# TOWARDS UNIVERSAL PARAPHRASTIC SENTENCE EMBEDDINGS

**John Wieting** **Mohit Bansal** **Kevin Gimpel** **Karen Livescu**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{jwieting,mbansal,kgimpel,klivescu}@ttic.edu



## ABSTRACT

We consider the problem of learning general-purpose, paraphrastic sentence embeddings based on supervision from the Paraphrase Database (Ganitkevitch et al., 2013). We compare six compositional architectures, evaluating them on annotated textual similarity datasets drawn both from the same distribution as the training data and from a wide range of other domains. We find that the most complex architectures, such as long short-term memory (LSTM) recurrent neural networks, perform best on the in-domain data. However, in out-of-domain scenarios, simple architectures such as word averaging vastly outperform LSTMs. Our simplest averaging model is even competitive with systems tuned for the particular tasks while also being extremely efficient and easy to use.

In order to better understand how these architectures compare, we conduct further experiments on three supervised NLP tasks: sentence similarity, entailment, and sentiment classification. We again find that the word averaging models perform well for sentence similarity and entailment, outperforming LSTMs. However, on sentiment classification, we find that the LSTM performs very strongly—even recording new state-of-the-art performance on the Stanford Sentiment Treebank.

We then demonstrate how to combine our pretrained sentence embeddings with these supervised tasks, using them both as a prior and as a black box feature extractor. This leads to performance rivaling the state of the art on the SICK similarity and entailment tasks. We release all of our resources to the research community[1] with the hope that they can serve as the new baseline for further work on universal sentence embeddings.

## 1 INTRODUCTION

Word embeddings have become ubiquitous in natural language processing (NLP). Several researchers have developed and shared word embeddings trained on large datasets (Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014), and these have been used effectively for many downstream tasks (Turian et al., 2010; Socher et al., 2011; Kim, 2014; Bansal et al., 2014; Tai et al., 2015). There has also been recent work on creating representations for word sequences such as phrases or sentences. Many functional architectures have been proposed to model compositionality in such sequences, ranging from those based on simple operations like addition (Mitchell & Lapata, 2010; Yu & Dredze, 2015; Iyyer et al., 2015) to those based on richly-structured functions like recursive neural networks (Socher et al., 2011), convolutional neural networks (Kalchbrenner et al., 2014), and recurrent neural networks using long short-term memory (LSTM) (Tai et al., 2015). However, there is little work on learning sentence representations that can be used across domains with the same ease and effectiveness as word embeddings. In this paper, we explore compositional models that can encode arbitrary word sequences into a vector with the property that sequences with similar meaning have high cosine similarity, and that can, importantly, also transfer easily across domains. We consider six compositional architectures based on neural networks and train them on noisy phrase pairs from the Paraphrase Database (PPDB; Ganitkevitch et al., 2013).

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

We consider models spanning the range of complexity from word averaging to LSTMs. With the simplest word averaging model, there are no additional compositional parameters. The only parameters are the word vectors themselves, which are learned to produce effective sequence embeddings when averaging is performed over the sequence. We add complexity by adding layers, leading to variants of deep averaging networks (Iyyer et al., 2015). We next consider several recurrent network variants, culminating in LSTMs because they have been found to be effective for many types of sequential data (Graves et al., 2008; 2013; Greff et al., 2015), including text (Sutskever et al., 2014; Vinyals et al., 2014; Xu et al., 2015a; Hermann et al., 2015; Ling et al., 2015; Wen et al., 2015).

To evaluate our models, we consider two tasks drawn from the same distribution as the training data, as well as 22 SemEval textual similarity datasets from a variety of domains (such as news, tweets, web forums, and image and video captions). Interestingly, we find that the LSTM performs well on the in-domain task, but performs much worse on the out-of-domain tasks. We discover surprisingly strong performance for the models based on word averaging, which perform well on both the in-domain and out-of-domain tasks, beating the best LSTM model by 16.5 Pearson's $r$ on average. Moreover, we find that learning word embeddings in the context of vector averaging performs much better than simply averaging pretrained, state-of-the-art word embeddings. Our average Pearson's $r$ over all 22 SemEval datasets is 17.1 points higher than averaging GloVe vectors[2] and 12.8 points higher than averaging PARAGRAM-SL999 vectors.[3]

Our final sentence embeddings[4] place in the top 25% of all submitted systems in every SemEval STS task from 2012 through 2015, being best or tied for best on 4 of the datasets.[5] This is surprising because the submitted systems were designed for those particular tasks, with access to training and tuning data specifically developed for each task.

While the above experiments focus on transfer, we also consider the fully supervised setting (Table 5). We compare the same suite of compositional architectures for three supervised NLP tasks: sentence similarity and textual entailment using the 2014 SemEval SICK dataset (Marelli et al., 2014), and sentiment classification using the Stanford Sentiment Treebank (Socher et al., 2013). We again find strong performance for the word averaging models for both similarity and entailment, outperforming the LSTM. However, for sentiment classification, we see a different trend. The LSTM now performs best, achieving 89.2% on the coarse-grained sentiment classification task. This result, to our knowledge, is the new state of the art on this task.

We then demonstrate how to combine our PPDB-trained sentence embedding models with supervised NLP tasks. We first use our model as a prior, yielding performance on the similarity and entailment tasks that rivals the state of the art. We also use our sentence embeddings as an effective black box feature extractor for downstream tasks, comparing favorably to recent work (Kiros et al., 2015).

We release our strongest sentence embedding model, which we call PARAGRAM-PHRASE XXL, to the research community.[6] Since it consists merely of a new set of word embeddings, it is extremely efficient and easy to use for downstream applications. Our hope is that this model can provide a new simple and strong baseline in the quest for universal sentence embeddings.

## 2 RELATED WORK

Researchers have developed many ways to embed word sequences for NLP. They mostly focus on the question of compositionality: given vectors for words, how should we create a vector for a word sequence? Mitchell & Lapata (2008; 2010) considered bigram compositionality, comparing many functions for composing two word vectors into a single vector to represent their bigram. Follow-up work by Blacoe & Lapata (2012) found again that simple operations such as vector ad-

[2]We used the publicly available 300-dimensional vectors that were trained on the 840 billion token Common Crawl corpus, available at `http://nlp.stanford.edu/projects/glove/`.

[3]These are 300-dimensional vectors from Wieting et al. (2015) and are available at `http://ttic.uchicago.edu/~wieting`. They give human-level performance on two commonly used word similarity datasets, WordSim353 (Finkelstein et al., 2001) and Simlex-999 (Hill et al., 2015).

[4]Denoted PARAGRAM-PHRASE-XXL and discussed in Section 4.3.

[5]As measured by the average Pearson's $r$ over all datasets in each task; see Table 4.

[6]Available at `http://ttic.uchicago.edu/~wieting`.

dition performed strongly. Many other compositional architectures have been proposed. Some have been based on distributional semantics (Baroni et al., 2014; Paperno et al., 2014; Polajnar et al., 2015; Tian et al., 2015), while the current trend is toward development of neural network architectures. These include neural bag-of-words models (Kalchbrenner et al., 2014), deep averaging networks (DANs) (Iyyer et al., 2015), feature-weighted averaging (Yu & Dredze, 2015), recursive neural networks based on parse structure (Socher et al., 2011; 2012; 2013; İrsoy & Cardie, 2014; Wieting et al., 2015), recursive networks based on non-syntactic hierarchical structure (Zhao et al., 2015; Chen et al., 2015b), convolutional neural networks (Kalchbrenner et al., 2014; Kim, 2014; Hu et al., 2014; Yin & Schütze, 2015; He et al., 2015), and recurrent neural networks using long short-term memory (Tai et al., 2015; Ling et al., 2015; Liu et al., 2015). In this paper, we compare six architectures: word averaging, word averaging followed by a single linear projection, DANs, and three variants of recurrent neural networks, including LSTMs.[7]

Most of the work mentioned above learns compositional models in the context of *supervised* learning. That is, a training set is provided with annotations and the composition function is learned for the purposes of optimizing an objective function based on those annotations. The models are then evaluated on a test set drawn from the same distribution as the training set.

