Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings

John Wieting **Kevin Gimpel**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{jwieting,kgimpel}@ttic.edu

Abstract

We consider the problem of learning general-purpose, paraphrastic sentence embeddings, revisiting the setting of Wieting et al. (2016b). While they found LSTM recurrent networks to underperform word averaging, we present several developments that together produce the opposite conclusion. These include training on sentence pairs rather than phrase pairs, averaging states to represent sequences, and regularizing aggressively. These improve LSTMs in both transfer learning and supervised settings. We also introduce a new recurrent architecture, the GATED RECURRENT AVERAGING NETWORK, that is inspired by averaging and LSTMs while outperforming them both. We analyze our learned models, finding evidence of preferences for particular parts of speech and dependency relations. [1]

1 Introduction

Modeling sentential compositionality is a fundamental aspect of natural language semantics. Researchers have proposed a broad range of compositional functional architectures (Mitchell and Lapata, 2008; Socher et al., 2011; Kalchbrenner et al., 2014) and evaluated them on a large variety of applications. Our goal is to learn a general-purpose sentence embedding function that can be used unmodified for measuring semantic textual similarity (STS) (Agirre et al., 2012) and can also serve as a useful initialization for downstream tasks. We wish to learn this embedding function such that sentences with high semantic similarity have high cosine similarity in the embedding space. In particular, we focus on the setting of Wieting et al. (2016b), in which models are trained on noisy paraphrase pairs and evaluated on both STS and supervised semantic tasks.

Surprisingly, Wieting et al. found that simple embedding functions—those based on averaging word vectors—outperform more powerful architectures based on long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). In this paper, we revisit their experimental setting and present several techniques that together improve the performance of the LSTM to be superior to word averaging.

We first change data sources: rather than train on noisy phrase pairs from the Paraphrase Database (PPDB; Ganitkevitch et al., 2013), we use noisy *sentence* pairs obtained automatically by aligning Simple English to standard English Wikipedia (Coster and Kauchak, 2011). Even though this data was intended for use by text simplification systems, we find it to be efficient and effective for learning sentence embeddings, outperforming much larger sets of examples from PPDB.

We then show how we can modify and regularize the LSTM to further improve its performance. The main modification is to simply average the hidden states instead of using the final one. For regularization, we experiment with two kinds of dropout and also with randomly scrambling the words in each input sequence. We find that these techniques help in the transfer learning setting and on two supervised semantic similarity datasets as well. Further gains are obtained on the supervised tasks by initializing with our models from the transfer setting.

Inspired by the strong performance of both averaging and LSTMs, we introduce a novel recurrent neural network architecture which we call

[1] Trained models and code are available at http://ttic.uchicago.edu/~wieting.

the GATED RECURRENT AVERAGING NETWORK (GRAN). The GRAN outperforms averaging and the LSTM in both the transfer and supervised learning settings, forming a promising new recurrent architecture for semantic modeling.

2 Related Work

Modeling sentential compositionality has received a great deal of attention in recent years. A comprehensive survey is beyond the scope of this paper, but we mention popular functional families: neural bag-of-words models (Kalchbrenner et al., 2014), deep averaging networks (DANs) (Iyyer et al., 2015), recursive neural networks using syntactic parses (Socher et al., 2011, 2012, 2013; İrsoy and Cardie, 2014), convolutional neural networks (Kalchbrenner et al., 2014; Kim, 2014; Hu et al., 2014), and recurrent neural networks using long short-term memory (Tai et al., 2015; Ling et al., 2015; Liu et al., 2015). Simple operations based on vector addition and multiplication typically serve as strong baselines (Mitchell and Lapata, 2008, 2010; Blacoe and Lapata, 2012).

Most work cited above uses a supervised learning framework, so the composition function is learned discriminatively for a particular task. In this paper, we are primarily interested in creating general purpose, domain independent embeddings for word sequences. Several others have pursued this goal (Socher et al., 2011; Le and Mikolov, 2014; Pham et al., 2015; Kiros et al., 2015; Hill et al., 2016; Arora et al., 2017; Pagliardini et al., 2017), though usually with the intent to extract useful features for supervised sentence tasks rather than to capture semantic similarity.

An exception is the work of Wieting et al. (2016b). We closely follow their experimental setup and directly address some outstanding questions in their experimental results. Here we briefly summarize their main findings and their attempts at explaining them. They made the surprising discovery that word averaging outperforms LSTMs by a wide margin in the transfer learning setting. They proposed several hypotheses for why this occurs. They first considered that the LSTM was unable to adapt to the differences in sequence length between phrases in training and sentences in test. This was ruled out by showing that neither model showed any strong correlation between sequence length and performance on the test data.

