Unsupervised Learning of Syntactic Structure with Invertible Neural Projections

Junxian He   Graham Neubig   Taylor Berg-Kirkpatrick

Language Technologies Institute
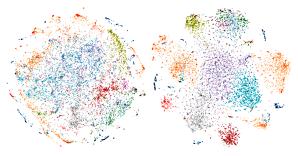School of Computer Science
Carnegie Mellon University
{junxianh, gneubig, tberg}@cs.cmu.edu

Abstract

Unsupervised learning of syntactic structure is typically performed using generative models with discrete latent variables and multinomial parameters. In most cases, these models have not leveraged continuous word representations. In this work, we propose a novel generative model that jointly learns discrete syntactic structure and continuous word representations in an unsupervised fashion by cascading an *invertible* neural network with a structured generative prior. We show that the invertibility condition allows for efficient exact inference and marginal likelihood computation in our model so long as the prior is well-behaved. In experiments we instantiate our approach with both Markov and tree-structured priors, evaluating on two tasks: part-of-speech (POS) induction, and unsupervised dependency parsing without gold POS annotation. On the Penn Treebank, our Markov-structured model surpasses state-of-the-art results on POS induction. Similarly, we find that our tree-structured model achieves state-of-the-art performance on unsupervised dependency parsing for the difficult training condition where neither gold POS annotation nor punctuation-based constraints are available.[1]

1 Introduction

Data annotation is a major bottleneck for the application of supervised learning approaches to many problems. As a result, unsupervised methods that learn directly from unlabeled data are increasingly important. For tasks related to unsupervised syntactic analysis, discrete generative models have dominated in recent years – for example, for both part-of-speech (POS) induction (Blunsom and Cohn, 2011; Stratos et al., 2016) and unsupervised dependency parsing (Klein and Manning,

(a) Traditional pre-trained skip-gram embeddings   (b) Learned latent embeddings from our approach

Figure 1: Visualization (t-SNE) of skip-gram embeddings (trained on one billion words with context window size equal to 1) and latent embeddings learned by our approach with a Markov-structured prior. Each node represents a word and is colored according to the most likely gold POS tag from the Penn Treebank (best seen in color).

2004; Cohen and Smith, 2009; Pate and Johnson, 2016). While similar models have had success on a range of unsupervised tasks, they have mostly ignored the apparent utility of continuous word representations evident from supervised NLP applications (He et al., 2017; Peters et al., 2018). In this work, we focus on leveraging and explicitly representing continuous word embeddings within unsupervised models of syntactic structure.

Pre-trained word embeddings from massive unlabeled corpora offer a compact way of injecting a prior notion of word similarity into models that would otherwise treat words as discrete, isolated categories. However, the specific properties of language captured by any particular embedding scheme can be difficult to control, and, further, may not be ideally suited to the task at hand. For example, pre-trained skip-gram embeddings (Mikolov et al., 2013) with small context window size are found to capture the syntactic properties of language well (Bansal et al., 2014; Lin et al., 2015). However, if our goal is to separate syntactic categories, this embedding space is not ideal – POS categories correspond to overlap-

---

[1] Code is available at https://github.com/jxhe/struct-learning-with-flow.
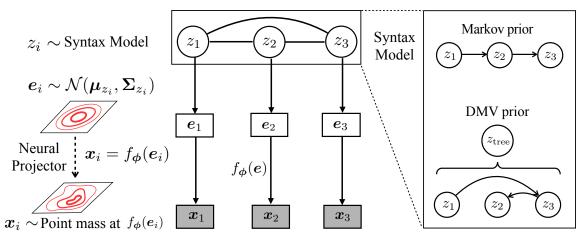
Figure 2: Depiction of proposed generative model. The syntax model is composed of discrete random variables, z_i. Each e_i is a latent continuous embeddings sampled from Gaussian distribution conditioned on z_i, while x_i is the observed embedding, deterministically derived from e_i. The left portion depicts how the neural projector maps the simple Gaussian to a more complex distribution in the output space. The right portion depicts two instantiations of the syntax model in our approach: one is Markov-structured and the other is DMV-structured. For DMV, z_{tree} is the latent dependency tree structure.

ping interspersed regions in the embedding space, evident in Figure 1(a).

In our approach, we propose to learn a new latent embedding space as a projection of pre-trained embeddings (depicted in Figure 1(b)), while *jointly* learning latent syntactic structure – for example, POS categories or syntactic dependencies. To this end, we introduce a new generative model (shown in Figure 2) that first generates a latent syntactic representation (e.g. a dependency parse) from a discrete structured prior (which we also call the "syntax model"), then, conditioned on this representation, generates a sequence of latent embedding random variables corresponding to each word, and finally produces the observed (pre-trained) word embeddings by projecting these latent vectors through a parameterized non-linear function. The latent embeddings can be jointly learned with the structured syntax model in a completely unsupervised fashion.

By choosing an invertible neural network as our non-linear projector, and then parameterizing our model in terms of the projection's inverse, we are able to derive tractable exact inference and marginal likelihood computation procedures so long as inference is tractable in the underlying syntax model. In §3.1 we show that this derivation corresponds to an alternate view of our approach whereby we jointly learn a mapping of observed word embeddings to a new embedding space that is more suitable for the syntax model, but include an additional Jacobian regularization term to prevent information loss.

