Deep Unordered Composition Rivals Syntactic Methods for Text Classification

Mohit Iyyer,[1] **Varun Manjunatha,**[1] **Jordan Boyd-Graber,**[2] **Hal Daumé III**[1]
[1]University of Maryland, Department of Computer Science and UMIACS
[2]University of Colorado, Department of Computer Science
{miyyer,varunm,hal}@umiacs.umd.edu, Jordan.Boyd.Graber@colorado.edu

Abstract

Many existing deep learning models for natural language processing tasks focus on learning the *compositionality* of their inputs, which requires many expensive computations. We present a simple deep neural network that competes with and, in some cases, outperforms such models on sentiment analysis and factoid question answering tasks while taking only a fraction of the training time. While our model is syntactically-ignorant, we show significant improvements over previous bag-of-words models by deepening our network and applying a novel variant of dropout. Moreover, our model performs better than syntactic models on datasets with high syntactic variance. We show that our model makes similar errors to syntactically-aware models, indicating that for the tasks we consider, nonlinearly transforming the input is more important than tailoring a network to incorporate word order and syntax.

1 Introduction

Vector space models for natural language processing (NLP) represent words using low dimensional vectors called embeddings. To apply vector space models to sentences or documents, one must first select an appropriate *composition function*, which is a mathematical process for combining multiple words into a single vector.

Composition functions fall into two classes: *unordered* and *syntactic*. Unordered functions treat input texts as bags of word embeddings, while syntactic functions take word order and sentence structure into account. Previously published experimental results have shown that syntactic functions outperform unordered functions on many tasks (Socher et al., 2013b; Kalchbrenner and Blunsom, 2013).

However, there is a tradeoff: syntactic functions require more training time than unordered composition functions and are prohibitively expensive in the case of huge datasets or limited computing resources. For example, the recursive neural network (Section 2) computes costly matrix/tensor products and nonlinearities at every node of a syntactic parse tree, which limits it to smaller datasets that can be reliably parsed.

We introduce a deep unordered model that obtains near state-of-the-art accuracies on a variety of sentence and document-level tasks with just minutes of training time on an average laptop computer. This model, the deep averaging network (**DAN**), works in three simple steps:

1. take the vector average of the embeddings associated with an input sequence of tokens
2. pass that average through one or more feed-forward layers
3. perform (linear) classification on the final layer's representation

The model can be improved by applying a novel dropout-inspired regularizer: for each training instance, randomly drop some of the tokens' embeddings before computing the average.

We evaluate **DAN**s on sentiment analysis and factoid question answering tasks at both the sentence and document level in Section 4. Our model's successes demonstrate that for these tasks, the choice of composition function is not as important as initializing with pretrained embeddings and using a deep network. Furthermore, **DAN**s, unlike more complex composition functions, can be effectively trained on data that have high syntactic variance. A qualitative analysis of the learned layers suggests that the model works by magnifying tiny but meaningful differences in the vector average through multiple hidden layers, and a detailed error analysis shows that syntactically-aware models actually make very similar errors to those of the more naïve DAN.

2 Unordered vs. Syntactic Composition

Our goal is to marry the speed of unordered functions with the accuracy of syntactic functions. In this section, we first describe a class of unordered composition functions dubbed “neural bag-of-words models” (NBOW). We then explore more complex syntactic functions designed to avoid many of the pitfalls associated with NBOW models. Finally, we present the deep averaging network (DAN), which stacks nonlinear layers over the traditional NBOW model and achieves performance on par with or better than that of syntactic functions.

2.1 Neural Bag-of-Words Models

For simplicity, consider text classification: map an input sequence of tokens X to one of k labels. We first apply a composition function g to the sequence of word embeddings $\boldsymbol{v}_w$ for $w \in X$. The output of this composition function is a vector $\boldsymbol{z}$ that serves as input to a logistic regression function.

In our instantiation of NBOW, g averages word embeddings[1]

$$\boldsymbol{z} = g(w \in X) = \frac{1}{|X|} \sum_{w \in X} \boldsymbol{v}_w. \qquad (1)$$

Feeding $\boldsymbol{z}$ to a softmax layer induces estimated probabilities for each output label

$$\hat{y} = \text{softmax}(\mathbf{W}_s \cdot \boldsymbol{z} + \boldsymbol{b}), \qquad (2)$$

where the softmax function is

$$\text{softmax}(\boldsymbol{q}) = \frac{\exp \boldsymbol{q}}{\sum_{j=1}^{k} \exp \boldsymbol{q}_j} \qquad (3)$$

$\mathbf{W}_s$ is a $k \times d$ matrix for a dataset with k output labels, and $\boldsymbol{b}$ is a bias term.

