Word Ordering as Unsupervised Learning Towards Syntactically Plausible Word Representations

Noriki Nishida and **Hideki Nakayama**
Graduate School of Information Science and Technology
The University of Tokyo
{nishida,nakayama}@nlab.ci.i.u-tokyo.ac.jp

Abstract

The research question we explore in this study is how to obtain syntactically plausible word representations without using human annotations. Our underlying hypothesis is that word ordering tests, or linearizations, is suitable for learning syntactic knowledge about words. To verify this hypothesis, we develop a differentiable model called Word Ordering Network (WON) that explicitly learns to recover correct word order while implicitly acquiring word embeddings representing syntactic knowledge. We evaluate the word embeddings produced by the proposed method on downstream syntax-related tasks such as part-of-speech tagging and dependency parsing. The experimental results demonstrate that the WON consistently outperforms both order-insensitive and order-sensitive baselines on these tasks.

1 Introduction

Distributed word representations have been successfully utilized to transfer lexical knowledge to downstream tasks in a semi-supervised manner, and well known to benefit various applications (Turian et al., 2010; Collobert et al., 2011; Socher et al., 2011). As different applications generally require different features, it is crucial to choose representations suitable for target downstream tasks.

The research question we want to explore in this study is how to obtain *syntactically plausible word representations* without human annotations, with a focus on syntax-related tasks (parsing, etc.). Whereas a variety of approaches related to *semantic* word embeddings have been proposed (Mikolov et al., 2013a,b; Pennington et al., 2014), it still remains unclear how we should obtain *syntactic* word embeddings from unannotated corpora.

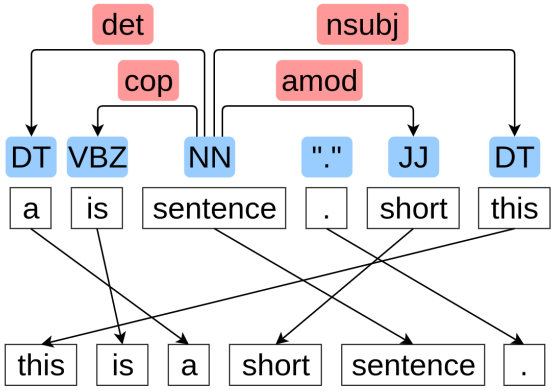

Figure 1: Illustration of the word ordering task. The goal of the word ordering task is to recover an original order given a set of shuffled tokens. The figure shows an example where original sentence is *"this is a short sentence."* To correctly reorder the tokens, syntactic knowledge about words (e.g. grammatical classes of words and their possible relations) is indispensable. In this study, we explore how well the word ordering task can be an objective to obtain syntactic word representations.

Word ordering tests, or linearizations, are commonly used to evaluate students' language proficiency. Suppose that we are given a set of randomly shuffled tokens {*"a", "is," "sentence," "short," "this," "."*}. In this case we can easily recover the original order: *"this is a short sentence."* We consider this doable thanks to our knowledge about grammatical classes (e.g., part-of-speech (POS) tags) of words and their possible relations. We depict the above explanation in Figure 1. Of course, it might not be necessary for machines to mimic exactly the same reasoning pro-

Proceedings of the The 8th International Joint Conference on Natural Language Processing, pages 70–79,
Taipei, Taiwan, November 27 – December 1, 2017

cess in humans. However, syntactic knowledge about words is crucial for both humans and machines to solve the word ordering task.

Inspired by this observation, in this study, we develop an end-to-end model called the Word Ordering Network (WON) that explicitly learns to recover correct word orders while implicitly acquiring word embeddings representing syntactic information. Our underlying hypothesis is that the word ordering task can be an objective for learning syntactic knowledge about words. The WON receives a set of shuffled tokens and first transforms them independently to low-dimensional continuous vectors, which are then aggregated to produce a single summarization vector. We formalize the word ordering task as a sequential prediction problem of a permutation matrix. We use a recurrent neural network (RNN) (Elman, 1990) with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) and a soft attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) that constructs rows of permutation matrices sequentially conditioned on summarization vectors.

