Sparse Overcomplete Word Vector Representations

Manaal Faruqui Yulia Tsvetkov Dani Yogatama Chris Dyer Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
{mfaruqui,ytsvetko,dyogatama,cdyer,nasmith}@cs.cmu.edu

Abstract

Current distributed representations of words show little resemblance to theories of lexical semantics. The former are dense and uninterpretable, the latter largely based on familiar, discrete classes (e.g., supersenses) and relations (e.g., synonymy and hypernymy). We propose methods that transform word vectors into sparse (and optionally binary) vectors. The resulting representations are more similar to the interpretable features typically used in NLP, though they are discovered automatically from raw corpora. Because the vectors are highly sparse, they are computationally easy to work with. Most importantly, we find that they outperform the original vectors on benchmark tasks.

1 Introduction

Distributed representations of words have been shown to benefit NLP tasks like parsing (Lazaridou et al., 2013; Bansal et al., 2014), named entity recognition (Guo et al., 2014), and sentiment analysis (Socher et al., 2013). The attraction of word vectors is that they can be derived directly from raw, unannotated corpora. Intrinsic evaluations on various tasks are guiding methods toward discovery of a representation that captures many facts about lexical semantics (Turney, 2001; Turney and Pantel, 2010).

Yet word vectors do not look anything like the representations described in most lexical semantic theories, which focus on identifying classes of words (Levin, 1993; Baker et al., 1998; Schuler, 2005) and relationships among word meanings (Miller, 1995). Though expensive to construct, conceptualizing word meanings symbolically is important for theoretical understanding and also when we incorporate lexical semantics into computational models where interpretability is desired. On the surface, discrete theories seem incommensurate with the distributed approach, a problem now receiving much attention in computational linguistics (Lewis and Steedman, 2013; Kiela and Clark, 2013; Vecchi et al., 2013; Grefenstette, 2013; Lewis and Steedman, 2014; Paperno et al., 2014).

Our contribution to this discussion is a new, principled sparse coding method that transforms any distributed representation of words into sparse vectors, which can then be transformed into binary vectors (§2). Unlike recent approaches of incorporating semantics in distributional word vectors (Yu and Dredze, 2014; Xu et al., 2014; Faruqui et al., 2015), the method does not rely on any external information source. The transformation results in longer, sparser vectors, sometimes called an "overcomplete" representation (Olshausen and Field, 1997). Sparse, overcomplete representations have been motivated in other domains as a way to increase separability and interpretability, with each instance (here, a word) having a small number of active dimensions (Olshausen and Field, 1997; Lewicki and Sejnowski, 2000), and to increase stability in the presence of noise (Donoho et al., 2006).

Our work builds on recent explorations of sparsity as a useful form of inductive bias in NLP and machine learning more broadly (Kazama and Tsujii, 2003; Goodman, 2004; Friedman et al., 2008; Glorot et al., 2011; Yogatama and Smith, 2014, *inter alia*). Introducing sparsity in word vector dimensions has been shown to improve dimension interpretability (Murphy et al., 2012; Fyshe et al., 2014) and usability of word vectors as features in downstream tasks (Guo et al., 2014). The word vectors we produce are more than 90% sparse; we also consider binarizing transformations that bring them closer to the categories and relations of lex-

ical semantic theories. Using a number of state-of-the-art word vectors as input, we find consistent benefits of our method on a suite of standard benchmark evaluation tasks (§3). We also evaluate our word vectors in a word intrusion experiment with humans (Chang et al., 2009) and find that our sparse vectors are more interpretable than the original vectors (§4).

We anticipate that sparse, binary vectors can play an important role as features in statistical NLP models, which still rely predominantly on discrete, sparse features whose interpretability enables error analysis and continued development. We have made an implementation of our method publicly available.[1]

2 Sparse Overcomplete Word Vectors

We consider methods for transforming dense word vectors to sparse, binary overcomplete word vectors. Fig. 1 shows two approaches. The one on the top, method A, converts dense vectors to sparse overcomplete vectors (§2.1). The one beneath, method B, converts dense vectors to sparse and binary overcomplete vectors (§2.2 and §2.4).

Let V be the vocabulary size. In the following, $\mathbf{X} \in \mathbb{R}^{L \times V}$ is the matrix constructed by stacking V non-sparse "input" word vectors of length L (produced by an arbitrary word vector estimator). We will refer to these as initializing vectors. $\mathbf{A} \in \mathbb{R}^{K \times V}$ contains V sparse overcomplete word vectors of length K. "Overcomplete" representation learning implies that $K > L$.