We train the NBOW model to minimize cross-entropy error, which for a single training instance with ground-truth label y is

$$\ell(\hat{y}) = \sum_{p=1}^{k} y_p \log(\hat{y}_p). \qquad (4)$$

Before we describe our deep extension of the NBOW model, we take a quick detour to discuss syntactic composition functions. Connections to other representation frameworks are discussed further in Section 4.

2.2 Considering Syntax for Composition

Given a sentence like “You’ll be more entertained getting hit by a bus”, an unordered model like NBOW might be deceived by the word “entertained” to return a positive prediction. In contrast, syntactic composition functions rely on the order and structure of the input to learn how one word or phrase affects another, sacrificing computational efficiency in the process. In subsequent sections, we argue that this complexity is not matched by a corresponding gain in performance.

Recursive neural networks (RecNNs) are syntactic functions that rely on natural language’s inherent structure to achieve state-of-the-art accuracies on sentiment analysis tasks (Tai et al., 2015). As in NBOW, each word type has an associated embedding. However, the composition function g now depends on a *parse tree* of the input sequence. The representation for any internal node in a binary parse tree is computed as a nonlinear function of the representations of its children (Figure 1, left). A more powerful RecNN variant is the recursive neural tensor network (RecNTN), which modifies g to include a costly tensor product (Socher et al., 2013b).

While RecNNs can model complex linguistic phenomena like negation (Hermann et al., 2013), they require much more training time than NBOW models. The nonlinearities and matrix/tensor products at each node of the parse tree are expensive, especially as model dimensionality increases. RecNNs also require an error signal at *every* node. One root softmax is not strong enough for the model to learn compositional relations and leads to worse accuracies than standard bag-of-words models (Li, 2014). Finally, RecNNs require relatively consistent syntax between training and test data due to their reliance on parse trees and thus cannot effectively incorporate out-of-domain data, as we show in our question-answering experiments. Kim (2014) shows that some of these issues can be avoided by using a convolutional network instead of a RecNN, but the computational complexity increases even further (see Section 4 for runtime comparisons).

What contributes most to the power of syntactic

[1] Preliminary experiments indicate that averaging outperforms the vector sum used in NBOW from Kalchbrenner et al. (2014).

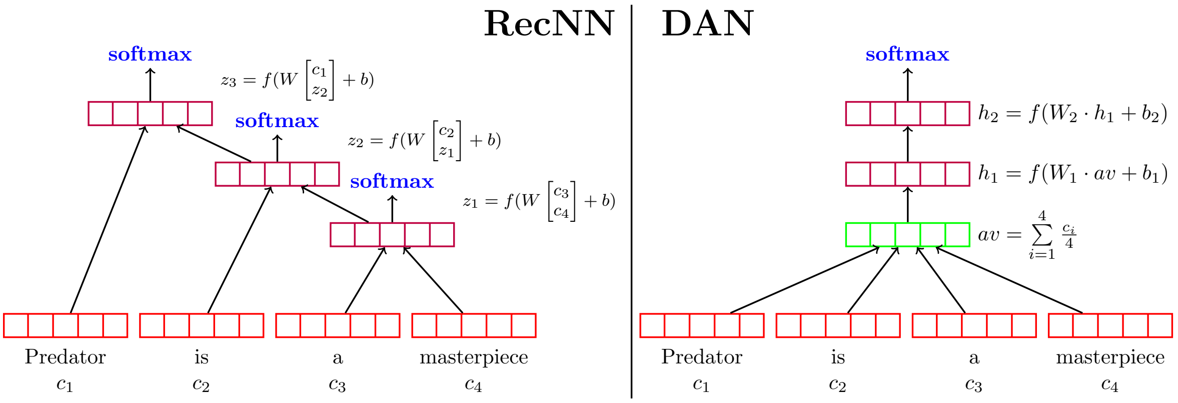
RecNN
DAN
softmax
$z_3 = f(W \begin{bmatrix} c_1 \\ z_2 \end{bmatrix} + b)$
softmax
$z_2 = f(W \begin{bmatrix} c_2 \\ z_1 \end{bmatrix} + b)$
softmax
$z_1 = f(W \begin{bmatrix} c_3 \\ c_4 \end{bmatrix} + b)$
Predator c_1
is c_2
a c_3
masterpiece c_4
softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator c_1
is c_2
a c_3
masterpiece c_4

Figure 1: On the left, a RecNN is given an input sentence for sentiment classification. Softmax layers are placed above every internal node to avoid vanishing gradient issues. On the right is a two-layer DAN taking the same input. While the RecNN has to compute a nonlinear representation (purple vectors) for every node in the parse tree of its input, this DAN only computes two nonlinear layers for every possible input.

functions: the compositionality or the nonlinearities? Socher et al. (2013b) report that removing the nonlinearities from their RecNN models drops performance on the Stanford Sentiment Treebank by over 5% absolute accuracy. Most unordered functions are linear mappings between bag-of-words features and output labels, so might they suffer from the same issue? To isolate the effects of syntactic composition from the nonlinear transformations that are crucial to RecNN performance, we investigate how well a deep version of the NBOW model performs on tasks that have recently been dominated by syntactically-aware models.