We evaluate the proposed word embeddings on downstream syntax-related tasks such as POS tagging and dependency parsing. The experimental results demonstrate that the WON outperforms both order-insensitive and order-sensitive baselines, and successfully yields the highest performance. In addition, we also evaluate the WON on traditional word-level benchmarks, such as word analogy and word similarity tasks. Combined with semantics-oriented embeddings by a simple fine-tuning technique, the WON gives competitive or better performances than the other baselines. Interestingly, we find that the WON has a potential to refine and improve semantic features. Moreover, we qualitatively analyze the feature space produced by the WON and find that the WON tends to capture not only syntactic but also semantic regularities between words. The source code of this work is available online. [1]

2 The Proposed Method

In this section, we formulate the WON which implicitly acquires syntactic word embeddings through learning to solve word ordering problems.

[1] https://github.com/norikinishida/won

2.1 Embedding Layer

Given a set of shuffled tokens $\mathcal{X} = \{w_1, \ldots, w_N\}$, the WON first transforms every single symbol w_c into a low-dimensional continuous vector, i.e.,

$$\boldsymbol{e}_c = F(w_c) \in \mathbb{R}^D, \tag{1}$$

where F is a learnable function. Please note that the number of tokens N in the input $\mathcal{X}$ can vary in the word ordering task.

2.2 Aggregation

To perform reordering on a set of shuffled embeddings $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N\}$, we aggregate the embeddings and compute a single summarization vector. The aggregation function is a sum of word embeddings followed by a non-linear transformation:

$$\tilde{\boldsymbol{e}} = \tanh(\boldsymbol{W}_{\mathrm{a}} \sum_{c=1}^{N} \boldsymbol{e}_c + \boldsymbol{b}_{\mathrm{a}}) \in \mathbb{R}^D, \tag{2}$$

where $\boldsymbol{W}_a \in \mathbb{R}^{D \times D}$ and $\boldsymbol{b}_a \in \mathbb{R}^D$ are a projection matrix and bias vector, respectively.

2.3 Prediction of a Permutation Matrix

We formalize a reordering problem as a prediction task of a permutation matrix.

A permutation matrix is a square binary matrix and every row and column contains exactly one entry of 1 and 0s elsewhere. The left-multiplication of a matrix $\boldsymbol{E} \in \mathbb{R}^{N \times D}$ by a permutation matrix $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ rearranges the rows of the matrix $\boldsymbol{E}$, e.g.

$$\begin{pmatrix} \boldsymbol{e}_1^\top \\ \boldsymbol{e}_2^\top \\ \boldsymbol{e}_3^\top \\ \boldsymbol{e}_4^\top \end{pmatrix} = \boldsymbol{P}\boldsymbol{E} \tag{3}$$

$$= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{e}_3^\top \\ \boldsymbol{e}_1^\top \\ \boldsymbol{e}_4^\top \\ \boldsymbol{e}_2^\top \end{pmatrix}. \tag{4}$$

Equation 4 gives an example where $\boldsymbol{E} = (\boldsymbol{e}_3, \boldsymbol{e}_1, \boldsymbol{e}_4, \boldsymbol{e}_2)^\top$, and the original sentence (correct order) is w_1, w_2, w_3, w_4.

In the word ordering task, one of the issues in predicting permutation matrices is that the number of tokens N changes according to the variable lengths of input sentences. Therefore, it is impossible to define and train learning models that have fixed-dimensional outputs such as multi-layer perceptrons.