2.1 Sparse Coding

In sparse coding (Lee et al., 2006), the goal is to represent each input vector $\mathbf{x}_i$ as a sparse linear combination of basis vectors, $\mathbf{a}_i$. Our experiments consider four initializing methods for these vectors, discussed in Appendix A. Given $\mathbf{X}$, we seek to solve

$$\arg\min_{\mathbf{D},\mathbf{A}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_2^2 + \lambda\Omega(\mathbf{A}) + \tau\|\mathbf{D}\|_2^2, \quad (1)$$

where $\mathbf{D} \in \mathbb{R}^{L \times K}$ is the dictionary of basis vectors. λ is a regularization hyperparameter, and Ω is the regularizer. Here, we use the squared loss for the reconstruction error, but other loss functions could also be used (Lee et al., 2009). To obtain sparse word representations we will impose an ℓ_1 penalty on $\mathbf{A}$. Eq. 1 can be broken down into loss for each word vector which can be optimized separately in parallel (§2.3):

$$\arg\min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{V} \|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 + \lambda\|\mathbf{a}_i\|_1 + \tau\|\mathbf{D}\|_2^2 \quad (2)$$

where $\mathbf{m}_i$ denotes the ith column vector of matrix $\mathbf{M}$. Note that this problem is not convex. We refer to this approach as **method A**.

2.2 Sparse Nonnegative Vectors

Nonnegativity in the feature space has often been shown to correspond to interpretability (Lee and Seung, 1999; Cichocki et al., 2009; Murphy et al., 2012; Fyshe et al., 2014; Fyshe et al., 2015). To obtain nonnegative sparse word vectors, we use a variation of the nonnegative sparse coding method (Hoyer, 2002). Nonnegative sparse coding further constrains the problem in Eq. 2 so that $\mathbf{D}$ and $\mathbf{a}_i$ are nonnegative. Here, we apply this constraint only to the representation vectors $\{\mathbf{a}_i\}$. Thus, the new objective for nonnegative sparse vectors becomes:

$$\arg\min_{\mathbf{D}\in\mathbb{R}_{\geq 0}^{L\times K},\mathbf{A}\in\mathbb{R}_{\geq 0}^{K\times V}} \sum_{i=1}^{V} \|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|_2^2 + \lambda\|\mathbf{a}_i\|_1 + \tau\|\mathbf{D}\|_2^2 \quad (3)$$

This problem will play a role in our second approach, **method B**, to which we will return shortly. This nonnegativity constraint can be easily incorporated during optimization, as explained next.

2.3 Optimization

We use online adaptive gradient descent (AdaGrad; Duchi et al., 2010) for solving the optimization problems in Eqs. 2–3 by updating $\mathbf{A}$ and $\mathbf{D}$. In order to speed up training we use asynchronous updates to the parameters of the model in parallel for every word vector (Duchi et al., 2012; Heigold et al., 2014).

However, directly applying stochastic subgradient descent to an ℓ_1-regularized objective fails to produce sparse solutions in bounded time, which has motivated several specialized algorithms that target such objectives. We use the AdaGrad variant of one such learning algorithm, the regularized dual averaging algorithm (Xiao, 2009), which keeps track of the online average gradient at time t: $\bar{g}_t = \frac{1}{t}\sum_{t'=1}^{t} g_{t'}$ Here, the subgradients do not include terms for the regularizer; they are derivatives of the unregularized objective ($\lambda = 0, \tau = 0$)