3 Deep Averaging Networks

The intuition behind deep feed-forward neural networks is that each layer learns a more abstract representation of the input than the previous one (Bengio et al., 2013). We can apply this concept to the NBOW model discussed in Section 2.1 with the expectation that each layer will increasingly magnify small but meaningful differences in the word embedding average. To be more concrete, take s_1 as the sentence "I really loved Rosamund Pike's performance in the movie Gone Girl" and generate s_2 and s_3 by replacing "loved" with "liked" and then again by "despised". The vector averages of these three sentences are almost identical, but the averages associated with the synonymous sentences s_1 and s_2 are slightly more similar to each other than they are to s_3's average.

Could adding depth to NBOW make small such distinctions as this one more apparent? In Equation 1, we compute $\boldsymbol{z}$, the vector representation for input text X, by averaging the word vectors $\boldsymbol{v}_{w \in X}$. Instead of directly passing this representation to an output layer, we can further transform $\boldsymbol{z}$ by adding more layers before applying the softmax. Suppose we have n layers, $\boldsymbol{z}_{1 \dots n}$. We compute each layer

$$\boldsymbol{z}_i = g(\boldsymbol{z}_{i-1}) = f(\mathbf{W}_i \cdot \boldsymbol{z}_{i-1} + \boldsymbol{b}_i) \qquad (5)$$

and feed the final layer's representation, $\boldsymbol{z}_n$, to a softmax layer for prediction (Figure 1, right).

This model, which we call a deep averaging network (DAN), is still unordered, but its depth allows it to capture subtle variations in the input better than the standard NBOW model. Furthermore, computing each layer requires just a single matrix multiplication, so the complexity scales with the number of layers rather than the number of nodes in a parse tree. In practice, we find no significant difference between the training time of a DAN and that of the shallow NBOW model.

3.1 Word Dropout Improves Robustness

Dropout regularizes neural networks by randomly setting hidden and/or input units to zero with some probability p (Hinton et al., 2012; Srivastava et al., 2014). Given a neural network with n units, dropout prevents overfitting by creating an ensemble of 2^n different networks that share parameters, where each network consists of some combination of dropped and undropped units. Instead of dropping units, a natural extension for the DAN model is to randomly drop word tokens' entire *word embeddings* from the vector average. Using this method, which we call *word dropout*, our network theoretically sees $2^{|X|}$ different token sequences for each input X.

We posit a vector $\boldsymbol{r}$ with $|X|$ independent Bernoulli trials, each of which equals 1 with probability p. The embedding $\boldsymbol{v}_w$ for token w in X is dropped from the average if $\boldsymbol{r}_w$ is 0, which exponentially increases the number of unique examples the network sees during training. This allows us to modify Equation 1:

$$\boldsymbol{r}_w \sim \text{Bernoulli}(p) \tag{6}$$

$$\hat{X} = \{w | w \in X \text{ and } \boldsymbol{r}_w > 0\} \tag{7}$$

$$\boldsymbol{z} = g(w \in X) = \frac{\sum_{w \in \hat{X}} \boldsymbol{v}_w}{|\hat{X}|}. \tag{8}$$

Depending on the choice of p, many of the "dropped" versions of an original training instance will be very similar to each other, but for shorter inputs this is less likely. We might drop a very important token, such as "horrible" in "the crab rangoon was especially horrible"; however, since the number of word types that are predictive of the output labels is low compared to non-predictive ones (e.g., neutral words in sentiment analysis), we always see improvements using this technique.

Theoretically, word dropout can also be applied to other neural network-based approaches. However, we observe no significant performance differences in preliminary experiments when applying word dropout to leaf nodes in RecNNs for sentiment analysis (dropped leaf representations are set to zero vectors), and it slightly hurts performance on the question answering task.

4 Experiments

We compare DANs to both the shallow NBOW model as well as more complicated syntactic models on sentence and document-level sentiment analysis and factoid question answering tasks. The DAN architecture we use for each task is almost identical, differing across tasks only in the type of output layer and the choice of activation function. Our results show that DANs outperform other bag-of-words models and many syntactic models with very little training time.[2] On the question-answering task, DANs effectively train on out-of-domain data, while RecNNs struggle to reconcile the syntactic differences between the training and test data.

