Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features

Matteo Pagliardini*
Iprova SA, Switzerland
mpagliardini@iprova.com

Prakhar Gupta*
EPFL, Switzerland
prakhar.gupta@epfl.ch

Martin Jaggi
EPFL, Switzerland
martin.jaggi@epfl.ch

Abstract

The recent tremendous success of unsupervised word embeddings in a multitude of applications raises the obvious question if similar methods could be derived to improve embeddings (i.e. semantic representations) of word sequences as well. We present a simple but efficient unsupervised objective to train distributed representations of sentences. Our method outperforms the state-of-the-art unsupervised models on most benchmark tasks, highlighting the robustness of the produced general-purpose sentence embeddings.

1 Introduction

Improving unsupervised learning is of key importance for advancing machine learning methods, as to unlock access to almost unlimited amounts of data to be used as training resources. The majority of recent success stories of deep learning does not fall into this category but instead relied on supervised training (in particular in the vision domain). A very notable exception comes from the text and natural language processing domain, in the form of semantic word embeddings trained unsupervised (Mikolov et al., 2013b,a; Pennington et al., 2014). Within only a few years from their invention, such word representations – which are based on a simple matrix factorization model as we formalize below – are now routinely trained on very large amounts of raw text data, and have become ubiquitous building blocks of a majority of current state-of-the-art NLP applications.

While very useful semantic representations are available for words, it remains challenging to produce and learn such semantic embeddings for longer pieces of text, such as sentences, paragraphs or entire documents. Even more so, it remains a key goal to learn such general-purpose representations in an unsupervised way.

Currently, two contrary research trends have emerged in text representation learning: On one hand, a strong trend in deep-learning for NLP leads towards increasingly powerful and complex models, such as recurrent neural networks (RNNs), LSTMs, attention models and even Neural Turing Machine architectures. While extremely strong in expressiveness, the increased model complexity makes such models much slower to train on larger datasets. On the other end of the spectrum, simpler "shallow" models such as matrix factorizations (or bilinear models) can benefit from training on much larger sets of data, which can be a key advantage, especially in the unsupervised setting.

Surprisingly, for constructing sentence embeddings, naively using averaged word vectors was shown to outperform LSTMs (see Wieting et al. (2016b) for plain averaging, and Arora et al. (2017) for weighted averaging). This example shows potential in exploiting the trade-off between model complexity and ability to process huge amounts of text using scalable algorithms, towards the simpler side. In view of this trade-off, our work here further advances unsupervised learning of sentence embeddings. Our proposed model can be seen as an extension of the C-BOW (Mikolov et al., 2013b,a) training objective to train sentence instead of word embeddings. We demonstrate that the empirical performance of our resulting general-purpose sentence embeddings very significantly exceeds the state of the art, while keeping the model simplicity as well as training and inference complexity exactly as low as in averaging methods (Wieting et al., 2016b; Arora et al., 2017), thereby also putting the work by (Arora et al., 2017) in perspective.

* indicates equal contribution

Proceedings of NAACL-HLT 2018, pages 528–540
New Orleans, Louisiana, June 1 - 6, 2018.

Contributions. The main contributions in this work can be summarized as follows:

- **Model.** We propose **Sent2Vec**[1], a simple unsupervised model allowing to compose sentence embeddings using word vectors along with n-gram embeddings, simultaneously training composition and the embedding vectors themselves.

- **Efficiency & Scalability.** The computational complexity of our embeddings is only $\mathcal{O}(1)$ vector operations per word processed, both during training and inference of the sentence embeddings. This strongly contrasts all neural network based approaches, and allows our model to learn from extremely large datasets, in a streaming fashion, which is a crucial advantage in the unsupervised setting. Fast inference is a key benefit in downstream tasks and industry applications.

- **Performance.** Our method shows significant performance improvements compared to the current state-of-the-art unsupervised and even semi-supervised models. The resulting general-purpose embeddings show strong robustness when transferred to a wide range of prediction benchmarks.

2 Model

Our model is inspired by simple matrix factor models (bilinear models) such as recently very successfully used in unsupervised learning of word embeddings (Mikolov et al., 2013b,a; Pennington et al., 2014; Bojanowski et al., 2017) as well as supervised of sentence classification (Joulin et al., 2017). More precisely, these models can all be formalized as an optimization problem of the form

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S \in \mathcal{C}} f_S(\boldsymbol{U}\boldsymbol{V}\boldsymbol{\iota}_S) \qquad (1)$$

for two parameter matrices $\boldsymbol{U} \in \mathbb{R}^{k \times h}$ and $\boldsymbol{V} \in \mathbb{R}^{h \times |\mathcal{V}|}$, where $\mathcal{V}$ denotes the vocabulary. Here, the columns of the matrix $\boldsymbol{V}$ represent the learnt source word vectors whereas those of $\boldsymbol{U}$ represent the target word vectors. For a given sentence S, which can be of arbitrary length, the indicator vector $\boldsymbol{\iota}_S \in \{0,1\}^{|\mathcal{V}|}$ is a binary vector encoding S (bag of words encoding).

Fixed-length context windows S running over the corpus are used in word embedding methods as in C-BOW (Mikolov et al., 2013b,a) and GloVe (Pennington et al., 2014). Here we have $k = |\mathcal{V}|$ and each cost function $f_S : \mathbb{R}^k \to \mathbb{R}$ only depends on a single row of its input, describing the observed target word for the given fixed-length context S. In contrast, for sentence embeddings which are the focus of our paper here, S will be entire sentences or documents (therefore variable length). This property is shared with the supervised FastText classifier (Joulin et al., 2017), which however uses soft-max with $k \ll |\mathcal{V}|$ being the number of class labels.

2.1 Proposed Unsupervised Model

We propose a new unsupervised model, **Sent2Vec**, for learning universal sentence embeddings. Conceptually, the model can be interpreted as a natural extension of the word-contexts from C-BOW (Mikolov et al., 2013b,a) to a larger sentence context, with the sentence words being specifically optimized towards additive combination over the sentence, by means of the unsupervised objective function.

