Binary Paragraph Vectors

Karol Grzegorczyk and **Marcin Kurdziel**
AGH University of Science and Technology
Department of Computer Science
Krakow, Poland
{kgr,kurdziel}@agh.edu.pl

Abstract

Recently Le & Mikolov described two log-linear models, called Paragraph Vector, that can be used to learn state-of-the-art distributed representations of documents. Inspired by this work, we present *Binary Paragraph Vector* models: simple neural networks that learn short binary codes for fast information retrieval. We show that binary paragraph vectors outperform autoencoder-based binary codes, despite using fewer bits. We also evaluate their precision in transfer learning settings, where binary codes are inferred for documents unrelated to the training corpus. Results from these experiments indicate that binary paragraph vectors can capture semantics relevant for various domain-specific documents. Finally, we present a model that simultaneously learns short binary codes and longer, real-valued representations. This model can be used to rapidly retrieve a short list of highly relevant documents from a large document collection.

1 Introduction

One of the significant challenges in contemporary information processing is the sheer volume of available data. Gantz and Reinsel (2012), for example, claim that the amount of digital data in the world doubles every two years. This trend underpins efforts to develop algorithms that can efficiently search for relevant information in huge datasets. One class of such algorithms, represented by, e.g., Locality Sensitive Hashing (Indyk and Motwani, 1998), relies on hashing data into short, locality-preserving binary codes (Wang et al., 2014). The codes can then be used to group the data into buckets, thereby enabling sublinear search for relevant information, or for fast comparison of data items. Most of the algorithms from this family are data-oblivious, i.e. can generate hashes for any type of data. Nevertheless, some methods target specific kind of input data, like text or image.

In this work we focus on learning binary codes for text documents. An important work in this direction has been presented by Salakhutdinov and Hinton (2009). Their *semantic hashing* leverages autoencoders with sigmoid bottleneck layer to learn binary codes from a word-count bag-of-words (BOW) representation. Salakhutdinov & Hinton report that binary codes allow for up to 20-fold improvement in document ranking speed, compared to real-valued representation of the same dimensionality. Moreover, they demonstrate that semantic hashing codes used as an initial document filter can improve precision of TF-IDF-based retrieval. Learning binary representation from BOW, however, has its disadvantages. First, word-count representation, and in turn the learned codes, are not in itself stronger than TF-IDF. Second, BOW is an inefficient representation: even for moderate-size vocabularies BOW vectors can have thousands of dimensions. Learning fully-connected autoencoders for such high-dimensional vectors is impractical. Salakhutdinov & Hinton restricted the BOW vocabulary in their experiments to 2000 most frequent words.

Binary codes have also been applied to cross-modal retrieval where text is one of the modalities. Specifically, Wang et al. (2013) incorporated tag information that often accompany text documents, while Masci et al. (2014) employed siamese neural networks to learn single binary representation for text and image data.

Recently several works explored simple neural models for unsupervised learning of distributed representations of words, sentences and documents. Mikolov et al. (2013) proposed log-linear models that learn distributed representations of words by predicting a central word from its context (CBOW model) or by predicting context words given the central word (Skip-gram model). The CBOW model was then extended by Le and Mikolov (2014) to learn distributed representations of documents. Specifically, they proposed Paragraph Vector Distributed Memory (PV-DM) model, in which the central word is predicted given the context words and the document vector. During training, PV-DM learns the word embeddings and the parameters of the softmax that models the conditional probability distribution for the central words. During inference, word embeddings and softmax weights are fixed, but the gradients are backpropagated to the inferred document vector. In addition to PV-DM, Le & Mikolov studied also a simpler model, namely Paragraph Vector Distributed Bag of Words (PV-DBOW). This model predicts words in the document given only the document vector. It therefore disregards context surrounding the predicted word and does not learn word embeddings. Le & Mikolov demonstrated that paragraph vectors outperform BOW and bag-of-bigrams in information retrieval task, while using only few hundreds of dimensions. These models are also amendable to learning and inference over large vocabularies. Original CBOW network used hierarchical softmax to model the probability distribution for the central word. One can also use noise-contrastive estimation (Gutmann and Hyvärinen, 2010) or importance sampling (Cho et al., 2015) to approximate the gradients with respect to the softmax logits.

An alternative approach to learning representation of pieces of text has been recently described by Kiros et al. (2015). Networks proposed therein, inspired by the Skip-gram model, learn to predict surrounding sentences given the center sentence. To this end, the center sentence is encoded by an encoder network and the surrounding sentences are predicted by a decoder network conditioned on the center sentence code. Once trained, these models can encode sentences without resorting to backpropagation inference. However, they learn representations at the sentence level but not at the document level.