Recent work has sought to take advantage of word embeddings in unsupervised generative models with alternate approaches (Lin et al., 2015; Tran et al., 2016; Jiang et al., 2016; Han et al., 2017). Lin et al. (2015) build an HMM with Gaussian emissions on observed word embeddings, but they do not attempt to learn new embeddings. Tran et al. (2016), Jiang et al. (2016), and Han et al. (2017) extend HMM or dependency model with valence (DMV) (Klein and Manning, 2004) with multinomials that use word (or tag) embeddings in their parameterization. However, they do not represent the embeddings as latent variables.

In experiments, we instantiate our approach using both a Markov-structured syntax model and a tree-structured syntax model – specifically, the DMV. We evaluate on two tasks: part-of-speech (POS) induction and unsupervised dependency parsing without gold POS tags. Experimental results on the Penn Treebank (Marcus et al., 1993) demonstrate that our approach improves the basic HMM and DMV by a large margin, leading to the state-of-the-art results on POS induction, and state-of-the-art results on unsupervised dependency parsing in the difficult training scenario where neither gold POS annotation nor punctuation-based constraints are available.

2 Model

As an illustrative example, we first present a baseline model for Markov syntactic structure (POS induction) that treats a sequence of pre-trained word embeddings as observations. Then, we propose our novel approach, again using Markov structure, that introduces latent word embedding variables and a neural projector. Lastly, we extend our approach to more general syntactic structures.

2.1 Example: Gaussian HMM

We start by describing the Gaussian hidden Markov model introduced by Lin et al. (2015), which is a locally normalized model with multinomial transitions and Gaussian emissions. Given a sentence of length ℓ, we denote the latent POS tags as $\boldsymbol{z} = \{z_i\}_{i=1}^{\ell}$, observed (pre-trained) word embeddings as $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^{\ell}$, transition parameters as $\boldsymbol{\theta}$, and Gaussian emission parameters as $\boldsymbol{\eta}$. The joint distribution of data and latent variables factors as:

$$p(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\eta}) = \prod_{i=1}^{\ell} p_{\boldsymbol{\theta}}(z_i|z_{i-1}) p_{\boldsymbol{\eta}}(\boldsymbol{x}_i|z_i), \quad (1)$$

where $p_{\boldsymbol{\theta}}(z_i|z_{i-1})$ is the multinomial transition probability and $p_{\boldsymbol{\eta}}(\boldsymbol{x}_i|z_i)$ is the multivariate Gaussian emission probability.

While the observed word embeddings do inform this model with a notion of word similarity – lacking in the basic multinomial HMM – the Gaussian emissions may not be sufficiently flexible to separate some syntactic categories in the complex pre-trained embedding space – for example the skip-gram embedding space as visualized in Figure 1(a) where different POS categories overlap. Next we introduce a new approach that adds flexibility to the emission distribution by incorporating new latent embedding variables.

2.2 Markov Structure with Neural Projector

To flexibly model observed embeddings and yield a new representation space that is more suitable for the syntax model, we propose to cascade a neural network as a projection function, deterministically transforming the simple space defined by the Gaussian HMM to the observed embedding space. We denote the latent embedding of the ith word in a sentence as $\boldsymbol{e}_i \in \mathbb{R}^{d_e}$, and the neural projection function as f, parameterized by ϕ. In the case of sequential Markov structure, our new model corresponds to the following generative process:

For each time step $i = 1, 2, \cdots, \ell$,

- Draw the latent state $z_i \sim p_{\boldsymbol{\theta}}(z_i|z_{i-1})$
- Draw the latent embedding $\boldsymbol{e}_i \sim \mathcal{N}(\boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i})$
- Deterministically produce embedding
  $\boldsymbol{x}_i = f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)$

The graphical model is depicted in Figure 2. The deterministic projection can also be viewed as sampling each observation from a point mass at $f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)$. The joint distribution of our model is:

$$\begin{aligned} &p(\boldsymbol{z}, \boldsymbol{e}, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\phi}) \\ &= \prod_{i=1}^{\ell} [p_{\boldsymbol{\theta}}(z_i|z_{i-1}) p_{\boldsymbol{\eta}}(\boldsymbol{e}_i|z_i) p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i)], \end{aligned} \quad (2)$$

where $p_{\boldsymbol{\eta}}(\cdot|z_i)$ is a conditional Gaussian distribution, and $p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i)$ is the Dirac delta function centered at $f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)$:

$$p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i) = \delta(\boldsymbol{x}_i - f_{\boldsymbol{\phi}}(\boldsymbol{e}_i)) = \begin{cases} \infty & \boldsymbol{x}_i = f_{\boldsymbol{\phi}}(\boldsymbol{e}_i) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

2.3 General Structure with Neural Projector

Our approach can be applied to a broad family of structured syntax models. We denote latent embedding variables as $\boldsymbol{e} = \{\boldsymbol{e}_i\}_{i=1}^{\ell}$, discrete latent variables in the syntax model as $\boldsymbol{z} = \{z_k\}_{k=1}^{K}$ ($K \geqslant \ell$), where $z_1, z_2, \ldots, z_\ell$ are conditioned to generate $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_\ell$. The joint probability of our model factors as:

$$\begin{aligned} p(\boldsymbol{z}, \boldsymbol{e}, \boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\phi}) = &\prod_{i=1}^{\ell} \big[p_{\boldsymbol{\eta}}(\boldsymbol{e}_i|z_i) p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i) \big] \\ &\cdot p_{\text{syntax}}(\boldsymbol{z}; \boldsymbol{\theta}), \end{aligned} \quad (4)$$

where $p_{\text{syntax}}(\boldsymbol{z}; \boldsymbol{\theta})$ represents the probability of the syntax model, and can encode any syntactic structure – though, its factorization structure will determine whether inference is tractable in our full model. As shown in Figure 2, we focus on two syntax models for syntactic analysis in this paper. The first is Markov-structured, which we use for POS induction, and the second is DMV-structured, which we use to learn dependency parses without supervision.

