Learning Text Pair Similarity with Context-sensitive Autoencoders

Hadi Amiri[1], **Philip Resnik**[1], **Jordan Boyd-Graber**[2], **Hal Daumé III**[1]
[1]Institute for Advanced Computer Studies
University of Maryland, College Park, MD
[2]Department of Computer Science
University of Colorado, Boulder, CO
{hadi,resnik,hal}@umd.edu,jordan.boyd.graber@colorado.edu

Abstract

We present a *pairwise context-sensitive Autoencoder* for computing text pair similarity. Our model encodes input text into *context-sensitive* representations and uses them to compute similarity between text pairs. Our model outperforms the state-of-the-art models in two semantic retrieval tasks and a contextual word similarity task. For retrieval, our unsupervised approach that merely ranks inputs with respect to the cosine similarity between their hidden representations shows comparable performance with the state-of-the-art supervised models and in some cases outperforms them.

1 Introduction

Representation learning algorithms learn representations that reveal intrinsic low-dimensional structure in data (Bengio et al., 2013). Such representations can be used to induce similarity between textual contents by computing similarity between their respective vectors (Huang et al., 2012; Silberer and Lapata, 2014).

Recent research has made substantial progress on semantic similarity using neural networks (Rothe and Schütze, 2015; Dos Santos et al., 2015; Severyn and Moschitti, 2015). In this work, we focus our attention on deep autoencoders and extend these models to integrate sentential or document *context* information about their inputs. We represent context information as low dimensional vectors that will be injected to deep autoencoders. To the best of our knowledge, this is the first work that enables integrating context into autoencoders.

In representation learning, context may appear in various forms. For example, the context of a current sentence in a document could be either its neighboring sentences (Lin et al., 2015; Wang and Cho, 2015), topics associated with the sentence (Mikolov and Zweig, 2012; Le and Mikolov, 2014), the document that contains the sentence (Huang et al., 2012), as well as their combinations (Ji et al., 2016). It is important to integrate context into neural networks because these models are often trained with only local information about their individual inputs. For example, recurrent and recursive neural networks only use local information about previously seen words in a sentence to predict the next word or composition.[1] On the other hand, context information (such as topical information) often capture global information that can guide neural networks to generate more accurate representations.

We investigate the utility of context information in three semantic similarity tasks: *contextual word sense similarity* in which we aim to predict semantic similarity between given word pairs in their sentential context (Huang et al., 2012; Rothe and Schütze, 2015), *question ranking* in which we aim to retrieve semantically equivalent questions with respect to a given test question (Dos Santos et al., 2015), and *answer ranking* in which we aim to rank single-sentence answers with respect to a given question (Severyn and Moschitti, 2015).

The contributions of this paper are as follows: (1) integrating context information into deep autoencoders and (2) showing that such integration improves the representation performance of deep autoencoders across several different semantic similarity tasks.

Our model outperforms the state-of-the-art su-

[1]For example, RNNs can predict the word "sky" given the sentence "clouds are in the __," but they are less accurate when longer history or global context is required, e.g. predicting the word "french" given the paragraph "I grew up in France. ... I speak fluent __."

pervised baselines in three semantic similarity tasks. Furthermore, the unsupervised version of our autoencoder show comparable performance with the supervised baseline models and in some cases outperforms them.

2 Context-sensitive Autoencoders

2.1 Basic Autoencoders

We first provide a brief description of basic autoencoders and extend them to context-sensitive ones in the next Section. Autoencoders are trained using a local unsupervised criterion (Vincent et al., 2010; Hinton and Salakhutdinov, 2006; Vincent et al., 2008). Specifically, the basic autoencoder in Figure 1(a) locally optimizes the hidden representation $\mathbf{h}$ of its input $\mathbf{x}$ such that $\mathbf{h}$ can be used to accurately reconstruct $\mathbf{x}$,

$$\mathbf{h} = \mathbf{g}(\mathbf{W}\mathbf{x} + \mathbf{b}_{\mathbf{h}}) \quad (1)$$

$$\hat{\mathbf{x}} = \mathbf{g}(\mathbf{W}'\mathbf{h} + \mathbf{b}_{\hat{\mathbf{x}}}), \quad (2)$$

where $\hat{\mathbf{x}}$ is the reconstruction of $\mathbf{x}$, the learning parameters $\mathbf{W} \in \mathbb{R}^{d' \times d}$ and $\mathbf{W}' \in \mathbb{R}^{d \times d'}$ are weight matrices, $\mathbf{b}_{\mathbf{h}} \in \mathbb{R}^{d'}$ and $\mathbf{b}_{\hat{\mathbf{x}}} \in \mathbb{R}^{d}$ are bias vectors for the hidden and output layers respectively, and $\mathbf{g}$ is a nonlinear function such as $\tanh(.)$.[2] Equation (1) *encodes* the input into an intermediate representation and Equation (2) *decodes* the resulting representation.

