context2vec: Learning Generic Context Embedding with Bidirectional LSTM

Oren Melamud
Computer Science Dept.
Bar-Ilan University
melamuo@cs.biu.ac.il

Jacob Goldberger
Faculty of Engineering
Bar-Ilan University
goldbej@eng.biu.ac.il

Ido Dagan
Computer Science Dept.
Bar-Ilan University
dagan@cs.biu.ac.il

Abstract

Context representations are central to various NLP tasks, such as word sense disambiguation, named entity recognition, co-reference resolution, and many more. In this work we present a neural model for efficiently learning a generic context embedding function from large corpora, using bidirectional LSTM. With a very simple application of our context representations, we manage to surpass or nearly reach state-of-the-art results on sentence completion, lexical substitution and word sense disambiguation tasks, while substantially outperforming the popular context representation of averaged word embeddings. We release our code and pre-trained models, suggesting they could be useful in a wide variety of NLP tasks.

1 Introduction

Generic word embeddings capture semantic and syntactic information about individual words in a compact low-dimensional representation. While they are trained to optimize a generic task-independent objective function, word embeddings were found useful in a broad range of NLP tasks, making an overall huge impact in recent years. A major advancement in this field was the introduction of highly efficient models, such as *word2vec* (Mikolov et al., 2013a) and *GloVe* (Pennington et al., 2014), for learning generic word embeddings from very large corpora. Capturing information from such corpora substantially increased the value of word embeddings to both unsupervised and semi-supervised NLP tasks.

To make inferences regarding a concrete target word instance, good representations of both the target word type and the given context are helpful. For example, in the sentence "*I can't find* [*April*]", we need to consider both the target word *April* and its context "*I can't find* [ ]" to infer that *April* probably refers to a person. This principle applies to various tasks, including word sense disambiguation, co-reference resolution and named entity recognition (NER).

Like target words, contexts are commonly represented via word embeddings. In an unsupervised setting, such representations were found useful for measuring context-sensitive similarity (Huang et al., 2012), word sense disambiguation (Chen et al., 2014), word sense induction (Kågebäck et al., 2015), lexical substitution (Melamud et al., 2015b), sentence completion (Liu et al., 2015) and more. The context representations used in such tasks are commonly just a simple collection of the individual embeddings of the neighboring words in a window around the target word, or a (sometimes weighted) average of these embeddings. We note that such approaches do not include any mechanism for optimizing the representation of the entire sentential context as a whole.

In supervised settings, various NLP systems use labeled data to learn how to consider context word representations in a more optimized task-specific way. This was done in tasks, such as chunking, NER, semantic role labeling, and co-reference resolution (Turian et al., 2010; Collobert et al., 2011; Melamud et al., 2016), mostly by considering the embeddings of words in a window around the target of interest. More recently, bidirectional recurrent neural networks, and specifically bidirectional LSTMs, were used in such tasks to learn

Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), pages 51–61,
Berlin, Germany, August 7-12, 2016.

internal representations of wider sentential contexts (Zhou and Xu, 2015; Lample et al., 2016). Since supervised data is usually limited in size, it has been shown that training such systems, using word embeddings that were pre-trained on large corpora, improves performance significantly. Yet, pre-trained word embeddings carry limited information regarding the inter-dependencies between target words and their sentential context as a whole. To model this (and more), the supervised systems still need to rely heavily on their albeit limited supervised data.

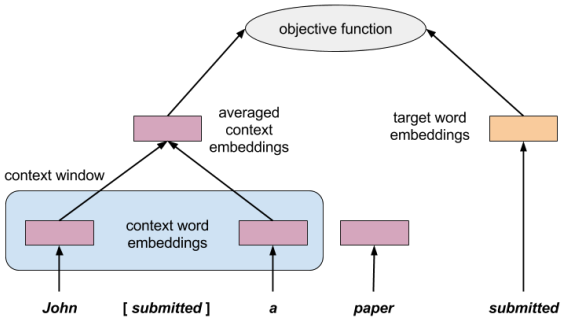
In this work we present *context2vec*, an unsupervised model and toolkit[1] for efficiently learning *generic* context embedding of wide sentential contexts, using bidirectional LSTM. Essentially, we use large plain text corpora to learn a neural model that embeds entire sentential contexts and target words in the same low-dimensional space, which is optimized to reflect inter-dependencies between targets and their entire sentential context as a whole. To demonstrate their high quality, we show that with a very simple application of our context representations, we are able to surpass or nearly reach state-of-the-art results on sentence completion, lexical substitution and word sense disambiguation tasks, while substantially outperforming the common average-of-word-embeddings representation (denoted *AWE*). We further hypothesize that both unsupervised and semi-supervised systems may benefit from using our pre-trained models, instead or in addition to individual pre-trained word embeddings.