The marginal data likelihood of our model is:

$$\begin{aligned} p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} \Big(&p_{\text{syntax}}(\boldsymbol{z}; \boldsymbol{\theta}) \\ &\cdot \prod_{i=1}^{\ell} \big[\underbrace{\int_{\boldsymbol{e}_i} p_{\boldsymbol{\eta}}(\boldsymbol{e}_i|z_i) p_{\boldsymbol{\phi}}(\boldsymbol{x}_i|\boldsymbol{e}_i) \mathrm{d}\boldsymbol{e}_i}_{p(\boldsymbol{x}_i|z_i)} \big] \Big). \end{aligned} \quad (5)$$

While the discrete variables $\boldsymbol{z}$ can be marginalized out with dynamic program in many cases, it is generally intractable to marginalize out the latent continuous variables, $\boldsymbol{e}_i$, for an arbitrary projection f in Eq. (5), which means inference and learning may be difficult. In §3, we address this issue by constraining f to be invertible, and show that this constraint enables tractable exact inference and marginal likelihood computation.

3 Learning & Inference

In this section, we introduce an invertibility condition for our neural projector to tackle the optimization challenge. Specifically, we constrain our neural projector with two requirements: (1) $\dim(\boldsymbol{x}) = \dim(\boldsymbol{e})$ and (2) f_ϕ^{-1} exists. Invertible transformations have been explored before in independent components analysis (Hyvärinen et al., 2004), gaussianization (Chen and Gopinath, 2001), and deep density models (Dinh et al., 2014, 2016; Kingma and Dhariwal, 2018), for unstructured data. Here, we generalize this style of approach to *structured* learning, and augment it with discrete latent variables (z_i). Under the invertibility condition, we derive a learning algorithm and give another view of our approach revealed by the objective function. Then, we present the architecture of a neural projector we use in experiments: a volume-preserving invertible neural network proposed by Dinh et al. (2014) for independent components estimation.

3.1 Learning with Invertibility

For ease of exposition, we explain the learning algorithm in terms of Markov structure without loss of generality. As shown in Eq. (5), the optimization challenge in our approach comes from the intractability of the marginalized emission factor $p(\boldsymbol{x}_i|z_i)$. If we can marginalize out $\boldsymbol{e}_i$ and compute $p(\boldsymbol{x}_i|z_i)$, then the posterior and marginal likelihood of our Markov-structured model can be computed with the forward-backward algorithm. We can apply Eq. (3) and obtain :

$$p(\boldsymbol{x}_i|z_i; \boldsymbol{\eta}, \boldsymbol{\phi}) = \int_{\boldsymbol{e}_i} p_{\boldsymbol{\eta}}(\boldsymbol{e}_i|z_i)\delta(\boldsymbol{x}_i - f_\phi(\boldsymbol{e}_i))\mathrm{d}\boldsymbol{e}_i.$$

By using the change of variable rule to the integration, which allows the integration variable $\boldsymbol{e}_i$ to be replaced by $\boldsymbol{x}_i' = f_\phi(\boldsymbol{e}_i)$, the marginal emission factor can be computed in closed-form when the invertibility condition is satisfied:

$$p(\boldsymbol{x}_i|\boldsymbol{z}_i; \boldsymbol{\eta}, \boldsymbol{\phi})$$
$$= \int_{\boldsymbol{x}_i'} p_{\boldsymbol{\eta}}(f_\phi^{-1}(\boldsymbol{x}_i')|z_i)\delta(\boldsymbol{x}_i - \boldsymbol{x}_i')\left|\det\frac{\partial f_\phi^{-1}}{\partial \boldsymbol{x}_i'}\right|\mathrm{d}\boldsymbol{x}_i'$$
$$= p_{\boldsymbol{\eta}}(f_\phi^{-1}(\boldsymbol{x}_i)|z_i)\left|\det\frac{\partial f_\phi^{-1}}{\partial \boldsymbol{x}_i}\right|, \qquad (6)$$

where $p_{\boldsymbol{\eta}}(\cdot|z)$ is a conditional Gaussian distribution, $\frac{\partial f_\phi^{-1}}{\partial \boldsymbol{x}_i}$ is the Jacobian matrix of function f_ϕ^{-1}

at $\boldsymbol{x}_i$, and $\left|\det\frac{\partial f_\phi^{-1}}{\partial \boldsymbol{x}_i}\right|$ represents the absolute value of its determinant. This Jacobian term is nonzero and differentiable if and only if f_ϕ^{-1} exists.