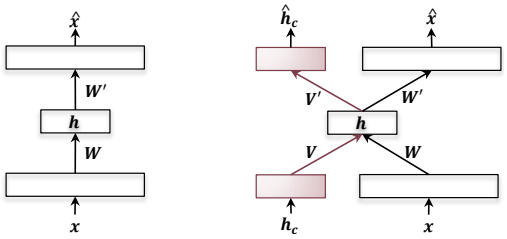
Training a single-layer autoencoder corresponds to optimizing the learning parameters to minimize the overall loss between inputs and their reconstructions. For real-valued $\mathbf{x}$, squared loss is often used, $l(\mathbf{x}) = ||\mathbf{x} - \hat{\mathbf{x}}||^2$, (Vincent et al., 2010):

$$\min_{\mathbf{\Theta}} \sum_{i=1}^{n} l(\mathbf{x}^{(i)}) \qquad \mathbf{\Theta} = \{\mathbf{W}, \mathbf{W}', \mathbf{b}_{\mathbf{h}}, \mathbf{b}_{\hat{\mathbf{x}}}\}. \quad (3)$$

This can be achieved using mini-batch stochastic gradient descent (Zeiler, 2012).

2.2 Integrating Context into Autoencoders

We extend the above basic autoencoder to integrate context information about inputs. We assume that—for each training example $\mathbf{x} \in \mathbb{R}^d$—we have a context vector $\mathbf{c}_{\mathbf{x}} \in \mathbb{R}^k$ that contains contextual information about the input.[3] The nature of this context vector depends on the input and target task. For example, neighboring words can be considered as the context of a target word in contextual word similarity task.



(a) Basic Autoencoder (b) Context Autoencoder

Figure 1: Schematic representation of basic and context-sensitive autoencoders: (a) Basic autoencoder maps its input $\mathbf{x}$ into the representation $\mathbf{h}$ such that it can reconstruct $\mathbf{x}$ with minimum loss, and (b) Context-sensitive autoencoder maps its inputs $\mathbf{x}$ and $\mathbf{h}_c$ into a context-sensitive representation $\mathbf{h}$ ($\mathbf{h}_c$ is the representation of the context information associated to $\mathbf{x}$).

We first learn the hidden representation $\mathbf{h}_c \in \mathbb{R}^{d'}$ for the given context vector $\mathbf{c}_{\mathbf{x}}$. For this, we use the same process as discussed above for the basic autoencoder where we use $\mathbf{c}_{\mathbf{x}}$ as the input in Equations (1) and (2) to obtain $\mathbf{h}_c$. We then use $\mathbf{h}_c$ to develop our context-sensitive autoencoder as depicted in Figure 1(b). This autoencoder maps its inputs $\mathbf{x}$ and $\mathbf{h}_c$ into a *context-sensitive* representation $\mathbf{h}$ as follows:

$$\mathbf{h} = \mathbf{g}(\mathbf{W}\mathbf{x} + \mathbf{V}\mathbf{h}_c + \mathbf{b}_{\mathbf{h}}) \quad (4)$$

$$\hat{\mathbf{x}} = \mathbf{g}(\mathbf{W}'\mathbf{h} + \mathbf{b}_{\hat{\mathbf{x}}}) \quad (5)$$

$$\hat{\mathbf{h}}_c = \mathbf{g}(\mathbf{V}'\mathbf{h} + \mathbf{b}_{\hat{\mathbf{h}}_c}). \quad (6)$$

Our intuition is that if $\mathbf{h}$ leads to a good reconstruction of its inputs, it has retained information available in the input. Therefore, it is a context-sensitive representation.

The loss function must then compute the loss between the input pair ($\mathbf{x}$, $\mathbf{h}_c$) and its reconstruction ($\hat{\mathbf{x}}$, $\hat{\mathbf{h}}_c$). For optimization, we can still use squared loss with a different set of parameters to minimize the overall loss on the training examples:

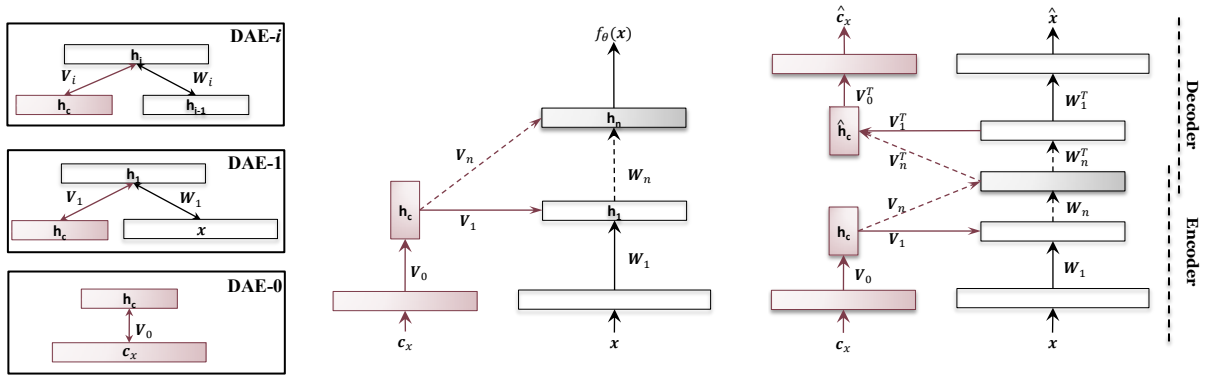
$$l(\mathbf{x}, \mathbf{h}_c) = ||\mathbf{x} - \hat{\mathbf{x}}||^2 + \lambda||\mathbf{h}_c - \hat{\mathbf{h}}_c||^2$$