In this work we present Binary Paragraph Vector models, an extensions to PV-DBOW and PV-DM that learn short binary codes for text documents. One inspiration for binary paragraph vectors comes from a recent work by Lin et al. (2015) on learning binary codes for images. Specifically, we introduce a sigmoid layer to the paragraph vector models, and train it in a way that encourages binary activations. We demonstrate that the resultant binary paragraph vectors significantly outperform semantic hashing codes. We also evaluate binary paragraph vectors in transfer learning settings, where training and inference are carried out on unrelated text corpora. Finally, we study models that simultaneously learn short binary codes for document filtering and longer, real-valued representations for ranking. While Lin et al. (2015) employed a supervised criterion to learn image codes, binary paragraph vectors remain unsupervised models: they learn to predict words in documents.

2 Binary paragraph vector models

The basic idea in binary paragraph vector models is to introduce a sigmoid nonlinearity before the softmax that models the conditional probability of words given the context. If we then enforce binary or near-binary activations in this nonlinearity, the probability distribution over words will be conditioned on a bit vector context, rather than real-valued representation. The inference in the model proceeds like in Paragraph Vector, except the document code is constructed from the sigmoid activations. After rounding, this code can be seen as a distributed binary representation of the document.

In the simplest Binary PV-DBOW model (Figure 1) the dimensionality of the real-valued document embeddings is equal to the length of the binary codes. Despite this low dimensional representation – a useful binary hash will typically have 128 or fewer bits – this model performed surprisingly well in our experiments. Note that we cannot simply increase the embedding dimension-

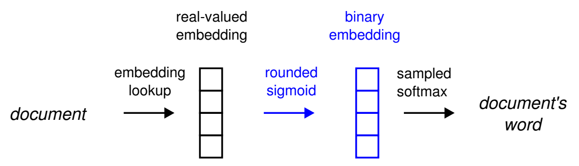


Figure 1: The Binary PV-DBOW model. Modifications to the original PV-DBOW model are highlighted.

ality in Binary PV-DBOW in order to learn better codes: binary vectors learned in this way would be too long to be useful in document hashing. The retrieval performance can, however, be improved by using binary codes for initial filtering of documents, and then using a representation with higher capacity to rank the remaining documents by their similarity to the query. Salakhutdinov and Hinton (2009), for example, used semantic hashing codes for initial filtering and TF-IDF for ranking. A similar document retrieval strategy can be realized with binary paragraph vectors. Furthermore, we can extend the Binary PV-DBOW model to simultaneously learn short binary codes and higher-dimensional real-valued representations. Specifically, in the Real-Binary PV-DBOW model (Figure 2) we introduce a linear projection between the document embedding matrix and the sigmoid nonlinearity. During training, we learn the softmax parameters and the projection matrix. During inference, softmax weights and the projection matrix are fixed. This way, we simultaneously obtain a high-capacity representation of a document in the embedding matrix, e.g. 300-dimensional real-valued vector, and a short binary representation from the sigmoid activations. One advantage of

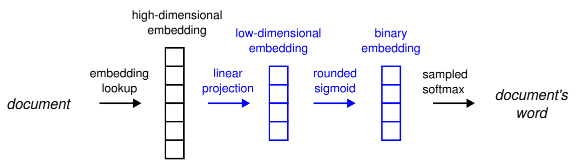


Figure 2: The Real-Binary PV-DBOW model. Modifications to the original PV-DBOW model are highlighted.

using the Real-Binary PV-DBOW model over two separate networks is that we need to store only one set of softmax parameters (and a small projection matrix) in the memory, instead of two large weight matrices. Additionally, only one model needs to be trained, rather than two distinct networks.

Binary document codes can also be learned by extending distributed memory models. Le and Mikolov (2014) suggest that in PV-DM, a context of the central word can be constructed by either concatenating or averaging the document vector and the embeddings of the surrounding words. However, in Binary PV-DM (Figure 3) we always construct the context by concatenating the relevant vectors before applying the sigmoid nonlinearity. This way, the length of binary codes is not tied to

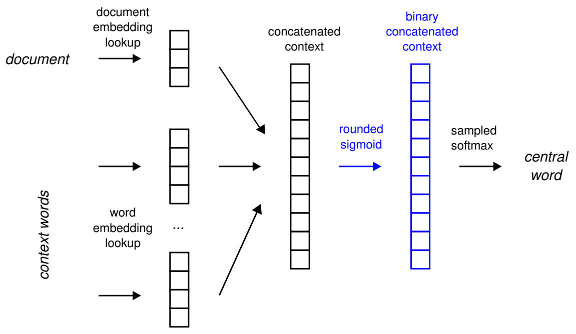


Figure 3: The Binary PV-DM model. Modifications to the original PV-DM model are highlighted.

the dimensionality of word embeddings.

