Convolutional Neural Network for Universal Sentence Embeddings

Xiaoqi Jiao[1] **Fang Wang**[1,2] **Dan Feng**[1]
[1]*Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System, (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China*
[2]*Shenzhen Huazhong University of Science and Technology Research Institute*
Corresponding author: wangfang@hust.edu.cn

Abstract

This paper proposes a simple CNN model for creating general-purpose sentence embeddings that can transfer easily across domains and can also act as effective initialization for downstream tasks. Recently, averaging the embeddings of words in a sentence has proven to be a surprisingly successful and efficient way of obtaining sentence embeddings. However, these models represent a sentence, only in terms of features of words or uni-grams in it. In contrast, our model (CSE) utilizes both features of words and n-grams to encode sentences, which is actually a generalization of these bag-of-words models. The extensive experiments demonstrate that CSE performs better than average models in transfer learning setting and exceeds the state of the art in supervised learning setting by initializing the parameters with the pre-trained sentence embeddings.

1 Introduction

Representing word sequences such as phrases and sentences plays an important role in natural language understanding systems. In recent years, many composition functions have been applied to word embeddings to obtain vectors for longer phrases or sentences, ranging from simple operations like addition (Mitchell and Lapata, 2008) to richly-structured functions like recursive neural networks (Socher et al., 2011), convolutional neural networks (Kalchbrenner et al., 2014), and recurrent neural networks (Tai et al., 2015).

In this paper, based on convolutional neural networks (CNN), we introduce an architecture that can use both features of words and features of n-grams to encode sentences into real-valued vectors with the property that sentences with similar meaning have high cosine similarity in the embedding space. Our goal is to learn general-purpose sentence representations that can be transferred for measuring semantic textual similarity (STS) (Agirre et al., 2012) and can also act as effective initialization for downstream tasks.

Originally, the CNN model is invented to extract local features in computer vision (Lecun et al., 1998). But it has been shown to be effective for NLP and has achieved excellent results in sentence modeling (Kalchbrenner et al., 2014), and other traditional NLP tasks (Collobert et al., 2011). To encode the meaning of a sentence of varying length into a fix-length vector, our simple CNN model first utilizes multiple filters with different size to extract possible features of n-grams in it. Then we obtain one feature corresponding to one filter through pooling operation. The intuition is to capture one aspect of semantic features of n-grams for each filter. Inspired by the strong performance of simply averaging word embeddings (Wieting et al., 2016), we also set filter size to 1 to extract word semantic information.

To evaluate our model, we consider two types of learning settings: transfer learning evaluation and supervised learning evaluation. For the transfer leaning setting, we first train our model on noisy phrase pairs from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013), then evaluate it on every SemEval STS task from 2012 through 2015 and the 2014 SemEval SICK dataset (Marelli et al., 2014). Particularly, we follow the experiment setting of Wieting et al. (2016), in which they compared various types of neural network architectures except CNN, spanning the range of complexity from word averaging to LSTMs

Proceedings of the 27th International Conference on Computational Linguistics, pages 2470–2481
Santa Fe, New Mexico, USA, August 20-26, 2018.

and showed the impressive performance of Paragram-Phrase (PP), a simple word average model. So our work can also serve as complementary to theirs. As results, we found our model performs better than their PP model in most evaluation datasets. Moreover, researchers have recently found that since models are ultimately tested on sentences, sentence representations trained on noisy sentence pairs, obtained automatically by aligning Simple English to standard English Wikipedia (Coster and Kauchak, 2011), perform much better than those trained on noisy phrase pairs from PPDB dataset (Wieting and Gimpel, 2017). So we conduct the same transfer learning experiment on noisy sentence pairs with our model which is initialized with the parameters obtained from PPDB, and again find our model outperforms two state-of-the-art models: ATT-CCG (Shaonan Wang, 2017) and GRAN (Wieting and Gimpel, 2017). For the supervised learning setting, we use the model from transfer setting as a prior and fine-tune this model on the SICK dataset in a supervised style. With useful initialization, we achieve new state-of-the-art results on this task.