[1] https://github.com/mfaruqui/sparse-coding

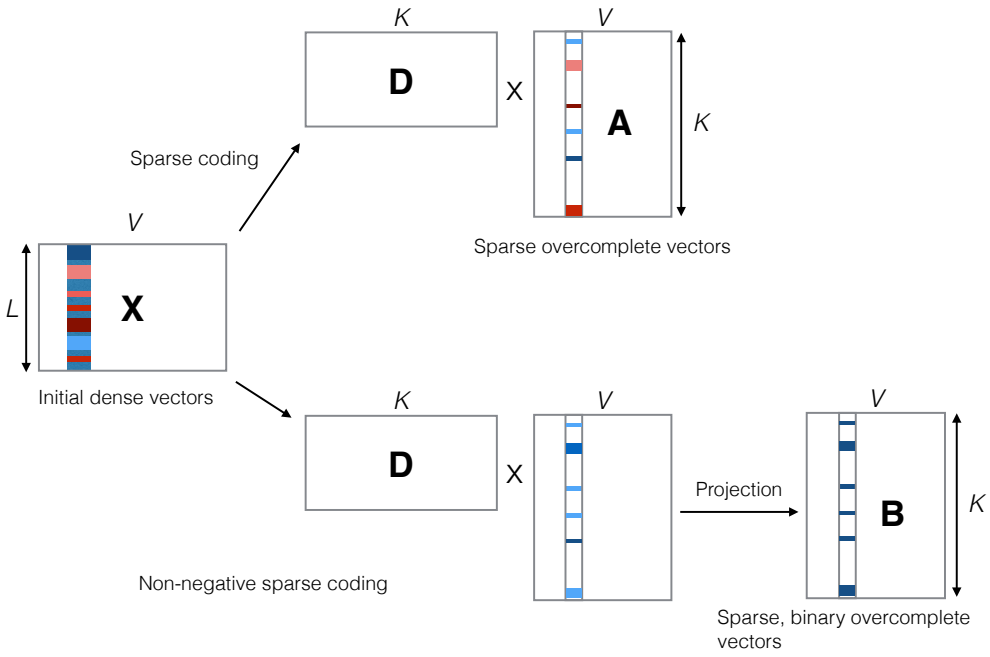

Figure 1: Methods for obtaining sparse overcomplete vectors (top, method A, §2.1) and sparse, binary overcomplete word vectors (bottom, method B, §2.2 and §2.4). Observed dense vectors of length L (left) are converted to sparse non-negative vectors (center) of length K which are then projected into the binary vector space (right), where $L \ll K$. $\mathbf{X}$ is dense, $\mathbf{A}$ is sparse, and $\mathbf{B}$ is the binary word vector matrix. Strength of colors signify the magnitude of values; negative is red, positive is blue, and zero is white.

with respect to $\mathbf{a}_i$. We define

$$\gamma = -\text{sign}(\bar{g}_{t,i,j}) \frac{\eta t}{\sqrt{G_{t,i,j}}} (|\bar{g}_{t,i,j}| - \lambda),$$

where $G_{t,i,j} = \sum_{t'=1}^{t} g_{t',i,j}^2$. Now, using the average gradient, the ℓ_1-regularized objective is optimized as follows:

$$a_{t+1,i,j} = \begin{cases} 0, & \text{if } |\bar{g}_{t,i,j}| \leq \lambda \\ \gamma, & \text{otherwise} \end{cases} \quad (4)$$

where, $a_{t+1,i,j}$ is the jth element of sparse vector $\mathbf{a}_i$ at the tth update and $\bar{g}_{t,i,j}$ is the corresponding average gradient. For obtaining nonnegative sparse vectors we take projection of the updated $\mathbf{a}_i$ onto $\mathbb{R}^K_{\geq 0}$ by choosing the closest point in $\mathbb{R}^K_{\geq 0}$ according to Euclidean distance (which corresponds to zeroing out the negative elements):

$$a_{t+1,i,j} = \begin{cases} 0, & \text{if } |\bar{g}_{t,i,j}| \leq \lambda \\ 0, & \text{if } \gamma < 0 \\ \gamma, & \text{otherwise} \end{cases} \quad (5)$$

2.4 Binarizing Transformation

Our aim with **method B** is to obtain word representations that can emulate the binary-feature space designed for various NLP tasks. We could

$\mathbf{X}$	L	λ	τ	K	% Sparse
Glove	300	1.0	10^{-5}	3000	91
SG	300	0.5	10^{-5}	3000	92
GC	50	1.0	10^{-5}	500	98
Multi	48	0.1	10^{-5}	960	93

Table 1: Hyperparameters for learning sparse overcomplete vectors tuned on the WS-353 task. Tasks are explained in §B. The four initial vector representations $\mathbf{X}$ are explained in §A.

hot, fresh, fish, 1/2, wine, salt
series, tv, appearances, episodes
1975, 1976, 1968, 1970, 1977, 1969
dress, shirt, ivory, shirts, pants
upscale, affluent, catering, clientele

Table 2: Highest frequency words in randomly picked word clusters of binary sparse overcomplete Glove vectors.

state this as an optimization problem:

$$\underset{\substack{\mathbf{D}\in\mathbb{R}^{L\times K}\\ \mathbf{B}\in\{0,1\}^{K\times V}}}{\arg\min} \sum_{i=1}^{V} \|\mathbf{x}_i - \mathbf{D}\mathbf{b}_i\|_2^2 + \lambda\|\mathbf{b}_i\|_1^1 + \tau\|\mathbf{D}\|_2^2 \tag{6}$$

where $\mathbf{B}$ denotes the binary (and also sparse) representation. This is an mixed integer bilinear program, which is NP-hard (Al-Khayyal and Falk, 1983). Unfortunately, the number of variables in the problem is $\approx KV$ which reaches 100 million when $V = 100,000$ and $K = 1,000$, which is intractable to solve using standard techniques.