Softmax layers in the models described above should be trained to predict words in documents given binary context vectors. Training should therefore encourage binary activations in the preceding sigmoid layers. This can be done in several ways. In semantic hashing autoencoders Salakhutdinov and Hinton (2009) added noise to the sigmoid coding layer. Error backpropagation then countered the noise, by forcing the activations to be close to 0 or 1. Another approach was used by Krizhevsky and Hinton (2011) in autoencoders that learned binary codes for small images. During the forward pass, activations in the coding layer were rounded to 0 or 1. Original (i.e. not rounded) activations were used when backpropagating errors. Alternatively, one could model the document codes with stochastic binary neurons. Learning in this case can still proceed with error backpropagation, provided that a suitable gradient estimator is used alongside stochastic activations. We experimented with the methods used in semantic hashing and Krizhevsky's autoencoders, as well as with the two biased gradient estimators for stochastic binary neurons discussed by Bengio et al. (2013). We also investigated the slope annealing trick (Chung et al., 2016) when training networks with stochastic binary activations. From our experience, binary paragraph vector models with rounded activations are easy to train and learn better codes than models with noise-based binarization or stochastic neurons. We therefore use Krizhevsky's binarization in our models.

3 Experiments

To assess the performance of binary paragraph vectors, we carried out experiments on three datasets: 20 Newsgroups[1], a cleansed version (also called v2) of Reuters Corpus Volume 1[2] (RCV1) and English Wikipedia[3]. As paragraph vectors can be trained with relatively large vocabularies, we did not perform any stemming of the source text. However, we removed stop words as well as words shorter than two characters and longer than 15 characters. Results reported by (Li et al., 2015) indicate that performance of PV-DBOW can be improved by including *n-grams* in the model. We therefore evaluated two variants of Binary PV-DBOW: one predicting words in documents and one predicting words and bigrams. Since 20 Newsgroups is a relatively small dataset, we used all words and bigrams from its documents. This amounts to a vocabulary with slightly over one million elements. For the RCV1 dataset we used words and bigrams with at least 10 occurrences in the text, which gives a vocabulary with approximately 800 thousands elements. In case of English Wikipedia we used words and bigrams with at least 100 occurrences, which gives a vocabulary with approximately 1.5 million elements.

The 20 Newsgroups dataset comes with reference train/test sets. In case of RCV1 we used half of the documents for training and the other half for evaluation. In case of English Wikipedia we held out for testing randomly selected 10% of the documents. We perform document retrieval by selecting queries from the test set and ordering other test documents according to the similarity of the inferred codes. We use Hamming distance for binary codes and cosine similarity for real-valued representations. Results are averaged over queries. We assess the performance of our models with precision-recall curves and two popular information retrieval metrics, namely mean average precision (MAP) and the normalized discounted cumulative gain at the 10th result (NDCG@10) (Järvelin and Kekäläinen, 2002). The results depend, of course, on the chosen document relevancy measure. Relevancy measure for the 20 Newsgroups dataset is straightforward: a retrieved document is relevant to the query if they both belong to the same newsgroup. In RCV1 each document belongs to a hierarchy of topics, making the definition of relevancy less obvious. In this case we adopted the relevancy measure used by Salakhutdinov and Hinton (2009). That is, the relevancy is calculated as the fraction of overlapping labels in a retrieved document and the query document. Overall, our selection of test datasets and relevancy measures for 20 Newsgroups and RCV1 follows Salakhutdinov and Hinton (2009), enabling comparison with semantic hashing codes. To assess the relevancy of articles in English Wikipedia we can employ categories assigned to them. However, unlike in RCV1, Wikipedia categories can have multiple parent categories and cyclic dependencies. Therefore, for this dataset we adopted a simplified relevancy measure: two articles are relevant if they share at least one category. We also removed from the test set categories with less than 20 documents as well as documents that were left with no categories. Overall, the relevancy is measured over more than $11,800$ categories, making English Wikipedia harder than the other two benchmarks.

We use AdaGrad (Duchi et al., 2011) for training and inference in all experiments reported in this work. During training we employ dropout (Srivastava et al., 2014) in the embedding layer. To facilitate models with large vocabularies, we approximate the gradients with respect to the softmax logits using the method described by Cho et al. (2015). Binary PV-DM networks use the same number of dimensions for document codes and word embeddings.

