Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations

John Wieting 1  Kevin Gimpel 2
1 Carnegie Mellon University, Pittsburgh, PA, 15213, USA
2 Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
jwieting@cs.cmu.edu, kgimpel@ttic.edu

Abstract

We extend the work of Wieting et al. (2017), back-translation a large parallel corpus to produce a dataset of more than 51 million English-English sentential paraphrase pairs in a dataset we call P A R A N M T - 5 0 M . We find this corpus to be cover many domains and styles of text, in addition to being rich in paraphrases with different sentence structure, and we release it to the community. and release it to the community. To show its utility, we use it to train paraphrastic sentence embeddings using only minor changes to the framework of Wieting et al. (2016b). The resulting embeddings outperform all supervised systems on every SemEval semantic textual similarity (STS) competition, and are a significant improvement in capturing paraphrastic similarity over all other sentence embeddings. We also show that our embeddings perform competitively on general sentence embedding tasks. We release the corpus, pretrained sentence embeddings, and code to generate them. We believe the corpus can be a valuable resource for automatic paraphrase generation and can provide a rich source of semantic information to improve downstream natural language understanding tasks. 1

1 Introduction

The natural language processing (NLP) community has benefited greatly from using pretrained word embeddings in downstream tasks. Recently, attention has been shifting from word embeddings to sentence embeddings. Most of this work is centered on general-purpose embeddings that can be used for any task, when they are used as features for a linear classifier or a shallow neural network (Kiros et al., 2015; Conneau et al., 2017). Our recent work has focused on paraphrastic sentence representations where sentences that have the same meaning lie close together in a vector space (Wieting et al., 2015, 2016b,a; Wieting and Gimpel, 2017; Wieting et al., 2017).

In this paper, we make two contributions. First, we scale up the work of Wieting et al. (2017). We use neural machine translation (NMT) to generate an automatic paraphrase corpus of over 51 million sentential paraphrase pairs. We call this dataset P A R A N M T - 5 0 M . We find this corpus to be of very high quality due to the state-of-the-art sentence embeddings that can be trained from it and from visual inspection. The dataset has the potential to be used for many tasks, from linguistically controlled paraphrase generation, style transfer, and sentence simplification to core NLP problems like neural machine translation. There are many rich paraphrases, from this corpus illustrating many paraphrase phenomena; we show examples and discuss its potential in Section 5.

Second, we show the utility of the corpus by using it to train state-of-the-art paraphrastic sentence embeddings. We perform a thorough search of the space, motivated by our prior work (Wieting et al., 2015, 2016b,a; Wieting and Gimpel, 2017; Wieting et al., 2017) and find that many neural architectures can be used to create strong embeddings. Moreover, given the large size of the paraphrase corpus, we found benefit from making a small change to the training procedure. That is, we increase the search space of negative examples during training. This presumably helps by ensuring the difficulty of minimizing the loss as the models improve over the course of learning.

1Generated corpus, trained models and code are available at www.cs.cmu.edu/~jwieting.
Our sentence embeddings would win every SemEval semantic textual similarity (STS) competition which have occurred from 2012-2016. These STS tasks have been the central task SemEval for the past 5 years, drawing dozens of participating institutions with over a hundred submissions each year. Since so many domains are covered in these datasets, they form a demanding evaluation for a general purpose sentence embedding model.

This is notable because most STS systems use curated lexical resources and supervised training data with manually-annotated similarities. Moreover, we do not even model the sentences jointly (i.e., using attention or features produced from both embeddings in the sentence pair), we just encode each sentence separately and then compute the cosine similarity between the embeddings.

We release our back-translated data and trained models for two main purposes. First we hope that the data can motivate new research directions and help create new and interesting NLP models, while adding a robustness to existing ones by incorporating this paraphrase knowledge. It is the largest collection of sentential paraphrases released to date. Secondly, since our pretrained sentence embeddings are state-of-the-art by a significant margin, we hope that these embeddings can be useful for many applications both as a sentence representation and as a general similarity metric. We are actively researching both of these directions to find downstream applications for both of these new resources.

2 Related Work

We describe related work in learning general-purpose sentence embeddings, work in automatically generating or discovering paraphrases, and finally prior work in leveraging neural machine translation for embedding learning.

Paraphrastic sentence embeddings. Our learning and evaluation setting is the same as that considered by Wieting et al. (2016b) and Wieting et al. (2016a), in which the goal is to learn paraphrastic sentence embeddings that can be used for downstream tasks. They trained models on PPDB and evaluated them using a suite of semantic textual similarity (STS) tasks and supervised semantic tasks. Others have begun to consider this setting as well (Arora et al., 2017).

Other work in learning general purpose sentence embeddings has used autoencoders (Socher et al., 2011; Hill et al., 2016), encoder-decoder architectures (Kiros et al., 2015), or other learning frameworks (Le and Mikolov, 2014; Pham et al., 2015; Conneau et al., 2017). Wieting et al. (2016b) and Hill et al. (2016) provide many empirical comparisons to this prior work. For conciseness, we compare only to the strongest configurations from their results.

Paraphrase generation and discovery. Many researchers have developed methods for generating or finding naturally-occurring sentential paraphrase pairs (Barzilay and McKeown, 2001; Dolan et al., 2004; Dolan and Brockett, 2005; Quirk et al., 2004; Coster and Kauchak, 2011; Xu et al., 2014, 2015; Lan et al., 2017).