$$\min_{\mathbf{\Theta}} \sum_{i=1}^{n} l(\mathbf{x}^{(i)}, \mathbf{h}_c^{(i)}) \quad (7)$$

$$\mathbf{\Theta} = \{\mathbf{W}, \mathbf{W}', \mathbf{V}, \mathbf{V}', \mathbf{b}_{\mathbf{h}}, \mathbf{b}_{\hat{\mathbf{x}}}, \mathbf{b}_{\hat{\mathbf{h}}_c}\},$$

[2] If the squared loss is used for optimization, as in Equation (3), nonlinearity is often not used in Equation (2) (Vincent et al., 2010).

[3] We slightly abuse the notation throughout this paper by referring to $\mathbf{c}_x$ or $\mathbf{h}_i$ as vectors, not elements of vectors.



(a) Network initialization (b) Stacking Denoising Autoencoders (c) Unrolling and Fine-tuning

Figure 2: Proposed framework for integrating context into deep autoencoders. Context layer ($\mathbf{c_x}$ and $\mathbf{h}_c$) and context-sensitive representation of input ($\mathbf{h}_n$) are shown in light red and gray respectively. (a) Pre-training properly initializes a stack of context-sensitive denoising autoencoders (DAE), (b) A context-sensitive deep autoencoder is created from properly initialized DAEs, (c) The network in (b) is unrolled and its parameters are fine-tuned for optimal reconstruction.

where $\lambda \in [0, 1]$ is a weight parameter that controls the effect of context information in the reconstruction process.

2.2.1 Denoising

Denoising autoencoders (DAEs) reconstruct an input from a *corrupted* version of it for more effective learning (Vincent et al., 2010). The corrupted input is then mapped to a hidden representation from which we obtain the reconstruction. However, the reconstruction loss is still computed with respect to the *uncorrupted* version of the input as before. Denoising autoencoders effectively learn representations by reversing the effect of the corruption process. We use *masking noise* to corrupt the inputs where a fraction η of input units are randomly selected and set to zero (Vincent et al., 2008).

2.2.2 Deep Context-Sensitive Autoencoders

Autoencoders can be stacked to create deep networks. A deep autoencoder is composed of multiple hidden layers that are stacked together. The initial weights in such networks need to be properly initialized through a greedy layer-wise training approach. Random initialization does not work because deep autoencoders converge to poor local minima with large initial weights and result in tiny gradients in the early layers with small initial weights (Hinton and Salakhutdinov, 2006).

Our deep context-sensitive autoencoder is composed of a stacked set of DAEs. As discussed above, we first need to properly initialize the learning parameters (weights and biases) associated to each DAE. As shown in Figure 2(a), we first train DAE-0, which initializes parameters associated to the context layer. The training procedure is exactly the same as training a basic autoencoder (Section 2.1 and Figure 1(a)).[4] We then treat $\mathbf{h}_c$ and $\mathbf{x}$ as "inputs" for DAE-1 and use the same approach as in training a context-sensitive autoencoder to initialize the parameters of DAE-1 (Section 2.2 and Figure 1(b)). Similarly, the i^{th} DAE is built on the output of the $(i-1)^{\text{th}}$ DAE and so on until the desired number of layers (e.g. n layers) are initialized. For denoising, the corruption is only applied on "inputs" of individual autoencoders. For example, when we are training DAE-i, $\mathbf{h}_{i-1}$ and $\mathbf{h}_c$ are first obtained from the original inputs of the network ($\mathbf{x}$ and $\mathbf{c_x}$) through a single forward pass and then their corrupted versions are computed to train DAE-i.

Figure 2(b) shows that the n properly initialized DAEs can be stacked to form a deep context-sensitive autoencoder. We *unroll* this network to fully optimize its weights through gradient descent and backpropagation (Vincent et al., 2010; Hinton and Salakhutdinov, 2006) .

2.2.3 Unrolling and Fine-tuning

We optimize the learning parameters of our initialized context-sensitive deep autoencoder by unfolding its n layers and making a $2n-1$ layer net-

[4] Figure 2(a) shows compact schematic diagrams of autoencoders used in Figures 1(a) and 1(b)

work whose lower layers form an "encoder" network and whose upper layers form a "decoder" network (Figure 2(c)). A global fine-tuning stage backpropagates through the entire network to fine-tune the weights for optimal reconstruction. In this stage, we update the network parameters again by training the network to minimize the loss between original inputs and their actual reconstruction. We backpropagate the error derivatives first through the decoder network and then through the encoder network. Each decoder layer tries to recover the input of its corresponding encoder layer. As such, the weights are initially symmetric and the decoder weights do need to be learned.