Model	RT	SST fine	SST bin	IMDB	Time (s)
DAN-ROOT	—	46.9	85.7	—	**31**
DAN-RAND	77.3	45.4	83.2	88.8	136
DAN	80.3	47.7	86.3	89.4	136
NBOW-RAND	76.2	42.3	81.4	88.9	91
NBOW	79.0	43.6	83.6	89.0	91
BiNB	—	41.9	83.1	—	—
NBSVM-bi	79.4	—	—	91.2	—
RecNN*	77.7	43.2	82.4	—	—
RecNTN*	—	45.7	85.4	—	—
DRecNN	—	49.8	86.6	—	431
TreeLSTM	—	**50.6**	86.9	—	—
DCNN*	—	48.5	86.9	89.4	—
PVEC*	—	48.7	87.8	**92.6**	—
CNN-MC	**81.1**	47.4	**88.1**	—	2,452
WRRBM*	—	—	—	89.2	—

Table 1: DANs achieve comparable sentiment accuracies to syntactic functions (bottom third of table) but require much less training time (measured as time of a single epoch on the SST fine-grained task). Asterisked models are initialized either with different pretrained embeddings or randomly.

4.1 Sentiment Analysis

Recently, syntactic composition functions have revolutionized both fine-grained and binary (positive or negative) sentiment analysis. We conduct sentence-level sentiment experiments on the Rotten Tomatoes (RT) movie reviews dataset (Pang and Lee, 2005) and its extension with phrase-level labels, the Stanford Sentiment Treebank (SST) introduced by Socher et al. (2013b). Our model is also effective on the document-level IMDB movie review dataset of Maas et al. (2011).

4.1.1 Neural Baselines

Most neural approaches to sentiment analysis are variants of either recursive or convolutional networks. Our recursive neural network baselines include standard RecNNs (Socher et al., 2011b), RecNTNs, the deep recursive network (DRecNN) proposed by İrsoy and Cardie (2014), and the TREE-LSTM of (Tai et al., 2015). Convolutional network baselines include the dynamic convolutional network (Kalchbrenner et al., 2014, DCNN) and the convolutional neural network multichannel (Kim, 2014, CNN-MC). Our other neural baselines are the sliding-window based paragraph vector (Le and Mikolov, 2014, PVEC)[3] and

[2] Code at http://github.com/miyyer/dan.

[3] PVEC is computationally expensive at both training and test time and requires enough memory to store a vector for every paragraph in the training data.

the word-representation restricted Boltzmann machine (Dahl et al., 2012, **WRRBM**), which only works on the document-level IMDB task.[4]

4.1.2 Non-Neural Baselines

We also compare to non-neural baselines, specifically the bigram naïve Bayes (**BINB**) and naïve Bayes support vector machine (**NBSVM-BI**) models introduced by Wang and Manning (2012), both of which are memory-intensive due to huge feature spaces of size $|V|^2$.

4.1.3 DAN Configurations

In Table 1, we compare a variety of **DAN** and **NBOW** configurations[5] to the baselines described above. In particular, we are interested in not only comparing **DAN** accuracies to those of the baselines, but also how initializing with pretrained embeddings and restricting the model to only root-level labels affects performance. With this in mind, the **NBOW-RAND** and **DAN-RAND** models are initialized with random 300-dimensional word embeddings, while the other models are initialized with publicly-available 300-d GloVe vectors trained over the Common Crawl (Pennington et al., 2014). The **DAN-ROOT** model only has access to sentence-level labels for SST experiments, while all other models are trained on labeled phrases (if they exist) in addition to sentences. We train all **NBOW** and **DAN** models using AdaGrad (Duchi et al., 2011).

We apply **DAN**s to documents by averaging the embeddings for all of a document's tokens and then feeding that average through multiple layers as before. Since the representations computed by **DAN**s are always d-dimensional vectors regardless of the input size, they are efficient with respect to both memory and computational cost. We find that the hyperparameters selected on the SST also work well for the IMDB task.

4.1.4 Dataset Details

We evaluate over both fine-grained and binary sentence-level classification tasks on the SST, and just the binary task on RT and IMDB. In the fine-grained SST setting, each sentence has a label from zero to five where two is the neutral class. For the binary task, we ignore all neutral sentences.[6]

4.1.5 Results

The **DAN** achieves the second best reported result on the RT dataset, behind only the significantly slower **CNN-MC** model. It's also competitive with more complex models on the SST and outperforms the **DCNN** and **WRRBM** on the document-level IMDB task. Interestingly, the **DAN** achieves good performance on the SST when trained with only sentence-level labels, indicating that it does not suffer from the vanishing error signal problem that plagues **RecNN**s. Since acquiring labelled phrases is often expensive (Sayeed et al., 2012; Iyyer et al., 2014b), this result is promising for large or messy datasets where fine-grained annotation is infeasible.