In this paper, in contrast, we are primarily interested in creating general purpose, domain independent embeddings for word sequences. There have been research efforts also targeting this goal. One approach is to train an autoencoder in an attempt to learn the latent structure of the sequence, whether it be a sentence with a parse tree (Socher et al., 2011), or a longer sequence such as a paragraph or document (Li et al., 2015b). Other recently proposed methods, including paragraph vectors (Le & Mikolov, 2014) and skip-thought vectors (Kiros et al., 2015), learn sequence representations that are predictive of words inside the sequence or in neighboring sequences. These methods produce generic representations that can be used to provide features for text classification or sentence similarity tasks. While skip-thought vectors capture similarity in terms of discourse context, in this paper we are interested in capturing paraphrastic similarity, i.e., whether two sentences have the same meaning.

Our learning formulation draws from a large body of related work on learning input representations in order to maximize similarity in the learned space (Weston et al., 2010; Yih et al., 2011; Huang et al., 2013; Hermann & Blunsom, 2014; Socher et al., 2014; Faruqui & Dyer, 2014; Bordes et al., 2014b;a; Lu et al., 2015), including our prior work (Wieting et al., 2015). We focus our exploration here on modeling and keep the learning methodology mostly fixed, though we do include certain choices about the learning procedure in our hyperparameter tuning space for each model.

## 3 Models and Training

Our goal is to embed sequences into a low-dimensional space such that cosine similarity in the space corresponds to the strength of the paraphrase relationship between the sequences. We experimented with six models of increasing complexity. The simplest model embeds a word sequence $x = \langle x_1, x_2, ..., x_n \rangle$ by averaging the vectors of its tokens. The only parameters learned by this model are the word embedding matrix $W_w$:

$$g_{\text{PARAGRAM-PHRASE}}(x) = \frac{1}{n}\sum_{i}^{n} W_w^{x_i}$$

where $W_w^{x_i}$ is the word embedding for word $x_i$. We call the learned embeddings PARAGRAM-PHRASE embeddings.

In our second model, we learn a projection in addition to the word embeddings:

$$g_{\text{proj}}(x) = W_p\left(\frac{1}{n}\sum_{i}^{n} W_w^{x_i}\right) + b$$

[7] In prior work, we experimented with recursive neural networks on binarized parses of the PPDB (Wieting et al., 2015), but we found that many of the phrases in PPDB are not sentences or even constituents, causing the parser to have unexpected behavior.

where $W_p$ is the projection matrix and $b$ is a bias vector.

Our third model is the deep averaging network (DAN) of Iyyer et al. (2015). This is a generalization of the above models that typically uses multiple layers as well as nonlinear activation functions. In our experiments below, we tune over the number of layers and choice of activation function.

Our fourth model is a standard recurrent network (RNN) with randomly initialized weight matrices and nonlinear activations:

$$h_t = f(W_x W_w^{x_t} + W_h h_{t-1} + b)$$
$$g_{\text{RNN}}(x) = h_{-1}$$

where $f$ is the activation function (either $\tanh$ or rectified linear unit; the choice is tuned), $W_x$ and $W_h$ are parameter matrices, $b$ is a bias vector, and $h_{-1}$ refers to the hidden vector of the last token.

Our fifth model is a special RNN which we call an *identity-RNN*. In the identity-RNN, the weight matrices are initialized to identity, the bias is initialized to zero, and the activation is the identity function. We divide the final output vector of the identity-RNN by the number of tokens in the sequence. Thus, before any updates to the parameters, the identity-RNN simply averages the word embeddings. We also regularize the identity-RNN parameters to their initial values. The idea is that, with high regularization, the identity-RNN is simply averaging word embeddings. However, it is a richer architecture and can take into account word order and hopefully improve upon the averaging baseline.

Our sixth and final model is the most expressive. We use long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997), a recurrent neural network (RNN) architecture designed to model sequences with long-distance dependencies. LSTMs have recently been shown to produce state-of-the-art results in a variety of sequence processing tasks (Chen et al., 2015a; Filippova et al., 2015; Xu et al., 2015c; Belinkov & Glass, 2015; Wang & Nyberg, 2015). We use the version from Gers et al. (2003) which has the following equations:

$$i_t = \sigma\left(W_{xi} W_w^{x_t} + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i\right)$$
$$f_t = \sigma\left(W_{xf} W_w^{x_t} + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f\right)$$
$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc} W_w^{x_t} + W_{hc} h_{t-1} + b_c\right)$$
$$o_t = \sigma\left(W_{xo} W_w^{x_t} + W_{ho} h_{t-1} + W_{co} c_t + b_o\right)$$
$$h_t = o_t \tanh(c_t)$$
$$g_{\text{LSTM}}(x) = h_{-1}$$

where $\sigma$ is the logistic sigmoid function. We found that the choice of whether or not to include the output gate had a significant impact on performance, so we used two versions of the LSTM model, one with the output gate and one without. For all models, we learn the word embeddings themselves, denoting the trainable word embedding parameters by $W_w$. We denote all other trainable parameters by $W_c$ ("compositional parameters"), though the PARAGRAM-PHRASE model has no compositional parameters. We initialize $W_w$ using some embeddings pretrained from large corpora.

### 3.1 TRAINING

We mostly follow the approach of Wieting et al. (2015). The training data consists of (possibly noisy) pairs taken directly from the original Paraphrase Database (PPDB) and we optimize a margin-based loss.

Our training data consists of a set $X$ of phrase pairs $\langle x_1, x_2 \rangle$, where $x_1$ and $x_2$ are assumed to be paraphrases. The objective function follows:

$$\min_{W_c, W_w} \frac{1}{|X|} \Bigg( \sum_{\langle x_1, x_2 \rangle \in X} \max(0, \delta - \cos(g(x_1), g(x_2)) + \cos(g(x_1), g(t_1))) \\ + \max(0, \delta - \cos(g(x_1), g(x_2)) + \cos(g(x_2), g(t_2))) \Bigg) \\ + \lambda_c \left\| W_c \right\|^2 + \lambda_w \left\| W_{w_{initial}} - W_w \right\|^2 \quad (1)$$

where $g$ is the embedding function in use (e.g., $g_{\text{LSTM}}$), $\delta$ is the margin, $\lambda_c$ and $\lambda_w$ 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 ($\cos(g(x_1), g(x_2))$) than either is to their respective negative examples $t_1$ and $t_2$, by a margin of at least $\delta$.

#### 3.1.1 SELECTING NEGATIVE EXAMPLES

To select $t_1$ and $t_2$ in Eq. 1, we tune the choice between two approaches. The first, MAX, simply chooses the most similar phrase in some set of phrases (other than those in the given phrase pair). For simplicity and to reduce the number of tunable parameters, we use the mini-batch for this set, but it could be a separate set. Formally, MAX corresponds to choosing $t_1$ for a given $\langle x_1, x_2 \rangle$ as follows:

$$t_1 = \underset{t:\langle t,\cdot\rangle \in X_b \setminus \{\langle x_1, x_2\rangle\}}{\operatorname{argmax}} \cos(g(x_1), g(t))$$

where $X_b \subseteq X$ is the current mini-batch. That is, we want to choose a negative example $t_i$ that is similar to $x_i$ according to the current model parameters. The downside of this approach is that we may occasionally choose a phrase $t_i$ that is actually a true paraphrase of $x_i$.

The second strategy selects negative examples using MAX with probability 0.5 and selects them randomly from the mini-batch otherwise. We call this sampling strategy MIX. We tune over the strategy in our experiments.

## 4 EXPERIMENTS

### 4.1 DATA

We experiment on 24 textual similarity datasets, covering many domains, including all datasets from every SemEval semantic textual similarity (STS) task (2012-2015). We also evaluate on the SemEval 2015 Twitter task (Xu et al., 2015b) and the SemEval 2014 Semantic Relatedness task (Marelli et al., 2014), as well as two tasks that use PPDB data (Wieting et al., 2015; Pavlick et al., 2015).