Formally, we learn a source (or context) embedding $\boldsymbol{v}_w$ and target embedding $\boldsymbol{u}_w$ for each word w in the vocabulary, with embedding dimension h and $k = |\mathcal{V}|$ as in (1). The sentence embedding is defined as the average of the source word embeddings of its constituent words, as in (2). We augment this model furthermore by also learning source embeddings for not only unigrams but also n-grams present in each sentence, and averaging the n-gram embeddings along with the words, i.e., the sentence embedding $\boldsymbol{v}_S$ for S is modeled as

$$\boldsymbol{v}_S := \frac{1}{|R(S)|}\boldsymbol{V}\boldsymbol{\iota}_{R(S)} = \frac{1}{|R(S)|}\sum_{w \in R(S)} \boldsymbol{v}_w \qquad (2)$$

where $R(S)$ is the list of n-grams (including unigrams) present in sentence S. In order to predict a missing word from the context, our objective models the softmax output approximated by negative sampling following (Mikolov et al., 2013b). For the large number of output classes $|\mathcal{V}|$ to be predicted, negative sampling is known to significantly improve training efficiency, see also (Goldberg and Levy, 2014). Given the binary logistic

[1] All our code and pre-trained models will be made publicly available on http://github.com/epfml/sent2vec

loss function $\ell : x \mapsto \log\left(1 + e^{-x}\right)$ coupled with negative sampling, our unsupervised training objective is formulated as follows:

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S\in\mathcal{C}} \sum_{w_t\in S} \Bigg(\ell\big(\boldsymbol{u}_{w_t}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big) + \sum_{w'\in N_{w_t}} \ell\big(-\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big)\Bigg)$$

where S corresponds to the current sentence and N_{w_t} is the set of words sampled negatively for the word $w_t \in S$. The negatives are sampled[2] following a multinomial distribution where each word w is associated with a probability $q_n(w) := \sqrt{f_w} \,/\, \big(\sum_{w_i\in\mathcal{V}} \sqrt{f_{w_i}}\big)$, where f_w is the normalized frequency of w in the corpus.

To select the possible target unigrams (positives), we use subsampling as in (Joulin et al., 2017; Bojanowski et al., 2017), each word w being discarded with probability $1 - q_p(w)$ where $q_p(w) := \min\big\{1, \sqrt{t/f_w} + t/f_w\big\}$. Where t is the subsampling hyper-parameter. Subsampling prevents very frequent words of having too much influence in the learning as they would introduce strong biases in the prediction task. With positives subsampling and respecting the negative sampling distribution, the precise training objective function becomes

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S\in\mathcal{C}} \sum_{w_t\in S} \Bigg(q_p(w_t)\ell\big(\boldsymbol{u}_{w_t}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big) + |N_{w_t}| \sum_{w'\in\mathcal{V}} q_n(w')\ell\big(-\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big)\Bigg) \tag{3}$$

2.2 Computational Efficiency

In contrast to more complex neural network based models, one of the core advantages of the proposed technique is the low computational cost for both inference and training. Given a sentence S and a trained model, computing the sentence representation $\boldsymbol{v}_S$ only requires $|S| \cdot h$ floating point operations (or $|R(S)| \cdot h$ to be precise for the n-gram case, see (2)), where h is the embedding dimension. The same holds for the cost of training with SGD on the objective (3), per sentence seen in the training corpus. Due to the simplicity of the model, parallel training is straight-forward using parallelized or distributed SGD.

Also, in order to store higher-order n-grams efficiently, we use the standard hashing trick, see e.g. (Weinberger et al., 2009), with the same hashing function as used in FastText (Joulin et al., 2017; Bojanowski et al., 2017).

2.3 Comparison to C-BOW

C-BOW (Mikolov et al., 2013b,a) aims to predict a chosen target word given its fixed-size context window, the context being defined by the average of the vectors associated with the words at a distance less than the window size hyper-parameter ws. If our system, when restricted to unigram features, can be seen as an extension of C-BOW where the context window includes the entire sentence, in practice there are few important differences as C-BOW uses important tricks to facilitate the learning of word embeddings. C-BOW first uses frequent word subsampling on the sentences, deciding to discard each token w with probability $q_p(w)$ or alike (small variations exist across implementations). Subsampling prevents the generation of n-grams features, and deprives the sentence of an important part of its syntactical features. It also shortens the distance between subsampled words, implicitly increasing the span of the context window. A second trick consists of using dynamic context windows: for each subsampled word w, the size of its associated context window is sampled uniformly between 1 and ws. Using dynamic context windows is equivalent to weighing by the distance from the focus word w divided by the window size (Levy et al., 2015). This makes the prediction task local, and go against our objective of creating sentence embeddings as we want to learn how to compose all n-gram features present in a sentence. In the results section, we report a significant improvement of our method over C-BOW.

2.4 Model Training

Three different datasets have been used to train our models: the Toronto book corpus[3], Wikipedia sentences and tweets. The Wikipedia and Toronto books sentences have been tokenized using the Stanford NLP library (Manning et al., 2014), while for tweets we used the NLTK tweets tokenizer (Bird et al., 2009). For training, we select a

[2] To efficiently sample negatives, a pre-processing table is constructed, containing the words corresponding to the square root of their corpora frequency. Then, the negatives N_{w_t} are sampled uniformly at random from the negatives table except the target w_t itself, following (Joulin et al., 2017; Bojanowski et al., 2017).

[3] http://www.cs.toronto.edu/~mbweb/

sentence randomly from the dataset and then proceed to select all the possible target unigrams using subsampling. We update the weights using SGD with a linearly decaying learning rate.

Also, to prevent overfitting, for each sentence we use dropout on its list of n-grams $R(S) \setminus \{U(S)\}$, where $U(S)$ is the set of all unigrams contained in sentence S. After empirically trying multiple dropout schemes, we find that dropping K n-grams ($n > 1$) for each sentence is giving superior results compared to dropping each token with some fixed probability. This dropout mechanism would negatively impact shorter sentences. The regularization can be pushed further by applying L1 regularization to the word vectors. Encouraging sparsity in the embedding vectors is particularly beneficial for high dimension h. The additional soft thresholding in every SGD step adds negligible computational cost. See also Appendix B. We train two models on each dataset, one with unigrams only and one with unigrams and bigrams. All training parameters for the models are provided in Table 5 in the appendix. Our C++ implementation builds upon the FastText library (Joulin et al., 2017; Bojanowski et al., 2017). We will make our code and pre-trained models available open-source.