4.1.6 Timing Experiments

DANs require less time per epoch and—in general—require fewer epochs than their syntactic counterparts. We compare **DAN** runtime on the SST to publicly-available implementations of syntactic baselines in the last column of Table 1; the reported times are for a single epoch to control for hyperparameter choices such as learning rate, and all models use 300-d word vectors. Training a **DAN** on just sentence-level labels on the SST takes under five minutes on a single core of a laptop; when labeled phrases are added as separate training instances, training time jumps to twenty minutes.[7] All timing experiments were performed on a single core of an Intel I7 processor with 8GB of RAM.

4.2 Factoid Question Answering

DANs work well for sentiment analysis, but how do they do on other NLP tasks? We shift gears to a paragraph-length factoid question answering task and find that our model outperforms other unordered functions as well as a more complex syntactic **RecNN** model. More interestingly, we find that unlike the **RecNN**, the **DAN** significantly benefits from out-of-domain Wikipedia training data.

Quiz bowl is a trivia competition in which players are asked four-to-six sentence questions about entities (e.g., authors, battles, or events). It is an ideal task to evaluate **DAN**s because there is prior

[4] The **WRRBM** is trained using a slow Metropolis-Hastings algorithm.

[5] Best hyperparameters chosen by cross-validation: three 300-d ReLu layers, word dropout probability $p = 0.3$, L2 regularization weight of 1e-5 applied to all parameters

[6] Our fine-grained SST split is {train: 8,544, dev: 1,101, test: 2,210}, while our binary split is {train: 6,920, dev:872, test:1,821}. Split sizes increase by an order of magnitude when labeled phrases are added to the training set. For RT, we do 10-fold CV over a balanced binary dataset of 10,662 sentences. Similarly, for the IMDB experiments we use the provided balanced binary training set of 25,000 documents.

[7] We also find that **DAN**s take significantly fewer epochs to reach convergence than syntactic models.

Model	Pos 1	Pos 2	Full	Time(s)
BoW-DT	35.4	57.7	60.2	—
IR	37.5	65.9	71.4	N/A
QANTA	47.1	72.1	73.7	314
DAN	46.4	70.8	71.8	**18**
IR-WIKI	53.7	**76.6**	**77.5**	N/A
QANTA-WIKI	46.5	72.8	73.9	1,648
DAN-WIKI	**54.8**	75.5	77.1	119

Table 2: The DAN achieves slightly lower accuracies than the more complex QANTA in much less training time, even at early sentence positions where compositionality plays a bigger role. When Wikipedia is added to the training set (bottom half of table), the DAN outperforms QANTA and achieves comparable accuracy to a state-of-the-art information retrieval baseline, which highlights a benefit of ignoring word order for this task.

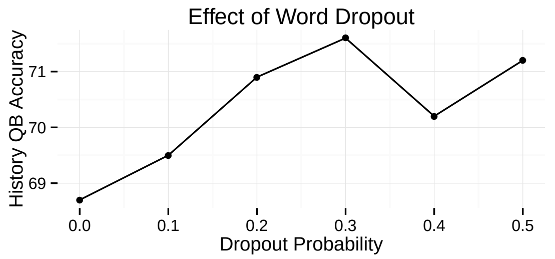

Figure 2: Randomly dropping out 30% of words from the vector average is optimal for the quiz bowl task, yielding a gain in absolute accuracy of almost 3% on the quiz bowl question dataset compared to the same model trained with no word dropout.

work using both syntactic and unordered models for quiz bowl question answering. In Boyd-Graber et al. (2012), naïve Bayes bag-of-words models (BOW-DT) and sequential language models work well on easy questions but poorly on harder ones. A dependency-tree RecNN called QANTA proposed in Iyyer et al. (2014a) shows substantial improvements, leading to the hypothesis that correctly modeling compositionality is crucial for answering hard questions.

4.2.1 Dataset and Experimental Setup

To test this, we train a DAN over the history questions from Iyyer et al. (2014a).[8] This dataset is augmented with 49,581 sentence/page-title pairs from the Wikipedia articles associated with the answers in the dataset. For fair comparison with QANTA, we use a normalized tanh activation function at the last layer instead of ReLu, and we also change the output layer from a softmax to the margin ranking loss (Weston et al., 2011) used in QANTA. We initialize the DAN with the same pretrained 100-d word embeddings that were used to initialize QANTA.

We also evaluate the effectiveness of word dropout on this task in Figure 2. Cross-validation indicates that $p = 0.3$ works best for question answering, although the improvement in accuracy is negligible for sentiment analysis. Finally, continuing the trend observed in the sentiment experiments, DAN converges much faster than QANTA.

4.2.2 DANs Improve with Noisy Data

Table 2 shows that while DAN is slightly worse than QANTA when trained only on question-answer pairs, it improves when trained on additional out-of-domain Wikipedia data (DAN-WIKI), reaching performance comparable to that of a state-of-the-art information retrieval system (IR-WIKI). QANTA, in contrast, barely improves when Wikipedia data is added (QANTA-WIKI) possibly due to the syntactic differences between Wikipedia text and quiz bowl question text.