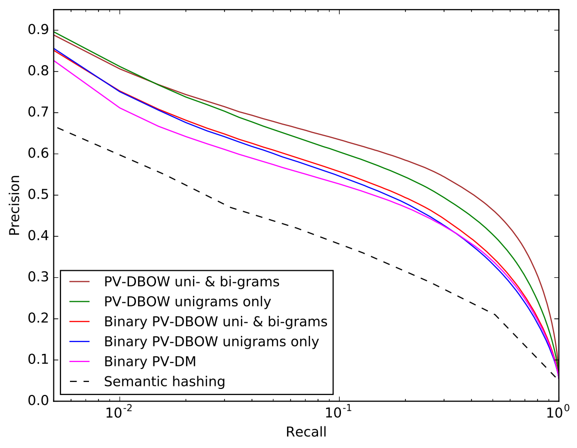
Performance of 128- and 32-bit binary paragraph vector codes is reported in Table 1 and in Figure 4. For comparison we also report performance of real-valued paragraph vectors. Note that the binary codes perform very well, despite their far lower capacity: on 20 Newsgroups and RCV1 the 128-bit Binary PV-DBOW trained with bigrams approaches the performance of the real-valued paragraph vectors, while on English Wikipedia its performance is slightly lower. Furthermore, Binary PV-DBOW with bigrams outperforms semantic hashing codes: comparison of precision-recall curves from Figures 4a and 4b with Salakhutdinov and Hinton (2009, Figures 6 & 7) shows that 128-bit codes learned with this model outperform 128-bit semantic hashing codes on 20 Newsgroups and RCV1. Moreover, the 32-bit codes from this model outperform 128-bit semantic hashing codes on the RCV1 dataset, and

[1]Available at `http://qwone.com/~jason/20Newsgroups`

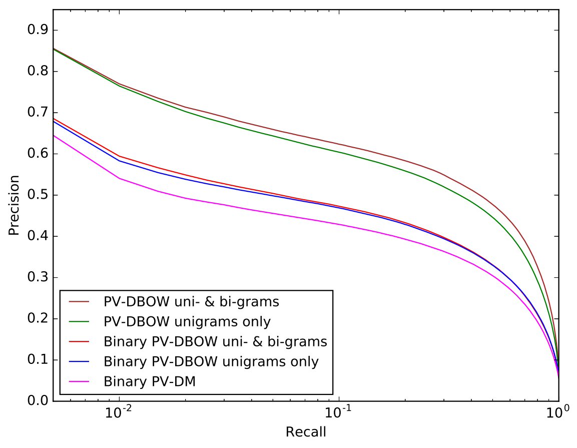
[2]Available at `http://trec.nist.gov/data/reuters/reuters.html`