A more tractable relaxation to this hard problem is to first constrain the continuous representation $\mathbf{A}$ to be nonnegative (i.e, $\mathbf{a}_i \in \mathbb{R}^K_{\geq 0}$; §2.2). Then, in order to avoid an expensive computation, we take the nonnegative word vectors obtained using Eq. 3 and project nonzero values to 1, preserving the 0 values. Table 2 shows a random set of word clusters obtained by (i) applying our method to Glove initial vectors and (ii) applying k-means clustering ($k = 100$). In §3 we will find that these vectors perform well quantitatively.

2.5 Hyperparameter Tuning

Methods A and B have three hyperparameters: the ℓ_1-regularization penalty λ, the ℓ_2-regularization penalty τ, and the length of the overcomplete word vector representation K. We perform a grid search on $\lambda \in \{0.1, 0.5, 1.0\}$ and $K \in \{10L, 20L\}$, selecting values that maximizes performance on one "development" word similarity task (WS-353, discussed in §B) while achieving at least 90% sparsity in overcomplete vectors. τ was tuned on one collection of initializing vectors (Glove, discussed in §A) so that the vectors in $\mathbf{D}$ are near unit norm. The four vector representations and their corresponding hyperparameters selected by this procedure are summarized in Table 1. These hyperparameters were chosen for method A and retained for method B.

3 Experiments

Using methods A and B, we constructed sparse overcomplete vector representations $\mathbf{A}$ and $\mathbf{B}$ resp., starting from four initial vector representations $\mathbf{X}$; these are explained in Appendix A. We used one benchmark evaluation (WS-353) to tune hyperparameters, resulting in the settings shown in Table 1; seven other tasks were used to evaluate the quality of the sparse overcomplete representations. The first of these is a word similarity task, where the score is correlation with human judgments, and the others are classification accuracies of an ℓ_2-regularized logistic regression model trained using the word vectors. These tasks are described in detail in Appendix B.

3.1 Effects of Transforming Vectors

First, we quantify the effects of our transformations by comparing their output to the initial ($\mathbf{X}$) vectors. Table 3 shows consistent improvements of sparsifying vectors (method A). The exceptions are on the SimLex task, where our sparse vectors are worse than the skip-gram initializer and on par with the multilingual initializer. Sparsification is beneficial across all of the text classification tasks, for all initial vector representations. On average across all vector types and all tasks, sparse overcomplete vectors outperform their corresponding initializers by 4.2 points.[2]

Binarized vectors (from method B) are also usually better than the initial vectors (also shown in Table 3), and tend to outperform the sparsified variants, except when initializing with Glove. On average across all vector types and all tasks, binarized overcomplete vectors outperform their corresponding initializers by 4.8 points and the continuous, sparse intermediate vectors by 0.6 points.

From here on, we explore more deeply the sparse overcomplete vectors from method A (denoted by $\mathbf{A}$), leaving binarization (method B) aside.

3.2 Effect of Vector Length

How does the length of the overcomplete vector (K) affect performance? We focus here on the Glove vectors, where $L = 300$, and report average performance across all tasks. We consider $K = \alpha L$ where $\alpha \in \{1, 2, 3, 5, 10, 15, 20\}$. Figure 2 plots the average performance across tasks against α. The earlier selection of $K = 3,000$ ($\alpha = 10$) gives the best result; gains are monotonic in α to that point and then begin to diminish.

3.3 Alternative Transformations

We consider two alternative transformations. The first preserves the original vector length but

[2] We report correlation on a 100 point scale, so that the average which includes accuracuies and correlation is equally representatitve of both.