They next examined whether the LSTM was overfitting on the training data, but then showed that both models achieve similar values of the training objective and similar performance on *in-domain* held-out test sets. Lastly, they considered whether their hyperparameters were inadequately tuned, but extensive hyperparameter tuning did not change the story. Therefore, the reason for the performance gap, and how to correct it, was left as an open problem. This paper takes steps toward addressing that problem.

3 Models and Training

3.1 Models

Our goal is to embed a word sequence s into a fixed-length vector. We focus on three compositional models in this paper, all of which use words as the smallest unit of compositionality. We denote the tth word in s as s_t, and we denote its word embedding by x_t.

Our first two models have been well-studied in prior work, so we describe them briefly. The first, which we call AVG, simply averages the embeddings x_t of all words in s. The only parameters learned in this model are those in the word embeddings themselves, which are stored in the word embedding matrix W_w. This model was found by Wieting et al. (2016b) to perform very strongly for semantic similarity tasks.

Our second model uses a long short-term memory (LSTM) recurrent neural network (Hochreiter and Schmidhuber, 1997) to embed s. We use the LSTM variant from Gers et al. (2003) including its "peephole" connections. We consider two ways to obtain a sentence embedding from the LSTM. The first uses the final hidden vector, which we denote h_{-1}. The second, denoted LSTMAVG, averages all hidden vectors of the LSTM. In both variants, the learnable parameters include both the LSTM parameters W_c and the word embeddings W_w.

Inspired by the success of the two models above, we propose a third model, which we call the GATED RECURRENT AVERAGING NETWORK (GRAN). The GATED RECURRENT AVERAGING NETWORK combines the benefits of AVG and LSTMs. In fact it reduces to AVG if the output of the gate is all ones. We first use an LSTM to generate a hidden vector, h_t, for each word s_t in

s. Then we use h_t to compute a gate that will be elementwise-multiplied with x_t, resulting in a new, gated hidden vector a_t for each step t:

$$a_t = x_t \odot \sigma(W_x x_t + W_h h_t + b) \qquad (1)$$

where W_x and W_h are parameter matrices, b is a parameter vector, and σ is the elementwise logistic sigmoid function. After all a_t have been generated for a sentence, they are averaged to produce the embedding for that sentence. This model includes as learnable parameters those of the LSTM, the word embeddings, and the additional parameters in Eq. (1). For both the LSTM and GRAN models, we use W_c to denote the "compositional" parameters, i.e., all parameters other than the word embeddings.

The motivation for the GRAN is that we are contextualizing the word embeddings prior to averaging. The gate can be seen as an attention, attending to the prior context of the sentence.[2]

We also experiment with four other variations of this model, though they generally were more complex and showed inferior performance. In the first, GRAN-2, the gate is applied to h_t (rather than x_t) to produce a_t, and then these a_t are averaged as before.

GRAN-3 and GRAN-4 use two gates: one applied to x_t and one applied to a_{t-1}. We tried two different ways of computing these gates: for each gate i, $\sigma(W_{x_i} x_t + W_{h_i} h_t + b_i)$ (GRAN-3) or $\sigma(W_{x_i} x_t + W_{h_i} h_t + W_{a_i} a_{t-1} + b_i)$ (GRAN-4). The sum of these two terms comprised a_t. In this model, the last average hidden state, a_{-1}, was used as the sentence embedding after dividing it by the length of the sequence. In these models, we are additionally keeping a running average of the embeddings that is being modified by the context at every time step. In GRAN-4, this running average is also considered when producing the contextualized word embedding.

Lastly, we experimented with a fifth GRAN, GRAN-5, in which we use two gates, calculated by $\sigma(W_{x_i} x_t + W_{h_i} h_t + b_i)$ for each gate i. The first is applied to x_t and the second is applied to h_t. The output of these gates is then summed. Therefore GRAN-5 can be reduced to either word-averaging or averaging LSTM states, depending on the behavior of the gates. If the first gate is all ones and the second all zeros throughout the sequence, the model is equivalent to word-averaging. Conversely, if the first gate is all zeros and the second is all ones throughout the sequence, the model is equivalent to averaging the LSTM states. Further analysis of these models is included in Section 4.