3 Related Work

We discuss existing models which have been proposed to construct sentence embeddings. While there is a large body of works in this direction – several among these using e.g. labelled datasets of paraphrase pairs to obtain sentence embeddings in a supervised manner (Wieting et al., 2016a,b; Conneau et al., 2017) to learn sentence embeddings – we here focus on unsupervised, task-independent models. While some methods require ordered raw text i.e., a coherent corpus where the next sentence is a logical continuation of the previous sentence, others rely only on raw text i.e., an unordered collection of sentences. Finally, we also discuss alternative models built from structured data sources.

3.1 Unsupervised Models Independent of Sentence Ordering

The **ParagraphVector DBOW** model (Le and Mikolov, 2014) is a log-linear model which is trained to learn sentence as well as word embeddings and then use a softmax distribution to predict words contained in the sentence given the sentence vector representation. They also propose a different model **ParagraphVector DM** where they use n-grams of consecutive words along with the sentence vector representation to predict the next word.

(Lev et al., 2015) also presented an early approach to obtain compositional embeddings from word vectors. They use different compositional techniques including static averaging or Fisher vectors of a multivariate Gaussian to obtain sentence embeddings from word2vec models.

Hill et al. (2016a) propose a **Sequential (Denoising) Autoencoder, S(D)AE**. This model first introduces noise in the input data: Firstly each word is deleted with probability p_0, then for each non-overlapping bigram, words are swapped with probability p_x. The model then uses an LSTM-based architecture to retrieve the original sentence from the corrupted version. The model can then be used to encode new sentences into vector representations. In the case of $p_0 = p_x = 0$, the model simply becomes a Sequential Autoencoder. Hill et al. (2016a) also propose a variant **(S(D)AE + embs.)** in which the words are represented by fixed pre-trained word vector embeddings.

Arora et al. (2017) propose a model in which sentences are represented as a weighted average of fixed (pre-trained) word vectors, followed by post-processing step of subtracting the principal component. Using the generative model of (Arora et al., 2016), words are generated conditioned on a sentence "discourse" vector $\boldsymbol{c}_s$:

$$Pr[w \,|\, \boldsymbol{c}_s] = \alpha f_w + (1-\alpha)\frac{\exp(\tilde{\boldsymbol{c}}_s^\top \boldsymbol{v}_w)}{Z_{\tilde{\boldsymbol{c}}_s}},$$

where $Z_{\tilde{\boldsymbol{c}}_s} := \sum_{w \in \mathcal{V}} \exp(\tilde{\boldsymbol{c}}_s^\top \boldsymbol{v}_w)$ and $\tilde{\boldsymbol{c}}_s := \beta \boldsymbol{c}_0 + (1-\beta)\boldsymbol{c}_s$ and α, β are scalars. $\boldsymbol{c}_0$ is the common discourse vector, representing a shared component among all discourses, mainly related to syntax. It allows the model to better generate syntactical features. The αf_w term is here to enable the model to generate some frequent words even if their matching with the discourse vector $\tilde{\boldsymbol{c}}_s$ is low.

Therefore, this model tries to generate sentences as a mixture of three type of words: words matching the sentence discourse vector $\boldsymbol{c}_s$, syntactical words matching $\boldsymbol{c}_0$, and words with high f_w. (Arora et al., 2017) demonstrated that for this model, the MLE of $\tilde{\boldsymbol{c}}_s$ can be approximated by $\sum_{w \in S} \frac{a}{f_w + a} \boldsymbol{v}_w$, where a is a scalar. The sentence

discourse vector can hence be obtained by subtracting $\boldsymbol{c}_0$ estimated by the first principal component of $\tilde{\boldsymbol{c}}_s$'s on a set of sentences. In other words, the sentence embeddings are obtained by a weighted average of the word vectors stripping away the syntax by subtracting the common discourse vector and down-weighting frequent tokens. They generate sentence embeddings from diverse pre-trained word embeddings among which are unsupervised word embeddings such as GloVe (Pennington et al., 2014) as well as supervised word embeddings such as paragram-SL999 (PSL) (Wieting et al., 2015) trained on the Paraphrase Database (Ganitkevitch et al., 2013).

In a very different line of work, **C-PHRASE** (Pham et al., 2015) relies on additional information from the syntactic parse tree of each sentence, which is incorporated into the C-BOW training objective.

Huang and Anandkumar (2016) show that single layer CNNs can be modeled using a tensor decomposition approach. While building on an unsupervised objective, the employed dictionary learning step for obtaining phrase templates is task-specific (for each use-case), not resulting in general-purpose embeddings.

3.2 Unsupervised Models Depending on Sentence Ordering

The **SkipThought** model (Kiros et al., 2015) combines sentence level models with recurrent neural networks. Given a sentence S_i from an ordered corpus, the model is trained to predict S_{i-1} and S_{i+1}.

FastSent (Hill et al., 2016a) is a sentence-level log-linear bag-of-words model. Like SkipThought, it uses adjacent sentences as the prediction target and is trained in an unsupervised fashion. Using word sequences allows the model to improve over the earlier work of paragraph2vec (Le and Mikolov, 2014). (Hill et al., 2016a) augment FastSent further by training it to predict the constituent words of the sentence as well. This model is named **FastSent + AE** in our comparisons.

Compared to our approach, **Siamese C-BOW** (Kenter et al., 2016) shares the idea of learning to average word embeddings over a sentence. However, it relies on a Siamese neural network architecture to predict surrounding sentences, contrasting our simpler unsupervised objective.

Note that on the character sequence level instead of word sequences, FastText (Bojanowski et al., 2017) uses the same conceptual model to obtain better word embeddings. This is most similar to our proposed model, with two key differences: Firstly, we predict from source word sequences to target words, as opposed to character sequences to target words, and secondly, our model is averaging the source embeddings instead of summing them.

3.3 Models requiring structured data

DictRep (Hill et al., 2016b) is trained to map dictionary definitions of the words to the pre-trained word embeddings of these words. They use two different architectures, namely BOW and RNN (LSTM) with the choice of learning the input word embeddings or using them pre-trained. A similar architecture is used by the **CaptionRep** variant, but here the task is the mapping of given image captions to a pre-trained vector representation of these images.