[3]A snapshot from April 5th, 2016

(a) 20 Newsgroups

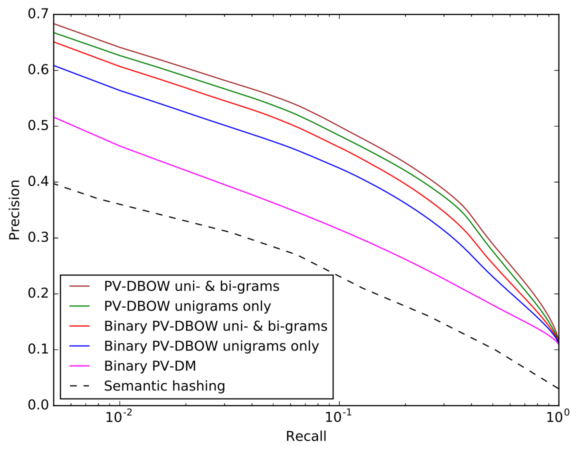


128 dimensional codes

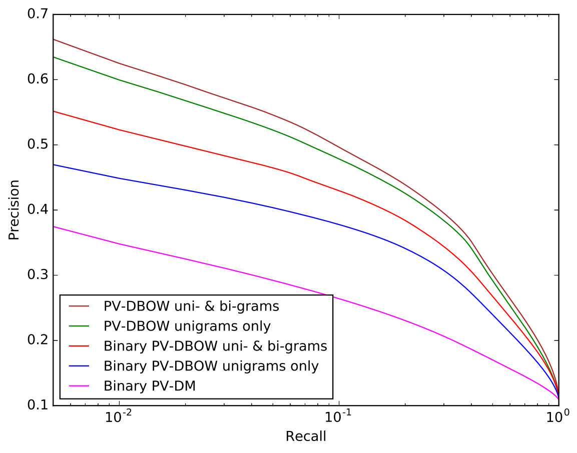


32 dimensional codes

(b) RCV1

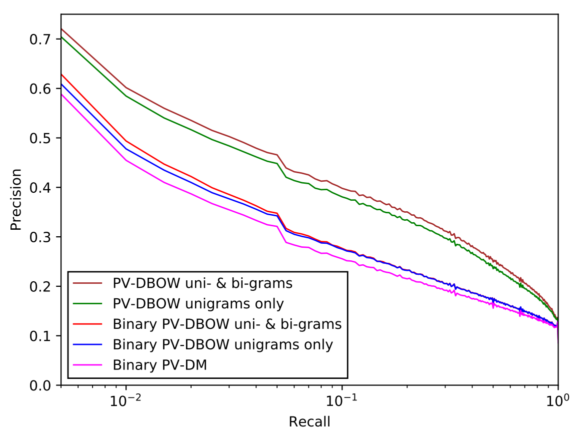


128 dimensional codes

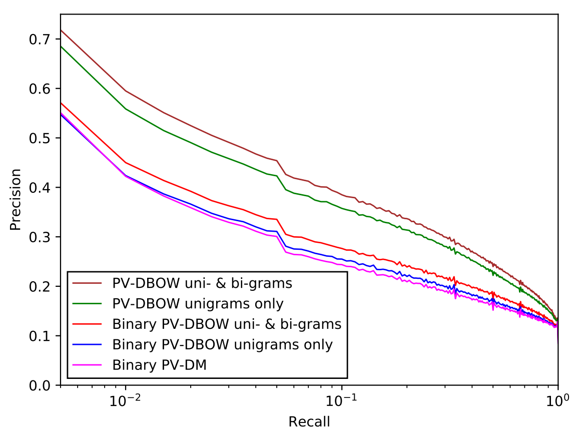


32 dimensional codes

(c) English Wikipedia



128 dimensional codes



32 dimensional codes

Figure 4: Precision-recall curves for the 20 Newsgroups, RCV1 and the English Wikipedia. Cosine similarity was used with real-valued representations and the Hamming distance with binary codes. For comparison we also included semantic hashing results reported by Salakhutdinov and Hinton (2009, Figures 6 & 7).

Code size	Model	With bigrams	20 Newsgroups		RCV1		English Wikipedia	
			MAP	NDCG@10	MAP	NDCG@10	MAP	NDCG@10
128	PV-DBOW	no	0.4	0.75	0.25	0.79	0.25	0.59
		yes	0.45	0.75	0.27	0.8	0.26	0.6
	Binary PV-DBOW	no	0.34	0.69	0.22	0.74	0.18	0.48
		yes	**0.35**	**0.69**	**0.24**	**0.77**	**0.18**	**0.49**
	PV-DM	N/A	0.41	0.73	0.23	0.78	0.24	0.59
	Binary PV-DM		0.34	0.65	0.18	0.69	0.16	0.46
32	PV-DBOW	no	0.43	0.71	0.26	0.75	0.23	0.55
		yes	0.46	0.72	0.27	0.77	0.25	0.58
	Binary PV-DBOW	no	0.32	0.53	0.22	0.6	0.16	0.41
		yes	**0.32**	**0.54**	**0.25**	**0.66**	**0.17**	**0.44**
	PV-DM	N/A	0.43	0.7	0.23	0.77	0.23	0.55
	Binary PV-DM		0.29	0.49	0.17	0.53	0.15	0.41

Table 1: Information retrieval results. The best results with binary models are highlighted.

on the 20 Newsgroups dataset give similar precision up to approximately 3% recall and better precision for higher recall levels. Note that the difference in this case lies not only in retrieval precision: the short 32-bit Binary PV-DBOW codes are more efficient for indexing than long 128-bit semantic hashing codes.

We also compared binary paragraph vectors against codes constructed by first inferring short, real-valued paragraph vectors and then using a separate hashing algorithm for binarization. When the dimensionality of the paragraph vectors is equal to the size of binary codes, the number of network parameters in this approach is similar to that of Binary PV models. We experimented with two standard hashing algorithms, namely random hyperplane projection (Charikar, 2002) and iterative quantization (Gong and Lazebnik, 2011). Paragraph vectors in these experiments were inferred using PV-DBOW with bigrams. Results reported in Table 2 show no benefit from using a separate algorithm for binarization. On the 20 Newsgroups and RCV1 datasets Binary PV-DBOW yielded higher MAP than the two baseline approaches. On English Wikipedia iterative quantization achieved MAP equal to Binary PV-DBOW, while random hyperplane projection yielded lower MAP. Some gain in precision of top hits can be observed for iterative quantization, as indicated by NDCG@10. However, precision of top hits can also be improved by querying with Real-Binary PV-DBOW model (Section 3.2). It is also worth noting that end-to-end inference in Binary PV models is more convenient than inferring real-valued vectors and then using another algorithm for hashing.

Li et al. (2015) argue that PV-DBOW outperforms PV-DM on a sentiment classification task, and demonstrate that the performance of PV-DBOW can be improved by including bigrams in the vocabulary. We observed similar results with Binary PV models. That is, including bigrams in the vocabulary usually improved retrieval precision. Also, codes learned with Binary PV-DBOW provided higher retrieval precision than Binary PV-DM codes. Furthermore, to choose the context size for the Binary PV-DM models, we evaluated several networks on validation sets taken out of the training data. The best results were obtained with a minimal one-word, one-sided context window. This is the distributed memory architecture most similar to the Binary PV-DBOW model.

