6-2015

Learning Word Representations with Hierarchical Sparse Coding

Dani Yogatama
Carnegie Mellon University

Manaal Faruqui
Carnegie Mellon University

Chris Dyer
Carnegie Mellon University, cdyer@cs.cmu.edu

Noah A. Smith
Carnegie Mellon University, nasmith@cs.cmu.edu

Published In
Journal of Machine Learning Research : Workshop and Conference Proceedings, 37, 87-96.

Learning Word Representations with Hierarchical Sparse Coding

Dani Yogatama DYOGATAMA@CS.CMU.EDU
Manaal Faruqui MFARUQUI@CS.CMU.EDU
Chris Dyer CDYER@CS.CMU.EDU
Noah A. Smith NASMITH@CS.CMU.EDU
Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

We propose a new method for learning word representations using hierarchical regularization in sparse coding inspired by the linguistic study of word meanings. We show an efficient learning algorithm based on stochastic proximal methods that is significantly faster than previous approaches, making it possible to perform hierarchical sparse coding on a corpus of billions of word tokens. Experiments on various benchmark tasks—word similarity ranking, syntactic and semantic analogies, sentence completion, and sentiment analysis—demonstrate that the method outperforms or is competitive with state-of-the-art methods.

1. Introduction

When applying machine learning to text, the classic categorical representation of words as indices of a vocabulary fails to capture syntactic and semantic similarities that are easily discoverable in data (e.g., *pretty*, *beautiful*, and *lovely* have similar meanings, opposite to *unattractive*, *ugly*, and *repulsive*). In contrast, recent approaches to word representation learning apply neural networks to obtain low-dimensional, continuous embeddings of words (Bengio et al., 2003; Mnih & Teh, 2012; Collobert et al., 2011; Huang et al., 2012; Mikolov et al., 2010; 2013a; Pennington et al., 2014).

In this work, we propose an alternative approach based on decomposition of a high-dimensional matrix capturing surface statistics of association between a word and its "contexts" with sparse coding. As in past work, contexts are words that occur nearby in running text (Turney & Pantel, 2010). Learning is performed by minimizing a reconstruction loss function to find the best factorization of the input matrix.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

The key novelty in our method is to govern the relationships among dimensions of the learned word vectors, introducing a hierarchical organization imposed through a structured penalty known as the group lasso (Yuan & Lin, 2006). The idea of regulating the order in which variables enter a model was first proposed by Zhao et al. (2009), and it has since been shown useful for other applications (Jenatton et al., 2011). Our approach is motivated by coarse-to-fine organization of words' meanings often found in the field of lexical semantics (see §2.2 for a detailed description), which mirrors evidence for distributed nature of hierarchical concepts in the brain (Raposo et al., 2012). Related ideas have also been explored in syntax (Petrov & Klein, 2008). It also has a foundation in cognitive science, where hierarchical structures have been proposed as representations of semantic cognition (Collins & Quillian, 1969). We propose a stochastic proximal algorithm for hierarchical sparse coding that is suitable for problems where the input matrix is very large and sparse. Our algorithm enables application of hierarchical sparse coding to learn word representations from a corpus of billions of word tokens and 400,000 word types.

On standard evaluation tasks—word similarity ranking, analogies, sentence completion, and sentiment analysis—we find that our method outperforms or is competitive with the best published representations. Our word representations are available at: `http://www.ark.cs.cmu.edu/dyogatam/wordvecs/`.

2. Model

2.1. Background and Notation

The observable representation of word v is taken to be a vector $\mathbf{x}_v \in \mathbb{R}^C$ of cooccurrence statistics with C different contexts. Most commonly, each context is a possible neighboring word within a fixed window.[1] Following many

[1] Others include: global context (Huang et al., 2012), multilingual context (Faruqui & Dyer, 2014), geographic context (Bamman et al., 2014), brain activation (Fyshe et al., 2014), and second-order context (Schutze, 1998).

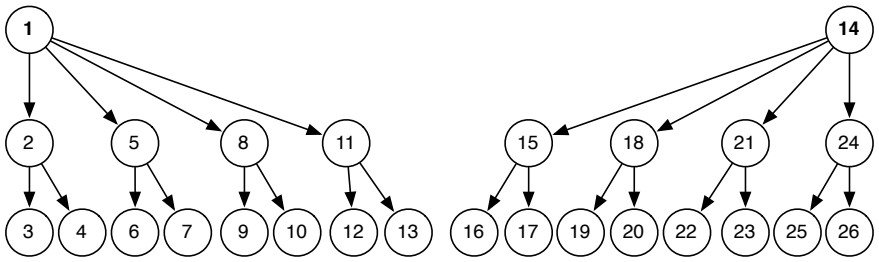


Figure 1. An example of a regularization forest that governs the order in which variables enter the model. In this example, 1 needs to be selected (nonzero) for $2, 3, \ldots, 13$ to be selected. However, $1, 2, \ldots, 13$ have nothing to do with the variables in the second tree: $14, 15, \ldots, 26$. See text for details.

others, we let $x_{v,c}$ be the pointwise mutual information (PMI) between the occurrence of context word c within a five-word window of an occurrence of word v (Turney & Pantel, 2010; Murphy et al., 2012; Faruqui & Dyer, 2014; Levy & Goldberg, 2014).