The most common syntactic structures in quiz bowl sentences are imperative constructions such as “Identify this British author who wrote Wuthering Heights”, which are almost never seen in Wikipedia. Furthermore, the subject of most quiz bowl sentences is a pronoun or pronomial mention referring to the answer, a property that is not true of Wikipedia sentences (e.g., “Little of Emily’s work from this period survives, except for poems spoken by characters.”). Finally, many Wikipedia sentences do not uniquely identify the title of the page they come from, such as the following sentence from Emily Brontë’s page: “She does not seem to have made any friends outside her family.” While noisy data affect both DAN and QANTA, the latter is further hampered by the syntactic divergence between quiz bowl questions and Wikipedia, which may explain the lack of improvement in accuracy.

[8] The training set contains 14,219 sentences over 3,761 questions. For more detail about data and baseline systems, see Iyyer et al. (2014a).

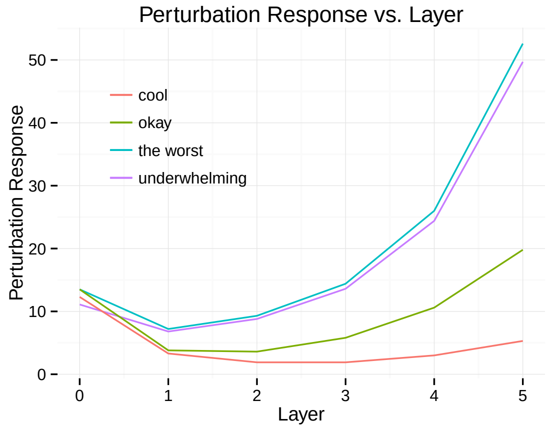

Figure 3: Perturbation response (difference in 1-norm) at each layer of a 5-layer **DAN** after replacing *awesome* in *the film's performances were awesome* with four words of varying sentiment polarity. While the shallow **NBOW** model does not show any meaningful distinctions, we see that as the network gets deeper, negative sentences are increasingly different from the original positive sentence.

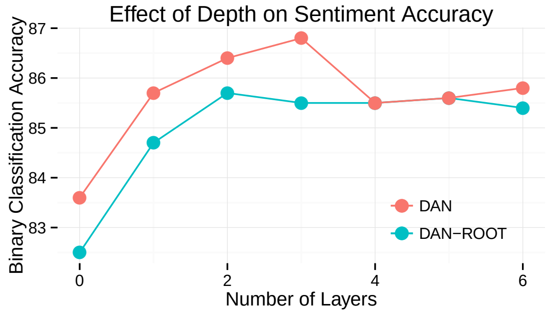

Figure 4: Two to three layers is optimal for the **DAN** on the SST binary sentiment analysis task, but adding any depth at all is an improvement over the shallow **NBOW** model.

5 How Do DANs Work?

In this section we first examine how the deep layers of the **DAN** amplify tiny differences in the vector average that are predictive of the output labels. Next, we compare **DAN**s to **DRecNN**s on sentences that contain negations and contrastive conjunctions and find that both models make similar errors despite the latter's increased complexity. Finally, we analyze the predictive ability of unsupervised word embeddings on a simple sentiment task in an effort to explain why initialization with these embeddings improves the **DAN**.

5.1 Perturbation Analysis

Following the work of İrsoy and Cardie (2014), we examine our network by measuring the response at each hidden layer to perturbations in an input sentence. In particular, we use the template *the film's performances were awesome* and replace the final word with increasingly negative polarity words (*cool*, *okay*, *underwhelming*, *the worst*). For each perturbed sentence, we observe how much the hidden layers differ from those associated with the original template in 1-norm.

Figure 3 shows that as a **DAN** gets deeper, the differences between negative and positive sentences become increasingly amplified. While nonexistent in the shallow **NBOW** model, these differences are visible even with just a single hidden layer, thus explaining why deepening the **NBOW** improves sentiment analysis as shown in Figure 4.

5.2 Handling Negations and "but": Where Syntax is Still Needed

While **DAN**s outperform other bag-of-words models, how can they model linguistic phenomena such as negation without considering word order? To evaluate **DAN**s over tougher inputs, we collect 92 sentences, each of which contains at least one negation and one contrastive conjunction, from the dev and test sets of the SST.[9] Our fine-grained accuracy is *higher* on this subset than on the full dataset, improving almost five percent absolute accuracy to 53.3%. The **DRecNN** model of İrsoy and Cardie (2014) obtains a similar accuracy of 51.1%, contrary to our intuition that syntactic functions should outperform unordered functions on sentences that clearly require syntax to understand.[10]

Are these sentences truly difficult to classify? A close inspection reveals that both the **DAN** and the **DRecNN** have an overwhelming tendency to predict negative sentiment (60.9% and 55.4% of the time for the **DAN** and **DRecNN** respectively) when they see a negation compared to positive sentiment (35.9% for **DAN**s, 34.8% for **DRecNN**s). If we further restrict our subset of sentences to only those with positive ground truth labels, we find that while both models struggle, the **DRecNN** obtains 41.7% accuracy, outperforming the **DAN**'s 37.5%.