4 Evaluation Tasks

We use a standard set of supervised as well as unsupervised benchmark tasks from the literature to evaluate our trained models, following (Hill et al., 2016a). The breadth of tasks allows to fairly measure generalization to a wide area of different domains, testing the general-purpose quality (universality) of all competing sentence embeddings. For downstream supervised evaluations, sentence embeddings are combined with logistic regression to predict target labels. In the unsupervised evaluation for sentence similarity, correlation of the cosine similarity between two embeddings is compared to human annotators.

Downstream Supervised Evaluation. Sentence embeddings are evaluated for various supervised classification tasks as follows. We evaluate paraphrase identification (MSRP) (Dolan et al., 2004), classification of movie review sentiment (MR) (Pang and Lee, 2005), product reviews (CR) (Hu and Liu, 2004), subjectivity classification (SUBJ) (Pang and Lee, 2004), opinion polarity (MPQA) (Wiebe et al., 2005) and question type classification (TREC) (Voorhees, 2002). To classify, we use the code provided by (Kiros et al., 2015) in the same manner as in (Hill et al., 2016a). For the MSRP dataset, containing pairs of sentences (S_1, S_2) with associated paraphrase label, we generate feature vectors by concatenating

their Sent2Vec representations $|\boldsymbol{v}_{S_1} - \boldsymbol{v}_{S_2}|$ with the component-wise product $\boldsymbol{v}_{S_1} \odot \boldsymbol{v}_{S_2}$. The predefined training split is used to tune the L2 penalty parameter using cross-validation and the accuracy and F1 scores are computed on the test set. For the remaining 5 datasets, Sent2Vec embeddings are inferred from input sentences and directly fed to a logistic regression classifier. Accuracy scores are obtained using 10-fold cross-validation for the MR, CR, SUBJ and MPQA datasets. For those datasets nested cross-validation is used to tune the L2 penalty. For the TREC dataset, as for the MRSP dataset, the L2 penalty is tuned on the predefined train split using 10-fold cross-validation, and the accuracy is computed on the test set.

Unsupervised Similarity Evaluation. We perform unsupervised evaluation of the learnt sentence embeddings using the sentence cosine similarity, on the STS 2014 (Agirre et al., 2014) and SICK 2014 (Marelli et al., 2014) datasets. These similarity scores are compared to the gold-standard human judgements using Pearson's r (Pearson, 1895) and Spearman's ρ (Spearman, 1904) correlation scores. The SICK dataset consists of about 10,000 sentence pairs along with relatedness scores of the pairs. The STS 2014 dataset contains 3,770 pairs, divided into six different categories on the basis of the origin of sentences/phrases, namely Twitter, headlines, news, forum, WordNet and images.

5 Results and Discussion

In Tables 1 and 2, we compare our results with those obtained by (Hill et al., 2016a) on different models. Table 3 in the last column shows the dramatic improvement in training time of our models (and other C-BOW-inspired models) in contrast to neural network based models. All our Sent2Vec models are trained on a machine with 2x Intel Xeon E5−2680v3, 12 cores @2.5GHz. Along with the models discussed in Section 3, this also includes the sentence embedding baselines obtained by simple averaging of word embeddings over the sentence, in both the C-BOW and skip-gram variants. TF-IDF BOW is a representation consisting of the counts of the 200,000 most common feature-words, weighed by their TF-IDF frequencies. To ensure coherence, we only include unsupervised models in the main paper. Performance of supervised and semi-supervised models on these evaluations can be observed in Tables 6 and 7 in the appendix.

Downstream Supervised Evaluation Results. On running supervised evaluations and observing the results in Table 1, we find that on an average our models are second only to SkipThought vectors. Also, both our models achieve state of the art results on the CR task. We also observe that on half of the supervised tasks, our unigrams + bigram model is the best model after SkipThought. Our models are weaker on the MSRP task (which consists of the identification of labelled paraphrases) compared to state-of-the-art methods. However, we observe that the models which perform very strongly on this task end up faring very poorly on the other tasks, indicating a lack of generalizability.

On rest of the tasks, our models perform extremely well. The SkipThought model is able to outperform our models on most of the tasks as it is trained to predict the previous and next sentences and a lot of tasks are able to make use of this contextual information missing in our Sent2Vec models. For example, the TREC task is a poor measure of how one predicts the content of the sentence (the question) but a good measure of how the next sentence in the sequence (the answer) is predicted.

Unsupervised Similarity Evaluation Results. In Table 2, we see that our Sent2Vec models are state-of-the-art on the majority of tasks when comparing to all the unsupervised models trained on the Toronto corpus, and clearly achieve the best averaged performance. Our Sent2Vec models also on average outperform or are at par with the C-PHRASE model, despite significantly lagging behind on the STS 2014 WordNet and News subtasks. This observation can be attributed to the fact that a big chunk of the data that the C-PHRASE model is trained on comes from English Wikipedia, helping it to perform well on datasets involving definition and news items. Also, C-PHRASE uses data three times the size of the Toronto book corpus. Interestingly, our model outperforms C-PHRASE when trained on Wikipedia, as shown in Table 3, despite the fact that we use no parse tree information.

Official STS 2017 benchmark. In the official results of the most recent edition of the STS 2017 benchmark (Cer et al., 2017), our model also significantly outperforms C-PHRASE, and in fact delivers the best unsupervised baseline method.