2 *Context2vec*'s Neural Model

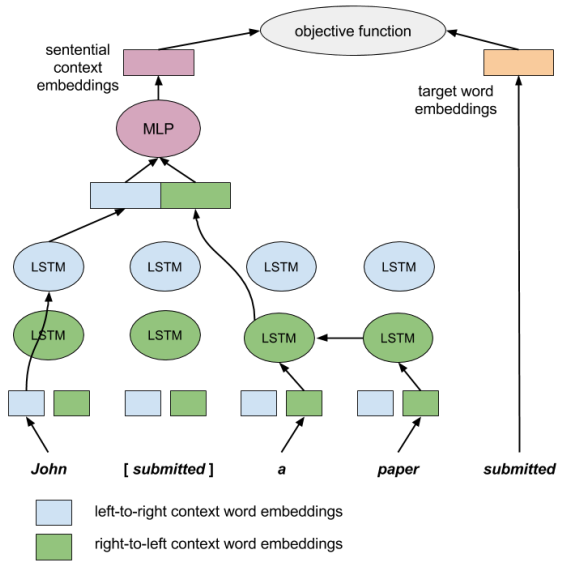
2.1 Model Overview

The main goal of our model is to learn a generic task-independent embedding function for variable-length sentential contexts around target words. To do this, we propose a neural network architecture, which is based on *word2vec*'s *CBOW* architecture (Mikolov et al., 2013a), but replaces its naive context modeling of averaged word embeddings in a fixed window, with a much more powerful neural model, using bidirectional LSTM. Our proposed architecture is illustrated in Figure 1, together with the analogical *word2vec* architecture. Both models learn context and target

[1]Source code and pre-trained models are available at: http://www.cs.biu.ac.il/nlp/resources/downloads/context2vec/



(a) word2vec *CBOW*



(b) *context2vec*

Figure 1: *word2vec* and *context2vec* architectures.

word representations at the same time, by embedding them into the same low-dimensional space, with the objective of having the context predict the target word via a log linear model. However, we utilize a much more powerful parametric model to capture the essence of sentential context.

The left-hand side of Figure 1b illustrates how *context2vec* represents sentential context. We use a bidirectional LSTM recurrent neural network, feeding one LSTM network with the sentence words from left to right, and another from right to left. The parameters of these two networks are completely separate, including two separate sets of left-to-right and right-to-left context word embeddings. To represent the context of a target word in a sentence (e.g. for "*John* [*submitted*] *a paper*"), we first concatenate the LSTM output vector representing its left-to-right context ("*John*") with the one representing its right-to-left context ("*a paper*"). With this, we aim to capture the relevant

information in the sentential context, even when it is remote from the target word. Next, we feed this concatenated vector into a multi-layer perceptron to be capable of representing non-trivial dependencies between the two sides of the context. We consider the output of this layer as the embedding of the entire joint sentential context around the target word. At the same time, the target word itself (right-hand side of Figure 1b) is represented with its own embedding, equal in dimensionality to that of the sentential context. We note that the only (yet crucial) difference between our model and *word2vec*'s *CBOW* (Figure 1a) is that *CBOW* represents the context around a target word as a simple average of the embeddings of the context words in a window around it, while *context2vec* utilizes a full-sentence neural representation of context.

Finally, to learn the parameters of our network, we use *word2vec*'s negative sampling objective function, with a positive pair being a target word and its entire sentential context, and respective k negative pairs as random target words, sampled from a (smoothed) unigram distribution over the vocabulary, paired with the same context. With this, we learn both the context embedding network parameters and the target word embeddings.

In contrast to *word2vec* and similar word embedding models that use context modeling mostly internally and consider the target word embeddings as their main output, our primary focus is the context representation. Our model achieves its objective by assigning similar embeddings to sentential contexts and their associated target words. Further, similar to the case in *word2vec* models, this indirectly results in assigning similar embeddings to target words that are associated with similar sentential contexts, and conversely to sentential contexts that are associated with similar target words. We will show in the following sections how these properties make our model useful.