The most relevant work uses bilingual corpora, e.g., Bannard and Callison-Burch (2005), culminating in the Paraphrase Database (PPDB; Ganitkevitch et al., 2013). Our goals are similar to those of PPDB, which has likewise been generated for many languages (Ganitkevitch and Callison-Burch, 2014) since it only needs parallel text.

Prior work has shown that PPDB can be used for learning embeddings for words and phrases (Faruqui et al., 2015; Wieting et al., 2015). However, when learning sentence embeddings, Wieting and Gimpel (2017) showed that PPDB is not as effective as sentential paraphrases, especially for recurrent networks. These results are intuitive because the phrases in PPDB are short and often cut across constituent boundaries. For sentential paraphrases, Wieting and Gimpel (2017) used a dataset developed for text simplification by Coster and Kauchak (2011). It was created by aligning sentences from Simple English and standard English Wikipedia.

Neural machine translation for paraphrastic embedding learning. Sutskever et al. (2014) trained NMT systems and visualized part of the space of the source language encoder for their English→French system. Hill et al. (2016) evaluated the encoders of English-to-X NMT systems as sentence representations, finding them to perform poorly compared to several other methods based on unlabeled data. Mallinson et al. (2017) adapted trained NMT models to produce sentence similarity scores in semantic evaluations. They used pairs of NMT systems, one to translate an
English sentence into multiple foreign translations and the other to then translate back to English. Wieting et al. (2017) only uses the NMT system to generate training data for training sentence embeddings, rather than using it in a similarity model itself. This permitted us to decouple decisions made in designing the NMT architecture from decisions about which models we will use for learning sentence embeddings. Thus we can benefit from orthogonal work in both NMT and in designing neural architectures to embed sentences. In other recent work, Suzuki et al. (2017) similarly use NMT to generate paraphrase corpora. Other work has used neural MT architectures and training settings to obtain better word embeddings (Hill et al., 2014a,b).

This work extends Wieting et al. (2017), scaling it up to a larger corpus of machine translations and producing state-of-the-art paraphrastic sentence embeddings.

3 Neural Machine Translation

To create the data used for training our models we back-translated the Czeng1.6 corpus (Bojar et al., 2016). We chose this corpus over other parallel text as it has many desirable properties. First, the sentences are short, meaning that NMT systems can generate correct inputs for many sentences in the corpus. Second, there is a lot of diversity in the corpus. A large portion of it is movie subtitles which tend to use a wide vocabulary and have a diversity of sentence structures, however other domains are included as well. Lastly, there is a lot of data. PARAGRAM:50M consists of over 51 million training examples, so there is plenty of data to train a strong in-domain translation system. It took over 10,000 GPU hours to back-translate all of this data.

We used the pre-trained models from the neural machine translation system of Sennrich et al. (2017) to translate the Czech sentences into English. The model was trained on this data along with data from some smaller Czech sources (160k from common-crawl, 650k from Europarl and 190k from News). We used beam search with a beam size of 12 and selected the highest scoring translation from the beam to be the translation.

Wieting et al. (2017) investigated methods to filter back-translated text. We experimented with simple approaches like using length cutoffs, which were found to be very effective, and more sophisticated ones like using a classifier to discern generated text from source text and using this classifier as an indicator of the quality of a translation.

We focused on extracting quality paraphrases from the back-translated text instead of trying to find both quality and lexically different paraphrases. We experimented with three simple approaches: We first used the translation scores from decoding. Secondly, we used trigram overlap filtering as was done by Wieting et al. (2017). Lastly, we experimented with using the PARAGRAM PHRASE embeddings from Wieting et al. (2015), with higher scoring pairs indicating a stronger paraphrase relationship between the sentences.

We then proceeded to find the quality training data. To do this, we ranked all 51M+ paraphrase pairs in the corpus by whichever of the three metrics we were using, and then split the data into tenths (so the first tenth contains the bottom 10% scored paraphrases, the second contains those in the bottom 10-20%, etc.). We then used the models to select the optimal score ranges on a small sample of labeled data. For more details about the corpus including statistics, comparisons to the other parallel corpora, and ablations on the data selection, see Section 5.

4 Models and Training

4.1 Models

We wish to embed a word sequence s into a fixed-length vector. We denote the tth word in s as st, and we denote its word embedding by xt. We focus on three main models in this paper, though we also experiment with combining them in various ways.

The first model, which we call Avg, simply averages the embeddings xt 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 Wa. This model was found by Wieting et al. (2016b) to per-

\[^2\] We also experimented with using our highest performing model from Section 6. This is similar to self-training in which a model is trained (typically with a small amount of supervision) and then used to label unlabeled data; the most confidently-labeled data is then added to the training set with the predicted labels. In this case, there is no supervision, but with stronger paraphrase models doing the ranking, the selected paraphrase pairs should be of higher quality.

\[^3\] We use the small English STS task from SemEval 2017 (Cer et al., 2017), which focused more on cross lingual sentence similarity, but had a track for English-English similarity consisting of 250 examples.
form very strongly for semantic similarity tasks.

The second model is similar to word averaging, but instead of averaging word embeddings, we average character trigram embeddings (Huang et al., 2013). Wieting et al. (2016a) found this to perform strongly for sentence embeddings compared to other n-gram orders and compared to word-averaging.