After the training is complete, the hidden layer $\mathbf{h}_n$ contains a context-sensitive representation of the inputs $\mathbf{x}$ and $\mathbf{c_x}$.

2.3 Context Information

Context is task and data dependent. For example, a sentence or document that contains a target word forms the word's context.

When context information is not readily available, we use topic models to determine such context for individual inputs (Blei et al., 2003; Stevens et al., 2012). In particular, we use Non-Negative Matrix Factorization (NMF) (Lin, 2007): Given a training set with n instances, i.e., $\mathbf{X} \in \mathbb{R}^{v \times n}$, where v is the size of a global vocabulary and the scalar k is the number of topics in the dataset, we learn the topic matrix $\mathbf{D} \in \mathbb{R}^{v \times k}$ and context matrix $\mathbf{C} \in \mathbb{R}^{k \times n}$ using the following sparse coding algorithm:

$$\begin{aligned} \min_{\mathbf{D},\mathbf{C}} \quad & \|\mathbf{X} - \mathbf{DC}\|_F^2 + \mu\|\mathbf{C}\|_1, \\ s.t. \quad & \mathbf{D} \geq 0,\ \mathbf{C} \geq 0, \end{aligned} \tag{8}$$

where each column in $\mathbf{C}$ is a sparse representation of an input over all topics and will be used as global context information in our model. We obtain context vectors for test instances by transforming them according to the fitted NMF model on training data. We also note that advanced topic modeling approaches, such as syntactic topic models (Boyd-Graber and Blei, 2009), can be more effective here as they generate linguistically rich context information.

3 Text Pair Similarity

We present unsupervised and supervised approaches for predicting semantic similarity scores

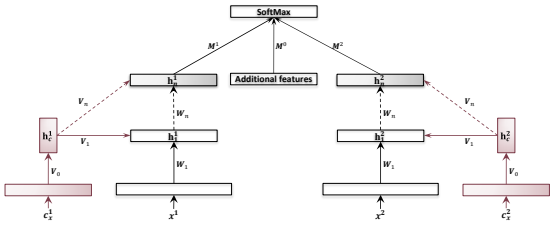


Figure 3: Pairwise context-sensitive autoencoder for computing text pair similarity.

for input texts (e.g., a pair of words) each with its corresponding context information. These scores will then be used to rank "documents" against "queries" (in retrieval tasks) or evaluate how predictions of a model correlate with human judgments (in contextual word sense similarity task).

In unsupervised settings, given a pair of input texts with their corresponding context vectors, ($\mathbf{x}^1$,$\mathbf{c}_{\mathbf{x}^1}$) and ($\mathbf{x}^2$,$\mathbf{c}_{\mathbf{x}^2}$), we determine their semantic similarity score by computing the cosine similarity between their hidden representations $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$ respectively.

In supervised settings, we use a copy of our context-sensitive autoencoder to make a pairwise architecture as depicted in Figure 3. Given ($\mathbf{x}^1$,$\mathbf{c}_{\mathbf{x}^1}$), ($\mathbf{x}^2$,$\mathbf{c}_{\mathbf{x}^2}$), and their binary relevance score, we use $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$ as well as additional features (see below) to train our pairwise network (i.e. further fine-tune the weights) to predict a similarity score for the input pair as follows:

$$rel(\mathbf{x}^1, \mathbf{x}^2) = softmax(\mathbf{M}^0\mathbf{a} + \mathbf{M}^1\mathbf{h}_n^1 + \mathbf{M}^2\mathbf{h}_n^2 + \mathbf{b}) \tag{9}$$

where $\mathbf{a}$ carries additional features, $\mathbf{M}$s are weight matrices, and $\mathbf{b}$ is the bias. We use the difference and similarity between the context-sensitive representations of inputs, $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$, as additional features:

$$\begin{aligned} \mathbf{h}_{sub} &= |\mathbf{h}_n^1 - \mathbf{h}_n^2| \\ \mathbf{h}_{dot} &= \mathbf{h}_n^1 \odot \mathbf{h}_n^2, \end{aligned} \tag{10}$$

where $\mathbf{h}_{sub}$ and $\mathbf{h}_{dot}$ capture the element-wise difference and similarity (in terms of the sign of elements in each dimension) between $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$, respectively. We expect elements in $\mathbf{h}_{sub}$ to be small for semantically similar and relevant inputs and large otherwise. Similarly, we expect elements in $\mathbf{h}_{dot}$ to be positive for relevant inputs and negative otherwise.

We can use any task-specific feature as additional features. This includes features from the

minimal edit sequences between parse trees of the input pairs (Heilman and Smith, 2010; Yao et al., 2013), lexical semantic features extracted from resources such as WordNet (Yih et al., 2013), or other features such as word overlap features (Severyn and Moschitti, 2015; Severyn and Moschitti, 2013). We can also use additional features (Equation 10), computed for BOW representations of the inputs $\mathbf{x}^1$ and $\mathbf{x}^2$. Such additional features improve the performance of our and baseline models.

4 Experiments

In this Section, we use t-test for significant testing and asterisk mark (*) to indicate significance at $\alpha = 0.05$.