In sparse coding, the goal is to represent each input vector $\mathbf{x} \in \mathbb{R}^C$ as a sparse linear combination of basis vectors. Given a stacked input matrix $\mathbf{X} \in \mathbb{R}^{C \times V}$, where V is the number of words, we seek to minimize:

$$\arg\min_{\mathbf{D} \in \mathcal{D}, \mathbf{A}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \lambda \Omega(\mathbf{A}), \tag{1}$$

where $\mathbf{D} \in \mathbb{R}^{C \times M}$ is the dictionary of basis vectors, $\mathcal{D}$ is the set of matrices whose columns have small (e.g., less than or equal to one) ℓ_2 norm, $\mathbf{A} \in \mathbb{R}^{M \times V}$ is the code matrix (each column of $\mathbf{A}$ represents a word), 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). Note that it is not necessary, although typical, for M to be less than C (when $M > C$, it is often called an overcomplete representation). The most common regularizer is the ℓ_1 penalty, which results in sparse codes. While structured regularizers are associated with sparsity as well (e.g., the group lasso encourages group sparsity), our motivation is to use Ω to encourage a coarse-to-fine organization of latent dimensions of the learned representations of words.

2.2. Structured Regularization for Word Representations

For $\Omega(\mathbf{A})$, we design a forest-structured regularizer that encourages the model to use some dimensions in the code space before using other dimensions. Consider the trees in Figure 1. In this example, there are 13 variables in each tree, and 26 variables in total (i.e., $M = 26$), each corresponding to a latent dimension for *one particular word*. These trees describe the order in which variables "enter the model" (i.e., take nonzero values). In general, a node may take a nonzero value only if its ancestors also do. For example, nodes 3 and 4 may only be nonzero if nodes 1 and 2 are also nonzero. Our regularizer for column v of $\mathbf{A}$, denoted by $\mathbf{a}_v$ (in this example, $\mathbf{a}_v \in \mathbb{R}^{26}$), for the trees in Figure 1 is:[2]

$$\Omega(\mathbf{a}_v) = \sum_{i=1}^{26} \|[a_{v,i}, \ \mathbf{a}_{v,\text{Descendants}(i)}]\|_2$$

where $\text{Descendants}(i)$ returns the (possibly empty) set of descendants of node i, and $[.]$ returns the subvector of $\mathbf{a}_v$ by considering only $a_{v,i}$ and $\mathbf{a}_{v,\text{Descendants}(i)}$.[3] Jenatton et al. (2011) proposed a related penalty with only one tree for learning image and document representations.

In the following, we discuss why organizing the code space this way is helpful in learning better word representations. Recall that the goal is to have a good dictionary $\mathbf{D}$ and code matrix $\mathbf{A}$. We apply the structured penalty to each column of $\mathbf{A}$. When we use the same structured penalty for these columns, we encode an additional shared constraint that the dimensions of $\mathbf{a}_v$ that correspond to top level nodes should focus on "general" contexts that are present in most words. In our case, this corresponds to contexts with extreme PMI values for many words, since they are the ones that incur the largest losses. As we go down the trees, more word-specific contexts can then be captured. As a result, we have better organization *across* words when learning their representations, which also translates to a more structured dictionary $\mathbf{D}$. Contrast this with the case when we use unstructured regularizers that penalize each dimension of $\mathbf{A}$ independently (e.g., lasso). In this case, each dimension of $\mathbf{a}_v$ has more flexibility to pay attention to any contexts (the only constraint that we encode is that the cardinality of the model should be small). We hypothesize that this is less appropriate for learning word representations, since the model has excessive freedom when learning $\mathbf{A}$ on noisy PMI values, which translates to poor $\mathbf{D}$.

The intuitive motivation for our regularizer comes from the field of lexical semantics, which often seeks to capture the relationships between words' meanings in hierarchically-

[2] $\Omega(\mathbf{A})$ is computed by adding components of $\Omega(\mathbf{a}_v)$ for all columns of $\mathbf{A}$.

[3] Note that if $\|[a_{v,i}, \mathbf{a}_{v,\text{Descendants}(i)}]\|_2$ is below a regularization threshold ($a_{v,i}$ is a zero node), $\|[\mathbf{a}_{v,\text{Descendants}(i)}]\|_2$ is also below the threshold (all its descendants are zero nodes as well). Conversely, if $\|[a_{v,i}, \mathbf{a}_{v,\text{Descendants}(i)}]\|_2$ is above the threshold ($a_{v,i}$ is a nonzero node), $\|[a_{v,\text{Parent}(i)}, a_{v,i}, \mathbf{a}_{v,\text{Descendants}(i)}]\|_2$ is also above the threshold ($a_{v,\text{Parent}(i)}$ is also a nonzero node).

organized lexicons. The best-known example is WordNet (Miller, 1995). Words with the same (or close) meanings are grouped together (e.g., *professor* and *prof* are synonyms), and fine-grained meaning groups ("synsets") are nested under coarse-grained ones (e.g., *professor* is a hyponym of *academic*). Our hierarchical sparse coding approach is still several steps away from inducing such a lexicon, but it seeks to employ the dimensions of a distributed word representation scheme in a similar coarse-to-fine way. In cognitive science, such hierarchical organization of semantic representations was first proposed by Collins & Quillian (1969).

2.3. Learning

Learning is accomplished by minimizing the function in Eq. 1, with the group lasso regularization function described in §2.2. The function is not convex with respect to $\mathbf{D}$ and $\mathbf{A}$, but it is convex with respect to each when the other is fixed. Alternating minimization routines have been shown to work reasonably well in practice for such problems (Lee et al., 2007), but they are too expensive here due to:

- The size of $\mathbf{X} \in \mathbb{R}^{C \times V}$ (C and V are each on the order of 10^5).
- The many overlapping groups in the structured regularizer $\Omega(\mathbf{A})$.

One possible solution is based on the online dictionary learning method of Mairal et al. (2010). For T iterations, we:

- Sample a mini-batch of words and (in parallel) solve for each one's $\mathbf{a}$ using proximal methods or alternating directions method of multipliers, shown to work well for overlapping group lasso problems (Jenatton et al., 2011; Qin & Goldfarb, 2012; Yogatama & Smith, 2014).
- Update $\mathbf{D}$ using the block coordinate descent algorithm of Mairal et al. (2010).