Vectors		SimLex Corr.	Senti. Acc.	TREC Acc.	Sports Acc.	Comp. Acc.	Relig. Acc.	NP Acc.	Average
Glove	**X**	36.9	77.7	76.2	95.9	79.7	86.7	77.9	76.2
	A	38.9	**81.4**	**81.5**	**96.3**	**87.0**	**88.8**	**82.3**	**79.4**
	B	**39.7**	81.0	81.2	95.7	84.6	87.4	81.6	78.7
SG	**X**	**43.6**	81.5	77.8	97.1	80.2	85.9	80.1	78.0
	A	41.7	**82.7**	81.2	**98.2**	84.5	86.5	81.6	79.4
	B	42.8	81.6	**81.6**	95.2	**86.5**	**88.0**	**82.9**	**79.8**
GC	**X**	9.7	68.3	64.6	75.1	60.5	76.0	79.4	61.9
	A	12.0	73.3	77.6	77.0	68.3	**81.0**	**81.2**	67.2
	B	**18.7**	**73.6**	**79.2**	**79.7**	**70.5**	79.6	79.4	**68.6**
Multi	**X**	**28.7**	75.5	63.8	83.6	64.3	81.8	79.2	68.1
	A	28.1	**78.6**	79.2	93.9	78.2	84.5	81.1	74.8
	B	**28.7**	77.6	**82.0**	**94.7**	**81.4**	**85.6**	**81.9**	**75.9**

Table 3: Performance comparison of transformed vectors to initial vectors **X**. We show sparse overcomplete representations **A** and also binarized representations **B**. Initial vectors are discussed in §A and tasks in §B.

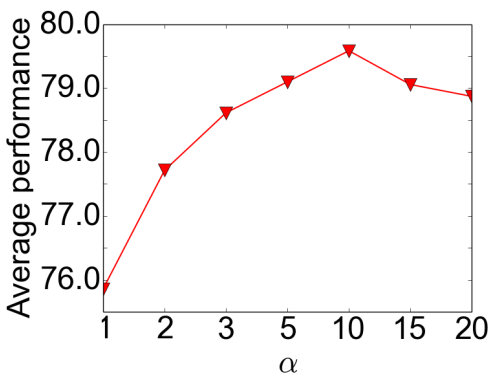

Figure 2: Average performace across all tasks for sparse overcomplete vectors (**A**) produced by Glove initial vectors, as a function of the ratio of K to L.

achieves a binary, sparse vector (**B**) by applying:

$$b_{i,j} = \begin{cases} 1 & \text{if } x_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

The second transformation was proposed by Guo et al. (2014). Here, the original vector length is also preserved, but sparsity is achieved through:

$$a_{i,j} = \begin{cases} 1 & \text{if } x_{i,j} \geq M^+ \\ -1 & \text{if } x_{i,j} \leq M^- \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where M^+ (M^-) is the mean of positive-valued (negative-valued) elements of **X**. These vectors are, obviously, not binary.

We find that on average, across initializing vectors and across all tasks that our sparse overcomplete (**A**) vectors lead to better performance than either of the alternative transformations.

4 Interpretability

Our hypothesis is that the dimensions of sparse overcomplete vectors are more interpretable than those of dense word vectors. Following Murphy et al. (2012), we use a word intrusion experiment (Chang et al., 2009) to corroborate this hypothesis. In addition, we conduct qualitative analysis of interpretability, focusing on individual dimensions.

4.1 Word Intrusion

Word intrusion experiments seek to quantify the extent to which dimensions of a learned word representation are coherent to humans. In one instance of the experiment, a human judge is presented with five words in random order and asked to select the "intruder." The words are selected by the experimenter by choosing one dimension j of the learned representation, then ranking the words on that dimension alone. The dimensions are chosen in decreasing order of the variance of their values across the vocabulary. Four of the words are the top-ranked words according to j, and the "true" intruder is a word from the bottom half of the list, chosen to be a word that appears in the top 10% of some other dimension. An example of an instance is:

naval, industrial, technological, marine, identity

$\mathbf{X}$:	Glove	SG	GC	Multi	Average
$\mathbf{X}$	76.2	78.0	61.9	68.1	71.0
Eq. 7	75.7	75.8	60.5	64.1	69.0
Eq. 8 (Guo et al., 2014)	75.8	76.9	60.5	66.2	69.8
$\mathbf{A}$	**79.4**	**79.4**	**67.2**	**74.8**	**75.2**

Table 4: Average performance across all tasks and vector models using different transformations.