4.1 Data and Context Information

We use three datasets: "SCWS" a word similarity dataset with ground-truth labels on similarity of pairs of target words in sentential context from Huang et al. (2012); "qAns" a TREC QA dataset with ground-truth labels for semantically relevant questions and (single-sentence) answers from Wang et al. (2007); and "qSim" a community QA dataset crawled from Stack Exchange with ground-truth labels for semantically equivalent questions from Dos Santos et al. (2015). Table 1 shows statistics of these datasets. To enable direct comparison with previous work, we use the same training, development, and test data provided by Dos Santos et al. (2015) and Wang et al. (2007) for qSim and qAns respectively and the entire data of SCWS (in unsupervised setting).

We consider local and global context for target words in SCWS. The local context of a target word is its ten neighboring words (five before and five after) (Huang et al., 2012), and its global context is a short paragraph that contains the target word (surrounding sentences). We compute average word embeddings to create context vectors for target words.

Also, we consider question title and body and answer text as input in qSim and qAns and use NMF to create global context vectors for questions and answers (Section 2.3).

4.2 Parameter Setting

We use pre-trained word vectors from GloVe (Pennington et al., 2014). However, because qSim questions are about specific technical topics, we only use GloVe as initialization.

Data	Split	#Pairs	%Rel
SCWS	All data	2003	100.0%
qAns	Train-All	53K	12.00%
	Train	4,718	7.400%
	Dev	1,148	19.30%
	Test	1,517	18.70%
qSim	Train	205K	0.048%
	Dev	43M	0.001%
	Test	82M	0.001%

Table 1: Data statistics. (#Pairs: number of word-word pairs in SCWS, question-answer pairs in qAns, and question-question pairs in qSim; %Rel: percentage of positive pairs.)

For the unsupervised SCWS task, following Huang et al. (2012), we use 100-dimensional word embeddings, $d = 100$, with hidden layers and context vectors of the same size, $d' = 100$, $k = 100$. In this unsupervised setting, we set the weight parameter $\lambda = .5$, masking noise $\eta = 0$, depth of our model $n = 3$. Tuning these parameters will further improve the performance of our model.

For qSim and qAns, we use 300-dimensional word embeddings, $d = 300$, with hidden layers of size $d' = 200$. We set the size of context vectors k (number of topics) using the reconstruction error of NMF on training data for different values of k. This leads to $k = 200$ for qAns and $k = 300$ for qSim. We tune the other hyper-parameters (η, n, and λ) using development data.

We set each input $\mathbf{x}$ (target words in SCWS, question titles and bodies in qSim, and question titles and single-sentence answers in qAns) to the average of word embeddings in the input. Input vectors could be initialized through more accurate approaches (Mikolov et al., 2013b; Li and Hovy, 2014); however, averaging leads to reasonable representations and is often used to initialize neural networks (Clinchant and Perronnin, 2013; Iyyer et al., 2015).

4.3 Contextual Word Similarity

We first consider the contextual word similarity task in which a model should predict the semantic similarity between words in their sentential context. For this evaluation, we compute Spearman's ρ correlation (Kokoska and Zwillinger, 2000) between the "relevance scores" predicted by different models and human judgments (Section 3).

The state-of-the-art model for this task is a semi-supervised approach (Rothe and Schütze, 2015). This model use resources like WordNet

to compute embeddings for different senses of words. Given a pair of target words and their context (neighboring words and sentences), this model represents each target word as the average of its sense embeddings weighted by cosine similarity to the context. The cosine similarity between the representations of words in a pair is then used to determine their semantic similarity. Also, the Skip-gram model (Mikolov et al., 2013a) is extended in (Neelakantan et al., 2014; Chen et al., 2014) to learn contextual word pair similarity in an unsupervised way.

Table 2 shows the performance of different models on the SCWS dataset. SAE, CSAE-LC, CSAE-LGC show the performance of our pairwise autoencoders without context, with local context, and with local and global context, respectively. In case of CSAE-LGC, we concatenate local and global context to create context vectors. CSAE-LGC performs significantly better than the baselines, including the semi-supervised approach in Rothe and Schütze (2015). It is also interesting that SAE (without any context information) outperforms the pre-trained word embeddings (Pre-trained embeds.).

Comparing the performance of CSAE-LC and CSAE-LGC indicates that global context is useful for accurate prediction of semantic similarity between word pairs. We further investigate these models to understand why global context is useful. Table 3 shows an example in which global context (words in neighboring sentences) effectively help to judge the semantic similarity between "Airport" and "Airfield." This is while local context (ten neighboring words) are less effective in helping the models to relate the two words.

Furthermore, we study the effect of global context in different POS tag categories. As Figure 4 shows global context has greater impact on A–A and N–N categories. We expect high improvement in the N–N category as noun senses are fairly self-contained and often refer to concrete things. Thus broader (not only local) context is needed to judge their semantic similarity. However, we don't know the reason for improvement on the A–A category as, in context, adjective interpretation is often affected by local context (e.g., the nouns that adjectives modify). One reason for improvement could be because adjectives are often interchangeable and this characteristic makes their meaning to be less sensitive to local context.