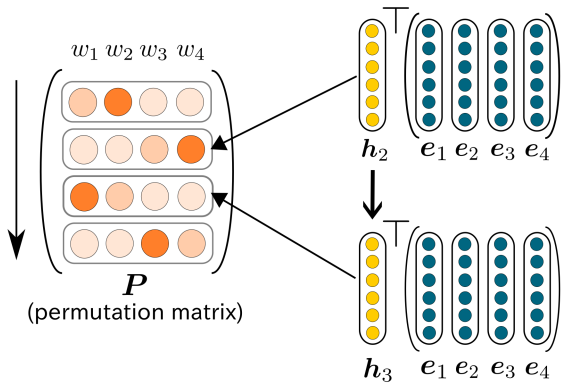

Figure 2: Visualization of our approach to sequentially predict a permutation matrix $\boldsymbol{P} \in \mathbb{R}^{N \times N}$. In this case, we are given $N = 4$ shuffled tokens (w_1, w_2, w_3, w_4). We first independently embeds each symbol to dense vectors $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4)$. Then, by using an RNN and a soft attention mechanism, we sequentially constructs the rows of the permutation matrix $\boldsymbol{P} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4)^\top$ for N steps through a scoring function. The vector $\boldsymbol{h}_r \in \mathbb{R}^D$ denotes the r-th hidden state of the RNN. One can interpret $\boldsymbol{p}_r$ as a selective probability distribution over the input tokens. For simplicity, in this figure, we ignore the projection matrix in the scoring function (Eq. 8).

Recently, Vinyals et al. (2015) proposed the *Pointer Networks* (PtrNets) that were successfully applied to geometric sorting problems. Inspired by the PtrNet, we develop an LSTM (Hochreiter and Schmidhuber, 1997) with a soft attention mechanism (Bahdanau et al., 2014; Luong et al., 2015). The LSTM constructs rows of a permutation matrix $\boldsymbol{P} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)^\top$ conditioned on a set of word embeddings $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$ calculated by Equation 1. If $\sum_{c=1}^{N} p_{r,c} = 1$ holds, one can interpret $p_{r,c}$ as the probability of the token w_c to be placed at r-th position. In Figure 2, we show a visualization of our approach to predict a permutation matrix with the LSTM.

The LSTM's r-th hidden state $\boldsymbol{h}_r \in \mathbb{R}^D$ and memory cells $\boldsymbol{c}_r \in \mathbb{R}^D$ are computed as follows:

$$\boldsymbol{h}_r, \boldsymbol{c}_r = \begin{cases} \tilde{\boldsymbol{e}},\ \boldsymbol{0} & (r = 0) \\ F_{\mathrm{LSTM}}(\boldsymbol{e}_{i_{r-1}}, \boldsymbol{h}_{r-1}, \boldsymbol{c}_{r-1}) & (1 \leq r \leq N) \end{cases}, \tag{5}$$

where the function F_{LSTM} is a state-update function and $i_{r-1} \in \{1, \ldots, N\}$ denotes the index of the token $w_{i_{r-1}}$ that is placed at the previous position, i.e.,

$$i_{r-1} = \operatorname*{argmax}_{c \in \{1,\ldots,N\}} p_{r-1,c}. \tag{6}$$

Subsequently, we predict a selective distribution over the input tokens:

$$p_{r,c} = \frac{\exp(\mathrm{score}(\boldsymbol{h}_r, \boldsymbol{e}_c))}{\sum_{k=1}^{N} \exp(\mathrm{score}(\boldsymbol{h}_r, \boldsymbol{e}_k))}, \tag{7}$$

where the scoring function score computes the confidence of placing the token w_c at r-th position. We define the scoring function as a bilinear model as follows

$$\mathrm{score}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{W}_s \boldsymbol{v} \in \mathbb{R}. \tag{8}$$

where $\boldsymbol{W}_s \in \mathbb{R}^{D \times D}$ denotes a learnable matrix.

2.4 Objective Function

As the WON is designed to be fully differentiable, it can be trained with any gradient descent algorithms, such as RMSProp (Tieleman and Hinton, 2012). Given a set of shuffled tokens $\mathcal{X} = \{w_1, \ldots, w_N\}$, we define a loss function as the following negative log likelihood:

$$\mathcal{L}(\mathcal{X}) = \sum_{r=1}^{N} -\log p_{r,t_r} \tag{9}$$

where $t_r \in \{1, \ldots, N\}$ denotes the index of the ground-truth token that appears at r-th position in the original sentence. In other words, an ordered sequence $w_{t_1}, w_{t_2}, \ldots, w_{t_N}$ forms the original sentence.