2 Related Work

Obtaining the most useful distributed representations of phrases or sentences could ultimately have a significant impact on language understanding systems since it is phrases and sentences, rather than individual words, that encode the human-like general world knowledge (or common sense) (Norman, 1972). According to their purposes, sentence embeddings generally fall into two categories: task-specific sentence embeddings and general-purpose sentence embeddings.

The first consists of sentence embeddings trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. Researchers have proposed many models along this line, and they typically use recursive neural networks (Socher et al., 2012; Socher et al., 2013), convolutional neural networks (Kalchbrenner et al., 2014; dos Santos and Gatti, 2014; Kim, 2014) or recurrent neural networks with long short-term memory (LSTM) (Hochreiter et al., 1997; Chung et al., 2014) as an intermediate step in creating sentence embeddings to solve a variety of NLP tasks including paraphrase identification and sentiment classification (Yin and Schütze, 2015; Tan et al., 2016; Lin et al., 2017).

The other category consists of universal sentence embeddings, which are usually trained by unsupervised or semi-supervised learning and can be served as features for many other NLP tasks such as text classification and semantic textual similarity. This include recursive auto-encoders (Socher et al., 2011), ParagraphVector (Le and Mikolov, 2014), SkipThought vectors (Kiros et al., 2015), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), PP (Wieting et al., 2016), Sent2Vec (Pagliardini et al., 2017), GRAN (Wieting and Gimpel, 2017), etc.

In this paper, we also aim to learn general-purpose, paraphrastic sentence embeddings, the same purpose as Wieting et al. (2016). In their work, they demonstrate the strong performance of the PP model across a broad range of tasks and domains, but also show some limitations of this bag-of-words model. For future work, they leave a challenge: can we find a model that takes context into account while still generalizing as well as the PP model? Then in their following work (Wieting and Gimpel, 2017), they proposed a novel architecture, called GRAN that utilizes the context information to create gate for each word in a sentence. After obtaining gates which can be seen as a kind of attention mechanisms, they average the word embeddings, multiplied with their respective gates, to get the embedding of the sentence. Besides, there is another model, called ATT-CCG (Shaonan Wang, 2017), which also incorporates attention mechanism into PP model by using word attributes information (CCG supertag of words). Unlike GRAN and ATT-CCG, we propose a convolutional architecture, called Convolutional Sentence Encoder (CSE), which directly consider the semantics of n-grams by using convolving filters with different size. Further description of this model is included in Section 3.1.

3 Model and Training

3.1 Model

Figure 1 shows the architecture of the proposed model CSE. Our model encodes the meaning of a sentence of varying length into a fixed-length real-valued vector such that the semantic similarity between

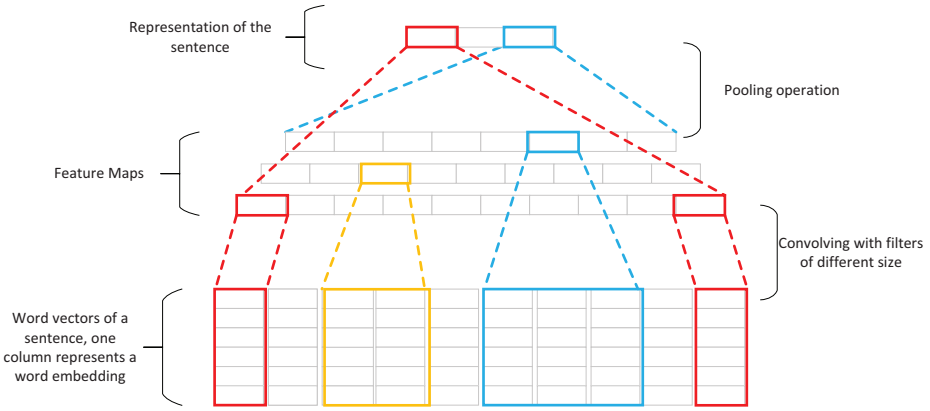

Figure 1: Model architecture

sentences can be measured by the cosine similarity of their corresponding vectors. We denote $\mathbf{x}_i \in \mathbb{R}^d$ as the d-dimensional word vector of the i-th word in a sentence. Then a sentence of length n (padded when necessary) is represented as