Model	**Context**	$\rho\times$**100**
Huang et al. (2012)	LGC	65.7
Chen et al. (2014)	LGC	65.4
Neelakantan et al. (2014)	LGC	69.3
Rothe and Schütze (2015)	LGC	69.8
Pre-trained embeds. (GloVe)	-	60.2
SAE	-	61.1
CSAE	LC	66.4
CSAE	LGC	**70.9***

Table 2: Spearman's ρ correlation between model predictions and human judgments in contextual word similarity. (LC: local context only, LGC: local and global context.)

...No cases in Gibraltar were reported. The **airport** is built on the isthmus which the Spanish Government claim not to have been ceded in the Treaty of Utrecht. Thus the integration of Gibraltar Airport in the Single European Sky system has been blocked by Spain. The 1987 agreement for joint control of the airport with...
...called "Tazi" by the German pilots. On 23 Dec 1942, the Soviet 24th Tank Corps reached nearby Skassirskaya and on 24 Dec, the tanks reached Tatsinskaya. Without any soldiers to defend the **airfield** it was abandoned under heavy fire. In a little under an hour, 108 Ju-52s and 16 Ju-86s took off for Novocherkassk – leaving 72 Ju-52s and many other aircraft burning on the ground. A new base was established...

Table 3: The importance of global context (neighboring sentences) in predicting the semantically similar words (Airport, Airfield).

4.4 Answer Ranking Performance

We evaluate the performance of our model in the answer ranking task in which a model should retrieve correct answers from a set of candidates for test questions. For this evaluation, we rank answers with respect to each test question according to the "relevance score" between question and each answer (Section 3).

The state-of-the-art model for answer ranking on qAns is a pairwise convolutional neural network (PCNN) presented in (Severyn and Moschitti, 2015). PCNN is a supervised model that first maps input question-answer pairs to hidden representations through a standard convolutional neural network (CNN) and then utilizes these representations in a pairwise CNN to compute a relevance score for each pair. This model also utilizes external *word overlap* features for each question-answer pair.[5] PCNN outperforms other competing CNN models (Yu et al., 2014) and models that use

[5]Word overlap and IDF-weighted word overlap computed for (a): all words, and (b): only non-stop words for each question-answer pair (Severyn and Moschitti, 2015).

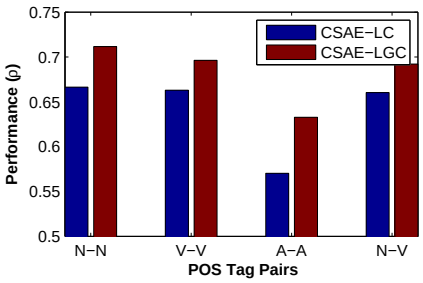


Figure 4: Effect of global context on contextual word similarity in different parts of speech (N: noun, V: verb, A: adjective). We only consider frequent categories.

Model	Train		Train-All	
	MAP	**MRR**	**MAP**	**MRR**
PCNN	62.58	65.91	67.09	72.80
SAE	65.69*	**71.70***	69.54*	75.47*
CSAE	**67.02***	70.99*	**72.29***	**77.29***
PCNN-WO	**73.29**	**79.62**	74.59	80.78
SAE-DST	72.53	76.97	76.38*	82.11*
CSAE-DST	71.26	76.88	**76.75***	**82.90***

Table 4: Answer ranking in *supervised* setting

Model	Train		Train-All	
	MAP	**MRR**	**MAP**	**MRR**
SAE	63.81	69.30	66.37	71.71
CSAE	**64.86***	**69.93***	**66.76***	**73.79***

Table 5: Answer ranking in *unsupervised* setting.

syntax and semantic features (Heilman and Smith, 2010; Yao et al., 2013).

Tables 4 and 5 show the performance of different models in terms of Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) in supervised and unsupervised settings. PCNN-WO and PCNN show the baseline performance with and without word overlap features. SAE and CSAE show the performance of our pairwise autoencoders without and with context information respectively. Their "X-DST" versions show their performance when additional features (Equation 10) are used. These features are computed for the hidden and BOW representations of question-answer pairs. We also include word overlap features as additional features.

Table 4 shows that SAE and CSAE consistently outperform PCNN, and SAE-DST and CSAE-DST outperform PCNN-WO when the models are trained on the larger training dataset, "Train-All." But PCNN shows slightly better performance than our model on "Train," the smaller training dataset. We conjecture this is because PCNN's convolution filter is wider (n-grams, $n > 2$) (Severyn and Moschitti, 2015).

Table 5 shows that the performance of *unsupervised* SAE and CSAE are comparable and in some cases better than the performance of the *supervised* PCNN model. We attribute the high performance of our models to context information that leads to richer representations of inputs.

Furthermore, comparing the performance of CSAE and SAE in both supervised and unsupervised settings in Tables 4 and 5 shows that context information consistently improves the MAP and MRR performance at all settings except for MRR on "Train" (supervised setting) that leads to a comparable performance. Context-sensitive representations significantly improve the performance of our model and often lead to higher MAP than the models that ignore context information.