3.2 Training

We follow the training procedure of Wieting et al. (2015) and Wieting et al. (2016b), described below. The training data consists of a set S of phrase or sentence pairs $\langle s_1, s_2\rangle$ from either the Paraphrase Database (PPDB; Ganitkevitch et al., 2013) or the aligned Wikipedia sentences (Coster and Kauchak, 2011) where s_1 and s_2 are assumed to be paraphrases. We optimize a margin-based loss:

$$\min_{W_c, W_w} \frac{1}{|S|} \Bigg(\sum_{\langle s_1, s_2\rangle \in S} \max(0, \delta - \cos(g(s_1), g(s_2)) + \cos(g(s_1), g(t_1))) + \max(0, \delta - \cos(g(s_1), g(s_2)) + \cos(g(s_2), g(t_2))) \Bigg) + \lambda_c \left\| W_c \right\|^2 + \lambda_w \left\| W_{w_{initial}} - W_w \right\|^2 \qquad (2)$$

where g is the model in use (e.g., AVG or LSTM), δ is the margin, λ_c and λ_w are regularization parameters, $W_{w_{initial}}$ is the initial word embedding matrix, and t_1 and t_2 are carefully-selected negative examples taken from a mini-batch during optimization. The intuition is that we want the two phrases to be more similar to each other $(\cos(g(s_1), g(s_2)))$ than either is to their respective negative examples t_1 and t_2, by a margin of at least δ.

3.2.1 Selecting Negative Examples

To select t_1 and t_2 in Eq. (2), we simply choose the most similar phrase in some set of phrases (other than those in the given phrase pair). For simplicity we use the mini-batch for this set, but it could be a different set. That is, we choose t_1 for a given $\langle s_1, s_2\rangle$ as follows:

$$t_1 = \operatorname*{argmax}_{t:\langle t, \cdot\rangle \in S_b \setminus \{\langle s_1, s_2\rangle\}} \cos(g(s_1), g(t))$$

where $S_b \subseteq S$ is the current mini-batch. That is, we want to choose a negative example t_i that is similar to s_i according to the current model. The downside is that we may occasionally choose a phrase t_i that is actually a true paraphrase of s_i.

[2] We tried a variant of this model without the gate. We obtain a_t from $f(W_x x_t + W_h h_t + b)$, where f is a nonlinearity, tuned over tanh and ReLU. The performance of the model is significantly worse than the GRAN in all experiments.

4 Experiments

Our experiments are designed to address the empirical question posed by Wieting et al. (2016b): why do LSTMs underperform AVG for transfer learning? In Sections 4.1.2-4.2, we make progress on this question by presenting methods that bridge the gap between the two models in the transfer setting. We then apply these same techniques to improve performance in the supervised setting, described in Section 4.3. In both settings we also evaluate our novel GRAN architecture, finding it to consistently outperform both AVG and the LSTM.

4.1 Transfer Learning

4.1.1 Datasets and Tasks

We train on large sets of noisy paraphrase pairs and evaluate on a diverse set of 22 textual similarity datasets, including all datasets from every SemEval semantic textual similarity (STS) task from 2012 to 2015. We also evaluate on the SemEval 2015 Twitter task (Xu et al., 2015) and the SemEval 2014 SICK Semantic Relatedness task (Marelli et al., 2014). Given two sentences, the aim of the STS tasks is to predict their similarity on a 0-5 scale, where 0 indicates the sentences are on different topics and 5 indicates that they are completely equivalent. We report the average Pearson's r over these 22 sentence similarity tasks.

Each STS task consists of 4-6 datasets covering a wide variety of domains, including newswire, tweets, glosses, machine translation outputs, web forums, news headlines, image and video captions, among others. Further details are provided in the official task descriptions (Agirre et al., 2012, 2013, 2014, 2015).

4.1.2 Experiments with Data Sources

We first investigate how different sources of training data affect the results. We try two data sources. The first is phrase pairs from the Paraphrase Database (PPDB). PPDB comes in different sizes (S, M, L, XL, XXL, and XXXL), where each larger size subsumes all smaller ones. The pairs in PPDB are sorted by a confidence measure and so the smaller sets contain higher precision paraphrases. PPDB is derived automatically from naturally-occurring bilingual text, and versions of PPDB have been released for many languages without the need for any manual annotation (Ganitkevitch and Callison-Burch, 2014).

	AVG	LSTM	LSTMAVG
PPDB	67.7	54.2	64.2
SimpWiki	68.4	59.3	67.5

Table 1: Test results on SemEval semantic textual similarity datasets (Pearson's $r \times 100$) when training on different sources of data: phrase pairs from PPDB or simple-to-standard English Wikipedia sentence pairs from Coster and Kauchak (2011).