Finally, we parallelize solving for all columns of $\mathbf{A}$, which are separable once $\mathbf{D}$ is fixed. In our experiments, we use this algorithm for a medium-sized corpus.

The main difficulty of learning word representations with hierarchical sparse coding is, again, that the size of the input matrix can be very large. When we use neighboring words as the contexts, the numbers of rows and columns are the size of the vocabulary. For a medium-sized corpus with hundreds of millions of word tokens, we typically have one or two hundred thousand unique words, so the above algorithm is still applicable. For a large corpus with billions of word tokens, this number can easily double or triple, making learning very expensive. We propose an alternative learning algorithm for such cases.

Algorithm 1 Fast algorithm for learning word representations with the forest regularizer.

Input: matrix $\mathbf{X}$, regularization constant λ and τ, learning rate sequences $\eta_0, \ldots, \eta_T$, number of iterations T
Initialize $\mathbf{D}_0$ and $\mathbf{A}_0$ randomly
for $t = 1, \ldots, T$ {can be parallelized, see text for details} **do**
  Sample $x_{c,v}$ with probability proportional to its (absolute) value
  $\mathbf{d}_c = \mathbf{d}_c + 2\eta_t(\mathbf{a}_v(x_{c,v} - \mathbf{d}_c \cdot \mathbf{a}_v) - \tau \mathbf{d}_c)$
  $\mathbf{a}_v = \mathbf{a}_v + 2\eta_t(\mathbf{d}_c(x_{c,v} - \mathbf{d}_c \cdot \mathbf{a}_v))$
  for $m = 1, \ldots, M$ **do**
    $\text{prox}_{\Omega_m,\lambda}(\mathbf{a}_v)$,
      where $\Omega_m = \|\langle a_{v,m},\ \mathbf{a}_{v,\text{Descendants}(m)}\rangle\|_2$
  end for
end for

We rewrite Eq. 1 as:

$$\arg\min_{\mathbf{D},\mathbf{A}} \sum_{c,v} (x_{c,v} - \mathbf{d}_c \cdot \mathbf{a}_v)^2 + \lambda\Omega(\mathbf{A}) + \tau \sum_m \|\mathbf{d}_m\|_2^2$$

where (abusing notation) $\mathbf{d}_c$ denotes the c-th row vector of $\mathbf{D}$ and $\mathbf{d}_m$ denotes the m-th column vector of $\mathbf{D}$ (recall that $\mathbf{D} \in \mathbb{R}^{C \times M}$). At each iteration, we sample an entry $x_{c,v}$ and perform gradient updates to the corresponding row $\mathbf{d}_c$ and column $\mathbf{a}_v$. Instead of considering all elements of the input matrix, our algorithm allows approximating the solution by using only some (e.g., nonzero) entries of the input matrix $\mathbf{X}$ if necessary.

We directly penalize columns of $\mathbf{D}$ by their squared ℓ_2 norm as an alternative to constraining columns of $\mathbf{D}$ to have unit ℓ_2 norm. The advantage of this transformation is that we have eliminated a projection step for columns of $\mathbf{D}$. Instead, we can include the gradient of the penalty term in the stochastic gradient update. We apply the proximal operator associated with $\Omega(\mathbf{a}_v)$ as a composition of elementary proximal operators with no group overlaps, similar to Jenatton et al. (2011). This can be done by recursively visiting each node of a tree and applying the proximal operator for the group lasso penalty associated with that node (i.e., the group lasso penalty where the node is the topmost node and the group consists of the node and all of its descendants). The proximal operator associated with node m, denoted by $\text{prox}_{\Omega_m,\lambda}$, is simply the block-thresholding operator for node m and all its descendants.

Since each entry $x_{c,v}$ only depends on $\mathbf{d}_c$ and $\mathbf{a}_v$, we can sample multiple entries and perform the updates in parallel as long as they do not share c and v. In our case, where C and V are on the order of hundreds of thousands and we only have tens or hundreds of processors, finding elements that do not violate this constraint is easy. For example, there are typically a huge number of nonzero entries (on

the order of billions). Using a sampling procedure that favors entries with higher (absolute) PMI values can lead to reasonably good word representations faster, so we can sample an entry with probability proportional to its absolute value.[4] Algorithm 1 summarizes our method.

2.4. Convergence Analysis

We analyze the convergence of Algorithm 1 for the basic setting where we uniformly sample elements of the input matrix $\mathbf{X}$. Similar to Mairal et al. (2010), we can rewrite our optimization problem as: $\arg\min_{\mathbf{A}} \sum_{t=1}^{T} \mathcal{L}^t(\mathbf{A}) + \lambda\Omega(\mathbf{A})$, where $\mathcal{L}^t(\mathbf{A}) = \|\mathbf{X} - \mathbf{D}^t\mathbf{A}\|_F^2 + \tau\sum_m \|\mathbf{d}_m^t\|_2^2$, and $\mathbf{D}^t = \mathbf{D}^{t-1} + 2\eta_t((\mathbf{X} - \mathbf{D}^{t-1}\mathbf{A}^{t-1})\mathbf{A}^{t-1\top} - \tau\mathbf{D}^{t-1})$. Note that $\mathcal{L}^t(\mathbf{A})$ is a nonconvex (with respect to $\mathbf{A}$) continuously differentiable function, which is the loss at timestep t after performing the dictionary update step. For ease of exposition, in the followings, we assume $\mathbf{A}$ is a vector formed by stacking together columns of the matrix $\mathbf{A}$.