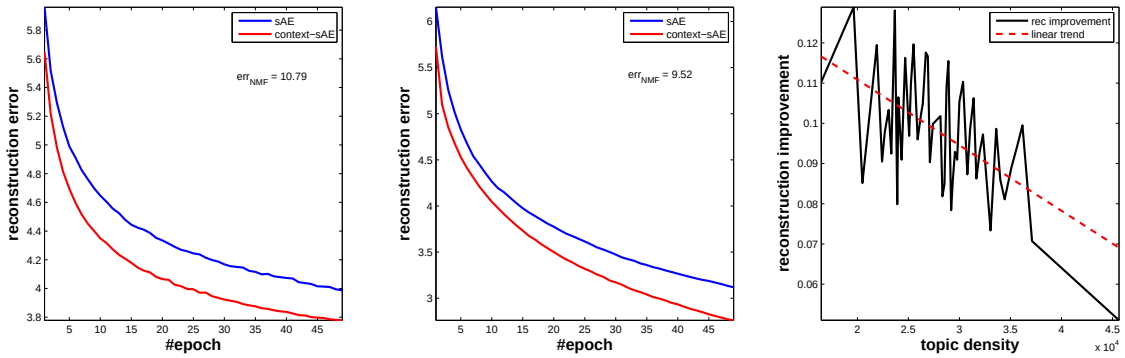
4.5 Question Ranking Performance

In the question ranking task, given a test question, a model should retrieve top-K questions that are semantically equivalent to the test question for $K = \{1, 5, 10\}$. We use qSim for this evaluation.

We compare our autoencoders against PCNN and PBOW-PCNN models presented in Dos Santos et al. (2015). PCNN is a pairwise convolutional neural network and PBOW-PCNN is a joint model that combines vector representations obtained from a pairwise bag-of-words (PBOW) network and a pairwise convolutional neural network (PCNN). Both models are supervised as they require similarity scores to train the network.

Table 6 shows the performance of different models in terms of Precision at Rank K, P@K. CSAE is more precise than the baseline; CSAE and CSAE-DST models consistently outperform the baselines on P@1, an important metric in search applications (CSAE also outperforms PCNN on P@5). Although context-sensitive models are more precise than the baselines at higher ranks, the PCNN and PBOW-PCNN models remain the best model for P@10.

Tables 6 and 7 show that context information consistently improves the results at all ranks in both supervised and unsupervised settings. The performance of the unsupervised SAE and CSAE models are comparable with the supervised PCNN model in higher ranks.



(a) qSim Dataset (b) qAns Dataset (c) qSim Dataset

Figure 5: Reconstruction Error and Improvement: (a) and (b) reconstruction error on qSim and qAns respectively. err_{NMF} shows the reconstruction error of NMF. Smaller error is better, (c) improvement in reconstruction error vs. topic density: greater improvement is obtained in topics with lower density.

Model	P@1	P@5	P@10
PCNN	20.0	33.8	**40.4**
SAE	16.8	29.4	32.8
CSAE	**21.4**	**34.9**	37.2
PBOW-PCNN	22.3	**39.7**	**46.4**
SAE-DST	22.2	35.9	42.0
CSAE-DST	**24.6**	37.9	38.9

Table 6: Question ranking in *supervised* setting

Model	P@1	P@5	P@10
SAE	17.3	32.4	32.8
CSAE	**18.6**	**33.2**	**34.1**

Table 7: Question ranking in *unsupervised* setting

5 Performance Analysis and Discussion

We investigate the effect of context information in reconstructing inputs and try to understand reasons for improvement in reconstruction error. We compute the average reconstruction error of SAE and CSAE (Equations (3) and (7)). For these experiments, we set $\lambda = 0$ in Equation (7) so that we can directly compare the resulting loss of the two models. CSAE will still use context information with $\lambda = 0$ but it *does not* backpropagate the reconstruction loss of context information.

Figures 5(a) and 5(b) show the average reconstruction error of SAE and CSAE on qSim and qAns datasets. Context information conistently improves reconstruction. The improvement is greater on qSim which contains smaller number of words per question as compared to qAns. Also, both models generate smaller reconstruction errors than NMF (Section 2.3). The lower performance of NMF is because it reconstructs inputs merely using global topics identified in datasets, while our models utilize both local and global information to reconstruct inputs.

5.1 Analysis of Context information

The improvement in reconstruction error mainly stems from areas in data where "topic density" is lower. We define topic density for a topic as the number of documents that are assigned to the topic by our topic model. We compute the average improvement in reconstruction error for each topic $\mathcal{T}_j$ using the loss functions for the basic and context-sensitive autoencoders:

$$\Delta_j = \frac{1}{|\mathcal{T}_j|} \sum_{\mathbf{x} \in \mathcal{T}_j} l(\mathbf{x}) - l(\mathbf{x}, \mathbf{h_x})$$

where we set $\lambda = 0$. Figure 5(c) shows improvement of reconstruction error versus topic density on qSim. Lower topic densities have greater improvement. This is because they have insufficient training data to train the networks. However, injecting context information improves the reconstruction power of our model by providing more information. The improvements in denser areas are smaller because neural networks can train effectively in these areas.[6]

5.2 Effect of Depth

The intuition behind deep autoencoders (and, generally, deep neural networks) is that each layer learns a more abstract representation of the input than the previous one (Hinton and Salakhutdinov, 2006; Bengio et al., 2013). We investigate

[6] We observed the same pattern in qAns.