Hashing algorithm	20 Newsgroups		RCV1		English Wikipedia	
	MAP	NDCG@10	MAP	NDCG@10	MAP	NDCG@10
Random hyperplane projection	0.27	0.53	0.21	0.66	0.16	0.44
Iterative quantization	0.31	0.58	0.23	0.68	0.17	0.46

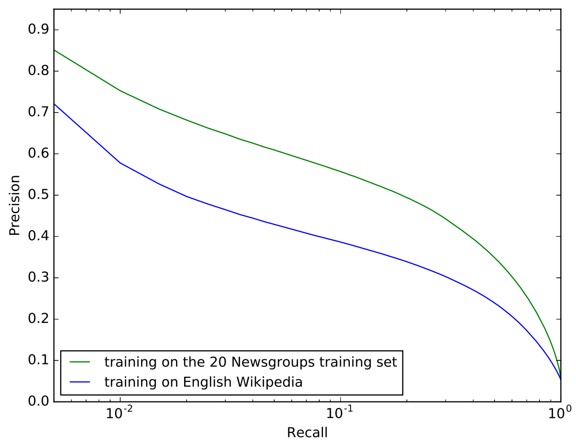
Table 2: Information retrieval results for 32-bit binary codes constructed by first inferring 32d real-valued paragraph vectors and then employing a separate hashing algorithm for binarization. Paragraph vectors were inferred using PV-DBOW with bigrams.

3.1 Transfer learning

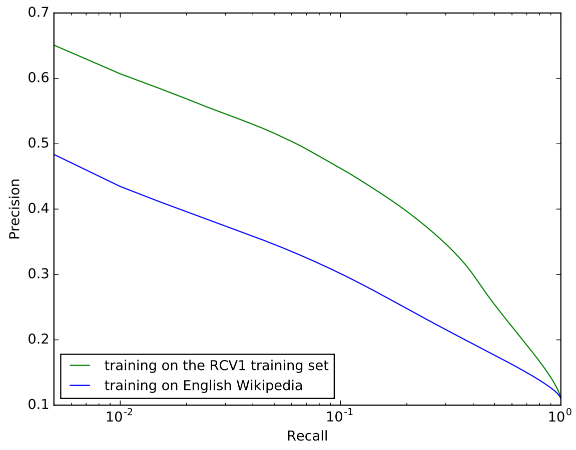
In the experiments presented thus far we had at our disposal training sets with documents similar to the documents for which we inferred binary codes. One could ask a question, if it is possible to use binary paragraph vectors without collecting a domain-specific training set? For example, what if we needed to hash documents that are not associated with any available domain-specific corpus? One solution could be to train the model with a big generic text corpus, that covers a wide variety of domains. Lau and Baldwin (2016) evaluated this approach for real-valued paragraph vectors, with promising results. It is not obvious, however, whether short binary codes would also perform well in similar settings. To shed light on this question we trained Binary PV-DBOW with bigrams on the English Wikipedia, and then inferred binary codes for the test parts of the 20 Newsgroups and RCV1 datasets. The results are presented in Table 3 and in Figure 5. The model trained on an unrelated text corpus gives lower retrieval precision than models with domain-specific training sets, which is not surprising. However, it still performs remarkably well, indicating that the semantics it captured can be useful for different text collections. Importantly, these results were obtained without domain-specific finetuning.

	MAP	NDCG@10
20 Newsgroups	0.24	0.51
RCV1	0.18	0.66

Table 3: Information retrieval results for the Binary PV-DBOW model trained on an unrelated text corpus. Results are reported for 128-bit codes.