Let us denote $\mathcal{L}(\mathbf{A}) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}^t(\mathbf{A})$. We show convergence of Algorithm 1 to a stationary point under the assumption that we have an unbiased estimate of the gradient with respect to $\mathbf{A}$: $\mathbb{E}[\bigtriangledown\mathcal{L}^t(\mathbf{A})] = \bigtriangledown\mathcal{L}(\mathbf{A})$. This can also be stated as $\mathbb{E}[\|\epsilon^t\|_2] = 0$, where $\|\epsilon^t\|_2 = \| \bigtriangledown \mathcal{L}(\mathbf{A}) - \bigtriangledown\mathcal{L}^t(\mathbf{A})\|_2$.

Our convergence proof uses the following definition of a stationary point and relies on a lemma from Sra (2012).

Definition 1. *A point $\mathbf{A}^*$ is a stationary point if and only if: $\mathbf{A}^* = prox_{\Omega,\lambda}(\mathbf{A}^* - \eta \bigtriangledown \mathcal{L}(\mathbf{A}^*))$.*

Lemma 2. *(Sra, 2012) Let F be a function with a (locally) Lipschitz continuous gradient with constant $L > 0$.*

$$F(\mathbf{A}^t) - F(\mathbf{A}^{t+1}) \geq \frac{2 - L\eta_t}{2\eta_t}\|\mathbf{A}^{t+1} - \mathbf{A}^t\|_2^2 - \|\epsilon^t\|_2\|\mathbf{A}^{t+1} - \mathbf{A}^t\|_2. \quad (2)$$

Theorem 3. *Let the assumption of an unbiased estimate of the gradient be satisfied and the learning rate satisfies $0 < \eta_t < \frac{2}{L}$. Algorithm 1 converges to a stationary point in expectation.*

Proof. We first show that $\mathcal{L}(\mathbf{A}^t) - \mathcal{L}(\mathbf{A}^{t+1})$ is bounded below in expectation. Since $\mathcal{L}$ has a Lipschitz continuous gradient, Lemma 2 already bounds $\mathcal{L}(\mathbf{A}^t) - \mathcal{L}(\mathbf{A}^{t+1})$. Let us denote the Lipschitz constant of $\mathcal{L}$ by L. Given our assumption about the error of the stochastic gradient (vanishing error), we have:

$$\mathbb{E}[\mathcal{L}(\mathbf{A}^t) - \mathcal{L}(\mathbf{A}^{t+1})] \geq \frac{2 - L\eta_t}{2\eta_t}\mathbb{E}[\|\mathbf{A}^{t+1} - \mathbf{A}^t\|_2^2]$$
$$= \frac{2 - L\eta_t}{2\eta_t}\mathbb{E}[\|\text{prox}_{\Omega,\lambda}(\mathbf{A}^t - \eta_t \bigtriangledown \mathcal{L}^t(\mathbf{A}^t)) - \mathbf{A}^t\|_2^2]$$

Since our learning rate satisfies $0 < \eta_t < \frac{2}{L}$, it is easy to show that the above is never negative. In order to show convergence, we then bound this quantity:

$$\sum_{t=1}^{T} \frac{2 - L\eta_t}{2\eta_t}\mathbb{E}[\|\text{prox}_{\Omega,\lambda}(\mathbf{A}^t - \eta_t \bigtriangledown \mathcal{L}^t(\mathbf{A}^t)) - \mathbf{A}^t\|_2^2]$$
$$\leq \sum_{t=1}^{T} \mathbb{E}[\mathcal{L}(\mathbf{A}^t) - \mathcal{L}(\mathbf{A}^{t+1})] = \mathbb{E}[\mathcal{L}(\mathbf{A}^1) - \mathcal{L}(\mathbf{A}_{T+1})]$$
$$\leq \mathbb{E}[\mathcal{L}(\mathbf{A}^1) - \mathcal{L}(\mathbf{A}^*)]$$

The right hand side (third line) is a positive constant and the left hand side (first line) is never negative, so $\mathbb{E}[\|\text{prox}_{\Omega,\lambda}(\mathbf{A}^t - \eta_t \bigtriangledown \mathcal{L}^t(\mathbf{A}^t)) - \mathbf{A}^t\|_2^2] \to 0$, which means that $\mathbf{A}_t$ converges to a stationary point in expectation based on the definition of a stationary point in Definition 1. □

3. Experiments

We present a controlled comparison of the forest regularizer against several strong baseline word representations learned on a fixed dataset, across several tasks. In §3.4 we compare to publicly available word vectors trained on different data.

3.1. Setup and Baselines

We use the WMT-2011 English news corpus as our training data.[5] The corpus contains about 15 million sentences and 370 million words. The size of our vocabulary is 180,834.[6]

In our experiments, we use forests similar to those in Figure 1 to organize the latent word space. Note that the example has 26 nodes (2 trees). We choose to evaluate performance with $M = 52$ (4 trees) and $M = 520$ (40 trees).[7] We denote the sparse coding method with regular ℓ_1 penalty by SC, and our method with structured regularization (§2.2) by FOREST. We set $\lambda = 0.1$. In this first set of experiments with a medium-sized corpus, we use the online learning algorithm of Mairal et al. (2010).

We compare with the following baseline methods:

- Turney & Pantel (2010): principal component analysis (PCA) by truncated singular value decomposition on

[4] In practice, we can use an even faster approximation of this sampling procedure by uniformly sampling a nonzero entry and multiplying its gradient by a scaling constant proportional to its absolute PMI value.

[5] http://www.statmt.org/wmt11/

[6] We replace words with frequency less than 10 with #rare# and numbers with #number#.

[7] In preliminary experiments we explored binary tree structures and found they did not work as well. We leave a more extensive exploration of tree structures to future work.

$\mathbf{X}^\top$. Note that this is also the same as minimizing the squared reconstruction loss in Eq. 1 without any penalty on $\mathbf{A}$.