The second source of data is a set of sentence pairs automatically extracted from Simple English Wikipedia and English Wikipedia articles by Coster and Kauchak (2011). This data was extracted for developing text simplification systems, where each instance pairs a simple and complex sentence representing approximately the same information. Though the data was obtained for simplification, we use it as a source of training data for learning paraphrastic sentence embeddings. The dataset, which we call SimpWiki, consists of 167,689 sentence pairs.

To ensure a fair comparison, we select a sample of pairs from PPDB XL such that the number of tokens is approximately the same as the number of tokens in the SimpWiki sentences.[3]

We use PARAGRAM-SL999 embeddings (Wieting et al., 2015) to initialize the word embedding matrix (W_w) for all models. For all experiments, we fix the mini-batch size to 100, and λ_c to 0. We tune the margin δ over $\{0.4, 0.6, 0.8\}$ and λ_w over $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 0\}$. We train AVG for 7 epochs, and the LSTM for 3, since it converges much faster and does not benefit from 7 epochs. For optimization we use Adam (Kingma and Ba, 2015) with a learning rate of 0.001. We use the 2016 STS tasks (Agirre et al., 2016) for model selection, where we average the Pearson's r over its 5 datasets. We refer to this type of model selection as *test*. For evaluation, we report the average Pearson's r over the 22 other sentence similarity tasks.

The results are shown in Table 1. We first note that, when training on PPDB, we find the same result as Wieting et al. (2016b): AVG outperforms the LSTM by more than 13 points. However, when training both on sentence pairs, the gap shrinks to about 9 points. It appears that part of the inferior performance for the LSTM in prior work was due

[3] The PPDB data consists of 1,341,188 phrase pairs and contains 3 more tokens than the SimpWiki data.

to training on phrase pairs rather than on sentence pairs. The AVG model also benefits from training on sentences, but not nearly as much as the LSTM.[4]

Our hypothesis explaining this result is that in PPDB, the phrase pairs are short fragments of text which are not necessarily constituents or phrases in any syntactic sense. Therefore, the sentences in the STS test sets are quite different from the fragments seen during training. We hypothesize that while word-averaging is relatively unaffected by this difference, the recurrent models are much more sensitive to overall characteristics of the word sequences, and the difference between train and test matters much more.

These results also suggest that the SimpWiki data, even though it was developed for text simplification, may be useful for other researchers working on semantic textual similarity tasks.

4.1.3 Experiments with LSTM Variations

We next compare LSTM and LSTMAVG. The latter consists of averaging the hidden vectors of the LSTM rather than using the final hidden vector as in prior work (Wieting et al., 2016b). We hypothesize that the LSTM may put more emphasis on the words at the end of the sentence than those at the beginning. By averaging the hidden states, the impact of all words in the sequence is better taken into account. Averaging also makes the LSTM more like AVG, which we know to perform strongly in this setting.

The results on AVG and the LSTM models are shown in Table 1. When training on PPDB, moving from LSTM to LSTMAVG improves performance by 10 points, closing most of the gap with AVG. We also find that LSTMAVG improves by moving from PPDB to SimpWiki, though in both cases it still lags behind AVG.

4.2 Experiments with Regularization

We next experiment with various forms of regularization. Previous work (Wieting et al., 2016b,a) only used L_2 regularization. Wieting et al. (2016b) also regularized the word embeddings back to their initial values. Here we use L_2 regularization as well as several additional regularization methods we describe below.

We try two forms of dropout. The first is just standard dropout (Srivastava et al., 2014) on the word embeddings. The second is "word dropout", which drops out entire word embeddings with some probability (Iyyer et al., 2015).

We also experiment with scrambling the inputs. For a given mini-batch, we go through each sentence pair and, with some probability, we shuffle the words in each sentence in the pair. When scrambling a sentence pair, we always shuffle both sentences in the pair. We do this before selecting negative examples for the mini-batch. The motivation for scrambling is to make it more difficult for the LSTM to memorize the sequences in the training data, forcing it to focus more on the identities of the words and less on word order. Hence it will be expected to behave more like the word averaging model.[5]

We also experiment with combining scrambling and dropout. In this setting, we tune over scrambling with either word dropout or dropout.

The settings for these experiments are largely the same as those of the previous section with the exception that we tune λ_w over a smaller set of values: $\{10^{-5}, 0\}$. When using L_2 regularization, we tune λ_c over $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. When using dropout, we tune the dropout rate over $\{0.2, 0.4, 0.6\}$. When using scrambling, we tune the scrambling rate over $\{0.25, 0.5, 0.75\}$. We also include a bidirectional model ("Bi") for both LSTMAVG and the GATED RECURRENT AVERAGING NETWORK. We tune over two ways to combine the forward and backward hidden states; the first simply adds them together and the second uses a single feedforward layer with a $\tanh$ activation.