2.2 Formal Specification and Analysis

We use a bidirectional LSTM recurrent neural network to obtain a sentence-level context representation. Let lLS be an LSTM reading the words of a given sentence from left to right, and let rLS be a reverse one reading the words from right to left. Given a sentence $w_{1:n}$, our 'shallow' bidirectional LSTM context representation for the target w_i is defined as the following vector concatenation:

$$\text{biLS}(w_{1:n}, i) = \text{lLS}(l_{1:i-1}) \oplus \text{rLS}(r_{n:i+1})$$

where l/r represent distinct left-to-right/right-to-left word embeddings of the sentence words.[2] This definition is a bit different than standard bidirectional LSTM, as we do not feed the LSTMs with the target word itself (i.e. the word in position i). Next, we apply the following non-linear function on the concatenation of the left and right context representations:

$$\text{MLP}(x) = L_2(\text{ReLU}(L_1(x)))$$

where MLP stands for Multi Layer Perceptron, ReLU is the Rectified Linear Unit activation function, and $L_i(x) = W_i x + b_i$ is a fully connected linear operation. Let $c = (w_1, ..., w_{i-1}, -, w_{i+1}, ..., w_n)$ be the sentential context of the word in position i. We define *context2vec*'s representation of c as:

$$\vec{c} = \text{MLP}(\text{biLS}(w_{1:n}, i)).$$

Next, we denote the embedding of a target word t as $\vec{t}$. We use the same embedding dimensionality for target and sentential context representations. To learn target word and context representations, we use the *word2vec* negative sampling objective function (Mikolov et al., 2013b):

$$S = \sum_{t,c} \left(\log \sigma(\vec{t} \cdot \vec{c}) + \sum_{i=1}^{k} \log \sigma(-\vec{t_i} \cdot \vec{c}) \right) \quad (1)$$

where the summation goes over each word token t in the training corpus and its corresponding (single) sentential context c, and σ is the sigmoid function. $t_1, ..., t_k$ are the negative samples, independently sampled from a smoothed version of the target words unigram distribution: $p_\alpha(t) \propto (\#t)^\alpha$, such that $0 \leq \alpha < 1$ is a smoothing factor, which increases the probability of rare words.

Levy and Goldberg (2014b) proved that when the objective function in Equation (1) is applied to single-word contexts, it is optimized when:

$$\vec{t} \cdot \vec{c} = \text{PMI}_\alpha(t, c) - \log(k) \quad (2)$$

where $\text{PMI}(t, c) = \log \frac{p(t,c)}{p_\alpha(t)p(c)}$ is the pointwise mutual information between the target word t and

[2] We pad every input sentence with special BOS and EOS words in positions 0 and $n + 1$, respectively.

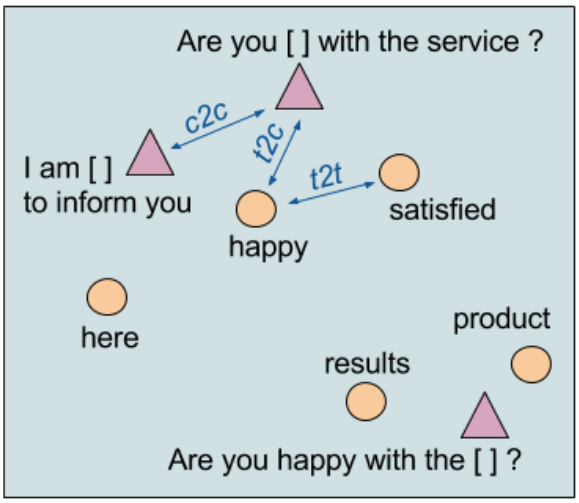


Figure 2: A 2D illustration of *context2vec*'s embedded space and similarity metrics. Triangles and circles denote sentential context embeddings and target word embeddings, respectively.

the context word c. The analysis presented in Levy and Goldberg (2014b) is valid for every co-occurrence matrix that describes the joint distribution of two random variables. Specifically, it can be applied to our case, where the context is not just a single word but an entire sentential context of a target word. Accordingly, we can view the target-context embedding obtained by our algorithm as a factorization of the PMI matrix between all possible target words and all possible different sentential contexts. Unlike the case of single-word contexts, it is not feasible to explicitly compute here this PMI matrix due to the exponential number of possible sentential contexts. However, the objective function that we optimize still aims to best approximate it. Based on the above analysis, we can expect the inner-product of our target and context embeddings to approximate $\text{PMI}_\alpha(c, t)$. We note that accordingly, with larger values of α, there will be more bias towards placing rare words closer to their associated contexts in this space.