The third model are various LSTM architectures (Hochreiter and Schmidhuber, 1997). We mainly experimented with LSTMs where we average the hidden states with some small amount of scrambling of the input words. This combination was proposed by Wieting and Gimpel (2017), where they found that averaging the LSTM hidden states was significantly more effective than using the last hidden state for paraphrastic transfer learning, and that scrambling led to significant gains as well. We also experiment with bidirectional LSTMs (BiLSTMs), averaging all hidden states. Unlike in (Conneau et al., 2017), we found this to outperform max-pooling for both paraphrastic similarity and general sentence embedding tasks. Similarly, we also experimented with concatenating the hidden states, prior to averaging, as was also done in (Conneau et al., 2017). However, we found that for a given fixed output dimension, this wasn’t as effective as just averaging the hidden states.

4.2 Training

We follow the training procedure of prior work (Wieting et al., 2015, 2016b), with one modification. We introduce “mega-batching” for selecting negative examples which results in significant performance gains. The training data is a set $S$ of paraphrastic pairs $⟨s_1, s_2⟩$ and we optimize a margin-based loss:

$$\min_{W_c, W_w} \frac{1}{|S|} \left( \sum_{⟨s_1, s_2⟩ \in S} \max(0, \delta - \cos(g(s_1), g(s_2))) ight. \\
+ \cos(g(s_1), g(t_1))) + \max(0, \delta - \cos(g(s_1), g(s_2))) \\
+ \cos(g(s_2), g(t_2))) + \lambda_c \|W_c\|^2 + \lambda_w \|W_{w\text{initial}} - W_w\|^2$$

where $g$ is the model (Avg or GRAN), $\delta$ is the margin, $\lambda_c$ and $\lambda_w$ are regularization parameters, $W_{w\text{initial}}$ is the initial word embedding matrix, and $t_1$ and $t_2$ are “negative examples” taken from a mini-batch during optimization. The intuition is that we want the two texts to be more similar to each other ($\cos(g(s_1), g(s_2))$) than either to their respective negative examples $t_1$ and $t_2$, by a margin of at least $\delta$. To select $t_1$ and $t_2$, we choose the most similar sentence in some set (other than those in the given pair). For simplicity we use the mini-batch for this set, i.e., we choose $t_1$ for a given $⟨s_1, s_2⟩$ as follows:

$$t_1 = \arg\max_{t: ⟨t, i⟩ \in S_b \cup \{(s_1, s_2)⟩} \cos(g(s_1), g(t))$$

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 $s_i$.

4.2.1 Mega-batching

Due to training on large amounts of sentence pairs, we find that the models require stronger negative examples to learn more discriminative paraphrasic sentence embeddings. Therefore, we find that tying together the $S_b \subseteq S$ with the mini-batchsize can lead to very large mini-batches which isn’t optimal for optimization.

Therefore, we propose mega-batching where we set a parameter $M$ that controls how many batches are gathered up, and then the negative sampling is done from this pool of batches all at once. Once every example in every batch has a negative example, the mega-batch is split back up into individual batches and training commences, this time with more challenging negative examples that help the model learn better paraphrastic representations of text. This can be seen in Table 5, where we show negative examples for different sizes of megabatches. From the table, it can be seen that as the megabatch size increases, the negative examples become much closer to the source sentence.¹ Megabatching is further analyzed in Section 6.4.

5 Generated Paraphrase Corpus

One of the keys to the success of this work, and eventual work that used back-translated text is finding the right data to use. We claim that the

¹We also experimented with augmenting the objective, with another term that tried to prevent false negative examples. We would select the nearest neighbor that had a cosine distance of at least $\delta$ from the gold paraphrase pair. However, we found this to be strictly worse than not including this term. Even when $\delta$ was 0. Perhaps this could work better if it was only introduced near the end of training.
so, what's half an hour?
well, don't worry. i've taken out tons and tons of guys. lots of guys.
it's gonna be ... classic.
greetings, all!
but she doesn't have much of a case.
it was good in spite of the taste.

Table 1: Example paraphrases from PARANMT-50M, separated into the reference and the translation of the source sentence.

| Source | Reference | Translation |
|--------|-----------|-------------|
| so, what's half an hour? | half an hour won't kill you. | |
| well, don’t worry. i’ve taken out tons and tons of guys. lots of guys. | don’t worry, i’ve done it to dozens of men. | |
| it’s gonna be ...... classic. | yeah, sure. it’s gonna be great. | |
| greetings, all! | hello everyone! | |
| but she doesn’t have much of a case. | but as far as the case goes , she doesn’t have much. | |
| it was good in spite of the taste. | despite the flavor, it felt good. | |

Table 2: Statistics on comparable parallel data resources. Standard deviations are in parenthesis.

| Statistics | Common-crawl | Czeng1.6 | Europarl | News |
|------------|--------------|----------|----------|------|
| Avg. Length | 24.0 (34.7)  | 15.1 (19.3) | 26.1 (15.9) | 25.2 (13.9) |
| Avg. IDF   | 7.7 (3.1)    | 7.1 (6.8)  | 7.1 (0.6)  | 7.1 (4.1)  |
| Vocabulary | 8.3 (0.16)   | 6.8 (0.05) | 7.0 (0.05) | 7.0 (0.12) |
| Parse Ent. | 3.5          | 4.1        | 3.0        | 3.4       |
| Num Ex.    | 0.10M        | 51.4M      | 0.65M      | 0.19M     |