We try two approaches for model selection. The first, *test*, is the same as was done in Section 4.1.2,

[4] We experimented with adding EOS tags at the end of training and test sentences, SOS tags at the start of training and test sentences, adding both, and adding neither. We treated adding these tags as hyperparameters and tuned over these four settings along with the other hyperparameters in the original experiment. Interestingly, we found that adding these tags, especially EOS, had a large effect on the LSTM when training on SimpWiki, improving performance by 6 points. When training on PPDB, adding EOS tags only improved performance by 1.6 points.

The addition of the tags had a smaller effect on LSTMAVG. Adding EOS tags improved performance by 0.3 points on SimpWiki and adding SOS tags on PPDB improved performance by 0.9 points.

[5] We also tried some variations on scrambling that did not yield significant improvements: scrambling after obtaining the negative examples, partially scrambling by performing n swaps where n comes from a Poisson distribution with a tunable λ, and scrambling individual sentences with some probability instead of always scrambling both in the pair.

Model	Regularization	Oracle	2016 STS
AVG	none	68.5	68.4
	dropout	68.4	68.3
	word dropout	68.3	68.3
LSTM	none	60.6	59.3
	L_2	60.3	56.5
	dropout	58.1	55.3
	word dropout	66.2	65.3
	scrambling	66.3	65.1
	dropout, scrambling	68.4	68.4
LSTMAVG	none	67.7	67.5
	dropout, scrambling	69.2	68.6
BiLSTMAVG	dropout, scrambling	**69.4**	**68.7**

Table 2: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) when experimenting with different regularization techniques.

Model	Oracle	STS 2016
GRAN (no reg.)	68.0	68.0
GRAN	69.5	**68.9**
GRAN-2	68.8	68.1
GRAN-3	69.0	67.2
GRAN-4	68.6	68.1
GRAN-5	66.1	64.8
BiGRAN	**69.7**	68.4

Table 3: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) for the GRAN architectures. The first row, marked as (no reg.) is the GRAN without any regularization. The other rows show the result of the various GRAN models using dropout and scrambling.

where we use the average Pearson's r on the 5 2016 STS datasets. The second tunes based on the average Pearson's r of all 22 datasets in our evaluation. We refer to this as *oracle*.

The results are shown in Table 2. They show that dropping entire word embeddings and scrambling input sequences is very effective in improving the result of the LSTM, while neither type of dropout improves AVG. Moreover, averaging the hidden states of the LSTM is the most effective modification to the LSTM in improving performance. All of these modifications can be combined to significantly improve the LSTM, finally allowing it to overtake AVG.

In Table 3, we compare the various GRAN architectures. We find that the GRAN provides a small improvement over the best LSTM configuration, possibly because of its similarity to AVG. It also outperforms the other GRAN models, despite being the simplest.

In Table 4, we show results on all individual STS evaluation datasets after using STS 2016 for model selection (unidirectional models only). The LSTMAVG and GATED RECURRENT AVERAGING NETWORK are more closely correlated in performance, in terms of Spearman's ρ and Pearson'r r, than either is to AVG. But they do differ significantly in some datasets, most notably in those comparing machine translation output with its reference. Interestingly, both the LSTMAVG and GATED RECURRENT AVERAGING NETWORK significantly outperform AVG in the datasets focused on comparing glosses like *OnWN* and *FNWN*. Upon examination, we found that these datasets, especially 2013 *OnWN*, contain examples of low similarity with high word overlap. For example, the pair ⟨*the act of preserving or protecting something.*, *the act of decreasing or reducing something.*⟩ from 2013 *OnWN* has a gold similarity score of 0.4. It appears that AVG was fooled by the high amount of word overlap in such pairs, while the other two models were better able to recognize the semantic differences.