Vectors	A1	A2	A3	Avg.	IAA	κ
$\mathbf{X}$	61	53	56	57	70	0.40
$\mathbf{A}$	71	70	72	71	77	0.45

Table 5: Accuracy of three human annotators on the word intrusion task, along with the average inter-annotator agreement (Artstein and Poesio, 2008) and Fleiss' κ (Davies and Fleiss, 1982).

(The last word is the intruder.)

We formed instances from initializing vectors and from our sparse overcomplete vectors ($\mathbf{A}$). Each of these two combines the four different initializers $\mathbf{X}$. We selected the 25 dimensions d in each case. Each of the 100 instances per condition (initial vs. sparse overcomplete) was given to three judges.

Results in Table 5 confirm that the sparse overcomplete vectors are more interpretable than the dense vectors. The inter-annotator agreement on the sparse vectors increases substantially, from 57% to 71%, and the Fleiss' κ increases from "fair" to "moderate" agreement (Landis and Koch, 1977).

4.2 Qualitative Evaluation of Interpretability

If a vector dimension is interpretable, the top-ranking words for that dimension should display semantic or syntactic groupings. To verify this qualitatively, we select five dimensions with the highest variance of values in initial and sparsified GC vectors. We compare top-ranked words in the dimensions extracted from the two representations. The words are listed in Table 6, a dimension per row. Subjectively, we find the semantic groupings better in the sparse vectors than in the initial vectors.

Figure 3 visualizes the sparsified GC vectors for six words. The dimensions are sorted by the average value across the three "animal" vectors. The animal-related words use many of the same dimensions (102 common active dimensions out of 500 total); in constrast, the three city names use mostly distinct vectors.

$\mathbf{X}$	combat, guard, honor, bow, trim, naval 'll, could, faced, lacking, seriously, scored see, n't, recommended, depending, part due, positive, equal, focus, respect, better sergeant, comments, critics, she, videos
$\mathbf{A}$	fracture, breathing, wound, tissue, relief relationships, connections, identity, relations files, bills, titles, collections, poems, songs naval, industrial, technological, marine stadium, belt, championship, toll, ride, coach

Table 6: Top-ranked words per dimension for initial and sparsified GC representations. Each line shows words from a different dimension.

5 Related Work

To the best of our knowledge, there has been no prior work on obtaining overcomplete word vector representations that are sparse and categorical. However, overcomplete features have been widely used in image processing, computer vision (Olshausen and Field, 1997; Lewicki and Sejnowski, 2000) and signal processing (Donoho et al., 2006). Nonnegative matrix factorization is often used for interpretable coding of information (Lee and Seung, 1999; Liu et al., 2003; Cichocki et al., 2009).

Sparsity constraints are in general useful in NLP problems (Kazama and Tsujii, 2003; Friedman et al., 2008; Goodman, 2004), like POS tagging (Ganchev et al., 2009), dependency parsing (Martins et al., 2011), text classification (Yogatama and Smith, 2014), and representation learning (Bengio et al., 2013; Yogatama et al., 2015). Including sparsity constraints in Bayesian models of lexical semantics like LDA in the form of sparse Dirichlet priors has been shown to be useful for downstream tasks like POS-tagging (Toutanova and Johnson, 2007), and improving interpretation (Paul and Dredze, 2012; Zhu and Xing, 2012).

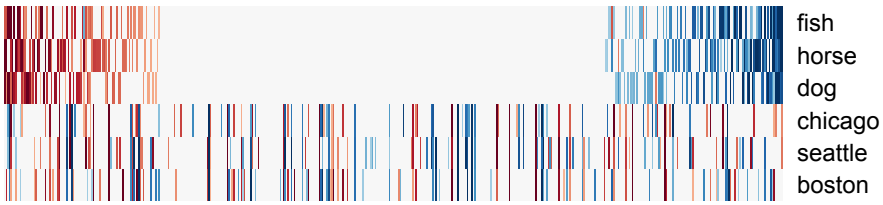

Figure 3: Visualization of sparsified GC vectors. Negative values are red, positive values are blue, zeroes are white.

6 Conclusion

We have presented a method that converts word vectors obtained using any state-of-the-art word vector model into sparse and optionally binary word vectors. These transformed vectors appear to come closer to features used in NLP tasks and outperform the original vectors from which they are derived on a suite of semantics and syntactic evaluation benchmarks. We also find that the sparse vectors are more interpretable than the dense vectors by humans according to a word intrusion detection test.