[4]For the Siamese C-BOW model trained on the Toronto

Data	Model	MSRP (Acc / F1)	MR	CR	SUBJ	MPQA	TREC	Average
Unordered Sentences: (Toronto Books; 70 million sentences, 0.9 Billion Words)	SAE	**74.3** / 81.7	62.6	68.0	86.1	76.8	80.2	74.7
	SAE + embs.	70.6 / 77.9	73.2	75.3	89.8	86.2	80.4	79.3
	SDAE	**76.4 / 83.4**	67.6	74.0	89.3	81.3	77.7	78.3
	SDAE + embs.	**73.7** / 80.7	74.6	78.0	90.8	**86.9**	78.4	80.4
	ParagraphVec DBOW	72.9 / 81.1	60.2	66.9	76.3	70.7	59.4	67.7
	ParagraphVec DM	73.6 / **81.9**	61.5	68.6	76.4	78.1	55.8	69.0
	Skipgram	69.3 / 77.2	73.6	77.3	89.2	85.0	82.2	78.5
	C-BOW	67.6 / 76.1	73.6	77.3	89.1	85.0	82.2	79.1
	Unigram TFIDF	73.6 / 81.7	73.7	79.2	90.3	82.4	**85.0**	80.7
	Sent2Vec uni.	72.2 / 80.3	75.1	**80.2**	90.6	**86.3**	83.8	**81.4**
	Sent2Vec uni. + bi.	72.5 / 80.8	**75.8**	**80.3**	**91.2**	85.9	**86.4**	**82.0**
Ordered Sentences: Toronto Books	SkipThought	73.0 / **82.0**	**76.5**	**80.1**	**93.6**	**87.1**	**92.2**	**83.8**
	FastSent	72.2 / 80.3	70.8	78.4	88.7	80.6	76.8	77.9
	FastSent+AE	71.2 / 79.1	71.8	76.7	88.8	81.5	80.4	78.4
2.8 Billion words	C-PHRASE	72.2 / 79.6	**75.7**	78.8	**91.1**	86.2	78.8	80.5

Table 1: Comparison of the performance of different models on different **supervised evaluation** tasks. An underline indicates the best performance for the dataset. Top 3 performances in each data category are shown in bold. The average is calculated as the average of accuracy for each category (For MSRP, we take the accuracy).)

	STS 2014						SICK 2014	
Model	News	Forum	WordNet	Twitter	Images	Headlines	Test + Train	Average
SAE	.17/.16	.12/.12	.30/.23	.28/.22	.49/.46	.13/.11	.32/.31	.26/.23
SAE + embs.	.52/.54	.22/.23	.60/.55	.60/.60	.64/.64	.41/.41	.47/.49	.50/.49
SDAE	.07/.04	.11/.13	.33/.24	.44/.42	.44/.38	.36/.36	.46/.46	.31/.29
SDAE + embs.	.51/.54	.29/.29	.56/.50	.57/.58	.59/.59	.43/.44	.46/.46	.49/.49
ParagraphVec DBOW	.31/.34	.32/.32	.53/.50	.43/.46	.46/.44	.39/.41	.42/.46	.41/.42
ParagraphVec DM	.42/.46	.33/.34	.51/.48	.54/.57	.32/.30	.46/.47	.44/.40	.43/.43
Skipgram	.56/.59	.42/.42	.73/**.70**	**.71**/.74	.65/.67	.55/.58	.60/.69	.60/.63
C-BOW	.57/.61	**.43/.44**	.72/.69	**.71/.75**	.71/.73	.55/.59	.60/.69	.60/.65
Unigram TF-IDF	.48/.48	.40/.38	.60/.59	.63/.65	.72/.74	.49/.49	.52/.58	.55/.56
Sent2Vec uni.	**.62/.67**	**.49/.49**	**.75/.72**	.70/**.75**	**.78/.82**	**.61/.63**	**.61/.70**	**.65/.68**
Sent2Vec uni. + bi.	**.62/.67**	**.51/.51**	.71/.68	.70/**.75**	**.75/.79**	.59/.62	**.62/.70**	**.65/.67**
SkipThought	.44/.45	.14/.15	.39/.34	.42/.43	.55/.60	.43/.44	.57/.60	.42/.43
FastSent	.58/.59	.41/.36	**.74/.70**	.63/.66	.74/.78	.57/.59	**.61/.72**	.61/.63
FastSent+AE	.56/.59	.41/.40	.69/.64	.70/.74	.63/.65	.58/.60	.60/.65	.60/.61
Siamese C-BOW[4]	.58/.59	.42/.41	.66/.61	**.71**/.73	.65/.65	**.63/.64**	–	–
C-PHRASE	**.69/.71**	**.43**/.41	**.76/.73**	.60/.65	**.75/.79**	**.60/.65**	.60/**.72**	**.63/.67**

Table 2: **Unsupervised Evaluation Tasks**: Comparison of the performance of different models on Spearman/Pearson correlation measures. An underline indicates the best performance for the dataset. Top 3 performances in each data category are shown in bold. The average is calculated as the average of entries for each correlation measure.

Macro Average. To summarize our contributions on both supervised and unsupervised tasks, in Table 3 we present the results in terms of the macro average over the averages of both supervised and unsupervised tasks along with the training times of the models[5]. For unsupervised tasks, averages are taken over both Spearman and Pearson scores. The comparison includes the best performing unsupervised and semi-supervised methods described in Section 3. For models trained on the Toronto books dataset, we report a 3.8 % points improvement over the state of the art. Considering all supervised, semi-supervised methods and all datasets compared in (Hill et al., 2016a), we report a 2.2 % points improvement.

We also see a noticeable improvement in accuracy as we use larger datasets like Twitter and Wikipedia. We furthermore see that the Sent2Vec models are faster to train when compared to methods like SkipThought and DictRep, owing to the SGD optimizer allowing a high degree of parallelizability.

We can clearly see Sent2Vec outperforming other unsupervised and even semi-supervised methods. This can be attributed to the superior generalizability of our model across supervised and unsupervised tasks.

Comparison with Arora et al. (2017). We also compare our work with Arora et al. (2017) who also use additive compositionality to obtain sentence embeddings. However, in contrast to our

corpus, supervised evaluation as well as similarity evaluation results on the SICK 2014 dataset are unavailable.