Dataset	LSTMAVG	AVG	GRAN
MSRpar	**49.0**	45.9	47.7
MSRvid	84.3	85.1	**85.2**
SMT-eur	**51.2**	47.5	49.3
OnWN	**71.5**	71.2	71.5
SMT-news	**68.0**	58.2	58.7
STS 2012 Average	**64.8**	61.6	62.5
headline	**77.3**	76.9	76.1
OnWN	81.2	72.8	**81.4**
FNWN	53.2	50.2	**55.6**
SMT	**40.7**	38.0	40.3
STS 2013 Average	63.1	59.4	**63.4**
deft forum	**56.6**	55.6	55.7
deft news	78.0	**78.5**	77.1
headline	74.5	**75.1**	72.8
images	84.7	85.6	**85.8**
OnWN	84.9	81.4	**85.1**
tweet news	76.3	**78.7**	78.7
STS 2014 Average	75.8	75.8	**75.9**
answers-forums	71.8	70.6	**73.1**
answers-students	71.1	**75.8**	72.9
belief	75.3	76.8	**78.0**
headline	79.5	**80.3**	78.6
images	85.8	**86.0**	85.8
STS 2015 Average	76.7	**77.9**	77.7
2014 SICK	71.3	72.4	**72.9**
2015 Twitter	**52.1**	**52.1**	50.2

Table 4: Results on SemEval textual similarity datasets (Pearson's $r \times 100$). The highest score in each row is in boldface.

4.3 Supervised Text Similarity

We also investigate if these techniques can improve LSTM performance on supervised semantic textual similarity tasks. We evaluate on two supervised datasets. For the first, we start with the 20 SemEval STS datasets from 2012-2015 and then use 40% of each dataset for training, 10% for validation, and the remaining 50% for testing. There are 4,481 examples in training, 1,207 in validation, and 6,060 in the test set. The second is the SICK 2014 dataset, using its standard training, validation, and test sets. There are 4,500 sentence pairs in the training set, 500 in the development set, and 4,927 in the test set. The SICK task is an easier learning problem since the training examples are all drawn from the same distribution, and they are mostly shorter and use simpler language. As these are supervised tasks, the sentence pairs in the training set contain manually-annotated semantic similarity scores.

We minimize the loss function[6] from Tai et al. (2015). Given a score for a sentence pair in the range $[1, K]$, where K is an integer, with sentence representations h_L and h_R, and model parameters θ, they first compute:

$$
\begin{aligned}
&h_\times = h_L \odot h_R, \;\; h_+ = |h_L - h_R|,\\
&h_s = \sigma\left(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}\right),\\
&\hat{p}_\theta = \text{softmax}\left(W^{(p)} h_s + b^{(p)}\right),\\
&\hat{y} = r^T \hat{p}_\theta,
\end{aligned}
$$

where $r^T = [1 \; 2 \; \ldots \; K]$. They then define a sparse target distribution p that satisfies $y = r^T p$:

$$
p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}
$$

for $1 \leq i \leq K$. Then they use the following loss, the regularized KL-divergence between p and $\hat{p}_\theta$:

$$
J(\theta) = \frac{1}{m} \sum_{k=1}^{m} \text{KL}\left(p^{(k)} \, \middle\| \, \hat{p}_\theta^{(k)}\right),
$$

where m is the number of training pairs.

We experiment with the LSTM, LSTMAVG, and AVG models with dropout, word dropout, and scrambling tuning over the same hyperparameter as in Section 4.2. We again regularize the word embeddings back to their initial state, tuning λ_w over $\{10^{-5}, 0\}$. We used the validation set for each respective dataset for model selection.

The results are shown in Table 5. The GATED RECURRENT AVERAGING NETWORK has the best performance on both datasets. Dropout helps the word-averaging model in the STS task, unlike in the transfer learning setting. The LSTM benefits slightly from dropout, scrambling, and averaging on their own individually with the exception of word dropout on both datasets and averaging on the SICK dataset. However, when combined, these modifications are able to significantly improve the performance of the LSTM, bringing it much closer in performance to AVG. This experiment indicates that these modifications when training LSTMs are beneficial outside the transfer learning setting, and can potentially be used to improve performance for the broad range of problems that use LSTMs to model sentences.

Model	Regularization	STS	SICK	Avg.
AVG	none	79.2	85.2	82.2
	dropout	80.7	84.5	82.6
	word dropout	79.3	81.8	80.6
LSTM	none	68.4	80.9	74.7
	dropout	69.6	81.3	75.5
	word dropout	68.0	76.4	72.2
	scrambling	74.2	84.4	79.3
	dropout, scrambling	75.0	84.2	79.6
LSTMAVG	none	69.0	79.5	74.3
	dropout	69.2	79.4	74.3
	word dropout	65.6	76.1	70.9
	scrambling	76.5	83.2	79.9
	dropout, scrambling	76.5	84.0	80.3
GRAN	none	79.7	85.2	82.5
	dropout	79.7	84.6	82.2
	word dropout	77.3	83.0	80.2
	scrambling	81.4	**85.3**	**83.4**
	dropout, scrambling	**81.6**	85.1	**83.4**

Table 5: Results from supervised training on the STS and SICK datasets (Pearson's $r \times 100$). The last column is the average result on the two datasets.