- Mikolov et al. (2010): a recursive neural network (RNN) language model. We obtain an implementation from `http://rnnlm.org/`.
- Mnih & Teh (2012): a log bilinear model that predicts a word given its context, trained using noise-contrastive estimation (NCE, Gutmann & Hyvarinen, 2010). We use our own implementation for this model.
- Mikolov et al. (2013a): a log bilinear model that predicts a word given its context (continuous bag of words, CBOW), trained using negative sampling (Mikolov et al., 2013b). We obtain an implementation from `https://code.google.com/p/word2vec/`.
- Mikolov et al. (2013a): a log bilinear model that predicts context words given a target word (skip gram, SG), trained using negative sampling (Mikolov et al., 2013b). We obtain an implementation from `https://code.google.com/p/word2vec/`.
- Pennington et al. (2014): a log bilinear model that is trained using AdaGrad (Duchi et al., 2011) to minimize a weighted square error on global (log) cooccurrence counts (global vectors, GV). We obtain an implementation from `http://nlp.stanford.edu/projects/glove/`.

Our focus here is on comparisons of model architectures. For a fair comparison, we train all competing methods on the same corpus using a context window of five words (left and right). For the baseline methods, we use default settings in the provided implementations (or papers, when implementations are not available and we reimplement the methods). We also trained the two baseline methods introduced by Mikolov et al. (2013a) with hierarchical softmax using a binary Huffman tree instead of negative sampling; consistent with Mikolov et al. (2013b), we found that negative sampling performs better and relegate hierarchical softmax results to supplementary materials.

3.2. Evaluation

We evaluate on the following benchmark tasks.

Word similarity. The first task evaluates how well the representations capture word similarity. For example *beautiful* and *lovely* should be closer in distance than *beautiful* and *science*. We evaluate on a suite of word similarity datasets, subsets of which have been considered in past work: WordSim 353 (Finkelstein et al., 2002), rare words (Luong et al., 2013), and many others; see supplementary materials for details. Following standard practice, for each competing model, we compute cosine distances between word pairs in word similarity datasets, then rank and report Spearman's rank correlation coefficient (Spearman, 1904) between the model's rankings and human rankings.

Syntactic and semantic analogies. The second evaluation dataset is two analogy tasks proposed by Mikolov et al. (2013a). These questions evaluate syntactic and semantic relations between words. There are 10,675 syntactic questions (e.g., *walking : walked :: swimming : swam*) and 8,869 semantic questions (e.g., *Athens : Greece :: Oslo :: Norway*). In each question, one word is missing, and the task is to correctly predict the missing word. We use the vector offset method (Mikolov et al., 2013a) that computes the vector $\mathbf{b} = \mathbf{a}_{\text{Athens}} - \mathbf{a}_{\text{Greece}} + \mathbf{a}_{\text{Oslo}}$. We only consider a question to be answered correctly if the returned vector ($\mathbf{b}$) has the highest cosine similarity to the correct answer (in this example, $\mathbf{a}_{\text{Norway}}$).

Sentence completion. The third evaluation task is the Microsoft Research sentence completion challenge (Zweig & Burges, 2011). In this task, the goal it to choose from a set of five candidate words which one best completes a sentence. For example: *Was she his {client, musings, discomfiture, choice, opportunity}, his friend, or his mistress?* (*client* is the correct answer). We choose the candidate with the highest average similarity to every other word in the sentence.[8]

Sentiment analysis. The last evaluation task is sentence-level sentiment analysis. We use the movie reviews dataset from Socher et al. (2013). The dataset consists of 6,920 sentences for training, 872 sentences for development, and 1,821 sentences for testing. We train ℓ_2-regularized logistic regression to predict binary sentiment, tuning the regularization strength on development data. We represent each example (sentence) as an M-dimensional vector constructed by taking the average of word representations of words appearing in that sentence.

The analogy, sentence completion, and sentiment analysis tasks are evaluated on prediction accuracy.

3.3. Results

Table 1 shows results on all evaluation tasks for $M = 52$ and $M = 520$. Runtime will be discussed in §3.5. In the similarity ranking and sentiment analysis tasks, our method performed the best in both low and high dimensional em-

[8] We note that unlike matrix decomposition based approaches, some of the neural network based models can directly compute the scores of context words given a possible answer (Mikolov et al., 2013a). We choose to use average similarities for a fair comparison of the representations.

Table 1. Summary of results. We report Spearman's correlation coefficient for the word similarity task and accuracies (%) for other tasks. Higher values are better (higher correlation coefficient or higher accuracy). The last two methods (columns) are new to this paper, and our proposed method is in the last column.

M	Task	PCA	RNN	NCE	CBOW	SG	GV	SC	FOREST
52	Word similarity	0.39	0.26	0.48	0.43	0.49	0.43	0.49	**0.52**
	Syntactic analogies	18.88	10.77	24.83	23.80	26.69	**27.40**	11.84	24.38
	Semantic analogies	8.39	2.84	25.29	8.45	19.49	**26.23**	4.50	9.86
	Sentence completion	27.69	21.31	**30.18**	25.60	26.89	25.10	25.10	28.88
	Sentiment analysis	74.46	64.85	70.84	68.48	71.99	72.60	75.51	**75.83**
520	Word similarity	0.50	0.31	0.59	0.53	0.58	0.51	0.58	**0.66**
	Syntactic analogies	40.67	22.39	33.49	52.20	**54.64**	44.96	22.02	48.00
	Semantic analogies	28.82	5.37	**62.76**	12.58	39.15	55.22	15.46	41.33
	Sentence completion	30.58	23.11	33.07	26.69	26.00	33.76	28.59	**35.86**
	Sentiment analysis	81.70	72.97	78.60	77.38	79.46	79.40	78.20	**81.90**

Table 2. Results on the syntactic and semantic analogies tasks with a bigger corpus ($M = 260$).