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_n, \tag{1}$$

Where $\oplus$ is the concatenation operator. In general, let $\mathbf{x}_{i:j}$ refers to the concatenation of words $\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_j$. A convolution operation involves a filter $\mathbf{w} \in \mathbb{R}^{h*d}$, which is applied to a window of h words to produce a new feature. For example, a feature c_i is generated from a window of words $\mathbf{x}_{i:i+h-1}$ by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b). \tag{2}$$

Here $b \in \mathbb{R}$ is a bias term and f is a activation function (either tanh or linear unit; linear unit is slightly better in our experiments, so we only report the linear unit results). This filter is applied to each possible window of words in the sentence $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$ to produce a feature map

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \tag{3}$$

with $\mathbf{c} \in \mathbb{R}^{n-h+1}$. Instead of using mean pooling, we then apply a sum pooling operation over the feature map for forcing our model to capture semantic similarity between words and phrases.[1] And we take the sum value $\hat{c} = sum(\mathbf{c})$ as the feature corresponding to this particular filter. The idea is to capture one aspect of n-gram semantic features for each feature map. This pooling scheme also naturally deals with variable sentence lengths.

We have described the process by which one feature is extracted from one filter. The model uses k filters (with varying window sizes) to obtain multiple features. These features form the representation of a sentence:

$$g(\mathbf{x}_{1:n}) = [\hat{c_1}, \hat{c_2}, \ldots, \hat{c_k}]. \tag{4}$$

For the learnable parameters in our model, we denote the word embeddings by W_w and all the other compositional parameters by W_c. In addition, We initialize W_w by some pretrained embeddings in all experiments.

3.2 Training

We follow the training procedure of Wieting et al. (2016), described below. The training data consists of a set S of phrase or sentence pairs $\langle s_1, s_2 \rangle$ from either PPDB (Ganitkevitch et al., 2013) or the aligned

[1] Suppose we have two sentences that are very different in length but have very close meanings. In order to get similar embeddings for them, our model must find a way to capture semantic similarity between words in the short sentence and n-grams in the long one.

Wikipedia sentences (Coster and Kauchak, 2011) where s_1 and s_2 are assumed to be paraphrases. We optimize a margin-based loss:

$$\min_{W_w, W_c} \frac{1}{|S|} \Bigg(\sum_{\langle s_1, s_2 \rangle \in S} \max(0, \delta - \cos(g(s_1), g(s_2)) + \cos(g(s_1), g(t_1))) + \max(0, \delta - \cos(g(s_1), g(s_2)) + \cos(g(s_2), g(t_2))) \Bigg) + \lambda_w \|W_{w_{initial}} - W_w\|^2 + \lambda_c \|W_c\|^2 \tag{5}$$

Where g is our sentence representation model, δ is the margin, λ_w and λ_c are the 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 nimi-batch during optimization. The intuition is that we want the two sentences to be more similar to each other $\cos(g(s_1), g(s_2))$ than either is to their respective negative examples t_1 and t_2, by a margin of at least δ.

Selecting Negative Examples: To select t_1 and t_2 in Eq. 5, we simply choose the most similar example in some set of phrases or sentences (other than those in the given pair). For simplicity, we use the mini-batch for this set, but it could be a different set. That is, we choose t_1 for a given $\langle s_1, s_2 \rangle$ as follows:

$$t_1 = \underset{t:\langle t, \cdot \rangle \in S_b \setminus \{\langle s_1, s_2 \rangle\}}{\arg\max} \cos(g(s_1), g(t))$$

where $S_b \in S$ 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.