2.3 Model Illustration

To demonstrate the qualities of the embedded space learned by *context2vec*, we illustrate three types of similarity metrics in that space: target-to-context (*t2c*), context-to-context (*c2c*) and target-to-target (*t2t*). All these are measured by the vector cosine value between the respective embedding representations. Only the latter target-to-target metric is the one typically used when illustrating and evaluating word embedding models, such as *word2vec*. Figure 2 provides a 2D illustration of such a space and respective metrics.

In Table 1 we show sentential contexts and the target words that are closest to them, using the target-to-context similarity metric with *context2vec* embeddings. As can be seen, the bidirectional LSTM modeling of *context2vec* is indeed capable in this case to capture long range dependencies, as well as to take both sides of the context into account. In Table 2 we show the closest target words to given contexts, using different *context2vec* models, each learned with a different negative sampling smoothing parameter α. This illustrates the bias that high α values introduce towards rare words, as predicted with the analysis in section 2.2.

Next, to illustrate the context-to-context similarity metric, we took the set of contexts for the target lemma *add* from the training set of Senseval-3 (Mihalcea et al., 2004). In Table 3 we show an example for a 'query' context from that set and the other two most similar contexts to it, based on *context2vec* and *AWE* (average of Skip-gram word embeddings) context representations. Melamud et al. (2015a) argues that since contexts induce meanings (or senses) for target words, a good context similarity measure should assign high similarity values to contexts that induce similar senses for the same target word. As can be seen in this example, *AWE*'s similarity measure seems to be influenced by the presence of the location names in the contexts, even though they have little effect on the perceived meaning of *add* in the sentences. Indeed, the sense of *add* in the closest contexts retrieved by *AWE* is different than that in the 'query' context. In this case, *context2vec*'s similarity measure was robust to this problem.

Finally, in Table 4, we show the closest target words to a few given target words, based on the target-to-target similarity metric. We compare *context2vec*'s target word embeddings to Skip-gram *word2vec* embeddings, trained with 2-word and 10-word windows. As can be seen, our model seems to better preserve the function of the given target words including part-of-speech and even tense, in comparison to the 2-word window model, and even more so compared to the 10-word window one. The intuition for this behavior is that Skip-gram literally skips words in the context

Sentential Context	Closest target words
This [ ] is due	item, fact-sheet, offer, pack, card
This [ ] is due not just to mere luck	offer, suggestion, announcement, item, prize
This [ ] is due not just to mere luck, but to outstanding work and dedication	award, prize, turnabout, offer, gift
[ ] is due not just to mere luck, but to outstanding work and dedication	it, success, this, victory, prize-money

Table 1: Closest target words to various sentential contexts, illustrating *context2vec*'s sensitivity to long range dependencies, and both sides of the target word.

α	John was [ ] last year
0.25	born, late, married, out, back
0.50	born, back, married, released, elected
0.75	born, interviewed, re-elected
1.00	starstruck, goal-less, unwed

Table 2: Closest target words to a given sentential context using different α values in *context2vec*.

around the target word and therefore may find, for instance, the contexts of *san* and *francisco* to be very similar. In contrast, our model considers only entire sentential contexts, taking context word order and position into consideration. Melamud et al. (2016) showed that target word embeddings, learned from context representations that are generated using n-gram language models, also exhibit function-preserving similarities, which is consistent with our observations.

2.4 Relation to Language Models

Our model is closely related to language models, as can be seen in section 2.2 and tables 1 and 2. In particular, it has a lot in common with LSTM-based language models, as both train LSTM neural networks with the objective to predict target words based on their (short and long range) context, and both use techniques, such as negative sampling, to address large vocabulary computational challenges during training (Jozefowicz et al., 2016). The main difference is that LSTM language models are mainly concerned with optimizing predictions of conditional probabilities for target words given their history, while our model is focused on deriving generally useful representations to whole history-and-future contexts of target words. We follow *word2vec*'s learning framework as it is known to produce high-quality representations for single words. It does so by having $\vec{t} \cdot \vec{c}$ approximate PMI(t, c) rather than $\log p(t|c)$.