Task	CBOW	SG	GV	FOREST
Syntactic	61.37	63.61	65.56	**65.63**
Semantic	23.13	54.41	**74.35**	52.88

beddings. In the sentence completion challenge, our method performed best in the high-dimensional case and second-best in the low-dimensional case. Importantly, FOREST outperforms PCA and unstructured sparse coding (SC) on every task. We take this collection of results as support for the idea that coarse-to-fine organization of latent dimensions of word representations captures the relationships between words' meanings better than unstructured organization.

Analogies Our results on the syntactic and semantic analogies tasks for all models are below state-of-the-art performance from previous work. We hypothesize that this is because performing well on these tasks requires training on a bigger corpus (in general, the bigger the training corpus is, the better the models will be for all tasks). We combine our WMT-2011 corpus with other news corpora and Wikipedia to obtain a corpus of 6.8 billion words. The size of the vocabulary of this corpus is 401,150. We retrain four models that are scalable to a corpus of this size: CBOW, SG, GV, and FOREST.[9] We select $M = 260$ to balance the trade-off between training time and performance ($M = 52$ does not perform as well, and $M = 520$ is computationally expensive). For FOREST, we use the fast learning algorithm in §2.3, since the online learning algorithm of Mairal et al. (2010) does not scale to a problem of this size. We report accuracies on the syntactic and semantic analogies tasks in Table 2. All models benefit significantly from a bigger corpus, and the performance levels are now comparable with previous work. On the syntactic analogies task, FOREST is competitive with GV and both models outperformed SG and CBOW. On the semantic analogies task, GV outperformed SG, FOREST, and CBOW.

[9]Our NCE implementation is not optimized and therefore not scalable.

3.4. Other Comparisons

In Table 3, we compare with five other baseline methods for which we do not train on our training data but pre-trained 50-dimensional word representations are available:

- Collobert et al. (2011): a neural network language model trained on Wikipedia data for 2 months (CW).[10]
- Huang et al. (2012): a neural network model that uses additional global document context (RNN-DC).[11]
- Mnih & Hinton (2008): a log bilinear model that predicts a word given its context, trained using hierarchical softmax (HLBL).[12]
- Murphy et al. (2012): a word representation trained using non-negative sparse embedding (NNSE) on dependency relations and document cooccurrence counts.[13] These vectors were learned using sparse coding, but using different contexts (dependency and document cooccurrences), a different training method, and with a nonnegativity constraint. Importantly, there is no hierarchy in the code space, as in FOREST.[14]
- Lebret & Collobert (2014): a word representation trained using Hellinger PCA (HPCA).[15]

These methods were all trained on different corpora, so they have different vocabularies that do not always include all

[10]http://ronan.collobert.com/senna/

[11]http://goo.gl/Wujc5G

[12]http://metaoptimize.com/projects/wordreprs/ (Turian et al., 2010)

[13]http://www.cs.cmu.edu/~bmurphy/NNSE/.

[14]We found that NNSE trained using our contexts performed very poorly; see supplementary materials.

[15]http://lebret.ch/words/

Table 3. Comparison to previously published word representations. The five right-most columns correspond to the tasks described above; parenthesized values are the number of in-vocabulary items that could be evaluated.

Models	M	V	W. Sim.	Syntactic	Semantic	Sentence	Sentiment
CW	50	130,000	(6,225) 0.51	(10,427) 12.34	(8,656) 9.33	(976) 24.59	69.36
RNN-DC		100,232	(6,137) 0.32	(10,349) 10.94	(7,853) 2.60	(964) 19.81	67.76
HLBL		246,122	(6,178) 0.11	(10,477) 8.98	(8,446) 1.74	(990) 19.90	62.33
NNSE		34,107	(3,878) 0.23	(5,114) 1.47	(1,461) 2.46	(833) 0.04	64.80
HPCA		178,080	(6,405) 0.29	(10,553) 10.42	(8,869) 3.36	(993) 20.14	67.49
FOREST	52	180,834	(6,525) **0.52**	(10,675) **24.38**	(8,733) **9.86**	(1,004) **28.88**	**75.83**

Table 4. Training time comparisons for skip gram (SG), glove (GV), and FOREST. For the medium corpus (370 million words), we learn FOREST with Mairal et al. (2010). This algorithm consists of two major steps: online learning of $\mathbf{D}$ and parallel learning of $\mathbf{A}$ with fixed $\mathbf{D}$ (see §2.3). "$*$" indicates that we only use 640 cores for the second step, since the first step only takes less than 2 hours even for $M = 520$. We can also see from this table that it becomes too expensive to use this algorithm for a bigger corpus. For the bigger corpus (6.8 billion words), we use Algorithm 1.

Models	Corpus	M	Cores	Time
SG	370M	52	16	1.5 hours
GV	370M	52	16	0.4 hours
FOREST	370M	52	640*	2.5 hours
SG	370M	520	16	5 hours
GV	370M	520	16	2.5 hours
FOREST	370M	520	640*	20 hours
SG	6.8B	260	16	6.5 hours
GV	6.8B	260	16	4 hours
FOREST	6.8B	260	16	4 hours

of the words found in the tasks. We estimate performance on the items for which prediction is possible, and show the count for each method in Table 3. This comparison should be interpreted cautiously since many experimental variables are conflated; nonetheless, FOREST performs strongly.

3.5. Discussion

In terms of running time, FOREST is reasonably fast to learn. See Table 4 for comparisons with other state-of-the-art methods.

Our method produces sparse word representations with exact zeros. We observe that the sparse coding method without a structured regularizer produces sparser representations, but it performs worse on our evaluation tasks, indicating that it zeroes out meaningful dimensions. For FOREST with $M = 52$ and $M = 520$, the average numbers of nonzero entries are 91% and 85% respectively. While our word representations are not extremely sparse, this makes intuitive sense since we try to represent about 180,000 contexts in only 52 (520) dimensions. We also did not tune λ. As we increase M, we get sparser representations.