4 Experiments

In this part, we present the results of our experiments. We first use the semantic textual similarity (STS) tasks from 2012 to 2016 (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015; Agirre et al., 2016) and the SemEval 2014 SICK (Marelli et al., 2014) test set to evaluate the performance of our model under the transfer learning setting. Then for supervised setting, we choose the SICK 2014 dataset with its standard train/dev/test split.

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. These STS datasets cover a broad range of domains, including news, image and video descriptions, glosses, web forum, twitter and so on. As for the supervised SICK task, it requires us to give a relatedness score for sentence pair on a 1-5 continuous scale similar as STS, but is an easier learning problem since the training and test examples are all drawn from the same distribution.

4.1 Transfer Learning

DataSets and Experimental Settings: As training data, we use noisy phrase pairs from Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) which is automatically derived from naturally-occurring bilingual text. PPDB comes in different sizes (S, M, L, XL, XXL, and XXXL), where each larger size subsumes all smaller ones but contains noisier paraphrases. Considering precision and data size, we choose the XL section of PPDB which contains 3,033,753 unique phrase pairs. We use PARAGRAM-SL999 embeddings (Wieting et al., 2015) to initialize the word embedding matrix (W_w) for the model, and fix the learning procedure, using Adam (Kingma and Ba, 2015) optimizer with a learning rate of 0.001. Moreover, we use filter windows (h) of 1, 2, 3 with 100 feature maps each[2], since we want the dimension of sentence vectors of CSE to be same as the baseline models. The reason we use filter window of 1 is inspired by the strong performance of the bag-of-words PP model. For hyperparameter tuning, we search $\delta \in \{0.4, 0.6, 0.8\}$, $\lambda_w \in \{10^{-4}, 10^{-5}, 10^{-6}\}$, $\lambda_c \in \{10^{-5}, 10^{-6}, 10^{-7}, 0\}$, and batch

[2]We put filter window size selecting part in Section 5.2 for further analyzing the difference between CSE and PP.

size over {32, 64, 128, 256}. We train the model on PPDB for 10 epochs, using STS 2016 datasets for validation, then evaluate on STS 2012-2015 tasks and the SICK test set.

Wieting and Gimpel (2017) found the sentences in the STS test sets are quite different from the short training fragments in PPDB, which may affect performance of models that are more sensitive to overall characteristics of the word sequences. Therefore, after training on PPDB, we train the model for 10 more epochs on another source of data, called SimpWiki, which is a set of sentence pairs, automatically extracted from Simple English Wikipedia and English Wikipedia articles by Coster and Kauchak (2011). This dataset, consists of 167,689 sentence pairs, has been proved useful for semantic textual similarity task by Wieting and Gimpel (2017), although it was extracted for developing text simplification systems. When continuing to train on this dataset, we initialize the model with the parameters learned from PPDB, use the hyperparameters that maximize Pearson's r on the 2016 STS tasks and try dropout (Srivastava et al., 2014) on the word embeddings. Then, we again test CSE on all STS tasks and the SICK test set.

Dataset	**PP**	**CSE**
MSRpar	42.6	**51.6**
MSRvid	74.5	**79.8**
SMT-eur	47.3	**50.7**
OnWN	**70.6**	69.9
SMT-news	58.4	**64.4**
STS 2012 Average	58.7	**63.3**
headline	72.4	**74.0**
OnWN	67.7	**72.1**
FNWN	**43.9**	29.8
STS 2013 Average	**61.3**	58.6
deft forum	48.7	**56.1**
deft news	**73.1**	70.0
headline	69.7	**70.9**
images	78.5	**81.6**
OnWN	78.8	**79.2**
tweet news	**76.4**	74.6
STS 2014 Average	70.9	**72.1**
answers-forums	**68.3**	66.2
answers-students	78.2	**78.4**
belief	**76.2**	68.5
headline	74.8	**76.5**
images	81.4	**82.9**
STS 2015 Average	**75.8**	74.5
2014 SICK	71.6	**71.9**
Total Average	67.7	**68.5**

Table 1: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) when models trained on PPDB XL only. The highest score in each row is in boldface.