Eq. (6) shows that we can directly calculate the marginal emission distribution $p(\boldsymbol{x}_i|z_i)$. Denote the marginal data likelihood of Gaussian HMM as $p_{\text{HMM}}(\boldsymbol{x})$, then the log marginal data likelihood of our model can be directly written as:

$$\log p(\boldsymbol{x}) = \log p_{\text{HMM}}(f_\phi^{-1}(\boldsymbol{x}))$$
$$+ \sum_{i=1}^\ell \log\left|\det\frac{\partial f_\phi^{-1}}{\partial \boldsymbol{x}_i}\right|, \qquad (7)$$

where $f_\phi^{-1}(\boldsymbol{x})$ represents the new sequence of embeddings after applying f_ϕ^{-1} to each $\boldsymbol{x}_i$. Eq. (7) shows that the training objective of our model is simply the Gaussian HMM log likelihood with an additional Jacobian regularization term. From this view, our approach can be seen as equivalent to reversely projecting the data through f_ϕ^{-1} to another manifold $\boldsymbol{e}$ that is directly modeled by the Gaussian HMM, with a regularization term. Intuitively, we optimize the reverse projection f_ϕ^{-1} to modify the $\boldsymbol{e}$ space, making it more appropriate for the syntax model. The Jacobian regularization term accounts for the volume expansion or contraction behavior of the projection. Maximizing it can be thought of as preventing information loss. In the extreme case, the Jacobian determinant is equal to zero, which means the projection is non-invertible and thus information is being lost through the projection. Such "information preserving" regularization is crucial during optimization, otherwise the trivial solution of always projecting data to the same single point to maximize likelihood is viable.[2]

More generally, for an arbitrary syntax model the data likelihood of our approach is:

$$p(\boldsymbol{x}) = \sum_z \Big(p_{\text{syntax}}(\boldsymbol{z})$$
$$\cdot \prod_{i=1}^\ell p_{\boldsymbol{\eta}}(f_\phi^{-1}(\boldsymbol{x}_i)|z_i)\left|\det\frac{\partial f_\phi^{-1}}{\partial \boldsymbol{x}_i}\right|\Big). \qquad (8)$$

If the syntax model itself allows for tractable inference and marginal likelihood computation, the same dynamic program can be used to marginalize out $\boldsymbol{z}$. Therefore, our joint model inherits the tractability of the underlying syntax model.

---

[2]For example, all $\boldsymbol{e}_i$ could learn to be zero vectors, leading to the trivial solution of learning zero mean and zero variance Gaussian emissions achieving infinite data likelihood.
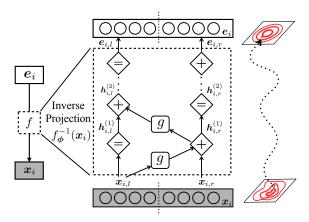
Figure 3: Depiction of the architecture of the inverse projection f_ϕ^{-1} that composes multiple volume-preserving coupling layers, with which we parameterize our model. On the right, we schematically depict how the inverse projection transforms the observed word embedding $\boldsymbol{x}_i$ to a point $\boldsymbol{e}_i$ in a new embedding space.

3.2 Invertible Volume-Preserving Neural Net

For the projection we can use an arbitrary invertible function, and given the representational power of neural networks they seem a natural choice. However, calculating the inverse and Jacobian of an arbitrary neural network can be difficult, as it requires that all component functions be invertible and also requires storage of large Jacobian matrices, which is memory intensive. To address this issue, several recent papers propose specially designed invertible networks that are easily trainable yet still powerful (Dinh et al., 2014, 2016; Jacobsen et al., 2018). Inspired by these works, we use the invertible transformation proposed by Dinh et al. (2014), which consists of a series of "coupling layers". This architecture is specially designed to guarantee a unit Jacobian determinant (and thus the invertibility property).

From Eq. (8) we know that only f_ϕ^{-1} is required for accomplishing learning and inference; we never need to explicitly construct f_ϕ. Thus, we directly define the architecture of f_ϕ^{-1}. As shown in Figure 3, the nonlinear transformation from the observed embedding $\boldsymbol{x}_i$ to $\boldsymbol{h}_i^{(1)}$ represents the first coupling layer. The input in this layer is partitioned into left and right halves of dimensions, $\boldsymbol{x}_{i,l}$ and $\boldsymbol{x}_{i,r}$, respectively. A single coupling layer is defined as:

$$\boldsymbol{h}_{i,l}^{(1)} = \boldsymbol{x}_{i,l}, \qquad \boldsymbol{h}_{i,r}^{(1)} = \boldsymbol{x}_{i,r} + g(\boldsymbol{x}_{i,l}), \qquad (9)$$

where $g : \mathbb{R}^{d_x/2} \to \mathbb{R}^{d_x/2}$ is the coupling function and can be any nonlinear form. This transformation satisfies $\dim(\boldsymbol{h}^{(1)}) = \dim(\boldsymbol{x})$, and Dinh et al. (2014) show that its Jacobian matrix is tri-

angular with all ones on the main diagonal. Thus the Jacobian determinant is always equal to one (i.e. volume-preserving) and the invertibility condition is naturally satisfied.

To be sufficiently expressive, we compose multiple coupling layers as suggested in Dinh et al. (2014). Specifically, we exchange the role of left and right half vectors at each layer as shown in Figure 3. For instance, from $\boldsymbol{x}_i$ to $\boldsymbol{h}_i^{(1)}$ the left subset $\boldsymbol{x}_{i,l}$ is unchanged, while from $\boldsymbol{h}_i^{(1)}$ to $\boldsymbol{h}_i^{(2)}$ the right subset $\boldsymbol{h}_{i,r}^{(1)}$ remains the same. Also note that composing multiple coupling layers does not change the volume-preserving and invertibility properties. Such a sequence of invertible transformations from the data space $\boldsymbol{x}$ to $\boldsymbol{e}$ is also called normalizing flow (Rezende and Mohamed, 2015).