We visualize our $M = 52$ word representations (FOREST) related to animals (10 words) and countries (10 words). We show the coefficient patterns for these words in Figure 2. We can see that in both cases, there are dimensions where the coefficient signs (positive or negative) agree for all 10 words (they are mostly on the right and left sides of the plots). Note that the dimensions where all the coefficients agree are not the same in animals and countries. The larger magnitude of the vectors for more abstract concepts (*animal*, *animals*, *country*, *countries*) is reminiscent of neural imaging studies that have found evidence of more global activation patterns for processing superordinate terms (Raposo et al., 2012). In Figure 3, we show tree visualizations of coefficients of word representations for *animal*, *horse*, and *elephant*. We show one tree for $M = 52$ (there are four trees in total, but other trees exhibit similar patterns). Coefficients that differ in sign mostly correspond to leaf nodes, validating our motivation that top level nodes should focus more on "general" contexts (for which they should be roughly similar for *animal*, *horse*, and *elephant*) and leaf nodes focus on word-specific contexts. One of the leaf nodes for animal is driven to zero, suggesting that more abstract concepts require fewer dimensions to explain.

For FOREST and SG with $M = 520$, we project the learned word representations into two dimensions using the t-SNE tool (van der Maaten & Hinton, 2008).[16] We show projections of words related to the concept *good* vs. *bad* in Figure 4.[17] See supplementary materials for *man* vs. *woman*, as well as 2-dimensional projections of NCE.

4. Conclusion

We introduced a new method for learning word representations based on hierarchical sparse coding. The regularizer encourages hierarchical organization of the latent dimensions of vector-space word embeddings. We showed that our method outperforms state-of-the-art methods on word similarity ranking, sentence completion, syntactic analogies, and sentiment analysis tasks.

[16] http://homepage.tudelft.nl/19j49/t-SNE.html

[17] Since t-SNE is a non-convex method, we run it 10 times and choose the plots with the lowest t-SNE error.

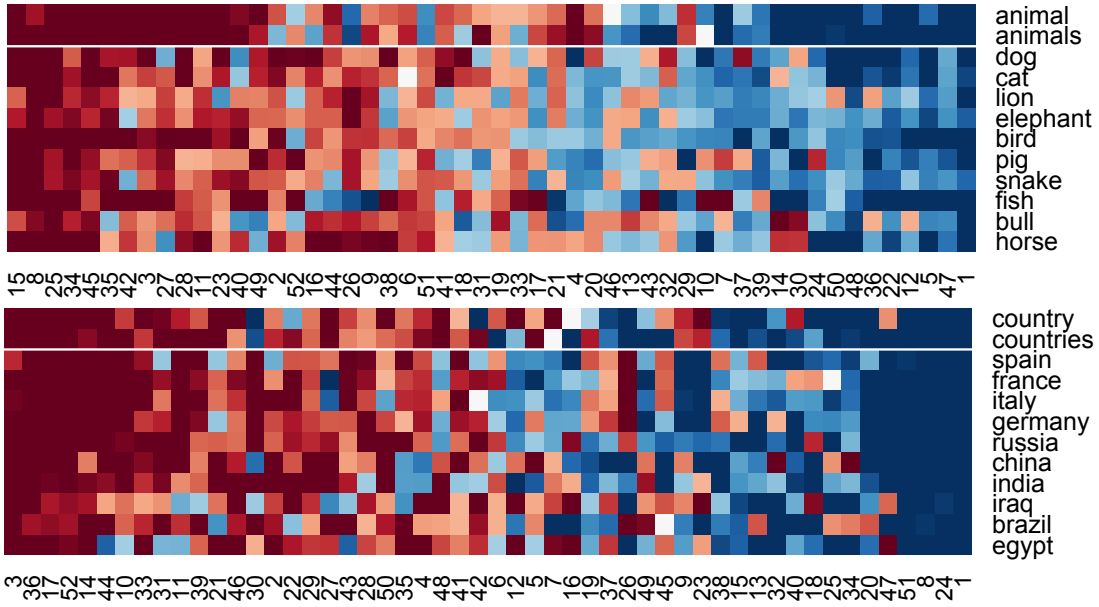
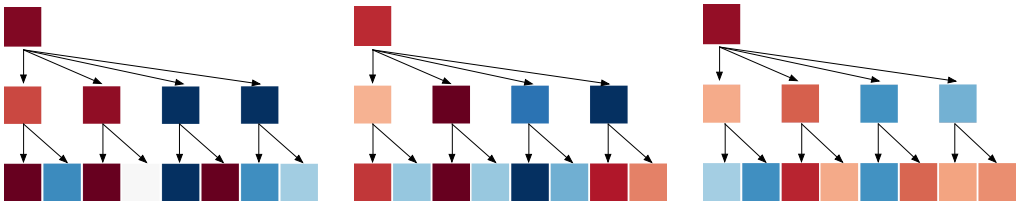


Figure 2. Heatmaps of word representations for 10 animals (top) and 10 countries (bottom) for $M = 52$ from FOREST. Red indicates negative values, blue indicates positive values (darker colors correspond to more extreme values); white denotes exact zero. The x-axis shows the original dimension index, we show the dimensions from the most negative (left) to the most positive (right), within each block, for readability.



(a) animal (b) horse (c) elephant

Figure 3. Tree visualizations of word representations for *animal* (left), *horse* (center), *elephant* (right) for $M = 52$. We use the same color coding scheme as in Figure 2. Here, we only show one tree (out of four), but other trees exhibit similar patterns.