Results: In Table 1, we show the results of our model when trained on PPDB XL only. Particularly, this set of results is the median of five runs with random initialisations of the learnable parameters. As baseline, we compare to PP model (Wieting et al., 2016) since it is simple and proved to performs strongly in the STS and SICK tasks, and more importantly, it uses the same training dataset as CSE, which ensures a fair comparison. As we can see, CSE outperforms the baseline in the majority of datasets (14 out of 20) and in total average. We attribute the strong performance of CSE to the functional architecture which represents a sentence in terms of features that depend on words and n-grams. In contrast, PP model only extracts the features of words. But on very few datasets such as 2013 FNWN, our model performs poorly. Upon examination, two characteristics of this dataset may explain the poor performance: a).it

contains many sentence pairs of low similarity with low word overlap. But it is more likely for CSE to overestimate the similarity of sentences for capturing extra grammatical features of n-grams. b).most sentence pairs are very different in length. Since addition pooling differentiates more the length of sentences than mean pooling(or average), it is more difficult for CSE to determine the similarity of this kind of sentence pairs.

Dataset	**GRAN**	**ATT-CCG**	**CSE**
MSRpar	47.7	**49.9**	48.6
MSRvid	85.2	84.2	**86.0**
SMT-eur	49.3	49.3	**49.8**
OnWN	71.5	**72.7**	72.6
SMT-news	58.7	**66.6**	57.8
STS 2012 Average	62.5	**64.5**	63.0
headline	76.1	73.6	**76.3**
OnWN	**81.4**	79.3	78.5
FNWN	**55.6**	50.7	51.5
STS 2013 Average	**71.0**	67.9	68.8
deft forum	55.7	55.2	**58.1**
deft news	**77.1**	75.5	**77.1**
headline	72.8	72.2	**74.6**
images	85.8	83.1	**86.2**
OnWN	**85.1**	84.1	84.2
tweet news	**78.7**	77.7	77.7
STS 2014 Average	75.9	74.6	**76.3**
answers-forums	73.1	69.2	**73.5**
answers-students	72.9	**78.3**	76.7
belief	78.0	**78.4**	76.8
headline	78.6	77.4	**79.8**
images	85.8	85.3	**86.0**
STS 2015 Average	77.7	77.7	**78.6**
answer	-	**64.5**	59.3
headlines	-	70.1	**75.5**
plagiarism	-	82.5	**82.6**
postediting	-	**83.0**	81.4
question	-	58.5	**73.2**
STS 2016 Average	-	71.8	**74.4**
2014 SICK	72.9	-	**73.0**

Table 2: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) after CSE trained on SimpWiki. The highest score in each row is in boldface.

After CSE continues to be trained on SimpWiki, we compare it with two state-of-the-art models: ATT-CCG and GRAN. From Table 2, in the 19 datesets, tested by all three models, CSE outperforms the two other models on 10 datasets. In addition, on SICK 2014 test set, our model performs slightly better than GRAN, and on STS 2016 tasks, our average Pearson's r is 2.6 points higher than ATT-CCG. Our hypothesis explaining the strong performance is that when training our model on SimpWiki, we initialize it with the parameters learned from PPDB, which makes learning process easier.

4.2 Supervised Learning

To investigate whether CSE can also get strong performance in supervised setting, we evaluate it on the SICK 2014 dataset which contains 4,500 sentence pairs in the training set, 500 in the development

set, and 4,927 in the test set. We minimize the objective function[3] from Tai et al. (2015). Given a sentence pair $\langle s_1, s_2 \rangle$ with representations $\langle h_1, h_2 \rangle$ and a relatedness score y in the range $[1, K]$, they first compute:

$$\begin{aligned} h_* &= h_1 \oplus h_2, \\ h_+ &= |h_1 - h_2|, \\ h_s &= \sigma\left(W^{(*)} h_* + W^{(+)} h_+ + b^{(h)}\right), \\ \hat{p_\theta} &= softmax\left(W^{(p)} h_s + b^{(p)}\right), \\ \hat{y} &= r^T \hat{p_\theta}, \end{aligned}$$