Table 3: Results of STS2017 for the data resources. The table shows that Czeng1.6 has the highest performance for all models.

| Data      | Word Avg | Trigram Avg | LSTM Avg |
|-----------|----------|-------------|---------|
| Common-crawl | 80.9     | 80.2        | 79.1    |
| Czeng1.6   | 83.6     | 82.2        | 82.5    |
| Europarl   | 78.9     | 78.0        | 80.4    |
| News       | 80.2     | 78.2        | 80.5    |

As the statistics indicate, Czeng1.6 isn’t especially more diverse in terms of vocabulary, though this could largely be due to all the rare entity tokens in the News and Common-crawl dataset. However, all corpora are significantly more diverse than Europarl, which tends to be very repetitive. However, Czeng1.6 has significantly shorter sentences than the corpus and more diverse structure due to its highest constituent parse entropy.

As another experiment, we use these small samples corpora for each data and trained 3 different models, Word Averaging, Trigram Averaging, and LSTM Averaging. In Table 3, the results on Czeng1.6 are the best for each model.

We hypothesize that the core strength of Czeng1.6 is more due to having better translations than any other innate property of the corpus. It has recently diverse in terms of vocabulary and has very diverse sentence structure, but the fact that the corpus has significantly shorter sentences than the other corpora, and has much more training data, means that its translations are going to be much better than that in the other corpora. In (Wieting et al., 2017), we noted that in their generated corpora, sentence length was the most important factor in filtering quality paraphrase data and is presumably due to how NMT performance worsens with longer sentences. Therefore, as obvious as it sounds, the better the translation of a corpus, the better the quality of the generated paraphrases.

We also show some illustrative examples from the corpus to show the diversity in paraphrase pairs present in the data. While the corpus has noise. We note that with improvements in NMT, the corpus quality can increase as NMT performance steadily rises. These examples are shown in Table 1 where we show the source sentence along with its translation.

6 Experiments

We now investigate how best to use our generated paraphrase data for training paraphrastic sentence embeddings as well as general-purpose sentence embeddings to be used as features for a wide range of tasks as has been done in (Kiros et al., 2015;
6.1 Evaluation

6.1.1 Paraphrastic Sentence Embedding Evaluation

We evaluate the quality of a paraphrase dataset by using the experimental setting of Wieting et al. (2016b). We use the paraphrases as training data to create paraphrastic sentence embeddings, then evaluate the embeddings on the SemEval semantic textual similarity (STS) tasks from 2012 to 2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016; Cer et al., 2017). Lastly, we evaluate on the STS Benchmark (Cer et al., 2017). The STS Benchmark was designed to be a standard for comparing sentence representation across a multitude of domains and across multiple modeling approaches (i.e. separate tracks for using the training data, modeling each sentence individually, etc.).

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. As our test set, we report the average Pearson’s $r$ over each year of these sentence similarity tasks from 2012-2016. We use the very small (250 examples) English track From SemEval 2017 as a development set.

6.1.2 Sentence Embedding Evaluation

We also evaluate our sentence embeddings on a range of tasks that have previously been used for evaluating sentence representations. These include sentiment analysis (MR, CR, SST), classifying the objectivity/subjectivity of a piece of text (SUBJ), opinion polarity (MPQA), question classification (TREC), paraphrase detection (MRPC), semantic relatedness (SICK), and textual entailment (SICK).

6.2 Experimental Setup

We use PARAGRAM-SL999 embeddings (Wieting et al., 2015) to initialize the word embedding matrix ($W_w$) for both models. For all experiments, we fix the mini-batch size to 100, $\lambda_w$ to 0, $\lambda_c$ to 0, and the margin $\delta$ to 0.4. We train all models for 5 epochs. For optimization we use Adam (Kingma and Ba, 2014) with a learning rate of 0.001.

For the recurrent neural networks, we set the scrambling rate to 0.3. We found, like in (Wieting and Gimpel, 2017), that scrambling significantly improves the results even though we are using far more sentence level data than was used in those previous experiments. We found a smaller value than the 0.5 primarily used in (Wieting and Gimpel, 2017) worked better most likely due to the larger amount of training data.

6.3 Filtering Data

In order to find the optimal training data to use for training our models, we first ranked all 51M+ generated paraphrase pairs from the NMT system (See Section 3 for details) by ranking them with 3 techniques: semantic similarity using the PARAGRAM-PHRASE model from (Wieting et al., 2016b), trigram-overlap, and translations score (the latter two were both used in (Wieting et al., 2017)).

Table 4 shows the results of the different filtering methods. We trained an LSTM Avg. model for a single epoch on 1M examples sampled from each of the ten folds for each filtering criteria. From the results, filtering by word similarity produces the highest quality data. We then randomly selected 5M examples, all where both sentence pairs had no more than 30 tokens, from the top two scoring folds. These 5M examples were the training data for our experiments.

6.4 Effect of Mega-Batching

The effect mega-batching has on our results is shown in Table 5. We trained on our 5M example training set for 5 epochs.