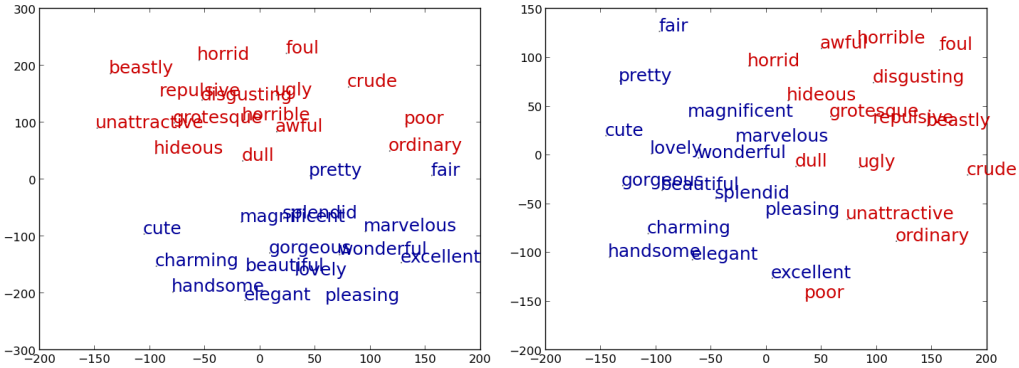


Figure 4. Two dimensional projections of the FOREST (left) and SG (right) word representations using the t-SNE tool (van der Maaten & Hinton, 2008). Words associated with "good" are colored in blue, words associated with "bad" are colored in red. We can see that in both cases most "good" and "bad" words are clustered together (in fact, they are linearly separated in the 2D space), except for *poor* in the SG case. See supplementary materials for more examples.

Acknowledgements

The authors thank Francis Bach for helpful discussions; anonymous reviewers, Sam Thomson, Bryan R. Routledge, Jesse Dodge, and Fei Liu for feedback on an earlier draft of this paper. This work was supported by the National Science Foundation through grant IIS-1352440, the Defense Advanced Research Projects Agency through grant FA87501420244, and computing resources provided by Google and the Pittsburgh Supercomputing Center.

References

Bamman, David, Dyer, Chris, and Smith, Noah A. Distributed representations of situated language. In *Proc. of ACL*, 2014.

Bengio, Yoshua, Ducharme, Rejean, Vincent, Pascal, and Jauvin, Chrstian. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Collins, Allan M. and Quillian, M. Ross. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behaviour*, 8:240–247, 1969.

Collobert, Ronan, Weston, Jason, Bottou, Leon, Karlen, Michael, Kavukcuoglu, Koray, and Kuska, Pavel. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2461–2505, 2011.

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

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

Finkelstein, Lev, Gabrilovich, Evgeniy, Matias, Yossi, Rivlin, Ehud, Solan, Zach, Wolfman, Gadi, and Ruppin, Eytan. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131, 2002.

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

Gutmann, Michael and Hyvarinen, Aapo. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. of AISTATS*, 2010.

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

Jenatton, Rodolphe, Mairal, Julien, Obozinski, Gullaume, and Bach, Francis. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12:2297–2334, 2011.

Lebret, Remi and Collobert, Ronan. Word embeddings through hellinger PCA. In *Proc. of EACL*, 2014.

Lee, Honglak, Battle, Alexis, Raina, Rajat, and Ng, Andrew Y. Efficient sparse coding algorithms. In *Proc. of NIPS*, 2007.

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

Levy, Omar and Goldberg, Yoav. Neural word embeddings as implicit matrix factorization. In *Proc. of NIPS*, 2014.

Luong, Minh-Thang, Socher, Richard, and Manning, Christopher D. Better word representations with recursive neural networks for morphology. In *Proc. of CONLL*, 2013.

Mairal, Julien, Bach, Francis, Ponce, Jean, and Sapiro, Guillermo. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

Mikolov, Tomas, Martin, Karafiat, Burget, Lukas, Cernocky, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In *Proc. of Interspeech*, 2010.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. In *Proc. of ICLR Workshop*, 2013a.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, 2013b.

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

Mnih, Andriy and Hinton, Geoffrey. A scalable hierarchical distributed language model. In *Proc. of NIPS*, 2008.

Mnih, Andriy and Teh, Yee Whye. A fast and simple algorithm for training neural probabilistic language models. In *Proc. of ICML*, 2012.

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

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

Petrov, S. and Klein, D. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proc. of EMNLP*, 2008.

Qin, Zhiwei (Tony) and Goldfarb, Donald. Structured sparsity via alternating direction methods. *Journal of Machine Learning Research*, 13:1435–1468, 2012.

Raposo, Ana, Mendes, Mafalda, and Marques, J. Frederico. The hierarchical organization of semantic memory: Executive function in the processing of superordinate concepts. *NeuroImage*, 59:1870–1878, 2012.

Schutze, Hinrich. Automatic word sense discrimination. *Computational Linguistics - Special issue on word sense disambiguation*, 24(1):97–123, 1998.

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

Spearman, Charles. The proof and measurement of association between two things. *The American Journal of Psychology*, 15:72–101, 1904.

Sra, Suvrit. Scalable nonconvex inexact proximal splitting. In *Proc. of NIPS*, 2012.

Turian, Joseph, Ratinov, Lev, and Bengio, Yoshua. Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL*, 2010.

Turney, Peter D. and Pantel, Patrick. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

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

Yogatama, Dani and Smith, Noah A. Making the most of bag of words: Sentence regularization with alternating direction method of multipliers. In *Proc. of ICML*, 2014.

Yuan, Ming and Lin, Yi. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68(1):49–67, 2006.

Zhao, Peng, Rocha, Gulherme, and Yu, Bin. The composite and absolute penalties for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497, 2009.

Zweig, Geoffry and Burges, Christopher J. C. The microsoft research sentence completion challenge. Technical report, Microsoft Research Technical Report MSR-TR-2011-129, 2011.