	PRC	PRC	PRC	dollar
Europe euro : USA ______?	dollar	Dusty	proposal	US	euros	dollar
Europe euro : Brazil ______?	real	Alegre	proposal	dollar	euros	euros
Europe euro : Canada ______?	dollar	Calgary	Calgary	dollar	dollar	dollar

For coarse resolutions (3-Bit) the regions in embedding space are too large to allow an identification of a country's currency. Because there are many words within the same distance to the target location, the most frequent one is retrieved as answer to the question. As the resolution increases, regions get smaller and thus more nuanced distances between word embeddings emerge, which yields not only increasingly accurate but also progressively more related answers.

7 Conclusion

In this paper, we explored three methods to post-process Skip-Gram word embeddings in order to identify means to reduce the amount of memory required to store the embedding matrix. Therefore, we evaluated the robustness of embeddings against a dimensionality-based (PCA), parameter-based (Pruning) and a resolution-based (Bit-Truncation) approach. The results indicate, that embeddings are most robust against Bit-Truncation and PCA-reduction and that

[4] http://tinyurl.com/jj-ecir2016-eval

preserving the number of distinguishable regions in embedding space is key for obtaining memory efficient (75% reduction) and accurate word vectors. Especially resource limited devices can benefit from these compact high-quality word features to improve NLP-tasks under memory constraints.

Acknowledgments. The presented work was developed within the EEXCESS project funded by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement number 600601.

References

1. Turney, P. D., Pantel, P.: From frequency to meaning: Vector space models of semantics. Journal of artificial intelligence research, 37(1), 141–188 (2010)
2. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. The Journal of Machine Learning Research, 3, 1137–1155 (2003)
3. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Jour. of Mac. Lea. Res., 12, 2493–2537 (2011)
4. Mnih, A., Hinton, G.: Three new graphical models for statistical language modelling. In: ICML, pp. 641–648, ACM, June (2007)
5. Huang, E. H., Socher, R., Manning, C. D., Ng, A. Y.: Improving word representations via global context and multiple word prototypes. In: 50th Annual Meeting of the Association for Computational Linguistics, pp. 873–882, ACL, July (2012)
6. Baroni, M., Dinu, G., Kruszewski, G.: Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: 52nd Annual Meeting of the Association for Computational Linguistics, pp. 238–247 (2014)
7. Mikolov T., Chen K., Corrado G., Dean J.: Efficient estimation of word representations in vector space. arXiv.org, 2013
8. Mikolov, T., Yih, W. T., Zweig, G.: Linguistic Regularities in Continuous Space Word Representations. In: HLT-NAACL, pp. 746–751, June (2013)
9. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in Neural Information Processing Systems, pp. 2177–2185 (2014)
10. Goldberg, Y., Levy, O.: word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722., (2014)
11. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research, 9(2579-2605), 85, (2008)
12. Chen, Y., Perozzi, B., Al-Rfou, R., Skiena, S.: The expressive power of word embeddings. ICML, Deep Learning for Audio, Speech and Lang. Proc. Workshop (2013)
13. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. In: 10th International Conference on World Wide Web, pp. 406–414, ACM, April (2001)
14. Bruni, E., Tran, N. K., Baroni, M.: Multimodal Distributional Semantics. J. Artif. Intell. Res.(JAIR), 49, pp. 1–47, (2014)
15. Schnabel, T., Labutov, I., Mimno, D., Joachims, T.: Evaluation methods for unsupervised word embeddings. In: Emp. Met. in Nat. Lang. Proc., (2015)
16. Ling, Y. T. M. F. W., Dyer, G. L. C.: Evaluation of Word Vector Representations by Subspace Alignment. In: Emp. Met. in Nat. Lang. Proc., pp. 2049–2054, ACL, Lisbon (2015)
17. Saad, Y.: Iterative methods for sparse linear systems. Siam (2003)