The first STS task was held in 2012 and these tasks have been held every year since. Given two sentences, the objective of the task is to predict how similar they are on a 0-5 scale, where 0 indicates the sentences are on different topics and 5 indicates that they are completely equivalent. Each STS task consists of 4-6 different datasets and the tasks cover a wide variety of domains which we have categorized below. Most submissions for these tasks use supervised models that are trained and tuned on either provided training data or similar datasets from older tasks. Details on the number of teams and submissions for each task and the performance of the submitted systems for each dataset are included in Table 1 and Table 2 respectively. For more details on these tasks please refer to the relevant publications for the 2012 (Agirre et al., 2012), 2013 (Agirre et al., 2013), 2014 (Agirre et al., 2014), and 2015 (Agirre et al., 2015) tasks.

| Dataset | No. of teams | No. of submissions |
|---|---|---|
| 2012 STS | 35 | 88 |
| 2013 STS | 34 | 89 |
| 2014 STS | 15 | 38 |
| 2015 STS | 29 | 74 |
| 2014 SICK | 17 | 66 |
| 2015 Twitter | 19 | 26 |

Table 1: Details on numbers of teams and submissions in the STS tasks used for evaluation.

Below are the textual domains contained in the STS tasks:
**News**: Newswire was used in the 2012 task (MSRpar) and the 2013 and 2014 tasks (deft news).
**Image and Video Descriptions**: Image descriptions generated via crowdsourcing were used in the 2013 and 2014 tasks (images). Video descriptions were used in the 2012 task (MSRvid).
**Glosses**: Glosses from WordNet, OntoNotes, and FrameNet were used in the 2012, 2013, and 2014 tasks (OnWN and FNWN).

**MT evaluation**: The output of machine translation systems with their reference translations was used in the 2012 task (SMT-eur and SMT-news) and the 2013 task (SMT).
**Headlines**: Headlines of news articles were used in the 2013, 2014, and 2015 tasks (headline).
**Web Forum**: Forum posts were used in the 2014 task (deft forum).
**Twitter**: Pairs containing a tweet related to a news headline and a sentence pertaining to the same news headline. This dataset was used in the 2014 task (tweet news).
**Belief**: Text from the Deft Committed Belief Annotation (LDC2014E55) was used in the 2015 task (belief).
**Questions and Answers**: Paired answers to the same question from StackExchange (answers-forums) and the BEETLE corpus (Dzikovska et al., 2010) (answers-students) were used in 2015.

For tuning, we use two datasets that contain PPDB phrase pairs scored by human annotators on the strength of their paraphrase relationship. One is a large sample of 26,456 annotated phrase pairs developed by Pavlick et al. (2015). The second, called Annotated-PPDB, was developed in our prior work (Wieting et al., 2015) and is a small set of 1,000 annotated phrase pairs that were filtered to focus on challenging paraphrase phenomena.

### 4.2 Transfer Learning

#### 4.2.1 Experimental Settings

As training data, we used the XL section[8] of PPDB which contains 3,033,753 unique phrase pairs. However, for hyperparameter tuning we only used 100k examples sampled from PPDB XXL and trained for 5 epochs. Then after finding the hyperparameters that maximize Spearman's $\rho$ on the Pavlick et al. PPDB task, we trained on the entire XL section of PPDB for 10 epochs. We used PARAGRAM-SL999 embeddings to initialize the word embedding matrix ($W_w$) for all models.

We chose the Pavlick et al. task for tuning because we wanted our entire procedure to only make use of PPDB and use no other resources. In particular, we did not want to use any STS tasks for training or hyperparameter tuning. We chose the Pavlick et al. dataset over Annotated-PPDB due to its larger size. But in practice the datasets are very similar and tuning on either produces similar results.

To learn model parameters for all experiments in this section, we minimize Eq. 1. Our models have the following tunable hyperparameters:[9] $\lambda_c$, the $\mathrm{L}_2$ regularizer on the compositional parameters $W_c$ (not applicable for the word averaging model), the pool of phrases used to obtain negative examples (coupled with mini-batch size $B$, to reduce the number of tunable hyperparameters), $\lambda_w$, the regularizer on the word embeddings, and $\delta$, the margin. We also tune over optimization method (either AdaGrad (Duchi et al., 2011) or Adam (Kingma & Ba, 2014)), learning rate (from $\{0.05, 0.005, 0.0005\}$), whether to clip the gradients with threshold 1 (Pascanu et al., 2012), and whether to use MIX or MAX sampling. For the classic RNN, we further tuned whether to use $\tanh$ or rectified linear unit activation functions; for the identity-RNN, we tuned $\lambda_c$ over $\{1000, 100, 10, 1\}$ because we wanted higher regularization on the composition parameters; for the DANs we tuned over activation function ($\tanh$ or rectified linear unit) and the number of layers (either 1 or 2); for the LSTMs we tuned on whether to include an output gate. We fix the output dimensionalities of all models that require doing so to the dimensionality of our word embeddings (300).

#### 4.2.2 Results

The results on all STS tasks as well as the SICK and Twitter tasks are shown in Table 2. We include results on the PPDB tasks in Table 3. In Table 2, we first show the median, $75^{\text{th}}$ percentile, and highest score from the official task rankings. We then report the performance of our seven models: PARAGRAM-PHRASE (PP), identity-RNN (iRNN), projection (proj.), deep-averaging network (DAN), recurrent neural network (RNN), LSTM with output gate (o.g.), and LSTM without output

[8]PPDB comes in different sizes (S, M, L, XL, XXL, and XXXL), where each larger size subsumes all smaller ones. The phrases are sorted by a confidence measure and so the smaller sets contain higher precision paraphrases.

[9]For $\lambda_c$ we searched over $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$, for $b$ we searched over $\{25, 50, 100\}$, for $\lambda_w$ we searched over $\{10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ as well as the setting in which we do not update $W_w$, and for $\delta$ we searched over $\{0.4, 0.6, 0.8\}$.

gate (no o.g.). We compare to three baselines: skip-thought vectors[10] (Kiros et al., 2015), denoted "ST", averaged GloVe[11] vectors (Pennington et al., 2014), and averaged PARAGRAM-SL999 vectors (Wieting et al., 2015), denoted "PSL". Note that the GloVe vectors were used to initialize the PARAGRAM-SL999 vectors which were, in turn, used to initialize our PARAGRAM-PHRASE embeddings. We compare to skip-thought vectors because trained models are publicly available and they show impressive performance when used as features on several tasks including textual similarity.