Where $r^T = [1\ 2 \ldots K]$, θ represents model parameters. They then define a sparse target distribution p that satisfies $y = r^T p$:

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

for $1 \leq i \leq K$. Finally, they use the regularized KL-divergence between p and $\hat{p_\theta}$ as cost function:

$$J(\theta) = \frac{1}{m} \sum_{k=1}^{m} KL\left(p^{(k)} \| \hat{p_\theta}^{(k)}\right) + \frac{\lambda}{2} \|\theta\|^2, \tag{6}$$

where m is the number of training pairs and the superscript k indicates the k-th sentence pair. For training, we initialize the supervised model with the parameters (including W_w and W_c) from Table 2, and regularize back to their initial values. Besides, we tune λ_w, λ_c and batch size same as in Section 4.1 and the extra hyperparameter $\lambda \in \{10^{-5}, 10^{-6}, 10^{-7}, 0\}$ in Eq. 6, referring to this setup as "universal". Again, we train the model for 10 epochs, using Adam optimizer with a learning rate of 0.001 and use Pearson's r index for validation. As a comparison, We also experiment with CSE with random initialization except initializing word embeddings W_w with PARAGRAM-SL999.

Model	r	ρ	*MSE*
Illinois-LH (Lai and Hockenmaier, 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al., 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al., 2014)	0.8268	0.7721	0.3224
ECNU (Zhao et al., 2014)	0.8414	-	-
Constituency Tree-LSTM (Tai et al., 2015)	0.8491	0.7873	0.2852
Dependency Tree-LSTM (Tai et al., 2015)	0.8676	0.8083	0.2532
CNN (He et al., 2015)	0.8686	0.8047	0.2606
PP$_{universal}$ (Wieting et al., 2016)	0.8684	-	-
GRAN$_{universal}$ (Wieting and Gimpel, 2017)	0.8600	-	-
CSE$_{random}$(This work)	0.8558	0.8059	0.2730
CSE$_{universal}$(This work)	**0.8696**	**0.8087**	**0.2480**

Table 3: Test results on the SICK dataset. Evaluation metrics are Pearson's r, Spearman's ρ, and mean squared error (*MSE*). The results of the first group are from SemEval 2014 systems. The best performance in each metric is in boldface.

The results are shown in Table 3. First, with random initialization, our model is not good enough, slightly better than models from SemEval 2014 systems and Constituency Tree-LSTM. But, initializing with universal parameters and regularizing back to them significantly improves the performance of our

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

model, exceeding the state of the art on all of the three evaluation metrics. The results demonstrate that our universal sentence embeddings carry rich semantic information and therefore can be used as effective features for downstream tasks.

5 Analysis

5.1 Training Data Analysis

The quality of sentence embedding depends heavily on training data source, particularly when experimenting in transfer learning setting, since test examples are not drawn from the same distribution. Wieting and Gimpel (2017) found changing training data from PPDB to SimpWiki is an effective method for improving the performance of sentence model. But, in this paper, we also want to know if our model can get extra boost by cumulative learning, that is, learning on phrase pairs first, then on sentence pairs.

Dateset	CSE$_{phrase}$	CSE$_{sentence}$	CSE$_{cumulative}$
STS 2012 Average	**63.3**	62.0	63.0
STS 2013 Average	58.6	67.3	**68.8**
STS 2014 Average	72.1	75.0	**76.3**
STS 2015 Average	74.5	77.4	**78.6**

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

To answer this question, we experiment with CSE, using phrase pairs dataset PPDB and sentence pairs dataset SimpWiki as training set. In Table 4, we show the results when CSE is trained on phrase or sentence pairs individually and a cumulative style of them two. First, we can see that it is better to use sentence pairs than phrase pairs for training since test sets are all sentences, which suggests CSE is sensitive to the overall characteristics of the word sequences, and the difference between train and test matters much. Otherwise, we find that CSE gains a large margin of performance improvement through the cumulative learning from phrase pairs to sentence pairs, justifying our claim that incremental learning is an important idea to gain extra improvements.