(a) 20 Newsgroups



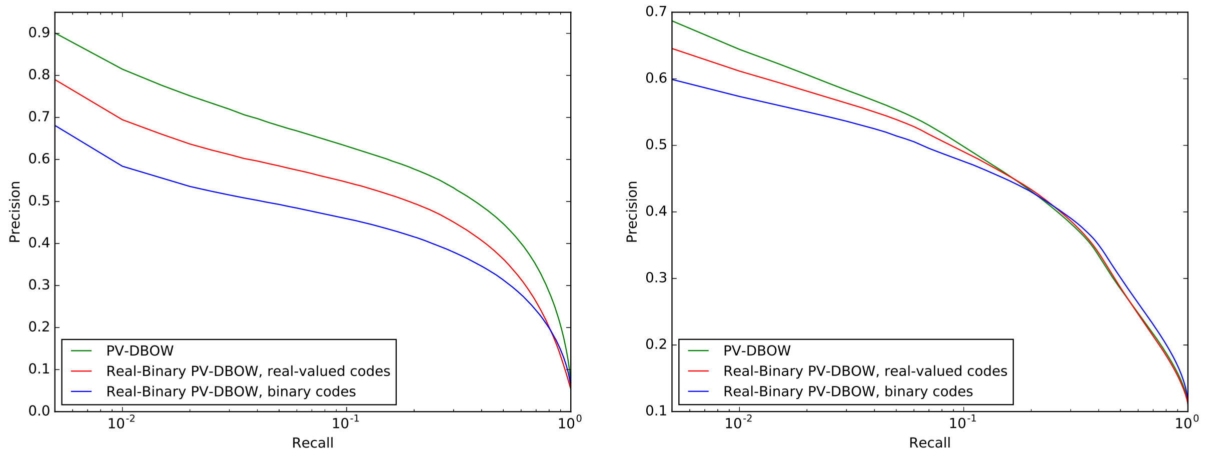
(b) RCV1

Figure 5: Precision-recall curves for the baseline Binary PV-DBOW models and a Binary PV-DBOW model trained on an unrelated text corpus. Results are reported for 128-bit codes.

3.2 Retrieval with Real-Binary models

As pointed out by Salakhutdinov and Hinton (2009), when working with large text collections one can use short binary codes for indexing and a representation with more capacity for ranking. Following this idea, we proposed Real-Binary PV-DBOW model (Section 2) that can simultaneously learn short binary codes and high-dimensional real-valued representations. We begin evaluation of this model by comparing retrieval precision of real-valued and binary representations learned by it. To this end, we trained a Real-Binary PV-DBOW model with 28-bit binary codes and 300-dimensional real-valued representations on the 20 Newsgroups and RCV1 datasets. Results are reported in Figure 6. The real-valued representations learned with this model give lower precision than PV-DBOW vectors but, importantly, improve precision over binary codes for top ranked documents. This justifies their use alongside binary codes.

Using short binary codes for initial filtering of documents comes with a tradeoff between the retrieval performance and the recall level. For example, one can select a small subset of similar documents by using 28–32 bit codes and retrieving documents within small Hamming distance to the query. This will improve retrieval performance, and possibly also precision, at the cost of recall. Conversely, short codes provide a less fine-grained hashing and can be used to index documents within larger Hamming distance to the



(a) 20 Newsgroups

(b) RCV1

Figure 6: Information retrieval results for binary and real-valued codes learned by the Real-Binary PV-DBOW model with bigrams. Results are reported for 28-bit binary codes and 300d real-valued codes. A 300d PV-DBOW model is included for reference.

query. They can therefore be used to improve recall at the cost of retrieval performance, and possibly also precision. For these reasons, we evaluated Real-Binary PV-DBOW models with different code sizes and under different limits on the Hamming distance to the query. In general, we cannot expect these models to achieve 100% recall under the test settings. Furthermore, recall will vary on query-by-query basis. We therefore decided to focus on the NDCG@10 metric in this evaluation, as it is suited for measuring model performance when a short list of relevant documents is sought, and the recall level is not known. MAP and precision-recall curves are not applicable in these settings.

Code size	Radius	NDCG@10					
		20 NG		RCV1		Wikipedia	
		A	B	A	B	A	B
28	1	0.79	0.85	0.77	0.85	0.66	0.7
	2	0.72	0.8	0.73	0.81	0.62	0.65
24		0.65	0.79	0.7	0.76	0.56	0.59
	3	0.63	0.76	0.69	0.74	0.5	0.55

Table 4: Information retrieval results for the Real-Binary PV-DBOW model. Real-valued representations have 300 dimensions. (A) Binary codes are used for selecting documents within a given Hamming distance to the query and real-valued representations are used for ranking. (B) For comparison, variant A was repeated with binary codes inferred using plain Binary PV-DBOW and real-valued representation inferred using original PV-DBOW model.

Information retrieval results for Real-Binary PV-DBOW are summarized in Table 4. The model gives higher NDCG@10 than 32-bit Binary PV-DBOW codes (Table 1). The difference is large when the initial filtering is restrictive, e.g. when using 28-bit codes and 1-2 bit Hamming distance limit. Real-Binary PV-DBOW can therefore be useful when one needs to quickly find a short list of relevant documents in a large text collection, and the recall level is not of primary importance. If needed, precision can be further improved by using plain Binary PV-DBOW codes for filtering and standard DBOW representation for raking (Table 4, column B). Note, however, that PV-DBOW model would then use approximately 10 times more parameters than Real-Binary PV-DBOW.

4 Conclusion

In this article we presented simple neural networks that learn short binary codes for text documents. Our networks extend Paragraph Vector by introducing a sigmoid nonlinearity before the softmax that predicts words in documents. Binary codes inferred with the proposed networks achieve higher retrieval precision than semantic hashing codes on two popular information retrieval benchmarks. They also retain a lot of their precision when trained on an unrelated text corpus. Finally, we presented a network that simultaneously learns short binary codes and longer, real-valued representations.