| Dataset | 50% | 75% | Max | PP | proj. | DAN | RNN | iRNN | LSTM (no o.g.) | LSTM (o.g.) | ST | GloVe | PSL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSRpar | 51.5 | 57.6 | 73.4 | 42.6 | 43.7 | 40.3 | 18.6 | 43.4 | 16.1 | 9.3 | 16.8 | **47.7** | 41.6 |
| MSRvid | 75.5 | 80.3 | 88.0 | **74.5** | 74.0 | 70.0 | 66.5 | 73.4 | 71.3 | 71.3 | 41.7 | 63.9 | 60.0 |
| SMT-eur | 44.4 | 48.1 | 56.7 | 47.3 | **49.4** | 43.8 | 40.9 | 47.1 | 41.8 | 44.3 | 35.2 | 46.0 | 42.4 |
| OnWN | 608 | 65.9 | 72.7 | **70.6** | 70.1 | 65.9 | 63.1 | 70.1 | 65.2 | 56.4 | 29.7 | 55.1 | 63.0 |
| SMT-news | 40.1 | 45.4 | 60.9 | 58.4 | **62.8** | 60.0 | 51.3 | 58.1 | 60.8 | 51.0 | 30.8 | 49.6 | 57.0 |
| STS 2012 Average | 54.5 | 59.5 | 70.3 | 58.7 | **60.0** | 56.0 | 48.1 | 58.4 | 51.0 | 46.4 | 30.8 | 52.5 | 52.8 |
| headline | 64.0 | 68.3 | 78.4 | 72.4 | 72.6 | 71.2 | 59.5 | **72.8** | 57.4 | 48.5 | 34.6 | 63.8 | 68.8 |
| OnWN | 52.8 | 64.8 | 84.3 | 67.7 | 68.0 | 64.1 | 54.6 | **69.4** | 68.5 | 50.4 | 10.0 | 49.0 | 48.0 |
| FNWN | 32.7 | 38.1 | 58.2 | 43.9 | **46.8** | 43.1 | 30.9 | 45.3 | 24.7 | 38.4 | 30.4 | 34.2 | 37.9 |
| SMT | 31.8 | 34.6 | 40.4 | 39.2 | **39.8** | 38.3 | 33.8 | 39.4 | 30.1 | 28.8 | 24.3 | 22.3 | 31.0 |
| STS 2013 Average | 45.3 | 51.4 | 65.3 | 55.8 | **56.8** | 54.2 | 44.7 | 56.7 | 45.2 | 41.5 | 24.8 | 42.3 | 46.4 |
| deft forum | 36.6 | 46.8 | 53.1 | 48.7 | **51.1** | 49.0 | 41.5 | 49.0 | 44.2 | 46.1 | 12.9 | 27.1 | 37.2 |
| deft news | 66.2 | 74.0 | 78.5 | **73.1** | 72.2 | 71.7 | 53.7 | 72.4 | 52.8 | 39.1 | 23.5 | 68.0 | 67.0 |
| headline | 67.1 | 75.4 | 78.4 | 69.7 | **70.8** | 69.2 | 57.5 | 70.2 | 57.5 | 50.9 | 37.8 | 59.5 | 65.3 |
| images | 75.6 | 79.0 | 83.4 | **78.5** | 78.1 | 76.9 | 67.6 | 78.2 | 68.5 | 62.9 | 51.2 | 61.0 | 62.0 |
| OnWN | 78.0 | 81.1 | 87.5 | 78.8 | **79.5** | 75.7 | 67.7 | 78.8 | 76.9 | 61.7 | 23.3 | 58.4 | 61.1 |
| tweet news | 64.7 | 72.2 | 79.2 | 76.4 | 75.8 | 74.2 | 58.0 | **76.9** | 58.7 | 48.2 | 39.9 | 51.2 | 64.7 |
| STS 2014 Average | 64.7 | 71.4 | 76.7 | 70.9 | **71.3** | 69.5 | 57.7 | 70.9 | 59.8 | 51.5 | 31.4 | 54.2 | 59.5 |
| answers-forums | 61.3 | 68.2 | 73.9 | **68.3** | 65.1 | 62.6 | 32.8 | 67.4 | 51.9 | 50.7 | 36.1 | 30.5 | 38.8 |
| answers-students | 67.6 | 73.6 | 78.8 | **78.2** | 77.8 | 78.1 | 64.7 | 78.2 | 71.5 | 55.7 | 33.0 | 63.0 | 69.2 |
| belief | 67.7 | 72.2 | 77.2 | **76.2** | 75.4 | 72.0 | 51.9 | 75.9 | 61.7 | 52.6 | 24.6 | 40.5 | 53.2 |
| headline | 74.2 | 80.8 | 84.2 | 74.8 | **75.2** | 73.5 | 65.3 | 75.1 | 64.0 | 56.6 | 43.6 | 61.8 | 69.0 |
| images | 80.4 | 84.3 | 87.1 | **81.4** | 80.3 | 77.5 | 71.4 | 81.1 | 70.4 | 64.2 | 17.7 | 67.5 | 69.9 |
| STS 2015 Average | 70.2 | 75.8 | 80.2 | **75.8** | 74.8 | 72.7 | 57.2 | 75.6 | 63.9 | 56.0 | 31.0 | 52.7 | 60.0 |
| 2014 SICK | 71.4 | 79.9 | 82.8 | 71.6 | **71.6** | 70.7 | 61.2 | 71.2 | 63.9 | 59.0 | 49.8 | 65.9 | 66.4 |
| 2015 Twitter | 49.9 | 52.5 | 61.9 | 52.9 | 52.8 | **53.7** | 45.1 | 52.9 | 47.6 | 36.1 | 24.7 | 30.3 | 36.3 |

Table 2: Results on SemEval textual similarity datasets (Pearson's $r \times 100$). The highest score in each row is in boldface (omitting the official task score columns).

The results in Table 2 show strong performance of our two simplest models: the PARAGRAM-PHRASE embeddings (PP) and our projection model (proj.). They outperform the other models on all but 5 of the 22 datasets. The iRNN model has the next best performance, while the LSTM models lag behind. These results stand in marked contrast to those in Table 3, which shows very similar performance across models on the in-domain PPDB tasks, with the LSTM models slightly outperforming the others. For the LSTM models, it is also interesting to note that removing the output gate results in stronger performance on the textual similarity tasks. Removing the output gate improves performance on 18 of the 22 datasets. The LSTM without output gate also performs reasonably well compared to our strong PARAGRAM-SL999 addition baseline, beating it on 12 of the 22 datasets.

### 4.3 PARAGRAM-PHRASE XXL

Since we found that PARAGRAM-PHRASE embeddings have such strong performance, we trained this model on more data from PPDB and also used more data for hyperparameter tuning. For tuning, we used all of PPDB XL and trained for 10 epochs, then trained our final model for 10 epochs on the entire phrase section of PPDB XXL, consisting of 9,123,575 unique phrase pairs.[12] We show the results of this improved model, which we call PARAGRAM-PHRASE XXL, in Table 4. We also report the median, $75^{\text{th}}$ percentile, and maximum score from our suite of textual similarity tasks.

[10]Note that we pre-processed the training data with the tokenizer from Stanford CoreNLP (Manning et al., 2014) rather than the included NLTK (Bird et al., 2009) tokenizer. We found that doing so significantly improves the performance of the skip-thought vectors.

[11]We used the publicly available 300-dimensional vectors that were trained on the 840 billion token Common Crawl corpus, available at `http://nlp.stanford.edu/projects/glove/`.

[12]We fixed batchsize to 100 and $\delta$ to 0.4, as these were the optimal values for the experiment in Table 2. Then, for $\lambda_w$ we searched over $\{10^{-6}, 10^{-7}, 10^{-8}\}$, and tuned over MIX and MAX sampling. To optimize, we used AdaGrad with a learning rate of 0.05.

| Model | Pavlick et al. (oracle) | Pavlick et al. (test) | Annotated-PPDB (test) |
|---|---|---|---|
| PARAGRAM-PHRASE | 60.3 | 60.0 | 53.5 |
| projection | 61.0 | 58.4 | 52.8 |
| DAN | 60.9 | 60.1 | 52.3 |
| RNN | 60.5 | 60.3 | 51.8 |
| iRNN | 60.3 | 60.0 | **53.9** |
| LSTM (no o.g.) | **61.6** | **61.3** | 53.4 |
| LSTM (o.g.) | 61.5 | 60.9 | 52.9 |
| skip-thought | 39.3 | 39.3 | 31.9 |
| GloVe | 44.8 | 44.8 | 25.3 |
| PARAGRAM-SL999 | 55.3 | 55.3 | 40.4 |

Table 3: Results on the PPDB tasks (Spearman's $\rho \times 100$). For the task in Pavlick et al. (2015), we include the oracle result (the max Spearman's $\rho$ on the dataset), since this dataset was used for model selection for all other tasks, as well as test results where models were tuned on Annotated-PPDB.