3 Related Work

Among the most popular methods for learning word embeddings are the *skip-gram* (SG) model and the *continuous bag-of-words* (CBOW) of Mikolov et al. (2013a,b), or the GloVe introduced by Pennington et al. (2014). These are formalized as simple log-bilinear models based on the inner product between two word vectors. The core idea is based on the *distributional hypothesis* (Harris, 1954; Firth, 1957), stating that words appearing in similar contexts tend to have similar meanings. For example, SG and CBOW are trained by making predictions of bag-of-words contexts appearing in a fixed-size window around target words, and vice versa. Although word embeddings produced by these models have been

shown to give improvements in a variety of downstream tasks, it still remains difficult for these models to learn syntactic word representations owing to their insensitivity to word order. As a consequence, word embeddings produced by these order-insensitive models are thus suboptimal for syntax-related tasks such as parsing (Andreas and Klein, 2014). In contrast, our method mainly focuses on word order information and utilize it in the learning process.

Ling et al. (2015b) introduced the *structured skip-gram* (SSG) model and the *continuous window* (CWindow) that extend SG and CBOW respectively. Let c be the window size. These models learn $2c$ context-embedding matrices to be aware of relative positions of context words in a window. The recent work of Trask et al. (2015) is also based on the same idea as SSG and CWindow. Ling et al. (2015a) proposed an approach to integrating an order-sensitive attention mechanism into CBOW, which allows for consideration of the contexts of words, and where the context words appear in a window. Bengio et al. (2003) presented a neural network language model (NNLM) where word embeddings are simultaneously learned along with a language model. One of the major shortcomings of these window-based approaches is that it is almost impossible to learn longer dependencies between words than the prefixed window size c. In contrast, the recurrent architecture allows the WON to take into account dependencies over an entire sentence.

Mikolov et al. (2010) applied an RNN for language modeling (RNNLM), and demonstrated that the word embeddings learned by the RNNLM capture both syntactic and semantic regularities. The main shortcoming of the RNNLM is that it is very slow to train unfortunately. This is a consequence of having to predict the probability distribution over an entire vocabulary V, which is generally very large in the real world. In contrast, the WON predicts the probability distribution over entire sentences, whose length N is usually less than $50 \ll |V|$. In our preliminary experiments, we found that the computation time for one iteration (= `forward` + `backward` + `parameter update`) of the WON is about 4 times faster than that of the RNNLM (LSTMLM).

Levy and Goldberg (2014) introduced dependency-based word embeddings. The method applies the *skip-gram with negative sampling* (SGNS) model (Mikolov et al., 2013b) to *syntactic contexts* derived from dependency parse-trees. Their method heavily relies on pre-trained dependency parsers to produce words' relations for each sentence in training corpora, thus encountering error propagation problems. In contrast, our method only requires raw corpora, and our aim is to produce word embeddings that improve syntax-related tasks, such as parsing, without using any human annotations.

The WON can be interpreted as a simplification of the recently proposed pointer network (PtrNet) (Vinyals et al., 2015). The main difference between the WON and the PtrNet is the *encoder* part. The PtrNet uses an RNN to encode an unordered set $\mathcal{X} = \{w_1, \ldots, w_N\}$ sequentially, i.e.,

$$\boldsymbol{e}_i = \mathrm{RNN}_{\mathrm{enc}}(w_i, \boldsymbol{e}_{i-1}). \tag{10}$$

In contrast, the WON treats each symbol independently (Eq. 1) and aggregates them with a simpler function (Eq. 2). In the word ordering task, the order of $\mathcal{X} = (w_1, \ldots, w_N)$ is meaningless because $\mathcal{X}$ is an out-of-order set. Nonetheless, according to Equation 10, the vector $\boldsymbol{e}_i$ depends on the input order of $w_1, \ldots, w_{i-1}$. Vinyals et al. (2015) evaluated the PtrNet on geometric sorting tasks (e.g., Travelling Salesman Problem) where each input w_i forms a continuous vector that represents the cartesian coordinate of the point (e.g., a city). However, in the word ordering task, Equation 10 suffers from the data sparseness problem, as each input w_i forms a high-dimensional discrete symbol.