5.2 Filter Size Selecting

Dateset	1-gram	2-gram	3-gram	4-gram	combined
STS 2012 Average	*61.82*	61.53	61.44	60.73	**62.87**
STS 2013 Average	*63.06*	62.14	62.56	62.37	**64.26**
STS 2014 Average	***75.32***	73.61	73.52	73.39	74.26
STS 2015 Average	*76.45*	76.14	76.06	76.11	**76.99**
Total Average	*70.13*	69.29	69.28	69.04	**70.40**

Table 5: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) where n-gram columns show the performance of CSE with filter windows of only size n. The highest score in each row is in boldface, and the highest score for the group of n-gram columns in each dateset is in italic

In this section, we explain how we choose the filter size of our model and analyze how big is the "n" on n-grams that it could be capturing. For comparison, we fix all the other hyperparameters except filter windows size.[4] Then we train the model with default Adam optimizer for 10 epochs, using SimpWiki as training data, STS 2016 for validation and STS 2012 to 2015 as test dataset. Results are shown in Table 5. From the first colunm group, we can see CSE with filter windows of size 1 performs much better than it with bigger window size, which indicates the meaning of words in a sentence plays a very important

[4]For qualitative explanation, we do not fine-tune the hyperparameters and simply fixed δ to 0.6, λ_w to 10^{-6}, λ_c to 0, and batch size to 100. Besides, we initialize the word embedding matrix (W_w) with PARAGRAM-SL999 embeddings and always keep the number of filters equal to 300.

role in composing the semantics of the sentence, which also proves why bag-of-words models like PP can have strong performance in STS tasks. In addition, with the window size increases, the performance of our model is getting poorer, and there is a significant drop in performance when n-gram comes to 4. Inspired by Kim (2014), we then try to combine filters with different window size for additional performance improvements. Simply choosing the three smallest filter windows size of 1, 2, 3 with 100 feature maps each leads us to better results, as shown in the *combined* column.

5.3 Error Analysis

#	Sentence A	Sentence B	CSE	PP	Gold
1	There are a lot of push up variations you can do and they all stress different muscle systems.	There are several different pushup variations out there and most of them provide a unique advantage.	**3.32**	3.14	4.2
2	the methodology takes much less time rather than naive methods.	this is a much quicker method than other more naive methods.	**4.88**	4.79	5.0
3	a boy plays with a noodle by the pool.	a boy plays with a foam noodle toy by a pool.	4.51	**4.61**	4.6
4	two hockey players fighting on the ice.	two hockey players in a struggle on the ice.	**4.78**	4.74	4.8
5	A group of sheep in a field.	A group of horses grazing in a field.	**3.86**	3.89	1.6
6	The lamb is looking at the camera.	A cat looking at the camera.	**3.89**	4.13	0.8
7	A person is riding their bike on a trail next to the woods.	A small child in a yellow shirt is holding their arms out to the sides.	2.63	**2.61**	0.4
8	A boy in a red sled is riding down the hill.	A multicolour dog in a red collar crouching on the grass.	3.29	**3.27**	0.0

Table 6: Illustrative sentence pairs from the STS datasets showing errors made by CSE and PP. The last three columns show the similarity score of CSE, the similarity score of PP, and the gold similarity score. Boldface indicates smaller error compared to gold standard scores.