4 Experiments

In this section, we first describe our datasets and experimental setup. We then instantiate our approach with Markov and DMV-structured syntax models, and report results on POS tagging and dependency grammar induction respectively. Lastly, we analyze the learned latent embeddings.

4.1 Data

For both POS tagging and dependency parsing, we run experiments on the Wall Street Journal (WSJ) portion of the Penn Treebank.[3] To create the observed data embeddings, we train skip-gram word embeddings (Mikolov et al., 2013) that are found to capture syntactic properties well when trained with small context window (Bansal et al., 2014; Lin et al., 2015). Following Lin et al. (2015), the dimensionality d_x is set to 100, and the training context window size is set to 1 to encode more syntactic information. The skip-gram embeddings are trained on the one billion word language modeling benchmark dataset (Chelba et al., 2013) in addition to the WSJ corpus.

4.2 General Experimental Setup

For the neural projector, we employ rectified networks as coupling function g following Dinh et al. (2014). We use a rectified network with an input layer, one hidden layer, and linear output units, the number of hidden units is set to the same as the number of input units. The number of coupling layers are varied as 4, 8, 16 for both tasks.

---

[3]Preprocessing is different for the two tasks, we describe the details in the following subsections.

We optimize marginal data likelihood directly using Adam (Kingma and Ba, 2014). For both tasks in the fully unsupervised setting, we do not tune the hyper-parameters using supervised data.

4.3 Unsupervised POS tagging

For unsupervised POS tagging, we use a Markov-structured syntax model in our approach, which is a popular structure for unsupervised tagging tasks (Lin et al., 2015; Tran et al., 2016).

Setup. Following existing literature, we train and test on the entire WSJ corpus (49208 sentences, 1M tokens). We use 45 tag clusters, the number of POS tags that appear in WSJ corpus. We train the discrete HMM and the Gaussian HMM (Lin et al., 2015) as baselines. For the Gaussian HMM, mean vectors of Gaussian emissions are initialized with the empirical mean of all word vectors with an additive noise. We assume diagonal covariance matrix for $p(e_i|z_i)$ and initialize it with the empirical variance of the word vectors. Following Lin et al. (2015), the covariance matrix is fixed during training. The multinomial probabilities are initialized as $\theta_{kv} \propto \exp(u_{kv})$, where $u_{kv} \sim U[0, 1]$. For our approach, we initialize the syntax model and Gaussian parameters with the pre-trained Gaussian HMM. The weights of layers in the rectified network are initialized from a uniform distribution with mean zero and a standard deviation of $\sqrt{1/n_{\text{in}}}$, where n_{in} is the input dimension.[4] We evaluate the performance of POS tagging with both Many-to-One (M-1) accuracy (Johnson, 2007) and V-Measure (VM) (Rosenberg and Hirschberg, 2007). Given a model we found that the tagging performance is well-correlated with the training data likelihood, thus we use training data likelihood as a unsupervised criterion to select the trained model over 10 random restarts after training 50 epochs. We repeat this process 5 times and report the mean and standard deviation of performance.

Results. We compare our approach with basic HMM, Gaussian HMM, and several state-of-the-art systems, including sophisticated HMM variants and clustering techniques with hand-engineered features. The results are presented in Table 1. Through the introduced latent embeddings and additional neural projection, our approach improves over the Gaussian HMM by 5.4 points in M-1 and 5.6 points in VM. Neural HMM

---

[4]This is the default parameter initialization in PyTorch.

System	M-1	VM
w/o hand-engineered features		
Discrete HMM	62.7	53.8
PYP-HMM (Blunsom and Cohn, 2011)	77.5	69.8
NHMM (basic) (Tran et al., 2016)	59.8	54.2
NHMM (+ Conv) (Tran et al., 2016)	74.1	66.1
NHMM (+ Conv & LSTM) (Tran et al., 2016)	79.1	71.7
Gaussian HMM (Lin et al., 2015)	75.4 (1.0)	68.5 (0.5)
Ours (4 layers)	79.5 (0.9)	73.0 (0.7)
Ours (8 layers)	**80.8** (1.3)	**74.1** (0.7)
Ours (16 layers)	73.2 (4.3)	70.5 (2.1)
w/ hand-engineered features		
Feature HMM (Berg-Kirkpatrick et al., 2010)	75.5	–
Brown (+ proto) (Christodoulopoulos et al., 2010)	76.1	68.8
Cluster (word-based) (Yatbaz et al., 2012)	80.2	72.1
Cluster (token-based) (Yatbaz et al., 2014)	79.5	69.1

Table 1: Unsupervised POS tagging results on entire WSJ, compared with other baselines and state-of-the-art systems. Standard deviation is given in parentheses when available.
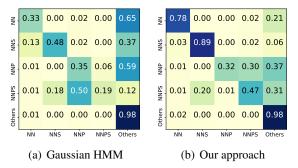

(a) Gaussian HMM   (b) Our approach

Figure 4: Normalized Confusion matrix for POS tagging experiments, row label represents the gold tag.