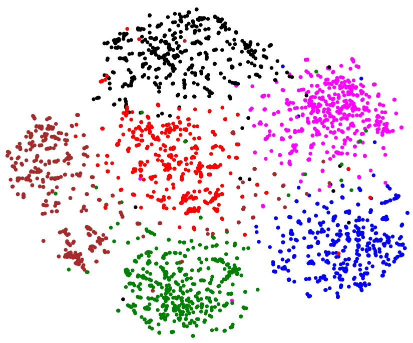
The best codes in our experiments were inferred with Binary PV-DBOW networks. The Binary PV-DM model did not perform so well. Li et al. (2015) made similar observations for Paragraph Vector models, and argue that in distributed memory model the word context takes a lot of the burden of predicting the central word from the document code. An interesting line of future research could, therefore, focus on models that account for word order, while learning good binary codes. It is also worth noting that Le and Mikolov (2014) constructed paragraph vectors by combining DM and DBOW representations. This strategy may proof useful also with binary codes, when employed with hashing algorithms designed for longer codes, e.g. with multi-index hashing (Norouzi et al., 2012).

Acknowledgments

This research is supported by National Science Centre, Poland grant no. 2013/09/B/ST6/01549 "Interactive Visual Text Analytics (IVTA): Development of novel, user-driven text mining and visualization methods for large text corpora exploration." This research was carried out with the support of the "HPC Infrastructure for Grand Challenges of Science and Engineering" project, co-financed by the European Regional Development Fund under the Innovative Economy Operational Programme. This research was supported in part by PL-Grid Infrastructure.

A Visualization of Binary PV codes

For an additional comparison with semantic hashing, we used t-distributed Stochastic Neighbor Embedding (van der Maaten and Hinton, 2008) to construct two-dimensional visualizations of codes learned by Binary PV-DBOW with bigrams. We used the same subsets of newsgroups and RCV1 topics that were used by Salakhutdinov and Hinton (2009, Figure 5). Codes learned by Binary PV-DBOW (Figure 7) appear slightly more clustered.


(a) A subset of the 20 Newsgroups dataset: green - soc.religion.christian, red - talk.politics.guns, blue - rec.sport.hockey, brown - talk.politics.mideast, magenta - comp.graphics, black - sci.crypt.

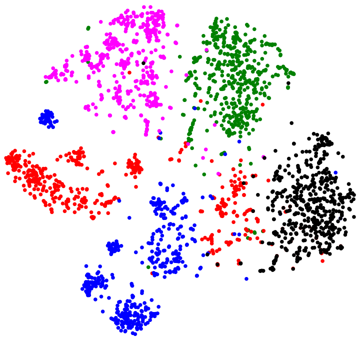

(b) A subset of the RCV1 dataset: green - disasters and accidents, red - government borrowing, blue - accounts/earnings, magenta - energy markets, black - EC monetary/economic.

Figure 7: t-SNE visualizations of 128 dimensional binary paragraph vector codes; the Hamming distance was used to calculate code similarity.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* .

Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, pages 380–388.

Sébastien Jean Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. ACL, volume 1, pages 1–10.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704* .

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

John Gantz and David Reinsel. 2012. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. Technical report, IDC.

Yunchao Gong and Svetlana Lazebnik. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, pages 817–824.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*. pages 297–304.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, pages 604–613.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)* 20(4):422–446.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Alex Krizhevsky and Geoffrey E Hinton. 2011. Using very deep autoencoders for content-based image retrieval. In *Proceedings of the 19th European Symposium on Artificial Neural Networks*. pages 489–494.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, pages 78–86.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning*. pages 1188–1196.

Bofang Li, Tao Liu, Xiaoyong Du, Deyuan Zhang, and Zhe Zhao. 2015. Learning document embeddings by predicting n-grams for sentiment classification of long movie reviews. *arXiv preprint arXiv:1512.08183* .

Kevin Lin, Huei Fang Yang, Jen Hao Hsiao, and Chu Song Chen. 2015. Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. pages 27–35.

Jonathan Masci, Michael M Bronstein, Alexander M Bronstein, and Jürgen Schmidhuber. 2014. Multimodal similarity-preserving hashing. *IEEE transactions on pattern analysis and machine intelligence* 36(4):824–830.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Mohammad Norouzi, Ali Punjani, and David J Fleet. 2012. Fast search in hamming space with multi-index hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pages 3108–3115.

Ruslan Salakhutdinov and Geoffrey E Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50(7):969–978.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927* .

Qifan Wang, Dan Zhang, and Luo Si. 2013. Semantic hashing using tags and topic modeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 213–222.