[5]time taken to train C-PHRASE models is unavailable

Type	Training corpus	Method	Supervised average	Unsupervised average	Macro average	Training time (in hours)
unsupervised	twitter (19.7B words)	**Sent2Vec uni. + bi.**	83.5	68.3	75.9	6.5*
unsupervised	twitter (19.7B words)	**Sent2Vec uni.**	82.2	69.0	75.6	3*
unsupervised	Wikipedia (1.7B words)	**Sent2Vec uni. + bi.**	83.3	66.2	74.8	2*
unsupervised	Wikipedia (1.7B words)	**Sent2Vec uni.**	82.4	66.3	74.3	3.5*
unsupervised	Toronto books (0.9B words)	**Sent2Vec books uni.**	81.4	66.7	74.0	1*
unsupervised	Toronto books (0.9B words)	**Sent2Vec books uni. + bi.**	82.0	65.9	74.0	1.2*
semi-supervised	structured dictionary dataset	DictRep BOW + emb	80.5	66.9	73.7	24**
unsupervised	2.8B words + parse info.	C-PHRASE	80.5	64.9	72.7	–
unsupervised	Toronto books (0.9B words)	C-BOW	79.1	62.8	70.2	2
unsupervised	Toronto books (0.9B words)	FastSent	77.9	62.0	70.0	2
unsupervised	Toronto books (0.9B words)	SkipThought	83.8	42.5	63.1	336**

Table 3: Best unsupervised and semi-supervised methods ranked by macro average along with their training times. ** indicates trained on GPU. * indicates trained on a single node using 30 threads. Training times for non-Sent2Vec models are due to Hill et al. (2016a). For CPU based competing methods, we were able to reproduce all published timings (+-10%) using our same hardware as for training Sent2Vec.

Dataset	Unsupervised GloVe (840B words) + WR	Semi-supervised PSL + WR	Sent2Vec Unigrams (19.7B words) Tweets Model	Sent2Vec Unigrams + Bigrams (19.7B words) Tweets Model
STS 2014	0.685	0.735	0.710	0.701
SICK 2014	0.722	0.729	0.710	0.715
Supervised average	0.815	0.807	0.822	0.835

Table 4: Comparison of the performance of the unsupervised and semi-supervised sentence embeddings by (Arora et al., 2017) with our models. Unsupervised comparisons are in terms of Pearson's correlation, while comparisons on supervised tasks are stating the average described in Table 1.

model, they use fixed, pre-trained word embeddings to build a weighted average of these embeddings using unigram probabilities. While we couldn't find pre-trained state of the art word embeddings trained on the Toronto books corpus, we evaluated their method using GloVe embeddings obtained from the larger Common Crawl Corpus[6], which is 42 times larger than our twitter corpus, greatly favoring their method over ours.

In Table 4, we report an experimental comparison to their model on unsupervised tasks. In the table, the suffix W indicates that their down-weighting scheme has been used, while the suffix R indicates the removal of the first principal component. They report values of $a \in [10^{-4}, 10^{-3}]$ as giving the best results and used $a = 10^{-3}$ for all their experiments. We observe that our results are competitive with the embeddings of Arora et al. (2017) for purely unsupervised methods. It is important to note that the scores obtained from supervised task-specific PSL embeddings trained for the purpose of semantic similarity outperform our method on both SICK and average STS 2014, which is expected as our model is trained purely unsupervised.

In order to facilitate a more detailed comparison, we also evaluated the unsupervised Glove + WR embeddings on downstream supervised tasks and compared them to our twitter models. To use Arora et al. (2017)'s method in a supervised setup, we precomputed and stored the common discourse vector $\boldsymbol{c}_0$ using 2 million random Wikipedia sentences. On an average, our models outperform their unsupervised models by a significant margin, this despite the fact that they used GloVe embeddings trained on larger corpora than ours (42 times larger). Our models also outperform their semi-supervised PSL + WR model. This indicates our model learns a more precise weighing scheme than the static one proposed by Arora et al. (2017).

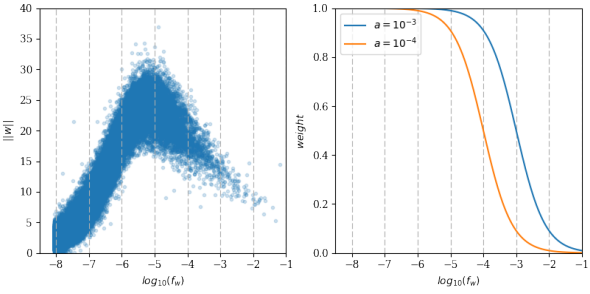

Figure 1: *Left figure:* the profile of the word vector L_2-norms as a function of $\log(f_w)$ for each vocabulary word w, as learnt by our unigram model trained on Toronto books. *Right figure:* down-weighting scheme proposed by Arora et al. (2017): $weight(w) = \frac{a}{a+f_w}$.

The effect of datasets and n-grams. Despite being trained on three very different datasets, all of our models generalize well to sometimes very

[6] http://www.cs.toronto.edu/~mbweb/

specific domains. Models trained on Toronto Corpus are the state-of-the-art on the STS 2014 images dataset even beating the supervised CaptionRep model trained on images. We also see that addition of bigrams to our models doesn't help much when it comes to unsupervised evaluations but gives a significant boost-up in accuracy on supervised tasks. We attribute this phenomenon to the ability of bigrams models to capture some non-compositional features missed by unigrams models. Having a single representation for *"not good"* or *"very bad"* can boost the supervised model's ability to infer relevant features for the corresponding classifier. For semantic similarity tasks however, the relative uniqueness of bigrams results in pushing sentence representations further apart, which can explain the average drop of scores for bigrams models on those tasks.

On learning the importance and the direction of the word vectors. Our model – by learning how to generate and compose word vectors – has to learn both the direction of the word embeddings as well as their norm. Considering the norms of the used word vectors as by our averaging over the sentence, we observe an interesting distribution of the "importance" of each word. In Figure 1 we show the profile of the L_2-norm as a function of $\log(f_w)$ for each $w \in \mathcal{V}$, and compare it to the static down-weighting mechanism of Arora et al. (2017). We can observe that our model is learning to down-weight frequent tokens by itself. It is also down-weighting rare tokens and the $norm$ profile seems to roughly follow Luhn's hypothesis (Luhn, 1958), a well known information retrieval paradigm, stating that mid-rank terms are the most significant to discriminate content.

6 Conclusion

In this paper, we introduce a novel, computationally efficient, unsupervised, C-BOW-inspired method to train and infer sentence embeddings. On supervised evaluations, our method, on an average, achieves better performance than all other unsupervised competitors with the exception of SkipThought. However, SkipThought vectors show a very poor performance on sentence similarity tasks while our model is state-of-the-art for these evaluations on average. Also, our model is generalizable, extremely fast to train, simple to understand and easily interpretable, showing the relevance of simple and well-grounded representation models in contrast to the models using deep architectures. Future work could focus on augmenting the model to exploit data with ordered sentences. Furthermore, we would like to investigate the model's ability to use pre-trained embeddings for downstream transfer learning tasks.