(NHMM) (Tran et al., 2016) is a baseline that also learns word representation jointly. Both their basic model and extended Conv version does not outperform the Gaussian HMM. Their best model incorporates another LSTM to model long distance dependency and breaks the Markov assumption, yet our approach still achieves substantial improvement over it without considering more context information. Moreover, our method outperforms the best published result that benefits from hand-engineered features (Yatbaz et al., 2012) by 2.0 points on VM.

Confusion Matrix. We found that most tagging errors happen in noun subcategories. Therefore, we do the one-to-one mapping between gold POS tags and induced clusters and plot the normalized confusion matrix of noun subcategories in Figure 4. The Gaussian HMM fails to identify "NN" and "NNS" correctly for most cases, and it often recognizes "NNPS" as "NNP". In contrast, our approach corrects these errors well.

4.4 Unsupervised Dependency Parsing without gold POS tags

For the task of unsupervised dependency parse induction, we employ the Dependency Model with Valence (DMV) (Klein and Manning, 2004) as the syntax model in our approach. DMV is a generative model that defines a probability distribution over dependency parse trees and syntactic categories, generating tokens and dependencies in a head-outward fashion. While, traditionally, DMV is trained using gold POS tags as observed syntactic categories, in our approach, we treat each tag as a latent variable, as described in §2.3.

Most existing approaches to this task are not fully unsupervised since they rely on gold POS tags following the original experimental setup for DMV. This is partially because automatically parsing from words is difficult even when using unsupervised syntactic categories (Spitkovsky et al., 2011a). However, inducing dependencies from words alone represents a more realistic experimental condition since gold POS tags are often unavailable in practice. Previous work that has trained from words alone often requires additional linguistic constraints (like sentence internal boundaries) (Spitkovsky et al., 2011a,b, 2012, 2013), acoustic cues (Pate and Goldwater, 2013), additional training data (Pate and Johnson, 2016), or annotated data from related languages (Cohen et al., 2011). Our approach is naturally designed to train on word embeddings directly, thus we attempt to induce dependencies without using gold POS tags or other extra linguistic information.

Setup. Like previous work we use sections 02-21 of WSJ corpus as training data and evaluate on section 23, we remove punctuations and train the models on sentences of length $\leqslant 10$, "head-percolation" rules (Collins, 1999) are applied to obtain gold dependencies for evaluation. We train basic DMV, extended DMV (E-DMV) (Headden III et al., 2009) and Gaussian DMV (which treats POS tag as unknown latent variables and generates observed word embeddings directly conditioned on them following Gaussian distribution) as baselines. Basic DMV and E-DMV are trained with Viterbi EM (Spitkovsky et al., 2010) on unsupervised POS tags induced from our Markov-structured model described in §4.3. Multinomial parameters of the syntax model in both Gaussian DMV and our model are initialized with the pre-trained DMV baseline. Other

System	$\leqslant 10$	all
w/o gold POS tags		
DMV (Klein and Manning, 2004)	49.6	35.8
E-DMV (Headden III et al., 2009)	52.1	38.2
UR-A E-DMV (Tu and Honavar, 2012)	58.9	46.1
CS* (Spitkovsky et al., 2013)	72.0*	64.4*
Neural E-DMV (Jiang et al., 2016)	55.3	42.7
CRFAE (Cai et al., 2017)	37.2	29.5
Gaussian DMV	55.4 (1.3)	43.1 (1.2)
Ours (4 layers)	58.4 (1.9)	46.2 (2.3)
Ours (8 layers)	**60.2** (1.3)	**47.9** (1.2)
Ours (16 layers)	54.1 (8.5)	43.9 (5.7)
w/ gold POS tags (for reference only)		
DMV (Klein and Manning, 2004)	55.1	39.7
UR-A E-DMV (Tu and Honavar, 2012)	71.4	57.0
MaxEnc (Le and Zuidema, 2015)	73.2	65.8
Neural E-DMV (Jiang et al., 2016)	72.5	57.6
CRFAE (Cai et al., 2017)	71.7	55.7
L-NDMV (Big training data) (Han et al., 2017)	77.2	63.2

Table 2: Directed dependency accuracy on section 23 of WSJ, evaluating on sentences of length $\leqslant 10$ and all lengths. Starred entries ($*$) denote that the system benefits from additional punctuation-based constraints. Standard deviation is given in parentheses when available.

parameters are initialized in the same way as in the POS tagging experiment. The directed dependency accuracy (DDA) is used for evaluation and we report accuracy on sentences of length $\leqslant 10$ and all lengths. We train the parser until training data likelihood converges, and report the mean and standard deviation over 20 random restarts.

Comparison with other related work. Our model directly observes word embeddings and does not require gold POS tags during training. Thus, results from related work trained on gold tags are not directly comparable. However, to measure how these systems might perform without gold tags, we run three recent state-of-the-art systems in our experimental setting: UR-A E-DMV (Tu and Honavar, 2012), Neural E-DMV (Jiang et al., 2016), and CRF Autoencoder (CRFAE) (Cai et al., 2017).[5] We use unsupervised POS tags (induced from our Markov-structured model) in place of gold tags.[6] We also train basic DMV on gold tags and include several state-of-the-art results on gold tags as reference points.