| Dataset | 50% | 75% | Max | PARAGRAM-PHRASE-XXL |
|---|---|---|---|---|
| MSRpar | 51.5 | 57.6 | 73.4 | 44.8 |
| MSRvid | 75.5 | 80.3 | 88.0 | 79.6 |
| SMT-eur | 44.4 | 48.1 | 56.7 | 49.5 |
| OnWN | 60.8 | 65.9 | 72.7 | 70.4 |
| SMT-news | 40.1 | 45.4 | 60.9 | **63.3** |
| STS 2012 Average | 54.5 | 59.5 | 70.3 | 61.5 |
| headline | 64.0 | 68.3 | 78.4 | 73.9 |
| OnWN | 52.8 | 64.8 | 84.3 | 73.8 |
| FNWN | 32.7 | 38.1 | 58.2 | 47.7 |
| SMT | 31.8 | 34.6 | 40.4 | **40.4** |
| STS 2013 Average | 45.3 | 51.4 | 65.3 | 58.9 |
| deft forum | 36.6 | 46.8 | 53.1 | **53.4** |
| deft news | 66.2 | 74.0 | 78.5 | 74.4 |
| headline | 67.1 | 75.4 | 78.4 | 71.5 |
| images | 75.6 | 79.0 | 83.4 | 80.4 |
| OnWN | 78.0 | 81.1 | 87.5 | 81.5 |
| tweet news | 64.7 | 72.2 | 79.2 | 77.4 |
| STS 2014 Average | 64.7 | 71.4 | 76.7 | 73.1 |
| answers-forums | 61.3 | 68.2 | 73.9 | 69.1 |
| answers-students | 67.6 | 73.6 | 78.8 | 78.0 |
| belief | 67.7 | 72.2 | 77.2 | **78.2** |
| headline | 74.2 | 80.8 | 84.2 | 76.4 |
| images | 80.4 | 84.3 | 87.1 | 83.4 |
| STS 2015 Average | 70.2 | 75.8 | 80.2 | 77.0 |
| 2014 SICK* | 71.4 | 79.9 | 82.8 | 72.7 |
| 2015 Twitter | 49.9 | 52.5 | 61.9 | 52.4 |

Table 4: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) for PARAGRAM-PHRASE XXL embeddings. Results that match or exceed the best shared task system are shown in bold. *For the 2014 SICK task, the median, $75^{\text{th}}$ percentile, and maximum include only the primary runs as the full set of results was not available.

PARAGRAM-PHRASE XXL matches or exceeds the best performance on 4 of the datasets (SMT-news, SMT, deft forum, and belief) and is within 3 points of the best performance on 8 out of 22. We have made this trained model available to the research community.[13]

### 4.4 USING REPRESENTATIONS IN LEARNED MODELS

We explore two natural questions regarding our representations learned from PPDB: (1) can these embeddings improve the performance of other models through initialization and regularization? (2) can they effectively be used as features for downstream tasks? To address these questions, we

[13]Available at `http://ttic.uchicago.edu/~wieting`.

| Task | word averaging | proj. | DAN | RNN | LSTM (no o.g.) | LSTM (o.g.) | w/ *universal* regularization |
|---|---|---|---|---|---|---|---|
| similarity (SICK) | **86.40** | 85.93 | 85.96 | 73.13 | 85.45 | 83.41 | **86.84** |
| entailment (SICK) | **84.6** | 84.0 | 84.5 | 76.4 | 83.2 | 82.0 | **85.3** |
| binary sentiment (SST) | 83.0 | 83.0 | 83.4 | 86.5 | 86.6 | **89.2** | 86.9 |

Table 5: Results from supervised training of each compositional architecture on similarity, entailment, and sentiment tasks. The last column shows results regularizing to our *universal* parameters from the models in Table 2. The first row shows Pearson's $r \times 100$ and the last two show accuracy.

used three tasks: The SICK similarity task, the SICK entailment task, and the Stanford Sentiment Treebank (SST) binary classification task (Socher et al., 2013). For the SICK similarity task, we minimize the objective function[14] 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 $\theta$, they first compute:

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

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 $\hat{p}_\theta$:

$$
J(\theta) = \frac{1}{m} \sum_{k=1}^{m} \mathrm{KL}\left(p^{(k)} \,\middle\|\, \hat{p}_\theta^{(k)}\right), \tag{2}
$$

where $m$ is the number of training pairs and where we always use $\mathrm{L}_2$ regularization on all compositional parameters[15] but omit these terms for clarity.

We use nearly the same model for the entailment task, with the only differences being that the final softmax layer has three outputs and the cost function is the negative log-likelihood of the class labels. For sentiment, since it is a binary sentence classification task, we first encoded the sentence and then used a fully-connected layer with a sigmoid activation followed by a softmax layer with two outputs. We used negative log-likelihood of the class labels as the cost function. All models use $\mathrm{L}_2$ regularization on all parameters, except for the word embeddings, which are regularized back to their initial values with an $\mathrm{L}_2$ penalty.

We first investigated how these models performed in the standard setting, without using any models trained using PPDB data. We tuned hyperparameters on the development set of each dataset[16] as well as on two optimization schemes: AdaGrad with learning rate of 0.05 and Adam with a learning rate of 0.001. We trained the models for 10 epochs and initialized the word embeddings with PARAGRAM-SL999 embeddings.

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

[15]Word embeddings are regularized toward their initial state.

[16]For all models, we tuned batch-size over $\{25, 50, 100\}$, output dimension over $\{50, 150, 300\}$, $\lambda_c$ over $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$, $\lambda_s = \lambda_c$, and $\lambda_w$ over $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ as well as the option of not updating the embeddings for all models except the word averaging model. We again fix the output dimensionalities of all models which require this specification, to the dimensionality of our word embeddings (300). Additionally, for the classic RNN, we further tuned whether to use $\tanh$ or rectified linear unit activation functions; for the DANs we tuned over activation function ($\tanh$ or rectified linear unit) and the number of layers (either 1 or 2).

The results are shown in Table 5. We find that using word averaging as the compositional architecture outperforms the other architectures for similarity and entailment. However, for sentiment classification, the LSTM is much stronger than the averaging models. This suggests that the superiority of a compositional architecture can vary widely depending on the evaluation, and motivates future work to compare these architectures on additional tasks.

These results are very competitive with the state of the art on these tasks. Recent strong results on the SICK similarity task include 86.86 using a convolutional neural network (He et al., 2015) and 86.76 using a tree-LSTM (Tai et al., 2015). For entailment, the best result we are aware of is 85.1 (Beltagy et al., 2015). On sentiment, the best previous result is 88.1 (Kim, 2014), which our LSTM surprisingly outperforms by a significant margin. We note that these experiments simply compare compositional architectures using only the provided training data for each task, tuning on the respective development sets. We did not use any PPDB data for these results, other than that used to train the initial PARAGRAM-SL999 embeddings. Our results appear to show that standard neural architectures can perform surprisingly well given strong word embeddings and thorough tuning over the hyperparameter space.

#### 4.4.1 REGULARIZATION AND INITIALIZATION TO IMPROVE TEXTUAL SIMILARITY MODELS

In this setting, we initialize each respective model to the parameters learned from PPDB (calling them *universal* parameters) and augment Eq. 2 with three separate regularization terms with the following weights: $\lambda_s$ which regularizes the classification parameters (the two layers used in the classification step after obtaining representations), $\lambda_w$ for regularizing the word parameters toward the learned $W_w$ from PPDB, and $\lambda_c$ for regularizing the compositional parameters (for all models except for the word averaging model) back to their initial values.[17] In all cases, we regularize to the universal parameters using $\mathrm{L}_2$ regularization.

The results are shown in the last column of Table 5, and we only show results for the best performing models on each task (word averaging for similarity/entailment, LSTM with output gate for sentiment). Interestingly, it seems that regularizing to our universal parameters significantly improves results for the similarity and entailment tasks which are competitive or better than the state-of-the-art, but harms the LSTM's performance on the sentiment classification task.

#### 4.4.2 REPRESENTATIONS AS FEATURES

| Task | PARAGRAM-PHRASE | | | skip-thought | |
|---|---|---|---|---|---|
| | 300 | 1200 | 2400 | uni-skip | bi-skip |
| similarity (SICK) | 82.15 | 82.85 | **84.94** | 84.77 | 84.05 |
| entailment (SICK) | 80.2 | 80.1 | **83.1** | - | - |
| binary sentiment (SST) | **79.7** | 78.8 | 79.4 | - | - |

Table 6: Results from supervised training on similarity, entailment, and sentiment tasks, except that we keep the sentence representations fixed to our PARAGRAM-PHRASE model. The first row shows Pearson's $r \times 100$ and the last two show accuracy, with boldface showing the highest score in each row.