Model	STS	SICK	Avg.
GRAN	**81.6**	85.3	**83.5**
GRAN-2	77.4	85.1	81.3
GRAN-3	81.3	85.4	83.4
GRAN-4	80.1	**85.5**	82.8
GRAN-5	70.9	83.0	77.0

Table 6: Results from supervised training on the STS and SICK datasets (Pearson's $r \times 100$) for the GRAN architectures. The last column is the average result on the two datasets.

In Table 6 we compare the various GRAN architectures under the same settings as the previous experiment. We find that the GRAN still has the best overall performance.

We also experiment with initializing the supervised models using our pretrained sentence model

[6] This objective function has been shown to perform very strongly on text similarity tasks, significantly better than squared or absolute error.

#	Sentence 1	Sentence 2	LAVG	AVG	Gold
1	the lamb is looking at the camera.	a cat looking at the camera.	**3.42**	4.13	0.8
2	he also said shockey is "living the dream life of a new york athlete.	"jeremy's a good guy," barber said, adding:"jeremy is living the dream life of the new york athlete.	**3.55**	4.22	2.75
3	bloomberg chips in a billion	bloomberg gives $1.1 b to university	**3.99**	3.04	4.0
4	in other regions, the sharia is imposed.	in other areas, sharia law is being introduced by force.	**4.44**	3.72	4.75
5	three men in suits sitting at a table.	two women in the kitchen looking at a object.	3.33	**2.79**	0.0
6	we never got out of it in the first place!	where does the money come from in the first place?	4.00	**3.33**	0.8
7	two birds interacting in the grass.	two dogs play with each other outdoors.	3.44	**2.81**	0.2

Table 7: Illustrative sentence pairs from the STS datasets showing errors made by LSTMAVG and AVG. The last three columns show the gold similarity score, the similarity score of LSTMAVG, and the similarity score of AVG. Boldface indicates smaller error compared to gold scores.

Model	Regularization	STS	SICK
AVG	dropout	80.7	84.5
	dropout, universal	**82.9**	85.6
LSTMAVG	dropout, scrambling	76.5	84.0
	dropout, scrambling, universal	81.3	85.2
GRAN	dropout, scrambling	81.6	85.1
	dropout, scrambling, universal	82.7	**86.0**

Table 8: Impact of initializing and regularizing toward universal models (Pearson's $r \times 100$) in supervised training.

parameters, for the AVG model (no regularization), LSTMAVG (dropout, scrambling), and GATED RECURRENT AVERAGING NETWORK (dropout, scrambling) models from Table 2 and Table 3. We both initialize and then regularize back to these initial values, referring to this setting as "universal".[7]

The results are shown in Table 8. Initializing and regularizing to the pretrained models significantly improves the performance for all three models, justifying our claim that these models serve a dual purpose: they can be used a black box semantic similarity function, and they possess rich knowledge that can be used to improve the performance of downstream tasks.

5 Analysis

5.1 Error Analysis

We analyze the predictions of AVG and the recurrent networks, represented by LSTMAVG, on the 20 STS datasets. We choose LSTMAVG as it correlates slightly less strongly with AVG than the GRAN on the results over all SemEval datasets used for evaluation. We scale the models' cosine similarities to lie within $[0, 5]$, then compare the predicted similarities of LSTMAVG and AVG to the gold similarities. We analyzed instances in which each model would tend to overestimate or underestimate the gold similarity relative to the other. These are illustrated in Table 7.

We find that AVG tends to overestimate the semantic similarity of a sentence pair, relative to LSTMAVG, when the two sentences have a lot of word or synonym overlap, but have either important differences in key semantic roles or where one sentence has significantly more content than the other. These phenomena are shown in examples 1 and 2 in Table 7. Conversely, AVG tends to underestimate similarity when there are one-word-to-multiword paraphrases between the two sentences as shown in examples 3 and 4.

LSTMAVG tends to overestimate similarity when the two inputs have similar sequences of syntactic categories, but the meanings of the sentences are different (examples 5, 6, and 7). Instances of LSTMAVG underestimating the similarity relative to AVG are relatively rare, and those that we found did not have any systematic patterns.

5.2 GRAN Gate Analysis

We also investigate what is learned by the gating function of the GATED RECURRENT AVERAGING NETWORK. We are interested to see whether its estimates of importance correlate with those of traditional syntactic and (shallow) semantic analysis.