Results. As shown in Table 2, our approach is able to improve over the Gaussian DMV by 4.8 points on length $\leqslant 10$ and 4.8 points on all

---

[5]For the three systems, we use implementations from the original papers (via personal correspondence with the authors), and tune their hyperparameters on section 22 of WSJ.

[6]Using words directly is not practical because these systems often require a transition probability matrix between input symbols, which requires too much memory.

System	M-1	VM
Ours (4 layers)	78.2	71.2
Ours (8 layers)	72.5	69.7
Ours (16 layers)	67.2	69.2

Table 3: Unsupervised POS tagging results of our approach on WSJ, with random initialization of syntax model.

System	M-1	VM
Gaussian HMM	72.0	65.0
Ours (4 layers)	76.4	69.3
Ours (8 layers)	76.8	69.4
Ours (16 layers)	67.3	62.0

Table 4: Unsupervised POS tagging results on WSJ, with fastText vectors as the observed embeddings.

System	$\leqslant 10$	all
Gaussian DMV	53.6	41.3
Ours (4 layers)	56.9	43.9
Ours (8 layers)	57.1	42.3
Ours (16 layers)	52.9	39.5

Table 5: Directed dependency accuracy on section 23 of WSJ, with fastText vectors as the observed embeddings.

lengths, which suggests the additional latent embedding layer and neural projector are helpful. The proposed approach yields, to the best of our knowledge,[7] state-of-the-art performance without gold POS annotation and without sentence-internal boundary information. DMV, UR-A E-DMV, Neural E-DMV, and CRFAE suffer a large decrease in performance when trained on unsupervised tags – an effect also seen in previous work (Spitkovsky et al., 2011a; Cohen et al., 2011). Since our approach induces latent POS tags jointly with dependency trees, it may be able to learn POS clusters that are more amenable to grammar induction than the unsupervised tags. We observe that CRFAE underperforms its gold-tag counterpart substantially. This may largely be a result of the model's reliance on prior linguistic rules that become unavailable when gold POS tag types are unknown. Many extensions to DMV can be considered orthogonal to our approach – they essentially focus on improving the syntax model. It is possible that incorporating these more sophisticated syntax models into our approach may lead to further improvements.

4.5 Sensitivity Analysis

Impact of Initialization. In the above experiments we initialize the structured syntax components with the pre-trained Gaussian or discrete baseline, which is shown as a useful technique to help train our deep models. We further study the results with fully random initialization. In the POS tagging experiment, we report the results in Table 3. While the performance with 4 layers is comparable to the pre-trained Gaussian initialization, deeper projections (8 or 16 layers) result in a dramatic drop in performance. This suggests that the structured syntax model with very deep projections is difficult to train from scratch, and a simpler projection might be a good compromise in the random initialization setting.

Different from the Markov prior in POS tag-

ging experiments, our parsing model seems to be quite sensitive to the initialization. For example, directed accuracy of our approach on sentences of length $\leqslant 10$ is below 40.0 with random initialization. This is consistent with previous work that has noted the importance of careful initialization for DMV-based models such as the commonly used harmonic initializer (Klein and Manning, 2004). However, it is not straightforward to apply the harmonic initializer for DMV directly in our model without using some kind of pre-training since we do not observe gold POS.

Impact of Observed Embeddings. We investigate the effect of the choice of pre-trained embedding on performance while using our approach. To this end, we additionally include results using fastText embeddings (Bojanowski et al., 2017) – which, in contrast with skip-gram embeddings, include character-level information. We set the context windows size to 1 and the dimension size to 100 as in the skip-gram training, while keeping other parameters set to their defaults. These results are summarized in Table 4 and Table 5. While fastText embeddings lead to reduced performance with our model, our approach still yields an improvement over the Gaussian baseline with the new observed embeddings space.

4.6 Qualitative Analysis of Embeddings

We perform qualitative analysis to understand how the latent embeddings help induce syntactic structures. First we filter out low-frequency words and punctuations in WSJ, and visualize the rest words (10k) with t-SNE (Maaten and Hinton, 2008) under different embeddings. We assign each word with its most likely gold POS tags in WSJ and color them according to the gold POS tags.

---

[7]We tried to be as thorough as possible in evaluation by running top performing systems using our more difficult training setup when this was feasible – but it was not possible to evaluate them all.

Target	Skip-gram	Markov Structure
come	go came follow coming sit	be go do give follow
singing	dancing sing drumming dance dances	dancing drumming marching playing recording
cigars	cigarettes sodas champagne cigar rum	sodas bottles drinks pills cigarettes
newer	flashier fancier conventional low-end new-generation	softer lighter thinner darker smoother
fanciest	priciest up-scale loveliest fancier high-end	liveliest priciest smartest best-run fastest-growing

Table 6: Target words and their 5 nearest neighbors, based on skip-gram embeddings and our learned latent embeddings with Markov-structured syntax model.
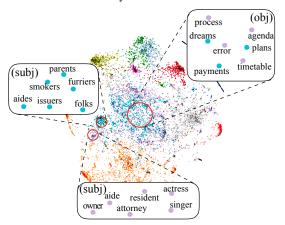

Figure 5: Visualization (t-SNE) of learned latent embeddings with DMV-structured syntax model. Each node represents a word and is colored according to the most likely gold POS tag in the Penn Treebank (best seen in color).