Acknowledgments

We are indebted to Piotr Bojanowski and Armand Joulin for helpful discussions. This project was supported by a Google Faculty Research Award.

References

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

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A Latent Variable Model Approach to PMI-based Word Embeddings. In *Transactions of the Association for Computational Linguistics*. pages 385–399.

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

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5:135–146.

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. In *SemEval-2017 - Proceedings of the 11th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, Vancouver, Canada, pages 1–14.

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

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.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*. pages 758–764.

Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv* .

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016a. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of NAACL-HLT*.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016b. Learning to understand phrases by embedding the dictionary. *TACL* 4:17–30.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 168–177.

Furong Huang and Animashree Anandkumar. 2016. Unsupervised Learning of Word-Sequence Representations from Scratch via Convolutional Tensor Decomposition. *arXiv* .

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Short Papers*. Valencia, Spain, pages 427–431.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. In *ACL - Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 941–951.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *NIPS 2015 - Advances in Neural Information Processing Systems 28*. pages 3294–3302.

Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML 2014 - Proceedings of the 31st International Conference on Machine Learning*. volume 14, pages 1188–1196.

Guy Lev, Benjamin Klein, and Lior Wolf. 2015. In defense of word embedding for generic text representation. In *International Conference on Applications of Natural Language to Information Systems*. Springer, pages 35–50.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development* 2(2):159–165.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*. pages 216–223.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS - Advances in Neural Information Processing Systems 26*. pages 3111–3119.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 271.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 115–124.

Karl Pearson. 1895. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London* 58:240–242.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

NT Pham, G Kruszewski, A Lazaridou, and M Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. *ACL/IJCNLP* .

R Tyrrell Rockafellar. 1976. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization* 14(5):877–898.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology* 15(1):72–101.

Ellen M Voorhees. 2002. Overview of the trec 2001 question answering track. In *NIST special publication*. pages 42–51.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pages 1113–1120.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* 39(2):165–210.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding Words and Sentences via Character n-grams. In *EMNLP - Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 1504–1515.

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

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

A Parameters for training models

Model	Embedding Dimensions	Minimum word count	Minimum Target word Count	Initial Lear ning Rate	Epochs	Subsampling hyper-parameter	Bigrams Dropped per sentence	Number of negatives sampled
Book corpus Sent2Vec unigrams	700	5	8	0.2	13	1×10^{-5}	-	10
Book corpus Sent2Vec unigrams + bigrams	700	5	5	0.2	12	5×10^{-6}	7	10
Wiki Sent2Vec unigrams	600	8	20	0.2	9	1×10^{-5}	-	10
Wiki Sent2Vec unigrams + bigrams	700	8	20	0.2	9	5×10^{-6}	4	10
Twitter Sent2Vec unigrams	700	20	20	0.2	3	1×10^{-6}	-	10
Twitter Sent2Vec unigrams + bigrams	700	20	20	0.2	3	1×10^{-6}	3	10

Table 5: Training parameters for the Sent2Vec models

B L1 regularization of models

Optionally, our model can be additionally improved by adding an L1 regularizer term in the objective function, leading to slightly better generalization performance. Additionally, encouraging sparsity in the embedding vectors is beneficial for memory reasons, allowing higher embedding dimensions h.

We propose to apply L1 regularization individually to each word (and n-gram) vector (both source and target vectors). Formally, the training objective function (3) then becomes

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{S \in \mathcal{C}} \sum_{w_t \in S} q_p(w_t) \Bigg(\Big(\ell\big(\boldsymbol{u}_{w_t}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big) + \tau(\|\boldsymbol{u}_{w_t}\|_1 + \|\boldsymbol{v}_{S\setminus\{w_t\}}\|_1) \Big) + |N_{w_t}| \sum_{w' \in \mathcal{V}} q_n(w') \Big(\ell\big(-\boldsymbol{u}_{w'}^\top \boldsymbol{v}_{S\setminus\{w_t\}}\big) + \tau(\|\boldsymbol{u}_{w'}\|_1) \Big) \Bigg) \tag{4}$$

where τ is the regularization parameter.

Now, in order to minimize a function of the form $f(\mathbf{z}) + g(\mathbf{z})$ where $g(\mathbf{z})$ is not differentiable over the domain, we can use the basic proximal-gradient scheme. In this iterative method, after doing a gradient descent step on $f(\mathbf{z})$ with learning rate α, we update $\mathbf{z}$ as

$$\mathbf{z}_{n+1} = prox_{\alpha,g}(\mathbf{z}_{n+\frac{1}{2}}) \tag{5}$$

where $prox_{\alpha,g}(\mathbf{x}) = \arg\min_{\mathbf{y}} \{g(\mathbf{y}) + \frac{1}{2\alpha}\|\mathbf{y}-\mathbf{x}\|_2^2\}$ is called the proximal function (Rockafellar, 1976) of g with α being the proximal parameter and $\mathbf{z}_{n+\frac{1}{2}}$ is the value of $\mathbf{z}$ after a gradient (or SGD) step on $\mathbf{z}_n$.

In our case, $g(\mathbf{z}) = \|\mathbf{z}\|_1$ and the corresponding proximal operator is given by

$$prox_{\alpha,g}(\mathbf{x}) = sign(\mathbf{x}) \odot \max(|\mathbf{x}_n| - \alpha, 0) \tag{6}$$

where $\odot$ corresponds to element-wise product.

Similar to the proximal-gradient scheme, in our case we can optionally use the thresholding operator on the updated word and n-gram vectors after an SGD step. The soft thresholding parameter used for this update is $\frac{\tau \cdot lr'}{|R(S\setminus\{w_t\})|}$ and $\tau \cdot lr'$ for the source and target vectors respectively where lr' is the current learning rate, τ is the $L1$ regularization parameter and S is the sentence on which SGD is being run.