In this section, we analyze the predictions of our model CSE and the average model PP on the STS datasets. After scaling the predicted cosine similarities into the range of [0, 5], We compare them to the gold standard scores. Examples are illustrated in Table 6. In the first group (examples 1 and 2), we found that when two sentences have a small amount of word overlap but similar meaning, CSE tends to make smaller error for taking the n-gram semantics into account. Then, just as we expected, when two sentences have a lot of word overlap, and have little differences in key semantic roles, PP performs better, as shown in example 3. But in example 4, we got the opposite result, our hypothesis explaining this result is that CNN architectures happen to be good at capturing semantic similarity between words and phrases (*fighting*, *in a struggle*). For the third group (examples 5 and 6), we inspect sentence pairs which have high word overlap rate but different meanings. Under this circumstance, the scores, predicted by CSE, are closer to the golds which suggests PP model is more easily fooled by the high amount of word overlap in such pairs and our model, CSE, is better able to recognize the semantic differences. But in last group (examples 7 and 8) where sentence pairs differ in meaning and have few words in common, CSE performs slightly worse than PP. We think this may due to its capability of capturing extra grammatical features, which leads it more likely to overestimate the similarity of sentences than bag-of-words models (or PP). In addition, from last two groups, both models tend to overestimate sentence pairs of low similarity which may be due to connections between words, whether grammatical or semantic.

6 Conclusion

We introduced a simple CNN architecture to create universal, paraphrastic sentence embeddings. Our model improves upon the state-of-the-art sentence representation models in transfer learning setting and exceeds the state of the art through initialization in the supervised setting. Furthermore, we analyzed the

advantages and disadvantages of our model, and found that it is better at capturing semantic similarity of two sentences than averaging models, especially when they have little word overlap but similar meanings. However, it tends to overestimate the low semantic similarity of a sentence pair. In addition, we released our trained model and codes to facilitate downstream tasks[5]. Future work could extend this model to related tasks including sentiment analysis, text classification and information retrieval.

Acknowledgements

We would like to thank the Writing Mentoring Program of COLING 2018, the anonymous mentor who carefully read our paper and gave very detailed feedback, and the reviewers for their insightful comments. This work was supported in part by National Key R&D Program of China NO.2018YFB10033005, NSFC No.61772216, National Defense Preliminary Research Project (31511010202), Hubei Province Technical Innovation Special Project (2017AAA129), Wuhan Application Basic Research Project (2017010201010103), Project of Shenzhen Technology Scheme JCYJ20170307172248636, Fundamental Research Funds for the Central Universities. This work was also supported by CERNET Innovation Project NGII20170120.

References

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*, SemEval '12, pages 385–393, Stroudsburg, PA, USA. Association for Computational Linguistics.

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*, pages 32–43. Association for Computational Linguistics.

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)*, pages 81–91. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability . In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation . In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511. Association for Computational Linguistics.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The Meaning Factory: Formal Semantics for Recognizing Textual Entailment and Determining Semantic Similarity . In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.*, 12:2493–2537, nov.

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*, pages 665–669. Association for Computational Linguistics.

[5]Trained model and code for training and evaluation are available at `https://github.com/XiaoqiJiao/COLING2018`

Cicero dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78. Dublin City University and Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks . In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586. Association for Computational Linguistics.

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*, pages 1367–1377. Association for Computational Linguistics.

Sepp Hochreiter, Sepp SchmidhuberHochreiter, and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sergio Jimenez, George Dueñas, Julia Baquero, and Alexander Gelbukh. 2014. UNAL-NLP: Combining Soft Cardinality Features for Semantic Textual Similarity, Relatedness and Entailment . In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

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

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A Denotational and Distributional Approach to Semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Yann Lecun, Leon Bottou, Y Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. 86:2278 – 2324, 12.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment . In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based Models of Semantic Composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June. Association for Computational Linguistics.

Donald A Norman. 1972. Memory, knowledge, and the answering of questions.

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

Chengqing Zong Shaonan Wang, Jiajun Zhang. 2017. Learning Sentence Representation with Guidance of Human Attention. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI-17)*, pages 4137–4143.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 801–809, USA. Curran Associates Inc.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

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)*, pages 1556–1566. Association for Computational Linguistics.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. 2017. Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2078–2088. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From Paraphrase Database to Compositional Paraphrase Model and Back. *Transactions of the Association of Computational Linguistics*, 3:345–358.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wenpeng Yin and Hinrich Schütze. 2015. 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*, pages 901–911.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. ECNU: One Stone Two Birds: Ensemble of Heterogenous Measures for Semantic Relatedness and Textual Entailment . In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277. Association for Computational Linguistics.