The results show that for both Word Avg. and BiLSTM Avg., larger megabatches increase performance. There isn’t much gain moving from 20 to 40, but these high megabatch sizes are

| Filtering Method | Para. Sim. | Trigram Ovl. | Trans. Score |
|------------------|------------|--------------|--------------|
| LSTM Avg.        | 83.8       | 83.4         | 83.4         |

Table 4: Best result on each fold of the data for each filtering method using for filtering the data.

| Mega-Batch Size | Word Avg. | BiLSTM Avg. |
|-----------------|-----------|-------------|
| 1               | 82.3      | 82.4        |
| 20              | 84.0      | 85.0        |
| 40              | 84.1      | 85.1        |

Table 5: Results on the STS tasks of word-averaging and BiLSTM models with different sizes of mega-batches.
Table 6: Example nearest neighbors for various mega-batch sizes for the BiLSTM Avg. model.

| Source Sentence | Negative Example |
|-----------------|------------------|
| Mega-batchsize of 1 | I mean, colonel. they know that I've been looking for her. sometimes the ball doesn't go down. |
| sir, I'm just trying to protect. | I'm looking at him, you know? I'll let it go a couple of rounds. |
| Mega-batchsize of 20 | I only ask that the baby be safe. I'm keeping him. I'll take two. |
| sir, I'm just trying to protect. I'm looking at him, you know? I'll let it go a couple of rounds. | Just trying to survive. On instinct. I looked at him with wonder. I want you to sit out a couple of rounds, all right? |
| Mega-batchsize of 40 | sir, I'm just trying to protect. I'm looking at him, you know? I'll let it go a couple of rounds. |

Table 6: Example nearest neighbors for various mega-batch sizes for the BiLSTM Avg. model.

a very significant improvement over not using megabatching at all (i.e. using a megabatch size of 1).

We also show how the negative examples change as a function of the mega-batch size in Table 6. We use the BiLSTM Avg. model and show the negative examples (nearest neighbors from nearly all examples\(^8\) in the megabatch) for different megabatch sizes for two sentences. As can be seen, having high mega-batch values produces more interesting negative examples and hence the model learns richer representations. This is likely more important when training the model on sentences since they have so much more space to have diverse meaning unlike prior work on learning on text snippets (Wieting et al., 2015, 2016b,a; Pham et al., 2015).

6.5 Simple Models

We first investigate how well simple models (Word Avg., Trigram Avg., and LSTM Avg.) do on their own. Table 7 shows the results on the STS tasks from 2012-2016. Table 9 shows results on the STS Benchmark.

Next we investigate how well our embeddings do on general sentence embedding tasks. Note that all of these simple models have 300 dimensions, which is far fewer than the several thousands (often 2400-4800) of dimensions used in most prior work.

6.5.1 Larger Recurrent Networks

Since LSTM Avg. model performed so well, we also investigated the effectiveness of scaling up the dimension of the output embedding and using BiLSTMs.

Interestingly, we found that higher dimensions did not increase performance on the STS tasks, but it did help on the general sentence embedding tasks.

6.6 Mixed Models

We next experimented with combining some of our models after noting that they have very different performance on different STS tasks. Table 10 shows the average absolute difference in Pearson’s \(r\) over all 25 datasets. Hence, combining models and training them jointly could result in overall improvement. From the table, it seems that the Trigram Avg. model compliments the two models using word embeddings, as the Trigram Avg. model can provide information on unknown words that are a limitation to both Word Avg. and LSTM Avg.

We experimented with two natural ways of combining the embeddings. The first was to simply train the models jointly and form the final embedding by adding the output of each model. This would leave us with a fixed 300 dimensional embedding. The second approach was to simply concatenate the output embeddings of each model, creating a final vector that had \(n \times 300\) dimensions where \(n\) is the number of models being jointly trained.

The mixed models have significant improvement over the individual models as well as the larger LSTMs and BiLSTMs. Interestingly, combining Trigram Avg. and Word Avg. results in the best performance. We did experiment with all combinations, but in the subsequent results only show the combination of Trigram Avg. and Word Avg. and the combination of all three models.

7 Discussion

There are several noteworthy facts about our experiments. First, it is interesting to note the effect of embedding dimensionality on performance in the various evaluations. For semantic similarity, the dimensionality of the embeddings does not appear to have a strong effect. However, for the general sentence embedding evaluation, larger dimensions nearly always improve overall performance. This is presumably due to the fact that having more features leads to more trainable parameters in the subsequent classifier, increasing the ability of the classifier to linearly separate the data.