To understand why a negation or contrastive conjunction triggers a negative sentiment prediction,

[9] We search for non-neutral sentences containing *not* / *n't*, and *but*. 48 of the sentences are positive while 44 are negative.

[10] Both models are initialized with pretrained 300-d GloVe embeddings for fair comparison.

Sentence	DAN	DRecNN	Ground Truth
a lousy movie that's not merely unwatchable, but also unlistenable	negative	negative	negative
if you're not a prepubescent girl, you'll be laughing at britney spears' movie-starring debut whenever it does n't have you impatiently squinting at your watch	negative	negative	negative
blessed with immense physical prowess he may well be, but ahola is simply not an actor	positive	neutral	negative
who knows what exactly godard is on about in this film, but his words and images do n't have to add up to mesmerize you.	positive	positive	positive
it's so good that its relentless, polished wit can withstand not only inept school productions, but even oliver parker's movie adaptation	negative	positive	positive
too bad, but thanks to some lovely comedic moments and several fine performances, it's not a total loss	negative	negative	positive
this movie was not good	negative	negative	negative
this movie was good	positive	positive	positive
this movie was bad	negative	negative	negative
the movie was not bad	negative	negative	positive

Table 3: Predictions of DAN and DRecNN models on real (top) and synthetic (bottom) sentences that contain negations and contrastive conjunctions. In the first column, words colored red individually predict the negative label when fed to a DAN, while blue words predict positive. The DAN learns that the negators *not* and *n't* are strong negative predictors, which means it is unable to capture double negation as in the last real example and the last synthetic example. The DRecNN does slightly better on the synthetic double negation, predicting a lower negative polarity.

we show six sentences from the negation subset and four synthetic sentences in Table 3, along with both models' predictions. The token-level predictions in the table (shown as colored boxes) are computed by passing each token through the DAN as separate test instances. The tokens *not* and *n't* are strongly predictive of negative sentiment. While this simplified "negation" works for many sentences in the datasets we consider, it prevents the DAN from reasoning about double negatives, as in "this movie was not bad". The DRecNN does slightly better in this case by predicting a lesser negative polarity than the DAN; however, we theorize that still more powerful syntactic composition functions (and more labelled instances of negation and related phenomena) are necessary to truly solve this problem.

5.3 Unsupervised Embeddings Capture Sentiment

Our model consistently converges slower to a worse solution (dropping 3% in absolute accuracy on coarse-grained SST) when we randomly initialize the word embeddings. This does not apply to just DANs; both convolutional and recursive networks do the same (Kim, 2014; İrsoy and Cardie, 2014). Why are initializations with these embeddings so crucial to obtaining good performance? Is it possible that unsupervised training algorithms are already capturing sentiment?

We investigate this theory by conducting a simple experiment: given a sentiment lexicon containing both positive and negative words, we train a logistic regression to discriminate between the associated word embeddings (without any fine-tuning). We use the lexicon created by Hu and Liu (2004), which consists of 2,006 positive words and 4,783 negative words. We balance and split the dataset into 3,000 training words and 1,000 test words. Using 300-dimensional GloVe embeddings pretrained over the Common Crawl, we obtain over 95% accuracy on the unseen test set, supporting the hypothesis that unsupervised pretraining over large corpora can capture properties such as sentiment.

Intuitively, after the embeddings are fine-tuned during DAN training, we might expect a decrease in the norms of stopwords and an increase in the norms of sentiment-rich words like "awesome" or "horrible". However, we find no significant differences between the L_2 norms of stopwords and words in the sentiment lexicon of Hu and Liu (2004).

6 Related Work

Our DAN model builds on the successes of both simple vector operations and neural network-based models for compositionality.

There are a variety of element-wise vector operations that could replace the average used in the DAN. Mitchell and Lapata (2008) experiment with many of them to model the compositionality of short phrases. Later, their work was extended to take into account the syntactic relation between words (Erk and Padó, 2008; Baroni and Zamparelli, 2010; Kartsaklis and Sadrzadeh, 2013) and grammars (Coecke et al., 2010; Grefenstette and Sadrzadeh, 2011). While the average works best for the tasks that we consider, Banea et al. (2014) find that simply summing word2vec embeddings outperforms all other methods on the SemEval 2014 phrase-to-word and sentence-to-phrase similarity tasks.