We also investigate how our PARAGRAM-PHRASE embeddings perform as features for supervised tasks. We use a similar set-up as in Kiros et al. (2015) and encode the sentences by averaging our PARAGRAM-PHRASE embeddings and then just learn the classification parameters without updating the embeddings. To provide a more apt comparison to skip-thought vectors, we also learned a linear projection matrix to increase dimensionality of our PARAGRAM-PHRASE embeddings. We chose 1200 and 2400 dimensions in order to both see the dependence of dimension on performance, and so that they can be compared fairly with skip-thought vectors. Note that 2400 dimensions is the same dimensionality as the uni-skip and bi-skip models in Kiros et al. (2015).

The 300 dimension case corresponds to the PARAGRAM-PHRASE embeddings from Table 2. We tuned our higher dimensional models on PPDB as described previously in Section 4.2.2 before train-

[17] We tuned $\lambda_s$ over $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$, $\lambda_c$ over $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$, and $\lambda_w$ over $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$. All other hyperparameters were tuned as previously described.

ing on PPDB XL.[18] Then we trained the same models for the similarity, entailment, and sentiment tasks as described in Section 4.4 for 20 epochs. We again tuned $\lambda_s$ over $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ and tuned over the two optimization schemes of AdaGrad with learning rate of 0.05 and Adam with a learning rate of 0.001. Note that we are not updating the word embeddings or the projection matrix during training.

The results are shown in Table 6. The similarity and entailment tasks show clear improvements as we project the embeddings into the 2400 dimensional space. In fact, our results outperform both types of skip-thought embeddings on the single task that we overlap. However, the sentiment task does not benefit from higher dimensional representations, which is consistent with our regularization experiments in which sentiment also did not show improvement. Therefore, it seems that our models learned from PPDB are more effective for *similarity* tasks than *classification* tasks, but this hypothesis requires further investigation.

## 5 DISCUSSION

It is interesting that the LSTM, with or without output gates, is outperformed by much simpler models on the similarity and entailment tasks studied in this paper. We now consider possible explanations for this trend.

The first hypothesis we test is based on length. Since PPDB contains short text snippets of a few words, the LSTM may not know how to handle the longer sentences that occur in our evaluation tasks. If this is true, the LSTM would perform much better on short text snippets and its performance would degrade as their length increases. To test this hypothesis, we took all 12,108 pairs from the 20 SemEval STS tasks and binned them by length.[19] We then computed the Pearson's $r$ for each bin. The results are shown in Table 7 and show that while the LSTM models do perform better on the shortest text pairs, they are still outperformed, at all lengths, by the PARAGRAM-PHRASE model.[20]

| Max Length | PARAGRAM-PHRASE | LSTM (no o.g.) | LSTM (o.g.) | PARAGRAM-SL999 |
|---|---|---|---|---|
| $\leq 4$ | **72.7** | 63.4 | 58.8 | 66.3 |
| 5 | **74.0** | 54.5 | 48.4 | 65.0 |
| 6 | **70.5** | 52.6 | 48.2 | 50.1 |
| 7 | **73.7** | 56.9 | 50.6 | 56.4 |
| 8 | **75.5** | 60.2 | 52.4 | 60.1 |
| 9 | **73.0** | 58.0 | 48.8 | 58.8 |
| $\geq 10$ | **72.6** | 55.6 | 53.8 | 58.4 |

Table 7: Performance (Pearson's $r \times 100$) as a function of the maximum number of tokens in the sentence pairs over all 20 SemEval STS datasets.

We next consider whether the LSTM has worse generalization due to overfitting on the training data. To test this, we analyzed how the models performed on the training data (PPDB XL) by computing the average difference between the cosine similarity of the gold phrase pairs and the negative examples.[21] We found that all models had very similar scores: 0.7535, 0.7572, 0.7565, and 0.7463 for PARAGRAM-PHRASE, projection, LSTM (o.g.), and LSTM (no o.g.). This, along with the similar performance of the models on the PPDB tasks in Table 3, suggests that overfitting is not the cause of the worse performance of the LSTM model.

Lastly, we consider whether the LSTM's weak performance was a result of insufficient tuning or optimization. We first note that we actually ran more hyperparameter tuning experiments for the

[18] Note that we fixed batch-size to 100, $\delta$ to 0.4, and used MAX sampling as these were the optimal parameters for the PARAGRAM-PHRASE embeddings. We tuned the other hyperparameters as described in Section 4.2.2 with the exception of $\lambda_c$ which was tuned over $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$.

[19] For each pair, we computed the number of tokens in each of the two pieces of text, took the max, and then binned based on this value.

[20] Note that for the analysis in Sections 5 and 6, the models used were selected from earlier experiments. They are not the same as those used to obtain the results in Table 2.

[21] More precisely, for each gold pair $\langle g_1, g_2 \rangle$, and $n_i$, the respective negative example of each $g_i$, we computed $2 \cdot \cos(g_1, g_2) - \cos(n_1, g_1) - \cos(n_2, g_2)$ and averaged this value over all pairs.

LSTM models than either the PARAGRAM-PHRASE or projection models, since we tuned the decision to use an output gate. Secondly, we note that Tai et al. (2015) had a similar LSTM result on the SICK dataset (Pearson's $r$ of 85.28 to our 85.45) to show that our LSTM implementation/tuning procedure is able to match or exceed performance of another published LSTM result. Thirdly, the similar performance across models on the PPDB tasks (Table 3) suggests that no model had a large advantage during tuning; all found hyperparameters that comfortably beat the PARAGRAM-SL999 addition baseline. Finally, we point out that we tuned over learning rate and optimization strategy, as well as experimented with clipping gradients, in order to rule out optimization issues.

### 5.1 UNDER-TRAINED EMBEDDINGS

One limitation of our new PARAGRAM-PHRASE vectors is that many of our embeddings are under-trained. The number of unique tokens occurring in our training data, PPDB XL, is 37,366. However, the number of tokens appearing more than 100 times is just 7,113. Thus, one clear source of improvement for our model would be to address under-trained embeddings for tokens appearing in our test data.

In order to gauge the effect under-trained embeddings and unknown words have on our model, we calculated the fraction of words in each of our 22 SemEval datasets that do not occur at least 100 times in PPDB XL along with our performance deviation from the $75^{\text{th}}$ percentile of each dataset. We found that this fraction had a Spearman's $\rho$ of -45.1 with the deviation from the $75^{\text{th}}$ percentile indicating that there is a significant negative correlation between the fraction of OOV words and performance on these STS tasks.

### 5.2 USING MORE PPDB

#### 5.2.1 PERFORMANCE VERSUS AMOUNT OF TRAINING DATA

Models in related work such as Kiros et al. (2015) and Li et al. (2015a) require significant training time on GPUs, on the order of multiple weeks. Moreover, dependence of model performance upon training data size is unclear. To investigate this dependence for our PARAGRAM-PHRASE model, we trained on different amounts of data and plotted the performance. The results are shown in Figure 1. We start with PPDB XL which has 3,033,753 unique phrase pairs and then divide by two until there are fewer than 10 phrase pairs.[22] For each data point (each division by two), we trained a model with that number of phrase pairs for 10 epochs. We use the average Pearson correlation for all 22 datasets in Table 2 as the dependent variable in our plot.

We experimented with two different ways of selecting training data. The first ("Ordered") retains the order of the phrase pairs in PPDB, which ensures the smaller datasets contain higher confidence phrase pairs. The second ("Random") randomly permutes PPDB XL before constructing the smaller datasets. In both methods, each larger dataset contains the previous one plus as many new phase pairs.