3 Evaluation Settings

We intend *context2vec*'s generic context embedding function to be integrated into various more optimized task-specific systems. However, to demonstrate its qualities independently, we address three different types of tasks by the simple means of measuring cosine distances between its embedded representations. Yet, we compare our performance against the state-of-the-art results of highly competitive task-optimized systems on each task. In addition we use *AWE* as a baseline representing a commonly used generic context representation, which like ours, can represent variable-length contexts with a fixed-size vector. Our evaluation includes the following tasks: sentence completion, lexical substitution and supervised word sense disambiguation (WSD).

3.1 Learning corpus

With the exception of the sentence completion task (MSCC), which comes with its own learning corpus, we used the two billion word ukWaC (Ferraresi et al., 2008) as our learning corpus. To speed-up the training of *context2vec*, we discarded all sentences that are longer than 64 words, reducing the size of the corpus by $\sim$10%. However, we train the embeddings used in the *AWE* baseline on the full corpus to not penalize it on account of our model. We lower-cased all text and considered any token with fewer than 100 occurrences as an unknown word. This yielded a vocabulary of a little over 180K words for the full corpus, and 160K words for the trimmed version.

3.2 Compared Methods

context2vec We implemented our model using the Chainer toolkit (Tokui et al., 2015), and Adam (Kingma and Ba, 2014) for optimization. To speed-up the learning time we used mini-batch training, where only sentences of equal length are

Query	Furthermore our work in Uganda and Romania [adds] a wider perspective.
context2vec closest	... themes in art have a fascination , since they [add] a subject interest to a viewer's enjoyment of artistic qualities.
	Richard is joining us every month to pass on tips , ideas and news from the world of horticulture , and [add] a touch of humour too
AWE closest	... the foreign ministers said political and economic reforms in Poland and Hungary had made considerable progress but [added] : the process remains fragile ...
	... Germany had announced the solution as a humanitarian act by the government, [adding] that it hoped Bonn in future would run its embassies in normal manner...

Table 3: An example for a given 'query' context and the two closest contexts to it, as 'retrieved' by *context2vec* similarity and *AWE* similarity.

context2vec	*word2vec*-w2	*word2vec*-w10	*context2vec*	*word2vec*-w2	*word2vec*-w10
flying			*syntactically*		
gliding	flew	flew	semantically	grammatically	semantically
sailing	fly	fly	lexically	phonologically	grammatically
diving	aerobatics	aeroplane	grammatically	semantically	syntax
flown	low-flying	flown	phonologically	ungrammatical	syntactic
travelling	flown	bi-plane	topologically	lexically	lexically
san			*prize*		
agios	francisco	francisco	prizes	prizes	prizes
aghios	diego	diego	award	prize-winner	winner
los	fransisco	fransisco	trophy	prizewinner	winners
tanjung	los	bernardino	medal	prize	prizewinner
puerto	obispo	los	prizewinner	prizewinners	prize.

Table 4: Top-5 closest target words to a few given target words.

assigned to the same batch. We discuss the hyper-parameters tuning of our model in section 4.1.

AWE We learned word embeddings with the popular *word2vec* Skip-gram model using standard hyperparameters: 600 dimensions, 10 negative samples, window-size 10 and 3/5 iterations for the ukWaC/MSCC learning corpora, respectively. Then we used a simple average of these embeddings as our *AWE* context representation.[3] In addition, we experimented with the following variations: (1) ignoring stopwords (2) performing a weighted average of the words in the context using tf-idf weights (3) considering just the 5-word window around the target word instead of the whole sentence. Specifically, in the WSD experiment the context provided for the target words is a full paragraph. Though it could be extended, *context2vec* is currently not designed to take advantage of such large context and therefore ignores all context outside of the sentence of the target word. However, for *AWE* we also experimented with the option of generating the context representation based on the entire paragraph. In all cases, the size (dimensionality) of the *AWE* context representation was equal to that of *context2vec*, and the context-to-target and context-to-context similarities were computed using vector cosine between the respective embedding representations, as with *context2vec*.