4 Experimental Setting

4.1 Dataset and Preprocessing

We used the English Wikipedia corpus as the training corpus. We lowercased and tokenized all tokens, and then replaced all digits with "7" (e.g., "ABC2017"→"ABC7777"). We built a vocabulary of the most frequent 300K words and replaced out-of-vocabulary tokens with a special "⟨UNK⟩" symbol. Subsequently, we appended special "⟨EOS⟩" symbols to the end of each sentence. The resulting corpus contains about 97 million sentences with about 2 billion tokens. We randomly extracted $5K$ sentences as the validation set.

4.2 Hyper Parameters

We set the dimensionality of word embeddings to 300. The dimensionality of the hidden states of the LSTM was 512. The $L2$ regularization term (called weight decay) was set to 4×10^{-6}. For the stochastic gradient descent algorithm, we used the SMORMS3 (Func, 2015), and the mini-batch size was set to 180.

4.3 Baselines

For a fair comparison, we trained the following order-insensitive/sensitive baselines on exactly the same pre-processed corpus described in Section 4.1.

- **SGNS** (Mikolov et al., 2013b): We used the `word2vec` implementation in Gensim[2] to train the Skip-Gram with Negative Sampling (SGNS). We set the window size to 5, and the number of negative samples to 5.
- **GloVe** (Pennington et al., 2014): GloVe's embeddings are trained by using the original implementation[3] provided by the authors. We set the window size to 15. In our preliminary experiments, we found that GloVe with a window size of 15 yields higher performances than that with a window size of 5.
- **SSG**, **CWindow** (Ling et al., 2015b): We built word embeddings by using the structured skip-gram (SSG) and the continuous window (CWindow). We used the original implementation[4] developed by the authors. The window size was 5, and the number of negative samples was 5.
- **LSTMLM**: We also compared the proposed method with the RNNLM (Mikolov et al., 2010) with LSTM units (LSTMLM). The hyper parameters were the same with that of the WON except for the mini-batch size. We used a mini-batch size of 100 for the LSTMLM.

5 Evaluation on Part-of-Speech Tagging

In this experiment, we evaluated the learned word embeddings by using them as pre-trained features in supervised POS tagging.

[2]https://radimrehurek.com/gensim/
[3]http://nlp. stanford.edu/projects/glove/
[4]https://github.com/wlin12/wang2vec

	Test Acc. (%)
SGNS (Mikolov et al., 2013b)	96.76
GloVe (Pennington et al., 2014)	96.31
SSG (Ling et al., 2015b)	96.94
CWindow (Ling et al., 2015b)	96.78
LSTMLM	96.92
WON	**97.04**

Table 1: Comparison results on part-of-speech tagging with different word embeddings. The dataset is the Wall Street Journal (WSJ) portion of the Penn Treebank (PTB) corpus. The evaluation metric is accuracy (%).

5.1 Supervised POS Tagger

In POS tagging, every token in a sentence is classified into its POS tag (`NN` for nouns, `VBD` for past tense verbs, `JJ` for adjectives, etc.). We first used the learned word embeddings to project three successive tokens (w_{i-1}, w_i, w_{i+1}) in an input sentence to feature vectors $(\boldsymbol{e}_{i-1}, \boldsymbol{e}_i, \boldsymbol{e}_{i+1})$ that are then concatenated and fed to a two-layer perceptron followed by a softmax function:

$$P(c|w_{i-1}, w_i, w_{i+1}) = \mathrm{MLP}([\boldsymbol{e}_{i-1}; \boldsymbol{e}_i; \boldsymbol{e}_{i+1}]), \quad (11)$$

where $[\cdot\ ;\ \cdot\ ;\ \cdot]$ denotes vector concatenation. The classifier MLP predicts the probability distribution over POS tags of the center token w_i. We put special padding symbols at the beginning and end of each sentence. The dimensionality of the hidden layer of the MLP was 300. The MLP classifier was trained via the SMORMS3 optimizer (Func, 2015) without updating the word embedding layer.

We used the Wall Street Journal (WSJ) portion of the Penn Treebank (PTB) corpus[5] (Marcus et al., 1993). We followed the standard section partition, which is to use sections 0-18 for training, sections 19-21 for validation, and sections 22-24 for testing. The dataset contains 45 tags. The evaluation metric was the word-level accuracy.