We observe that $L1$ regularization using the proximal step gives our models a small boost in performance. Also, applying the thresholding operator takes only $|R(S \setminus \{w_t\})| \cdot h$ floating point operations for the updating the word vectors corresponding to the sentence and $(|N|+1) \cdot h$ for updating the target

as well as the negative word vectors, where $|N|$ is the number of negatives sampled and h is the embedding dimension. Thus, performing $L1$ regularization using soft-thresholding operator comes with a small computational overhead.

We set τ to be 0.0005 for both the Wikipedia and the Toronto Book Corpus unigrams + bigrams models.

C Performance comparison with Sent2Vec models trained on different corpora

Data	Model	MSRP (Acc / F1)	MR	CR	SUBJ	MPQA	TREC	Average
Unordered Sentences: (Toronto Books)	**Sent2Vec uni.**	72.2 / 80.3	75.1	**80.2**	90.6	86.3	83.8	81.4
	Sent2Vec uni. + bi.	72.5 / 80.8	75.8	**80.3**	91.2	85.9	86.4	82.0
	Sent2Vec uni. + bi. $L1$-reg	71.6 / 80.1	76.1	**80.9**	91.1	86.1	86.8	82.1
Unordered sentences: Wikipedia (69 million sentences; 1.7 B words)	**Sent2Vec uni.**	71.8 / 80.2	**77.3**	**80.3**	92.0	**87.4**	85.4	82.4
	Sent2Vec uni. + bi.	72.4 / 80.8	**77.9**	**80.9**	92.6	86.9	89.2	83.3
	Sent2Vec uni. + bi. $L1$-reg	73.6 / 81.5	**78.1**	**81.5**	92.8	**87.2**	87.4	83.4
Unordered sentences: Twitter (1.2 billion sentences; 19.7 B words)	**Sent2Vec uni.**	71.5 / 80.0	**77.1**	**81.3**	90.8	**87.3**	85.4	82.2
	Sent2Vec uni. + bi.	72.4 / 80.6	**78.0**	**82.1**	91.8	86.7	89.8	83.5
Other structured Data Sources	CaptionRep BOW	73.6 / 81.9	61.9	69.3	77.4	70.8	72.2	70.9
	CaptionRep RNN	72.6 / 81.1	55.0	64.9	64.9	71.0	62.4	65.1
	DictRep BOW	73.7 / 81.6	71.3	75.6	86.6	82.5	73.8	77.3
	DictRep BOW+embs	68.4 / 76.8	76.7	78.7	90.7	87.2	81.0	80.5
	DictRep RNN	73.2 / 81.6	67.8	72.7	81.4	82.5	75.8	75.6
	DictRep RNN+embs.	66.8 / 76.0	72.5	73.5	85.6	85.7	72.0	76.0

Table 6: Comparison of the performance of different Sent2Vec models with different semi-supervised/supervised models on different **downstream supervised evaluation** tasks. An underline indicates the best performance for the dataset and Sent2Vec model performances are bold if they perform as well or better than all other non-Sent2Vec models, including those presented in Table 1.

	STS 2014						SICK 2014	Average
Model	News	Forum	WordNet	Twitter	Images	Headlines	Test + Train	
Sent2Vec book corpus uni.	.62/.67	**.49/.49**	.75/.72.	.70/**.75**	**.78/.82**	.61/.63	**.61**/.70	**.65**/.68
Sent2Vec book corpus uni. + bi.	.62/.67	**.51/.51**	.71/.68	.70/**.75**	.75/.79	.59/.62	**.62**/.70	**.65**/.67
Sent2Vec book corpus uni. + bi. $L1$-reg	.62/.68	**.51/.52**	.72/.70	.69/**.75**	.76/.81	.60/.63	**.62**/.71	**.66**/.68
Sent2Vec wiki uni.	.66/.71	.47/**.47**	.70/.68	.68/.72	.76/.79	**.63/.67**	**.64**/.71	**.65**/.68
Sent2Vec wiki uni. + bi.	.68/**.74**	**.50/.50**	.66/.64	.67/.72	.75/.79	.62/**.67**	**.63**/.71	**.65**/.68
Sent2Vec wiki uni. + bi. $L1$-reg	**.69/.75**	**.52/.52**	.72/.69	.67/.72	.76/.80	.61/**.66**	**.63/.72**	**.66**/.69
Sent2Vec twitter uni.	.67/**.74**	**.52/.53**	.75/.72	**.72/.78**	.77/.81	**.64/.68**	**.62**/.71	**.67/.71**
Sent2Vec twitter uni. + bi.	.68/**.74**	**.54/.54**	.72/.69	.70/**.77**	.76/.79	.62/**.67**	**.63/.72**	**.66/.70**
CaptionRep BOW	.26/.26	.29/.22	.50/.35	.37/.31	.78/.81	.39/.36	.45/.44	.54/.62
CaptionRep RNN	.05/.05	.13/.09	.40/.33	.36/.30	.76/.82	.30/.28	.36/.35	.51/.59
DictRep BOW	.62/.67	.42/.40	.81/.81	.62/.66	.66/.68	.53/.58	.61/.63	.58/.66
DictRep BOW + embs.	.65/.72	.49/.47	.85/.86	.67/.72	.71/.74	.57/.61	.61/.70	.62/.70
DictRep RNN	.40/.46	.26/.23	.78/.78	.42/.42	.56/.56	.38/.40	.47/.49	.49/.55
DictRep RNN + embs.	.51/.60	.29/.27	.80/.81	.44/.47	.65/.70	.42/.46	.52/.56	.49/.59

Table 7: **Unsupervised Evaluation**: Comparison of the performance of different Sent2Vec models with semi-supervised/supervised models on Spearman/Pearson correlation measures. An underline indicates the best performance for the dataset and Sent2Vec model performances are bold if they perform as well or better than all other non-Sent2Vec models, including those presented in Table 2.

D Dataset Description

	STS 2014						SICK 2014	Wikipedia	Twitter	Book Corpus
Sentence Length	News	Forum	WordNet	Twitter	Images	Headlines	Test + Train	Dataset	Dataset	Dataset
Average	17.23	10.12	8.85	11.64	10.17	7.82	9.67	25.25	16.31	13.32
Standard Deviation	8.66	3.30	3.10	5.28	2.77	2.21	3.75	12.56	7.22	8.94

Table 8: Average sentence lengths for the datasets used in the comparison.