3.3 Sentence Completion Challenge

The Microsoft Sentence Completion Challenge (MSCC) (Zweig and Burges, 2011) includes 1,040 items. Each item is a sentence with one word replaced by a gap, and the challenge is to identify the word, out of five choices, that is most meaningful and coherent as the gap-filler. While there is no official dev/test split for this dataset, we followed previous work (Mirowski and Vlachos, 2015) and used the first 520 sentences for parameter tuning and the rest as the test set.[4]

[3] We made some preliminary experiments using word embeddings learned with *word2vec*'s *CBOW* model, instead of Skip-gram, but this yielded worse results.

[4] Mikolov et al. (2013a) did not specify their dev/test split and all other works reported results only on the entire dataset.

The MSCC includes a learning corpus of 50 million words. To use this corpus for training our models, we first discarded all sentences longer than 128 words, which resulted in a negligible reduction of $\sim$ 1% in the size of the corpus. Then, we converted all text to lowercase and considered all words with frequency less than 3 as unknown, yielding a vocabulary of about 100K word types.

Finally, as the gap-filler, we simply choose the word whose target word embedding is the most similar to the embedding of the given context using the target-to-context similarity metric. We report the accuracy achieved in this task.

3.4 Lexical Substitution Task

The lexical substitution task requires finding a substitute word for a given target word in sentential context. The difference between this and the sentence completion task is that the substitute word needs not only to be coherent with the sentential context, but also preserve the meaning of the original word in that context. Most recent works evaluated their performance on a ranking variant of the lexical substitution task, which uses predefined candidate lists provided with the gold standard, and requires to rank them considering the sentential context. Performance in this task is reported with generalized average precision (GAP).[5] As in MSCC, in this evaluation we rank lexical substitutes according to the measured similarity between their target word embeddings and the embedding of the given sentential context.

We used two lexical substitution datasets in our experiments. The first is the dataset introduced in the lexical substitution task of SemEval 2007 (McCarthy and Navigli, 2007), denoted LST-07, split into 300 dev sentences and 1,710 test sentences. The second is a more recent 'all-words ' dataset (Kremer et al., 2014), denoted LST-14, with over 15K target word instances. It comes with a predefined 35%/65% split. We used the smaller set as the dev set for parameter tuning and the larger one as our test set.

3.5 Supervised WSD

In supervised WSD tasks, the goal is to determine the correct sense of words in context, based on a manually tagged training set. To classify a test word instance in context, we consider all of the tagged instances of the same word lemma in the training set, and find the instance whose context embedding is the most similar to the context embedding of the test instance using the context-to-context similarity metric. Then, we use the tagged senses[6] of that instance. We note that this is essentially the simplest form of a k-nearest-neighbor algorithm, with $k = 1$.

As our supervised WSD dataset we used the Senseval-3 lexical sample dataset (Mihalcea et al., 2004), denoted SE-3, which includes 7,860 train and 3,944 test instances. We used the training set for parameter tuning and report accuracy results on the test set.

context word units	300
LSTM hidden/output units	600
MLP input units	1200
MLP hidden units	1200
sentential context units	600
target word units	600
negative samples	10

Table 5: *context2vec* hyperparameters

4 Results

4.1 Development Experiments

The hyperparameters used in our reported experiments with *context2vec* are summarized in Table 5. In preliminary development experiments, we used only 200 units for representing sentential contexts, and then saw significant improvement in results, when moving to 600 units. Increasing the representation size to 1,000 units did not seem to further improve results.

With mini-batches of 1,000 sentences at a time, we started by training our models with a single iteration over the 2-billion-word ukWaC corpus. This took $\sim$30 hours, using a single Tesla K80 GPU. For the smaller 50-million-word MSCC learning corpus, a full iteration with a batch size of 100 took only about 3 hours. For this corpus, we started with 5 training iterations.

To explore the rare-word bias effect of the vocabulary smoothing factor α, we varied its value in our development experiments. The results appear in Table 6 on the left hand side. Since we preferred to keep our model as simple as possible, based on these results, we chose the single

[5] See Melamud et al. (2015a) for more of their setting details, which we followed here.