Secondly, while InferSent has strong performance across all tasks, our model obtains better results on semantic similarity. Table 7 shows a siz-
| Data | Dim. | 2012 | 2013 | 2014 | 2015 | 2016 |
|------|------|------|------|------|------|------|
| **Our Work** |      |      |      |      |      |      |
| **Simple Models** |      |      |      |      |      |      |
| Word Avg. | 300  | 66.2 | 61.8 | 76.2 | 79.3 | 77.5 |
| Trigram Avg. | 300  | 67.2 | 60.3 | 76.1 | 79.7 | **78.3** |
| LSTM Avg. | 300  | 67.0 | 62.3 | 76.3 | 78.5 | 76.0 |
| **Larger Recurrent Networks** |      |      |      |      |      |      |
| LSTM Avg. | 600  | 66.6 | 61.3 | 76.0 | 78.8 | 74.8 |
| LSTM Avg. | 900  | **68.0** | 60.4 | 76.3 | 78.8 | 75.9 |
| BiLSTM Avg. | 600  | 67.4 | 60.3 | 76.4 | 79.7 | 76.5 |
| BiLSTM Avg. | 900  | 67.4 | 60.2 | 76.1 | 79.5 | 76.5 |
| **Mixed Models** |      |      |      |      |      |      |
| Trigram-Word (addition) | 300  | 67.3 | **62.8** | **77.5** | 80.1 | 78.2 |
| Trigram-Word-LSTM (addition) | 300  | 67.1 | **62.8** | 76.8 | 79.2 | 77.0 |
| **Trigram-Word (concatenation)** |      |      |      |      |      |      |
| Trigram-Word-LSTM (concatenation) | 600  | 67.8 | 62.7 | 77.4 | **80.3** | 78.1 |
| Trigram-Word-LSTM (concatenation) | 900  | 67.7 | **62.8** | 76.9 | 79.8 | 76.8 |
| **STS Competitions** |      |      |      |      |      |      |
| 1<sup>st</sup> Place System |      |      | 64.8 | 62.0 | 74.3 | 79.0 | 77.7 |
| 2<sup>nd</sup> Place System |      |      | 63.4 | 59.1 | 74.2 | 78.0 | 75.7 |
| 3<sup>rd</sup> Place System |      |      | 64.1 | 58.3 | 74.3 | 77.8 | 75.7 |
| **Related Work** |      |      |      |      |      |      |
| InferSent (AllSNLI) (Conneau et al., 2017) | 4096 | 58.6 | 51.5 | 67.8 | 68.3 | 67.2 |
| InferSent (SNLI) (Conneau et al., 2017) | 4096 | 57.1 | 50.4 | 66.2 | 65.2 | 63.5 |
| FastSent (Hill et al., 2016) | 100  | -    | -    | 63   | -    | -    |
| DictRep (Hill et al., 2016) | 500  | -    | -    | 67   | -    | -    |
| CaptionRep (Hill et al., 2016) | 500  | -    | -    | 46   | -    | -    |
| SkipThought (Kiros et al., 2015) | 4800 | -    | -    | 29   | -    | -    |
| CPHASE (Pham et al., 2015) | -    | -    | -    | 65   | -    | -    |
| ParagraphVec (Le and Mikolov, 2014) | 100  | -    | -    | 42   | -    | -    |
| CBOW (from (Hill et al., 2016)) | 500  | -    | -    | 64   | -    | -    |
| Skipgram (from (Hill et al., 2016)) | -    | -    | -    | 62   | -    | -    |

Table 7: Results on the STS tasks of our simplest models. We compare to the top performing systems from each SemEval. Note that we are reporting the mean result, not the weighted mean used in the competition. Our best performing overall model, Trigram-Word is in bold.

The table shows a remarkable gap between our work and all previous work (outside of our own). Our models consistently perform around 10 points higher for each year of the STS tasks as well as on the STS Benchmark.

Regarding the supervised tasks, we note that some result trends appear to be influenced by data domain. InferSent is trained on a dataset of mostly captions, especially the model trained on just SNLI. Therefore, the datasets for the SICK relatedness and entailment evaluations are similar in domain to the training domain of InferSent, especially entailment since the target tasks are aligned. Our results on MRPC and entailment are significantly better than Skip-Thought vectors, and on tasks that are not on caption data (MRPC), we have performance competitive with that of InferSent.

To measure the effects of domain on the performance of InferSent, we performed additional experiments. We first compared the performance of both our model and their SNLI and ALLNLI models on all STS tasks from 2012-2016. We then compared the overall mean with that of the three caption STS datasets within the collection. The results are shown in Table 11. The InferSent models are much closer to the Trigram-Word model on the caption datasets than overall, and InferSent trained on SNLI shows the largest difference between its
Table 8: General sentence representations tasks for unsupervised, learning, unsupervised learning using structured resources, supervised transfer learning, and direct supervised learning.