We use the oracle trained GATED RECURRENT AVERAGING NETWORK from Table 3 and calculate the L_1 norm of the gate after embedding 10,000 sentences from English Wikipedia.[8] We also automatically tag and parse these sentences using the Stanford dependency parser (Manning et al., 2014). We then compute

[7] In these experiments, we tuned λ_w over $\{10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 0\}$ and λ_c over $\{10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 0\}$.

[8] We selected only sentences of less than or equal to 15 tokens to ensure more accurate parsing.

POS		Dep. Label	
top 10	bot. 10	top 10	bot. 10
NNP	TO	number	possessive
NNPS	WDT	nn	cop
CD	POS	num	det
NNS	DT	acomp	auxpass
VBG	WP	appos	prep
NN	IN	pobj	cc
JJ	CC	vmod	mark
UH	PRP	dobj	aux
VBN	EX	amod	expl
JJS	WRB	conj	neg

Table 9: POS tags and dependency labels with highest and lowest average GATED RECURRENT AVERAGING NETWORK gate L_1 norms. The lists are ordered from highest norm to lowest in the top 10 columns, and lowest to highest in the bottom 10 columns.

Dep. Label	Weight
xcomp	170.6
acomp	167.1
root	157.4
amod	143.1
advmod	121.6

Table 10: Average L_1 norms for adjectives (JJ) with selected dependency labels.

the average gate L_1 norms for particular part-of-speech tags, dependency arc labels, and their conjunction.

Table 9 shows the highest/lowest average norm tags and dependency labels. The network prefers nouns, especially proper nouns, as well as cardinal numbers, which is sensible as these are among the most discriminative features of a sentence.

Analyzing the dependency relations, we find that nouns in the object position tend to have higher weight than nouns in the subject position. This may relate to topic and focus; the object may be more likely to be the "new" information related by the sentence, which would then make it more likely to be matched by the other sentence in the paraphrase pair.

We find that the weights of adjectives depend on their position in the sentence, as shown in Table 10. The highest norms appear when an adjective is an xcomp, acomp, or root; this typically means it is residing in an object-like position in its clause. Adjectives that modify a noun (amod) have medium weight, and those that modify another adjective or verb (advmod) have low weight.

Lastly, we analyze words tagged as VBG, a highly ambiguous tag that can serve many syntactic roles in a sentence. As shown in Table 11, we find that when they are used to modify a noun (amod) or in the object position of a clause (xcomp, pcomp) they have high weight. Medium weight appears when used in verb phrases (root, vmod) and low weight when used as prepositions or auxiliary verbs (prep, auxpass).

Dep. Label	Weight
pcomp	190.0
amod	178.3
xcomp	176.8
vmod	170.6
root	161.8
auxpass	125.4
prep	121.2

Table 11: Average L_1 norms for words with the tag VBG with selected dependency labels.

6 Conclusion

We showed how to modify and regularize LSTMs to improve their performance for learning paraphrastic sentence embeddings in both transfer and supervised settings. We also introduced a new recurrent network, the GATED RECURRENT AVERAGING NETWORK, that improves upon both AVG and LSTMs for these tasks, and we release our code and trained models.

Furthermore, we analyzed the different errors produced by AVG and the recurrent methods and found that the recurrent methods were learning composition that wasn't being captured by AVG. We also investigated the GRAN in order to better understand the compositional phenomena it was learning by analyzing the L_1 norm of its gate over various inputs.

Future work will explore additional data sources, including from aligning different translations of novels (Barzilay and McKeown, 2001), aligning new articles of the same topic (Dolan et al., 2004), or even possibly using machine translation systems to translate bilingual text into paraphrastic sentence pairs. Our new techniques, combined with the promise of new data sources, offer a great deal of potential for improved universal paraphrastic sentence embeddings.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We thank the developers of Theano (Theano Development Team, 2016) and NVIDIA Corporation for donating GPUs used in this research.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. *Proceedings of SemEval* pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting on Association for Computational Linguistics*.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of HLT-NAACL*.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2003. Learning precise timing with LSTM recurrent networks. *The Journal of Machine Learning Research* 3.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8).

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science* 34(8).

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. *arXiv preprint arXiv:1703.02507* .

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

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 *Advances in Neural Information Processing Systems*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on 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. *The Journal of Machine Learning Research* 15(1).

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016b. Towards universal paraphrastic sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL (TACL)* .

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. SemEval-2015 task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*.