[6] There's one or more senses assigned to a each instance.

	context2vec					*AWE*			
	neg sampling parameter α				iters+	best	best	worst	worst
	0.25	0.50	0.75	1.00	0.75	config	result	config	result
MSCC-dev	52.5	56.5	**60.0**	52.7	**66.2**	sent+stop	**51.0**	W5	36.5
LST-07-dev	50.1	52.9	53.6	**54.3**	**55.4**	W5+stop	**45.8**	sent	40.0
LST-14-dev	48.2	**48.9**	48.0	46.1	48.3	sent+stop	**40.4**	sent	39.2
SE-3-dev	72.1	**72.4**	71.6	**72.5**	**72.6**	W5+tf-idf	**62.4**	sent	57.3

Table 6: Development set results. *iters+* denotes the best model found when running more training iterations with $\alpha = 0.75$. *AWE* config: *W5*/sent denotes using a 5-word-window/full-sentence, and *stop*/tf-idf denotes ignoring stop words or using tf-idf weights, respectively.

value $\alpha = 0.75$ for all of our test sets experiments. With this choice, we also tried training our models with more iterations and found that with 3 iterations over the ukWaC corpus and 10 iterations over the MSCC corpus we can obtain some further improvement in results, see *iters+* in Table 6.

The results of our experiments with all of the *AWE* variants, described in section 3.2, appear on the right hand side of Table 6. For brevity, we report only the best and worst configuration for each benchmark. As can be seen, in two out of four benchmarks, a window of 5 words yields better performance than a full sentential context, suggesting that the *AWE* representation is not very successful in leveraging effectively long range information. Removing stop words or using tf-idf weights improves performance significantly. However, the results are still much lower than the ones achieved with *context2vec*. To raise the bar, in each test-set experiment we used the best *AWE* configuration found for the corresponding development-set experiment.

4.2 Test Sets Results

The test set results are summarized in Table 7. First, we see that *context2vec* substantially outperforms *AWE* across all benchmarks. This suggests that our context representations are much better optimized for capturing sentential context information than *AWE*, at least for these tasks. Further, we see that with *context2vec* we either surpass or almost reach the state-of-the-art on all benchmarks. This is quite impressive, considering that all we did was measure cosine distances between *context2vec*'s representations to compete with more complex and task-optimized systems.

More specifically, in the sentence completion task (MSCC) the prior state-of-the-art result is due to Mikolov et al. (2013a) and was achieved by a weighted combination of scores from two different models: a recurrent neural network language model, and a Skip-gram model. The second-best result is due to Liu et al. (2015) and is based on word embeddings that are learned based on both corpora and structured knowledge resources, such as WordNet. *context2vec* outperforms both of them. In the lexical substitution tasks, the best prior results are due to Melamud et al. (2015a).[7] They employ an exemplar-based approach that requires keeping thousands of exemplar contexts for every target word type. The second-best is due to Melamud et al. (2015b). They propose a simple approach, but it requires dependency-parsed text as input. *context2vec* achieves comparable results with these works, using the same learning corpus. In the Senseval-3 supervised WSD task, the best result is due to Ando (2006) and the second-best to

	c2v iters+	*c2v*	*AWE*	S-1	S-2
MCSS					
test	**64.0**	62.7	48.4	-	-
all	**65.1**	61.3	49.7	58.9	56.2
LST-07					
test	**56.1**	54.8	41.9	55.2	-
all	**56.0**	54.6	42.5	55.1	53.6
LST-14					
test	47.7	47.3	38.1	**50.0**	-
all	47.9	47.5	38.9	**50.2**	48.3
SE-3					
test	72.8	71.2	61.4	**74.1**	73.6

Table 7: Results on test sets. *c2v* is *context2vec* and *iters+* denotes the model that was trained with more iterations. S-1/S-2 stand for the best/second-best prior result reported for the benchmark.

[7] Szarvas et al. (2013) achieved almost the same result, but with a supervised model, not directly compared to ours.

Rothe and Schütze (2015). *context2vec* is almost on par with these results, which were achieved with dedicated feature engineering and supervised machine learning models.

5 Related Work

Substitute vectors (Yuret, 2012) represent contexts as a probabilistic distribution over the potential gap-filler words for the target slot, pruned to its top-k most probable words. While using this representation showed interesting potential (Yatbaz et al., 2012; Melamud et al., 2015a), it can currently be generated efficiently only with n-gram language models and hence is limited to fixed-size context windows. It is also high dimensional and sparse, in contrast to our proposed representations.

Syntactic dependency context embeddings have been proposed recently (Levy and Goldberg, 2014a; Bansal et al., 2014). They depend on the availability of a high-quality dependency parser, and can be viewed as a 'bag-of-dependencies' rather than a single representation for the entire sentential context. However, we believe that incorporating such dependency-based information in our model is an interesting future direction.