| Model                                | MR  | CR  | SUBJ | MPQA | SST  | TREC | MRPC | SICK-R | SICK-E |
|--------------------------------------|-----|-----|------|------|------|------|------|--------|--------|
| **Unsupervised (Unordered Sentences)** |     |     |      |      |      |      |      |        |        |
| Unigram-TFIDF (Hill et al., 2016)    | 73.7| 79.2| 90.3 | 82.4 | -    | 85.0 | 73.6 | 81.7   | -      |
| ParagraphVec (Le and Mikolov, 2014)  | 60.2| 66.9| 76.3 | 70.7 | -    | 59.4 | 72.9 | 81.1   | -      |
| SDAE (Hill et al., 2016)             | 74.6| 78.0| 90.8 | 86.9 | -    | 78.4 | 73.7 | 80.7   | -      |
| **Unsupervised (Ordered Sentences)**  |     |     |      |      |      |      |      |        |        |
| FastSent (Hill et al., 2016)         | 70.8| 78.4| 88.7 | 80.6 | -    | 76.8 | 72.2 | 80.3   | -      |
| FastSent+AE (Hill et al., 2016)      | 71.8| 76.7| 88.8 | 81.5 | -    | 80.4 | 71.2 | 79.1   | -      |
| SkipThought (Kiros et al., 2015)     | 76.5| 80.1| 93.6 | 87.1 | 82.0 | 92.2 | 73.0 | 82.0   | 0.858  | 82.3   |
| **Unsupervised (Structured Resources)** |     |     |      |      |      |      |      |        |        |
| CaptionRep (Hill et al., 2016)       | 61.9| 69.3| 77.4 | 70.8 | -    | 72.2 | -   | -      | -      |
| DictRep (Hill et al., 2016)          | 76.7| 78.7| 90.7 | 87.2 | -    | 81.0 | 68.4 | 76.8   | -      |
| NMT En-to-Fr (Hill et al., 2016)     | 64.7| 70.1| 84.9 | 81.5 | -    | 82.8 | -   | -      | -      |
| BYTE mLSTM (Radford et al., 2017)    | 86.9| 91.4| 94.6 | 88.5 | -    | -   | 75.0 | 82.8   | 79.2   |
| **Simple Models (Our Work)**         |     |     |      |      |      |      |      |        |        |
| Word Avg. (300d)                     | 75.8| 80.5| 89.2 | 87.1 | 80.0 | 80.1 | 68.6 | 80.9   | 83.6   | 80.6   |
| Trigram Avg. (300d) (This work)      | 68.8| 75.5| 83.6 | 82.3 | 73.6 | 73.0 | 71.4 | 82.0   | 79.3   | 78.0   |
| LSTM Avg. (300d)                     | 73.8| 78.4| 88.5 | 86.5 | 80.6 | 76.8 | 73.6 | 82.3   | 83.9   | 81.9   |
| **Larger Recurrent Networks (Our Work)** |     |     |      |      |      |      |      |        |        |
| LSTM (600d)                          | 75.2| 80.5| 90.0 | 88.0 | 80.0 | 83.8 | 74.0 | 82.1   | 85.2   | 82.6   |
| LSTM (900d)                          | 75.8| 81.7| 90.5 | 87.4 | 81.6 | 84.4 | 74.8 | 83.0   | 86.0   | 83.0   |
| BiLSTM (600d)                        | 75.4| 80.8| 89.9 | 87.2 | 79.1 | 84.4 | 73.7 | 81.9   | 85.4   | 84.3   |
| BiLSTM (900d)                        | 75.6| 82.4| 90.6 | 87.7 | 81.3 | 87.4 | 75.0 | 82.9   | 85.8   | 84.4   |
| **Mixed Models (Our Work)**          |     |     |      |      |      |      |      |        |        |
| Trigram-Word (300d, addition)        | 74.8| 78.8| 88.5 | 87.4 | 78.7 | 79.0 | 71.4 | 81.4   | 83.2   | 80.6   |
| Trigram-Word-LSTM (300d, addition)   | 75.0| 80.7| 88.6 | 86.6 | 77.9 | 78.6 | 72.7 | 80.8   | 83.6   | 81.8   |
| Trigram-Word (600d, concatenation)   | 75.8| 80.5| 89.9 | 87.8 | 79.7 | 82.4 | 70.7 | 81.7   | 84.6   | 82.0   |
| Trigram-Word-LSTM (900d, concatenation) | 77.6| 81.4| 91.4 | 88.2 | 82.0 | 85.4 | 74.0 | 81.5   | 85.4   | 83.8   |
| BiLSTM AVG (4096)                    | 77.5| 82.6| 91.0 | 89.3 | 82.8 | 86.8 | 75.8 | 82.6   | 85.9   | 83.8   |
| BiLSTM MAX (4096)                    | 76.6| 83.4| 90.9 | 88.5 | 82.0 | 87.2 | 76.6 | 83.5   | 85.3   | 82.5   |
| **Supervised (Transfer)**            |     |     |      |      |      |      |      |        |        |
| BiLSTM-Max (on SST) (Conneau et al., 2017) | (*)| 83.7| 90.2 | 89.5 | (*) | 86.0 | 72.7 | 80.9   | 0.863  | 83.1   |
| BiLSTM-Max (on SNLI) (Conneau et al., 2017) | 79.9| 84.6| 92.1 | 89.8 | 83.3 | 88.7 | 75.1 | 82.3   | 0.885  | 86.3   |
| BiLSTM-Max (on AllNLI) (Conneau et al., 2017) | 81.1| 86.3| 92.4 | 90.2 | 84.6 | 88.2 | 76.2 | 83.1   | 0.884  | 86.3   |
| **Supervised (Direct)**              |     |     |      |      |      |      |      |        |        |
| Naive Bayes - SVM                    | 79.4| 81.8| 93.2 | 86.3 | 83.1 | -   | -   | -      | -      |
| AdaSent (Zhao et al., 2015)          | 83.1| 86.3| 95.5 | 93.3 | -    | 92.4 | -   | -      | -      |
| TF-KLD (Ji and Eisenstein, 2013)     | -   | -   | -   | -    | -    | 80.4| 85.9 | -      | -      |
| Illinois-LH (Lai and Hockenmaier, 2014) | - | -  | -  | -    | -    | -  | -   | 84.5   |        |
| Dependency Tree-LSTM (Tai et al., 2015) | - | -  | -  | -    | -    | -  | -   | 0.868  | -      |

Overall performance and its performance on caption data.