We make three observations about the plot in Figure 1. The first is that performance continually increases as more training data is added. This is encouraging as our embeddings can continually improve with more data. Secondly, we note the sizable improvement (4 points) over the PARAGRAM-SL999 baseline by training on just 92 phrase pairs from PPDB. Finally, we note the difference between randomly permuting the training data and using the order from PPDB (which reflects the confidence that the phrases in each pair possess the paraphrase relationship). Performance of the randomly permuted data is usually slightly better than that of the ordered data, until the performance gap vanishes once half of PPDB XL is used. We suspect this behavior is due to the *safe* phrase pairs that occur in the beginning of PPDB. These high-confidence phrase pairs usually have only slight differences and therefore are not as useful for training our model.

## 6 QUALITATIVE ANALYSIS

To explore other differences between our PARAGRAM-PHRASE vectors and the PARAGRAM-SL999 vectors that were used for initialization, we inspected lists of nearest neighbors in each vector space.

[22] The smallest dataset contained 5 pairs.

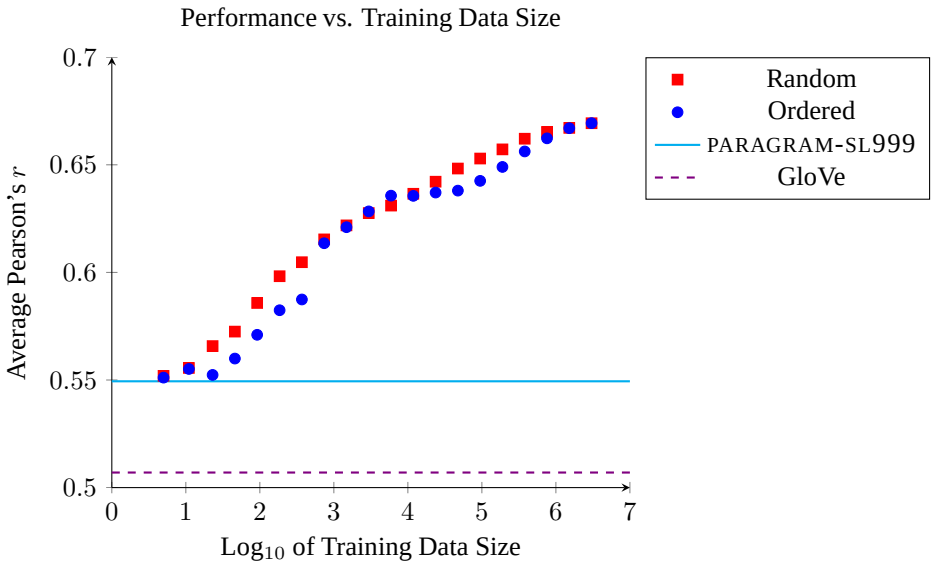


Figure 1: Performance of the PARAGRAM-PHRASE embeddings as measured by the average Pearson's $r$ on 22 textual similarity datasets versus the amount of training data from PPDB on a log scale. Each datapoint contains twice as much training data as the previous one. Random and Ordered refer to whether we shuffled the XL paraphrase pairs from PPDB or kept them in order. We also show baselines of averaging PARAGRAM-SL999 and GloVe embeddings.

| Word | PARAGRAM-PHRASE Nearest Neighbors | PARAGRAM-SL999 Nearest Neighbors |
|---|---|---|
| unlike | contrary, contrast, opposite, versa, conversely, opposed, contradiction | than, although, whilst, though, albeit, kinda, alike |
| 2 | 2.0, two, both, ii, 2nd, couple, 02 | 2.0, 3, 1, b, ii, two, 2nd |
| ladies | girls, daughters, honorable, females, girl, female, dear | gentlemen, colleague, fellow, girls, mr, madam, dear |
| lookin | staring, looking, watching, look, searching, iooking, seeking | doin, goin, talkin, sayin, comin, outta, somethin |
| disagree | agree, concur, agreeing, differ, accept | disagreement, differ, dispute, difference, disagreements |

Table 8: Nearest neighbors of PARAGRAM-PHRASE and PARAGRAM-SL999 word embeddings sorted by cosine similarity.

When obtaining nearest neighbors, we restricted our search to the 10,000 most common tokens in PPDB XL to ensure that the PARAGRAM-PHRASE vectors were not too under-trained. Some informative neighbors are shown in Table 8. In the first four rows, we see that the PARAGRAM-PHRASE embeddings have neighbors with a strong paraphrasing relationship. They tend to avoid having neighbors that are antonyms or co-hyponyms such as *unlike* and *alike* or *2* and *3* which are an issue for the PARAGRAM-SL999 embeddings. In contrast to the first four rows, the last row shows a problematic effect of our bag-of-words composition function: *agree* is the nearest neighbor of *disagree*. The reason for this is that there are numerous pairs in PPDB XL such as *i disagree* and *i do not agree* that encourage *disagree* and *agree* to have high cosine similarity. A model that takes context into account could resolve this issue. The difficulty would be finding a model that does so while still generalizing well, as we found that our PARAGRAM-PHRASE embeddings generalize better than learning a weight matrix or using a recurrent neural network. We leave this for future work.

When we take a closer look at our PARAGRAM-PHRASE embeddings, we find that information-bearing content words, such as *poverty*, *kidding*, *humanitarian*, *18*, and *july* have the largest $L_2$ norms, while words such as *of*, *it*, *to*, *hereby* and *the* have the smallest. Pham et al. (2015) noted this same phenomenon in their closely-related compositional model. Interestingly, we found that this weighting explains much of the success of our model. In order to quantify exactly how much, we calculated a weight for each token in our working vocabulary[23] simply by summing up the absolute

[23]This corresponds to the 42,091 tokens that appear in the intersection of our PARAGRAM-SL999 vocabulary, the test sets of all STS tasks in our evaluation, and PPDB XL plus an unknown word token.

value of all components of its PARAGRAM-PHRASE vector. Then we multiplied each weight by its corresponding PARAGRAM-SL999 word vector. We computed the average Pearson's $r$ over all 22 datasets in Table 2. The PARAGRAM-SL999 vectors have an average correlation of 54.94, the PARAGRAM-PHRASE vectors have 66.83, and the scaled PARAGRAM-SL999 vectors, where each is multiplied by its computed weight, have an average Pearson's $r$ of 62.64. Therefore, it can be surmised that at least 64.76% of the improvement over the initial PARAGRAM-SL999 vectors is due to weighting tokens by their importance.[24]

We also investigated the connection between these multiplicative weights and word frequency. To do so, we calculated the frequency of all tokens in PPDB XL.[25] We then normalized these by the total number of tokens in PPDB XL and used the reciprocal of these scores as the multiplicative weights. Thus less frequent words have more weight than more frequent words. With this baseline weighting method, the average Pearson's $r$ is 45.52, indicating that the weights we obtain for these words are more sophisticated than mere word frequency. These weights are potentially useful for other applications that can benefit from modeling word importance, such as information retrieval.

## 7 CONCLUSION

We introduced an approach to create universal sentence embeddings and propose our model as the new baseline for embedding sentences, as it is simple, efficient, and performs strongly across a broad range of tasks and domains. Moreover, our representations do not require the use of any neural network architecture. The embeddings can be simply averaged for a given sentence in an NLP application to create its sentence embedding. We also find that our representations can improve general text similarity and entailment models when used as a prior and can achieve strong performance even when used as fixed representations in a classifier. Future work will focus on improving our embeddings by effectively handling undertrained words as well as by exploring new models that generalize even better to the large suite of text similarity tasks used in our experiments.

### ACKNOWLEDGMENTS

We would like to thank Yoon Kim, the anonymous reviewers, and the area chair for their valuable comments. We would also like to thank the developers of Theano (Bergstra et al., 2010; Bastien et al., 2012) and thank NVIDIA Corporation for donating GPUs used in this research.

## REFERENCES

Agirre, Eneko, Diab, Mona, Cer, Daniel, and Gonzalez-Agirre, Aitor. 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, 2012.

Agirre, Eneko, Cer, Daniel, Diab, Mona, Gonzalez-Agirre, Aitor, and Guo, Weiwei. *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*, 2013.