5.2 Results & Discussion

Table 1 presents the comparison of the WON to the other baselines on the test split. The results demonstrate that the WON gives the highest performance, which supports our hypothesis that the word ordering task is effective for acquiring syntactic knowledge about words. We also

[5]We used the LDC99T42 Treebank release 3 version.

	Dev		Test	
	UAS	LAS	UAS	LAS
SGNS	91.56	90.09	91.11	89.89
GloVe	88.87	87.09	88.28	86.61
SSG	91.11	89.60	90.93	89.43
CWindow	91.23	89.69	91.16	89.67
LSTMLM	91.83	90.34	91.49	90.08
WON	**91.92**	**90.49**	**91.82**	**90.38**

Table 2: Results on dependency parsing with different word embeddings. The dataset was the WSJ portion of the PTB corpus. The evaluation metrics were Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS).

observe that the order-sensitive methods (WON, LSTMLM, and SSG) tend to outperform the order-insensitive methods (SGNS and GloVe), which indicates that, as we expect, word order information is crucial for learning syntactic word embeddings.

6 Evaluation on Dependency Parsing

In this experiment, as in Section 5, we evaluated the learned word embeddings on supervised dependency parsing.

6.1 Supervised Dependency Parser

Dependency parsing aims to identify syntactic relations between token pairs in a sentence. We used Stanford's neural network dependency parser (Chen and Manning, 2014)[6], whose word embeddings were initialized with the learned word embeddings. We followed all the default settings except for the word embedding size (`embeddingSize` = 300) and the number of training iterations (`maxIter` = 6000).

We used the WSJ portion of the PTB corpus and followed the standard splits of sections 2-21 for training, 22 for validation, and 23 for testing. We converted the treebank corpus to Stanford style dependencies using the Stanford converter. The parsing performances were evaluated in terms of Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS).

6.2 Results & Discussion

Table 2 shows the results of the different word embeddings on dependency parsing. First we observe that the WON consistently outperforms the baselines on both UAS and LAS. Next, by comparing the unlimited-context models (WON and LSTMLM) with the limited-context models (SGNS, GloVe, SSG, CWindow), we can notice that the former give higher parsing scores than the latter. These results are reasonable because the former can learn arbitrary-length syntactic dependencies between words without constraints from the fixed-size window size based on which the limited-window models are trained.

[6]http://nlp.stanford.edu/software/nndep.shtml

7 Fusion with Semantic Features

In various NLP tasks, both syntactic and semantic features can benefit performances. To enrich our syntax-oriented word embeddings with semantic information, in this section, we adopt a simple fine-tuning technique and verify its effectiveness. More precisely, we first initialize the word embeddings $\boldsymbol{W}$ with pre-trained parameters $\boldsymbol{W}_{\text{sem}}$ produced by a semantics-oriented model such as the SGNS. Subsequently we add the following penalty term to the loss function in Equation 9:

$$\lambda \|\boldsymbol{W} - \boldsymbol{W}_{\text{sem}}\|_{\mathcal{F}}^2, \tag{12}$$

where $\lambda \in \mathbb{R}$ is a hyper parameter to control the intensity of the penalty term in the learning process, and $\|\cdot\|_{\mathcal{F}}^2$ is the Frobenius norm. This term attempts to keep the word embeddings $\boldsymbol{W}$ close to the semantic representations $\boldsymbol{W}_{\text{sem}}$ while minimizing the syntax-oriented objective on the word ordering task. In our experiments, we used the SGNS's embeddings as $\boldsymbol{W}_{\text{sem}}$ and set λ to 1. The SGNS was trained as explained in Section 4.3.

In this section, we quantitatively evaluated the WON with the above fine-tuning technique on two major benchmarks: (1) word analogy task, and (2) word similarity task.