We also compared the performance of these models on the STS Benchmark under several conditions as shown in Table 12. We compare the models unsupervised performance on the entire dataset, the entire dataset without caption data and the performance of the models on just the caption data. We also include the performance on the supervised setting for these cases as well. From the table, it is clear that relative to our models, InferSent does much better on the caption data than the non-caption data, providing evidence of a bias.

Note that this bias is less for the model trained on AllSNLI, as it incorporates other domains.

Overall, we believe that there are many avenues to training sentence embeddings, each with their own strengths and weaknesses. Ours are specialized for semantic similarity, entailment, and paraphrasing. However, they do well on general tasks as well when scaled up, outperforming Skipthought and all of the earlier work on all tasks with the exception of X and Y. InferSent does well on lots of these tasks, better than any published system to date.

We would also like to emphasize that our embeddings only require cheap structured data. There are millions to billions of pairs of parallel...
Table 9: Pearson’s $r$ on the test set of the STS Benchmark.

| Model                  | Average Abs. Pearson’s r |
|------------------------|--------------------------|
| Word Avg. / Trigram Avg. | 2.75                     |
| Word Avg. / LSTM Avg.   | 2.17                     |
| Trigram Avg. / LSTM Avg. | 2.89                     |

Table 10: The average absolute differences in performance between each model and each other model over all 25 datasets in the STS competition data.

| Model                  | Mean Diff. | InferSent (AllNLI) | InferSent (SNLI) |
|------------------------|------------|--------------------|------------------|
|                         |            | Tr. (All)          | Tr. (Cap.)       | Tr. (No Cap.)   |
| MSRvid (2012) diff.     | 5.2        | 6.4                | 4.8              |
| Images (2014) diff.     | 6.4        | 4.8                |                  |
| Images (2015) diff.     | 3.6        | 3.0                |                  |

Table 11: Difference in performance (Pearson’s $r$) between Trigram-Word and InferSent models trained on two different datasets. The first row is the mean difference across all 25 datasets, then the following rows show differences on three individual datasets that are comprised of captions.

Table 12: Results on the STS benchmark comparing our Trigram-word model with Infersent. Tr. refers that the models were trained and tuned using the training and development data of the STS Benchmark. Cos. refers to the results on the test set with no training or tuning. We experiment with using all of the data (All), just the caption portion of the data (Cap.), and all of the data except for the caption portion (No Cap.).

It would be interesting to combine the best of these models that have various specialization in a single model. It seems that (Radford et al., 2017), Infersent, and our model are a natural fit - each with their own specializations. Combining these in a clever way, could make truly capable sentence embeddings that show strong performance on nearly all NLP tasks. It would be interesting to see if adding in this extra semantic information even improves the paraphrastic performance of the final embeddings.

8 Conclusion

This paper makes two valuable contributions. First, we release PARANMT-50M, a corpus of over 51M backtranslations from the Czeng1.6 corpus, which we release for general use. We also release code and trained models for state-of-the-art paraphrastic sentence embeddings, which also exhibit strong performance as general-purpose sentence representations for a multitude of tasks. We hope that this corpus, along with our improved paraphrastic sentence representations, can significantly improve the state-of-the-art in many research areas like machine translation, paraphrase generation, question-answering, among many others.
Acknowledgments

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 Marítxalar, Rada Mihalcea, German Rigau, Larraitz Uría, 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, 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.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics.

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. Association for Computational Linguistics, pages 50–57.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Vhariš. 2016. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, Text, Speech, and Dialogue: 19th International Conference, TSD 2016. Masaryk University, Springer International Publishing, Cham / Heidelberg / New York / Dordrecht / London, number 9924 in Lecture Notes in Computer Science, pages 231–238.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint arXiv:1708.00055 .

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. arXiv preprint arXiv:1705.02364 .

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. Association for Computational Linguistics, pages 665–669.

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. Association for Computational Linguistics, page 350.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In Proc. of IWP.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

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 Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. 2014a. Embedding word similarity with neural machine translation. arXiv preprint arXiv:1412.6448 .
Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014b. Not all neural embeddings are born equal. arXiv preprint arXiv:1410.0718 .

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).

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. 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.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In EMNLP. pages 891–896.

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

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. arXiv preprint arXiv:1506.06276 .

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In SemEval@ COLING. pages 329–334.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Copenhagen, Denmark.

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

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. pages 881–893.

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.

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

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. arXiv preprint arXiv:1703.02507 .

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

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).

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In Proceedings of EMNLP.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. arXiv preprint arXiv:1704.01444 .

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. arXiv preprint arXiv:1703.04357 .

Yang Shao. 2017. Hcti at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pages 130–133.

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.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems. pages 3104–3112.

Yui Suzuki, Tomoyuki Kajiwara, and Mamoru Komachi. 2017. Building a non-trivial paraphrase corpus using multiple machine translation systems. In Proceedings of ACL 2017. Student Research Workshop. Association for Computational Linguistics, Vancouver, Canada. pages 36–42. http://aclweb.org/anthology/P17-3007.
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 EMNLP*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016b. Towards universal paraphrastic sentence embeddings. In *Proceedings of ICLR*.

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)*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of ACL*.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *Proceedings of EMNLP*.

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)*.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics* 2:435–448.

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