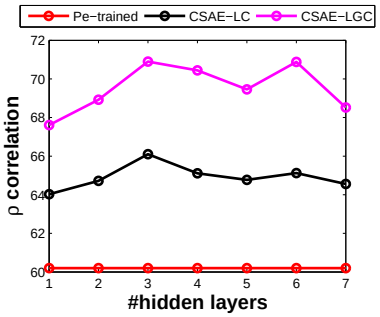


Figure 6: Effect of depth in contextual word similarity. Three hidden layers is optimal for this task.

if adding depth to our context-sensitive autoencoder will improve its performance in the contextual word similarity task.

Figure 6 shows that as we increase the depth of our autoencoders, their performances initially improve. The CSAE-LGC model that uses both local and global context benefits more from greater number of hidden layers than CSAE-LC that only uses local context. We attribute this to the use of global context in CSAE-LGC that leads to more accurate representations of words in their context. We also note that with just a single hidden layer, CSAE-LGC largely improves the performance as compared to CSAE-LC.

6 Related Work

Representation learning models have been effective in many tasks such as language modeling (Bengio et al., 2003; Mikolov et al., 2013b), topic modeling (Nguyen et al., 2015), paraphrase detection (Socher et al., 2011), and ranking tasks (Yih et al., 2013). We briefly review works that use context information for text representation.

Huang et al. (2012) presented an RNN model that uses document-level context information to construct more accurate word representations. In particular, given a sequence of words, the approach uses other words in the document as external (global) knowledge to predict the next word in the sequence. Other approaches have also modeled context at the document level (Lin et al., 2015; Wang and Cho, 2015; Ji et al., 2016).

Ji et al. (2016) presented a context-sensitive RNN-based language model that integrates representations of previous sentences into the language model of the current sentence. They showed that this approach outperforms several RNN language models on a text coherence task.

Liu et al. (2015) proposed a context-sensitive RNN model that uses Latent Dirichlet Allocation (Blei et al., 2003) to extract topic-specific word embeddings. Their best-performing model regards each topic that is associated to a word in a sentence as a pseudo word, learns topic and word embeddings, and then concatenates the embeddings to obtain topic-specific word embeddings.

Mikolov and Zweig (2012) extended a basic RNN language model (Mikolov et al., 2010) by an additional feature layer to integrate external information (such as topic information) about inputs into the model. They showed that such information improves the perplexity of language models.

In contrast to previous research, we integrate context into deep autoencoders. To the best of our knowledge, this is the first work to do so. Also, in this paper, we depart from most previous approaches by demonstrating the value of context information in *sentence-level* semantic similarity and ranking tasks such as QA ranking tasks. Our approach to the ranking problems, both for Answer Ranking and Question Ranking, is different from previous approaches in the sense that we judge the relevance between inputs based on their context information. We showed that adding sentential or document context information about questions (or answers) leads to better rankings.

7 Conclusion and Future Work

We introduce an effective approach to integrate sentential or document context into deep autoencoders and show that such integration is important in semantic similarity tasks. In the future, we aim to investigate other types of linguistic context (such as POS tag and word dependency information, word sense, and discourse relations) and develop a unified representation learning framework that integrates such linguistic context with representation learning models.

Acknowledgments

We thank anonymous reviewers for their thoughtful comments. This paper is based upon work supported, in whole or in part, with funding from the United States Government. Boyd-Graber is supported by NSF grants IIS/1320538, IIS/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 sponsors.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

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

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Jordan L Boyd-Graber and David M Blei. 2009. Syntactic topic models. In *Proceedings of NIPS*.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*.

Stéphane Clinchant and Florent Perronnin. 2013. Aggregating continuous word embeddings for information retrieval. *the Workshop on Continuous Vector Space Models and their Compositionality, ACL*.

Cicero Dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of ACL-IJCNLP*.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL*.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL-IJCNLP*.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2016. Document context language models. *ICLR (Workshop track)*.

S. Kokoska and D. Zwillinger. 2000. *CRC Standard Probability and Statistics Tables and Formulae, Student Edition*. Taylor & Francis.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of EMNLP*.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of EMNLP*.

Chuan-bi Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of AAAI*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Spoken Language Technologies*. IEEE.

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

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

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

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the EMNLP*.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *TACL*, 3:299–313.

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

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL-IJNLP*.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of EMNLP*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of SIGIR*.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of ACL*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*.

Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. 2012. Exploring topic coherence over many models and many topics. In *Proceedings of EMNLP-CNNL*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings ICML*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. *CoRR*, abs/1511.03729.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of EMNLP-CoNLL*.

Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of NAACL*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of ACL*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS, Deep Learning Workshop*.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.