7.1 Word Analogy

The word analogy task has been used in previous work to evaluate the ability of word embeddings to represent semantic and syntactic regularities. In this experiment, we used the word analogy dataset produced by Mikolov et al. (2013a). The dataset consists of questions like "A is to B what C is to ?," denoted as "A : B :: C : ?." The dataset contains about 20K such questions, divided into a *syntactic subset* and a *semantic subset*. The syntactic subset contains nine question types, such as `adjective-to-adverb` and `opposite`, while the semantic subset contains five question

Question Types	SGNS	GloVe	SSG	CWindow	LSTMLM	WON
adjective-to-adverb	24.1	23.3	**29.9**	12.1	4.3	**29.9**
opposite	36.2	29.9	37.0	11.7	15.0	**37.8**
comparative	85.7	79.5	88.5	73.5	55.3	**88.7**
superlative	59.3	49.1	**68.7**	43.8	22.4	62.8
present-participle	64.9	61.0	**73.6**	57.4	27.1	71.8
nationality-adjective	89.4	**92.2**	89.7	87.3	30.5	90.8
past-tense	58.0	52.2	59.0	54.0	33.1	**61.4**
plural	75.2	**83.0**	75.2	70.4	26.4	75.4
plural-verbs	78.9	56.0	**84.6**	64.6	61.0	82.9
capital-common	94.5	95.3	92.5	93.1	53.8	**95.5**
captal-world	87.8	**94.5**	84.0	66.6	22.1	82.6
currency	12.8	8.7	**14.0**	3.7	1.9	10.7
city	66.0	60.7	56.9	61.9	13.6	**67.4**
family	**84.2**	77.9	81.8	59.1	62.9	**84.2**
Total	69.9	68.3	69.7	58.2	27.0	**70.6**

Table 3: Results on the word analogy task (Mikolov et al., 2013a) with different word embeddings. The first upper block presents the results on nine syntactic question types. In the lower block we show the results on five semantic question types. The last row presents the total score. The evaluation metric is accuracy (%).

types such as `city-in-state` and `family`. Suppose that a vector $\boldsymbol{e}_w$ is a representation of a word w, and is normalized to unit norm. Following a previous work (Mikolov et al., 2013a), we answer an analogy question "A : B :: C : ?" by finding a word w^* that has the closest representation to $(\boldsymbol{e}_B - \boldsymbol{e}_A + \boldsymbol{e}_C)$ in terms of cosine similarity, i.e.,

$$w^* = \underset{w \in V \setminus \{A,B,C\}}{\operatorname{argmax}} \frac{(\boldsymbol{e}_B - \boldsymbol{e}_A + \boldsymbol{e}_C)^\top \boldsymbol{e}_w}{\|\boldsymbol{e}_B - \boldsymbol{e}_A + \boldsymbol{e}_C\|}, \quad (13)$$

where V denotes the vocabulary. The evaluation was performed using accuracy, which denotes the percentage of words predicted correctly.

In Table 3, we report the results of the different word embeddings on this task. As can be seen in the Table 3, the WON outperforms the baselines on four out of nine syntactic question types, and tends to yield higher accuracies by a large margin than the baselines except for the SSG. Our method and the SSG totally give the best performances on seven of nine syntactic question types. This tendency, as in Section 5.2, indicates that word order information is crucial to learn syntactic word embeddings. In regard to semantics, the WON achieves the best scores on three out of five semantic question types. Interestingly, on two semantic question types (`capital-common` and `city`), the WON outperforms the SGNS that was used to initialize our word embeddings. This result implies that the word ordering task has the potential to improve not only syntactic but also semantic features. Our method achieves the highest accuracy on the overall score.

	WS-353	MC	RG
SGNS	71.26	81.96	78.86
GloVe	62.54	71.57	75.54
SSG	**73.08**	81.78	**80.37**
CWindow	70.31	80.92	77.80
LSTMLM	53.34	66.76	63.23
WON	70.97	**82.43**	77.64

Table 4: Results on the word similarity task with different word embeddings. Spearman's rank correlation coefficents (%) are computed on three datasets: WS-353, MC, and RG.