A couple of recent works extended *word2vec*'s *CBOW* by replacing its internal context representation. Ling et al. (2015b) proposed a *continuous window*, which is a simple linear projection of the context window embeddings into a low dimensional vector. Ling et al. (2015a) proposed '*CBOW with attention*', which is used for finding the relevant features in a context window. In contrast to our model, both approaches confine the context to a fixed-size window. Furthermore, they limit their scope to using these context representations only internally to improve the learning of target words embeddings, rather than evaluate the benefit of using them directly in NLP tasks, as we do.

Kawakami and Dyer (2016) represent words in context using bidirectional LSTMs and multilingual supervision. In contrast, our model is focused on representing the context alone. Yet, as shown in our lexical substitution and word sense disambiguation evaluations, it can easily be used for modeling the meaning of words in context as well.

Finally, there is considerable work on using recurrent neural networks to represent word sequences, such as phrases or sentences (Socher et al., 2011; Kiros et al., 2015). We note that the techniques used for learning sentence representations have much in common with those we use for sentential context representations. Yet, sentential context representations aim to reflect the information in the sentence only inasmuch as it is relevant to the target slot. Specifically, different target positions in the same sentence can yield completely different context representations. In contrast, sentence representations aim to reflect the entire contents of the sentence.

6 Conclusions and Future Potential

We presented *context2vec*, a neural model that learns a generic embedding function for variable-length contexts of target words. We demonstrated that it can be trained in a reasonable time over billions of words and generate high quality context representations, which substantially outperform the traditional average-of-word-embeddings approach on three different tasks. As such, we hypothesize that it could contribute to various NLP systems that model context. Specifically, semi-supervised systems may benefit from using our model, as it may carry more useful information learned from large corpora, than individual pre-trained word embeddings do.

Acknowledgments

We thank our anonymous reviewers for their useful comments. This work was partially supported by the Israel Science Foundation grant 880/12 and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Rie Kubota Ando. 2006. Applying alternating structure optimization to word sense disambiguation. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 77–84. Association for Computational Linguistics.

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

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa.

2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4)*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Mikael Kågebäck, Fredrik Johansson, Richard Johansson, and Devdatt Dubhashi. 2015. Neural context embeddings for automatic discovery of word senses. In *Proceedings of NAACL*.

Kazuya Kawakami and Chris Dyer. 2016. Learning to represent words in context with multilingual supervision. In *Workshop in ICLR*.

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

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of NIPS*.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us-analysis of an all-words lexical substitution corpus. In *Proceedings of EACL*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of ACL*.

Omer Levy and Yoav Goldberg. 2014b. Neural word embeddings as implicit matrix factorization. In *Proceedings of NIPS*.

Wang Ling, Lin Chu-Cheng, Yulia Tsvetkov, and Silvio Amir. 2015a. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of EMNLP*.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015b. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of NAACL*.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. *Proceedings of ACL*.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of SemEval*.

Oren Melamud, Ido Dagan, and Jacob Goldberger. 2015a. Modeling word meaning in context with substitute vectors. In *Proceedings of ACL*.

Oren Melamud, Omer Levy, and Ido Dagan. 2015b. A simple word embedding model for lexical substitution. In *Proceedings of Workshop on Vector Space Modeling for NLP (VSM)*.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *Proceedings of NAACL*.

Rada Mihalcea, Timothy Anatolievich Chklovski, and Adam Kilgarriff. 2004. The senseval-3 english lexical sample task. ACL.

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 *Proceedings of NIPS*.

Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. *arXiv preprint arXiv:1507.01193*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings EMNLP*.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *Proceedings of ACL*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

György Szarvas, Róbert Busa-Fekete, and Eyke Hüllermeier. 2013. Learning to rank lexical substitutions. In *Proceedings of EMNLP*.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in NIPS*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semisupervised learning. In *Proceedings of ACL*.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning syntactic categories using paradigmatic representations of word context. In *Proceedings EMNLP*.

Deniz Yuret. 2012. FASTSUBS: An efficient and exact procedure for finding the most likely lexical substitutes based on an n-gram language model. *Signal Processing Letters, IEEE*, 19(11):725–728.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL*.

Geoffrey Zweig and Christopher JC Burges. 2011. The microsoft research sentence completion challenge. Technical report, Technical Report MSR-TR-2011-129, Microsoft.