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

[24]We also trained a model in which we only a learn a single multiplicative parameter for each word in our vocabulary, keeping the word embeddings fixed to the PARAGRAM-SL999 embeddings. We trained for 10 epochs on all phrase pairs in PPDB XL. The resulting average Pearson's $r$, after tuning on the Pavlick et al. PPDB task, was 62.06, which is slightly lower than using the absolute value of each PARAGRAM-PHRASE vector as its multiplicative weight.

[25]Tokens that did not appear in PPDB XL were assigned a frequency of 1.

Agirre, Eneko, Banea, Carmen, Cardie, Claire, Cer, Daniel, Diab, Mona, Gonzalez-Agirre, Aitor, Guo, Weiwei, Lopez-Gazpio, Inigo, Maritxalar, Montse, Mihalcea, Rada, Rigau, German, Uria, Larraitz, and Wiebe, Janyce. 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)*, 2015.

Bansal, Mohit, Gimpel, Kevin, and Livescu, Karen. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2014.

Baroni, Marco, Bernardi, Raffaela, and Zamparelli, Roberto. Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology*, 9, 2014.

Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian J., Bergeron, Arnaud, Bouchard, Nicolas, and Bengio, Yoshua. Theano: new features and speed improvements, 2012.

Belinkov, Yonatan and Glass, James. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Beltagy, Islam, Roller, Stephen, Cheng, Pengxiang, Erk, Katrin, and Mooney, Raymond J. Representing meaning with a combination of logical form and vectors. *arXiv preprint arXiv:1505.06816*, 2015.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

Bird, Steven, Klein, Ewan, and Loper, Edward. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

Blacoe, William and Lapata, Mirella. 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*, 2012.

Bordes, Antoine, Chopra, Sumit, and Weston, Jason. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014a.

Bordes, Antoine, Weston, Jason, and Usunier, Nicolas. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014b.

Chen, Xinchi, Qiu, Xipeng, Zhu, Chenxi, Liu, Pengfei, and Huang, Xuanjing. Long short-term memory neural networks for Chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015a.

Chen, Xinchi, Qiu, Xipeng, Zhu, Chenxi, Wu, Shiyu, and Huang, Xuanjing. Sentence modeling with gated recursive neural network. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015b.

Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, and Kuksa, Pavel. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12, 2011.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12, 2011.

Dzikovska, Myroslava O, Moore, Johanna D, Steinhauser, Natalie, Campbell, Gwendolyn, Farrow, Elaine, and Callaway, Charles B. Beetle II: a system for tutoring and computational linguistics experimentation. In *Proceedings of the ACL 2010 System Demonstrations*, 2010.

Faruqui, Manaal and Dyer, Chris. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 2014.

Filippova, Katja, Alfonseca, Enrique, Colmenares, Carlos A., Kaiser, Lukasz, and Vinyals, Oriol. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Finkelstein, Lev, Gabrilovich, Evgeniy, Matias, Yossi, Rivlin, Ehud, Solan, Zach, Wolfman, Gadi, and Ruppin, Eytan. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001.

Ganitkevitch, Juri, Durme, Benjamin Van, and Callison-Burch, Chris. Ppdb: The paraphrase database. In *HLT-NAACL*. The Association for Computational Linguistics, 2013.

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

Graves, Alex, Liwicki, Marcus, Bunke, Horst, Schmidhuber, Jürgen, and Fernández, Santiago. Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems 20*. 2008.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

Greff, Klaus, Srivastava, Rupesh Kumar, Koutník, Jan, Steunebrink, Bas R, and Schmidhuber, Jürgen. LSTM: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

He, Hua, Gimpel, Kevin, and Lin, Jimmy. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Hermann, Karl Moritz and Blunsom, Phil. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014.

Hermann, Karl Moritz, Kočiský, Tomáš, Grefenstette, Edward, Espeholt, Lasse, Kay, Will, Suleyman, Mustafa, and Blunsom, Phil. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, 2015.

Hill, Felix, Reichart, Roi, and Korhonen, Anna. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4), 2015.

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

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

Huang, Po-Sen, He, Xiaodong, Gao, Jianfeng, Deng, Li, Acero, Alex, and Heck, Larry. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, 2013.

İrsoy, Ozan and Cardie, Claire. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems 27*. 2014.

Iyyer, Mohit, Manjunatha, Varun, Boyd-Graber, Jordan, and Daumé III, Hal. 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)*, 2015.

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

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

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kiros, Ryan, Zhu, Yukun, Salakhutdinov, Ruslan, Zemel, Richard S, Torralba, Antonio, Urtasun, Raquel, and Fidler, Sanja. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

Le, Quoc V and Mikolov, Tomas. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

Li, Jiwei, Luong, Minh-Thang, and Jurafsky, Dan. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015a.

Li, Jiwei, Luong, Thang, and Jurafsky, Dan. A hierarchical neural autoencoder for paragraphs and documents. 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)*, 2015b.

Ling, Wang, Dyer, Chris, Black, Alan W, Trancoso, Isabel, Fermandez, Ramon, Amir, Silvio, Marujo, Luis, and Luis, Tiago. 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*, 2015.

Liu, Pengfei, Qiu, Xipeng, Chen, Xinchi, Wu, Shiyu, and Huang, Xuanjing. 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*, 2015.

Lu, Ang, Wang, Weiran, Bansal, Mohit, Gimpel, Kevin, and Livescu, Karen. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

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

Marelli, Marco, Bentivogli, Luisa, Baroni, Marco, Bernardi, Raffaella, Menini, Stefano, and Zamparelli, Roberto. 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)*, 2014.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013.

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

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

Paperno, Denis, Pham, Nghia The, and Baroni, Marco. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

Pavlick, Ellie, Rastogi, Pushpendre, Ganitkevich, Juri, Durme, Benjamin Van, and Callison-Burch, Chris. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2015.

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

Pham, Nghia The, Kruszewski, Germán, Lazaridou, Angeliki, and Baroni, Marco. 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)*, 2015.

Polajnar, Tamara, Rimell, Laura, and Clark, Stephen. An exploration of discourse-based sentence spaces for compositional distributional semantics. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, 2015.

Socher, Richard, Huang, Eric H, Pennin, Jeffrey, Manning, Christopher D, and Ng, Andrew Y. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, 2011.

Socher, Richard, Huval, Brody, Manning, Christopher D., and Ng, Andrew Y. 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*, 2012.

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

Socher, Richard, Karpathy, Andrej, Le, Quoc V., Manning, Christopher D., and Ng, Andrew Y. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2, 2014.

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

Tai, Kai Sheng, Socher, Richard, and Manning, Christopher D. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Tian, Ran, Okazaki, Naoaki, and Inui, Kentaro. The mechanism of additive composition. *arXiv preprint arXiv:1511.08407*, 2015.

Turian, Joseph, Ratinov, Lev-Arie, and Bengio, Yoshua. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010.

Vinyals, Oriol, Kaiser, Lukasz, Koo, Terry, Petrov, Slav, Sutskever, Ilya, and Hinton, Geoffrey. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*, 2014.

Wang, Di and Nyberg, Eric. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2015.

Wen, Tsung-Hsien, Gasic, Milica, Mrkšić, Nikola, Su, Pei-Hao, Vandyke, David, and Young, Steve. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Weston, Jason, Bengio, Samy, and Usunier, Nicolas. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1), 2010.

Wieting, John, Bansal, Mohit, Gimpel, Kevin, Livescu, Karen, and Roth, Dan. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3, 2015.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron C., Salakhutdinov, Ruslan, Zemel, Richard S., and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015a.

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

Xu, Yan, Mou, Lili, Li, Ge, Chen, Yunchuan, Peng, Hao, and Jin, Zhi. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015c.

Yih, Wen-tau, Toutanova, Kristina, Platt, John C., and Meek, Christopher. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, 2011.

Yin, Wenpeng and Schütze, Hinrich. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

Yu, Mo and Dredze, Mark. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3, 2015.

Zhao, Han, Lu, Zhengdong, and Poupart, Pascal. Self-adaptive hierarchical sentence model. In *Proceedings of IJCAI*, 2015.