7.2 Word Similarity

The word similarity benchmark is commonly used to evaluate word embeddings in terms of distributional semantic similarity. The word similarity datasets consist of triplets like (w_1, w_2, s), where $s \in \mathbb{R}$ is a human-annotated similarity score between two words (w_1, w_2). In this task, we compute cosine similarity between two word embeddings. The evaluation is performed with the Spearman's rank correlation coefficient between

Query	The 3 most similar words		
he	she	they	we
him	them	us	me
his	their	our	your
boy	kid	creature	girl
boys	ladies	guys	folks
dragon	werewolf	dwarf	vamp
dragons	robots	giants	spiders
city	village	library	palace
cities	countries	kingdoms	neighborhoods
drive	ride	walk	hike
drives	draws	pisses	causes
drove	rode	marched	strode
driving	traveling	walking	riding
driven	flown	propelled	shaken
happy	pleased	unhappy	thrilled
happier	crazier	prettier	tougher
happiest	hottest	toughest	coolest
good	nice	bad	decent
better	easier	worse	safer
best	worst	hardest	biggest
in	on	into	under

Table 5: Query words and their most similar words. Cosine similarities are computed between their embeddings produced by the WON.

the human-annotated similarities and the computed similarities.

Table 4 presents the results on three datasets: WordSim-353 (Finkelstein et al., 2001), MC (Miller and Charles, 1991), and RG (Rubenstein and Goodenough, 1965). we observe that the WON gives a slightly higher performance than the baselines on the MC dataset. On the other datasets, the SSG yields the best performances. These results are interesting because the two models rely on word order information while the word similarity task originally focuses on topical semantic similarities between words.

Further investigation into the interaction between syntactic and semantic representations would be interesting and needs to be explored.

8 Qualitative Analysis

In this section, we inspect the learned vector space by computing the similarities between word embeddings.

In this experiment we trained the WON on the BookCorpus (Zhu et al., 2015) that is preprocessed in the same way described in Section 4.1. The BookCorpus consists of a large collection of novels, which results in a grammatically sophisticated text corpus that would be suitable for qualitative analysis. Note that to clearly investigate the word embeddings produced by the WON we neither initialize our word embeddings with other models nor use fine-tuning techniques, as in experiments on downstream syntax-related tasks (Section 5 and Section 6). We choose queries focusing on (1) declension of personal pronouns, (2) singular and plural forms of nouns, (3) verb conjugation, (4) comparative/superlative forms of adjectives, and (5) prepositions.

Table 5 presents some representative queries for (1)-(5) and their respective most similar words in the learned vector space. First we can observe that our word embeddings produce a continuous vector space that successfully captures syntactic regularities. In addition to the syntactic regularities, interestingly, we found that the WON prefers to gather words in terms of those meanings or semantic categories.

9 Conclusion and Future Work

The research question we explored in this study was how to learn *syntactic word embeddings* without using any human annotations. Our underlying hypothesis is that the word odering task is suitable for obtaining syntactic knowledge about words. To verify this idea, we developed the WON, which implicitly learns syntactic word representations through learning to explicitly solve the word ordering task. The experimental results demonstrate that the WON gives improvements over baselines particularly on syntax-related tasks, such as part-of-speech tagging and dependency parsing. We can also observe that the WON, by combined with a simple fine-tuning technique, has the potential to refine not only syntactic but also semantic features.

It remains unclear how well order-sensitive models like the WON can learn syntactic knowledge about words in languages other than English. Especially, it is interesting to investigate cases on languages with richer morphology and freer word order. We leave this to future work.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive and helpful suggestions on this work. We also thank Makoto Miwa and Naoaki Okazaki for valuable comments and

discussion. This work was supported by JSPS KAKENHI Grant Number 16H05872 and JST CREST JPMJCR1304.

References

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

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

John R. Firth. 1957. *A synopsis of linguistic theory, 1930-1955*. Blackwell.

Simon Func. 2015. Smorms3 - blog entry: Rmsprop loses to smorms3 - beware the epsilon! *http://sifter.org/ simon/journal/20150420.html*.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

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

Over Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

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 the 2015 Conference of Empirical Methods in Natural Language Processing*.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015b. Two/too simple adaptation of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

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

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

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

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

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 the conference on empirical methods in natural language processing*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2).

Andrew Trask, David Gilmore, and Matthew Russell. 2015. Modeling order in neuralword embeddings at scale. In *Proceedings of The 32nd International Conference on Machine Learning*.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724*.