Acknowledgments

We thank Alona Fyshe for discussions on vector interpretability and three anonymous reviewers for their feedback. This research was supported in part by the National Science Foundation through grant IIS-1251131 and the Defense Advanced Research Projects Agency through grant FA87501420244. This work was supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF-10-1-0533.

A Initial Vector Representations (X)

Our experiments consider four publicly available collections of pre-trained word vectors. They vary in the amount of data used and the estimation method.

Glove. Global vectors for word representations (Pennington et al., 2014) are trained on aggregated global word-word co-occurrence statistics from a corpus. These vectors were trained on 6 billion words from Wikipedia and English Gigaword and are of length 300.[3]

Skip-Gram (SG). The word2vec tool (Mikolov et al., 2013) is fast and widely-used. In this model, each word's Huffman code is used as an input to a log-linear classifier with a continuous projection layer and words within a given context window are predicted. These vectors were trained on 100 billion words of Google news data and are of length 300.[4]

Global Context (GC). These vectors are learned using a recursive neural network that incorporates both local and global (document-level) context features (Huang et al., 2012). These vectors were trained on the first 1 billion words of English Wikipedia and are of length 50.[5]

Multilingual (Multi). Faruqui and Dyer (2014) learned vectors by first performing SVD on text in different languages, then applying canonical correlation analysis on pairs of vectors for words that align in parallel corpora. These vectors were trained on WMT-2011 news corpus containing 360 million words and are of length 48.[6]

B Evaluation Benchmarks

Our comparisons of word vector quality consider five benchmark tasks. We now describe the different evaluation benchmarks for word vectors.

Word Similarity. We evaluate our word representations on two word similarity tasks. The first is the WS-353 dataset (Finkelstein et al., 2001), which contains 353 pairs of English words that have been assigned similarity ratings by humans. This dataset is used to tune sparse vector learning hyperparameters (§2.5), while the remaining of the tasks discussed in this section are completely held out.

[3] http://www-nlp.stanford.edu/projects/glove/

[4] https://code.google.com/p/word2vec

[5] http://nlp.stanford.edu/~socherr/ACL2012_wordVectorsTextFile.zip

[6] http://cs.cmu.edu/~mfaruqui/soft.html

A more recent dataset, SimLex-999 (Hill et al., 2014), has been constructed to specifically focus on similarity (rather than relatedness). It contains a balanced set of noun, verb, and adjective pairs. We calculate cosine similarity between the vectors of two words forming a test item and report Spearman's rank correlation coefficient (Myers and Well, 1995) between the rankings produced by our model against the human rankings.

Sentiment Analysis (Senti). Socher et al. (2013) created a treebank of sentences annotated with fine-grained sentiment labels on phrases and sentences from movie review excerpts. The coarse-grained treebank of positive and negative classes has been split into training, development, and test datasets containing 6,920, 872, and 1,821 sentences, respectively. We use average of the word vectors of a given sentence as feature for classification. The classifier is tuned on the dev. set and accuracy is reported on the test set.

Question Classification (TREC). As an aid to question answering, a question may be classified as belonging to one of many question types. The TREC questions dataset involves six different question types, e.g., whether the question is about a location, about a person, or about some numeric information (Li and Roth, 2002). The training dataset consists of 5,452 labeled questions, and the test dataset consists of 500 questions. An average of the word vectors of the input question is used as features and accuracy is reported on the test set.

20 Newsgroup Dataset. We consider three binary categorization tasks from the 20 Newsgroups dataset.[7] Each task involves categorizing a document according to two related categories with training/dev./test split in accordance with Yogatama and Smith (2014): (1) Sports: baseball vs. hockey (958/239/796) (2) Comp.: IBM vs. Mac (929/239/777) (3) Religion: atheism vs. christian (870/209/717). We use average of the word vectors of a given sentence as features. The classifier is tuned on the dev. set and accuracy is reported on the test set.

NP bracketing (NP). Lazaridou et al. (2013) constructed a dataset from the Penn Treebank (Marcus et al., 1993) of noun phrases (NP) of length three words, where the first can be an adjective or a noun and the other two are nouns. The task is to predict the correct bracketing in the parse tree for a given noun phrase. For example, *local (phone company)* and *(blood pressure) medicine* exhibit *right* and *left* bracketing, respectively. We append the word vectors of the three words in the NP in order and use them as features for binary classification. The dataset contains 2,227 noun phrases split into 10 folds. The classifier is tuned on the first fold and cross-validation accuracy is reported on the remaining nine folds.