For our Markov-structured model, we have displayed the embedding space in Figure 1(b), where the gold POS clusters are well-formed. Further, we present five example target words and their five nearest neighbors in terms of cosine similarity. As shown in Table 6, the skip-gram embedding captures both semantic and syntactic aspects to some degree, yet our embeddings are able to focus especially on the syntactic aspects of words, in an unsupervised fashion without using any extra morphological information.

In Figure 5 we depict the learned latent embeddings with the DMV-structured syntax model. Unlike the Markov structure, the DMV structure maps a large subset of singular and plural nouns to the same overlapping region. However, two clusters of singular and plural nouns are actually separated. We inspect the two clusters and the overlapping region in Figure 5, it turns out that the nouns in the separated clusters are words that can appear as subjects and, therefore, for which verb agreement is important to model. In contrast, the nouns

in the overlapping region are typically objects. This demonstrates that the latent embeddings are focusing on aspects of language that are specifically important for modeling dependency without ever having seen examples of dependency parses.

Some previous work has deliberately created embeddings to capture different notions of similarity (Levy and Goldberg, 2014; Cotterell and Schütze, 2015), while they use extra morphology or dependency annotations to guide the embedding learning, our approach provides a potential alternative to create new embeddings that are guided by structured syntax model, only using unlabeled text corpora.

5 Related Work

Our approach is related to flow-based generative models, which are first described in NICE (Dinh et al., 2014) and have recently received more attention (Dinh et al., 2016; Jacobsen et al., 2018; Kingma and Dhariwal, 2018). This relevant work mostly adopts simple (e.g. Gaussian) and fixed priors and does not attempt to learn interpretable latent structures. Another related generative model class is variational auto-encoders (VAEs) (Kingma and Welling, 2013) that optimize a lower bound on the marginal data likelihood, and can be extended to learn latent structures (Miao and Blunsom, 2016; Yin et al., 2018). Against the flow-based models, VAEs remove the invertibility constraint but sacrifice the merits of exact inference and exact log likelihood computation, which potentially results in optimization challenges (Kingma et al., 2016). Our approach can also be viewed in connection with generative adversarial networks (GANs) (Goodfellow et al., 2014) that is a likelihood-free framework to learn implicit generative models. However, it is nontrivial for a gradient-based method like GANs to propagate gradients through discrete structures.

6 Conclusion

In this work, we define a novel generative approach to leverage continuous word representations for unsupervised learning of syntactic structure. Experiments on both POS induction and unsupervised dependency parsing tasks demonstrate the effectiveness of our proposed approach. Future work might explore more sophisticated invertible projections, or recurrent projections that jointly transform the entire input sequence.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of HLT-NAACL*, pages 582–590. Association for Computational Linguistics.

Phil Blunsom and Trevor Cohn. 2011. A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of ACL*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*.

Jiong Cai, Yong Jiang, and Kewei Tu. 2017. Crf autoencoder for unsupervised dependency parsing. In *Proceedings of EMNLP*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Scott Saobing Chen and Ramesh A Gopinath. 2001. Gaussianization. In *Advances in neural information processing systems*.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of EMNLP*.

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with nonparallel multilingual guidance. In *Proceedings of EMNLP*.

Shay B Cohen and Noah A Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of HLT-NAACL*.

Michael Collins. 1999. *HEAD-DRIVEN STATISTICAL MODELS FOR NATURAL LANGUAGE PARSING*. Ph.D. thesis, University of Pennsylvania.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of NAACL-HLT*.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*.

Wenjuan Han, Yong Jiang, and Kewei Tu. 2017. Dependency grammar induction with neural lexicalization and big training data. In *Proceedings of EMNLP*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of ACL*.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of HLT-NAACL*.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. 2004. *Independent component analysis*, volume 46. John Wiley & Sons.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. 2018. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of EMNLP*.

Mark Johnson. 2007. Why doesnt em find good hmm pos-taggers? In *Proceedings of the EMNLP-CoNLL*.

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

Diederik P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Diederik P Kingma and Max Welling. 2013. Autoencoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Dan Klein and Christopher D Manning. 2004. Corpusbased induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of NAACL-HLT*.

Omer Levy and Yoav Goldberg. 2014. Dependencybased word embeddings. In *Proceedings of ACL*.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. In *Proceedings of the NAACL-HLT*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*.

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.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of EMNLP*.

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

John K Pate and Sharon Goldwater. 2013. Unsupervised dependency parsing with acoustic cues. *Transactions of the Association for Computational Linguistics*.

John K Pate and Mark Johnson. 2016. Grammar induction from (lots of) words alone. In *Proceedings of COLING*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of HLT-NAACL*.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of ICML*.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of EMNLP-CoNLL*.

Valentin I Spitkovsky, Hiyan Alshawi, Angel X Chang, and Daniel Jurafsky. 2011a. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of EMNLP*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of CoNLL*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2012. Capitalization cues improve dependency grammar induction. In *Proceedings of NAACL-HLT Workshop on the Induction of Linguistic Structure*.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of EMNLP*.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL*.

Karl Stratos, Michael Collins, and Daniel Hsu. 2016. Unsupervised part-of-speech tagging with anchor hidden markov models. *Transactions of the Association for Computational Linguistics*.

Ke M Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of EMNLP-CoNLL*.

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

Mehmet Ali Yatbaz, Enis Rıfat Sert, and Deniz Yuret. 2014. Unsupervised instance-based part of speech induction using probable substitutes. In *Proceedings of COLING*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of ACL*.