Once we compute the embedding average in a DAN, we feed it to a deep neural network. In contrast, most previous work on neural network-based methods for NLP tasks explicitly model word order. Outside of sentiment analysis, RecNN-based approaches have been successful for tasks such as parsing (Socher et al., 2013a), machine translation (Liu et al., 2014), and paraphrase detection (Socher et al., 2011a). Convolutional networks also model word order in local windows and have achieved performance comparable to or better than that of RecNNs on many tasks (Collobert and Weston, 2008; Kim, 2014). Meanwhile, feedforward architectures like that of the DAN have been used for language modeling (Bengio et al., 2003), selectional preference acquisition (Van de Cruys, 2014), and dependency parsing (Chen and Manning, 2014).

7 Future Work

In Section 5, we showed that the performance of our DAN model worsens on sentences that contain lingustic phenomena such as double negation. One promising future direction is to cascade classifiers such that syntactic models are used only when a DAN is not confident in its prediction. We can also extend the DAN's success at incorporating out-of-domain training data to sentiment analysis: imagine training a DAN on labeled tweets for classification on newspaper reviews. Another potentially interesting application is to add gated units to a DAN,as has been done for recurrent and recursive neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014; Sutskever et al., 2014; Tai et al., 2015), to drop useless words rather than randomly-selected ones.

8 Conclusion

In this paper, we introduce the deep averaging network, which feeds an unweighted average of word vectors through multiple hidden layers before classification. The DAN performs competitively with more complicated neural networks that explicitly model semantic and syntactic compositionality. It is further strengthened by word dropout, a regularizer that reduces input redundancy. DANs obtain close to state-of-the-art accuracy on both sentence and document-level sentiment analysis and factoid question-answering tasks with much less training time than competing methods; in fact, all experiments were performed in a matter of minutes on a single laptop core. We find that both DANs and syntactic functions make similar errors given syntactically-complex input, which motivates research into more powerful models of compositionality.

Acknowledgments

We thank Ozan İrsoy not only for many insightful discussions but also for suggesting some of the experiments that we included in the paper. We also thank the anonymous reviewers, Richard Socher, Arafat Sultan, and the members of the UMD "Thinking on Your Feet" research group for their helpful comments. This work was supported by NSF Grant IIS-1320538. Boyd-Graber is also supported by NSF Grants CCF-1409287 and NCSE-1422492. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Carmen Banea, Di Chen, Rada Mihalcea, Claire Cardie, and Janyce Wiebe. 2014. Simcompass: Using deep learning word embeddings to assess cross-level similarity. In *SemEval*.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of Empirical Methods in Natural Language Processing*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*.

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.

Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daumé III. 2012. Besting the quiz master: Crowdsourcing incremental classification games. In *Proceedings of Empirical Methods in Natural Language Processing*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis (Lambek Festschirft)*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference of Machine Learning*.

George E Dahl, Ryan P Adams, and Hugo Larochelle. 2012. Training restricted boltzmann machines on word observations. In *Proceedings of the International Conference of Machine Learning*.

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

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of Empirical Methods in Natural Language Processing*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of Empirical Methods in Natural Language Processing*.

Karl Moritz Hermann, Edward Grefenstette, and Phil Blunsom. 2013. "not not bad" is not "bad": A distributional account of negation. *Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Knowledge Discovery and Data Mining*.

Ozan İrsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Proceedings of Advances in Neural Information Processing Systems*.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014a. A neural network for factoid question answering over paragraphs. In *Proceedings of Empirical Methods in Natural Language Processing*.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014b. Political ideology detection using recursive neural networks. In *Proceedings of the Association for Computational Linguistics*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *ACL Workshop on Continuous Vector Space Models and their Compositionality*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the Association for Computational Linguistics*.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of Empirical Methods in Natural Language Processing*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of Empirical Methods in Natural Language Processing*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference of Machine Learning*.

Jiwei Li. 2014. Feature weight tuning for recursive neural networks. *CoRR*, abs/1412.3714.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proceedings of the Association for Computational Linguistics*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the Association for Computational Linguistics*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the Association for Computational Linguistics*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Asad B. Sayeed, Jordan Boyd-Graber, Bryan Rusk, and Amy Weinberg. 2012. Grammatical structures for word-level sentiment detection. In *North American Association of Computational Linguistics*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Proceedings of Advances in Neural Information Processing Systems*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of Empirical Methods in Natural Language Processing*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing With Compositional Vector Grammars. In *Proceedings of the Association for Computational Linguistics*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of Empirical Methods in Natural Language Processing*.

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

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks.

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of Empirical Methods in Natural Language Processing*.

Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the Association for Computational Linguistics*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *International Joint Conference on Artificial Intelligence*.