[7] http://qwone.com/~jason/20Newsgroups

References

Faiz A. Al-Khayyal and James E. Falk. 1983. Jointly constrained biconvex programming. *Mathematics of Operations Research*, pages 273–286.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of ACL*.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proc. of ACL*.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *NIPS*.

Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. 2009. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. John Wiley & Sons.

Mark Davies and Joseph L Fleiss. 1982. Measuring agreement for multinomial data. *Biometrics*, pages 1047–1051.

David L. Donoho, Michael Elad, and Vladimir N. Temlyakov. 2006. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1).

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report EECS-2010-24, University of California Berkeley.

John C. Duchi, Alekh Agarwal, and Martin J. Wainwright. 2012. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of EACL*.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proc. of NAACL*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proc. of WWW*.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Alona Fyshe, Partha P. Talukdar, Brian Murphy, and Tom M. Mitchell. 2014. Interpretable semantic vectors from a joint model of brain- and text- based meaning. In *Proc. of ACL*.

Alona Fyshe, Leila Wehbe, Partha P. Talukdar, Brian Murphy, and Tom M. Mitchell. 2015. A compositional and interpretable semantic space. In *Proc. of NAACL*.

Kuzman Ganchev, Ben Taskar, Fernando Pereira, and João Gama. 2009. Posterior vs. parameter sparsity in latent variable models. In *NIPS*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proc. of ICML*.

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *Proc. of NAACL*.

E. Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. arXiv:1304.5823.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proc. of EMNLP*.

Georg Heigold, Erik McDermott, Vincent Vanhoucke, Andrew Senior, and Michiel Bacchiani. 2014. Asynchronous stochastic optimization for sequence training of deep neural networks. In *Proc. of ICASSP*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456.

Patrik O. Hoyer. 2002. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proc. of IEEE Workshop on*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proc. of ACL*.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proc. of EMNLP*.

Douwe Kiela and Stephen Clark. 2013. Detecting compositionality of multi-word expressions using nearest neighbours in vector space models. In *Proc. of EMNLP*.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Angeliki Lazaridou, Eva Maria Vecchi, and Marco Baroni. 2013. Fish transporters and miracle homes: How compositional distributional semantics can help NP parsing. In *Proc. of EMNLP*.

Daniel D. Lee and H. Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. 2006. Efficient sparse coding algorithms. In *NIPS*.

Honglak Lee, Rajat Raina, Alex Teichman, and Andrew Y. Ng. 2009. Exponential family sparse coding with application to self-taught learning. In *Proc. of IJCAI*.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Michael Lewicki and Terrence Sejnowski. 2000. Learning overcomplete representations. *Neural Computation*, 12(2):337–365.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the ACL*, 1:179–192.

Mike Lewis and Mark Steedman. 2014. Combining formal and distributional models of temporal and intensional semantics. In *Proc. of ACL*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proc. of COLING*.

Weixiang Liu, Nanning Zheng, and Xiaofeng Lu. 2003. Non-negative matrix factorization for visual coding. In *Proc. of ICASSP*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Structured sparsity in structured prediction. In *Proc. of EMNLP*.

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

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proc. of COLING*.

Jerome L. Myers and Arnold D. Well. 1995. *Research Design & Statistical Analysis*. Routledge.

Bruno A. Olshausen and David J. Field. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311 – 3325.

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proc. of ACL*.

Michael Paul and Mark Dredze. 2012. Factorial LDA: Sparse multi-dimensional text models. In *NIPS*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*.

Karin Kipper Schuler. 2005. *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Kristina Toutanova and Mark Johnson. 2007. A bayesian lda-based model for semi-supervised part-of-speech tagging. In *NIPS*.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning : Vector space models of semantics. *JAIR*, 37(1):141–188.

Peter D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proc. of ECML*.

Eva Maria Vecchi, Roberto Zamparelli, and Marco Baroni. 2013. Studying the recursive behaviour of adjectival modification with compositional distributional semantics. In *Proc. of EMNLP*.

Lin Xiao. 2009. Dual averaging methods for regularized stochastic learning and online optimization. In *NIPS*.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proc. of CIKM*.

Dani Yogatama and Noah A Smith. 2014. Linguistic structured sparsity in text categorization. In *Proc. of ACL*.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A. Smith. 2015. Learning word representations with hierarchical sparse coding. In *Proc. of ICML*.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proc. of ACL*.

Jun Zhu and Eric P Xing. 2012. Sparse topical coding. arXiv:1202.3778.