Unsupervised PO

Departme

Objectives

This work tries to inform the HMM model with word embeddings to solve the unsupervised POS

OS Induction with

Chao Zhao zhaochaocs@gmail.co

ent of Computer Science and Engineering, University of Cal

Latent syntac

He et al. [2] modify the generative model by adding a latent word embedding layer $\mathbf{z}$ between the dis-

Normalizing Flows

om

ifornia, Santa Cruz

CTIC EMBEDDING

By introducing $\boldsymbol{e}'_i = f(\boldsymbol{z}_i)$ and applying the change of variable rule discussed above, they obtain an an-

S

Baskin
Engineering
UC SANTA CRUZ

Results

The performance of the proposed model is tested on the WSJ dataset:

induction task. The highlights are as follows:

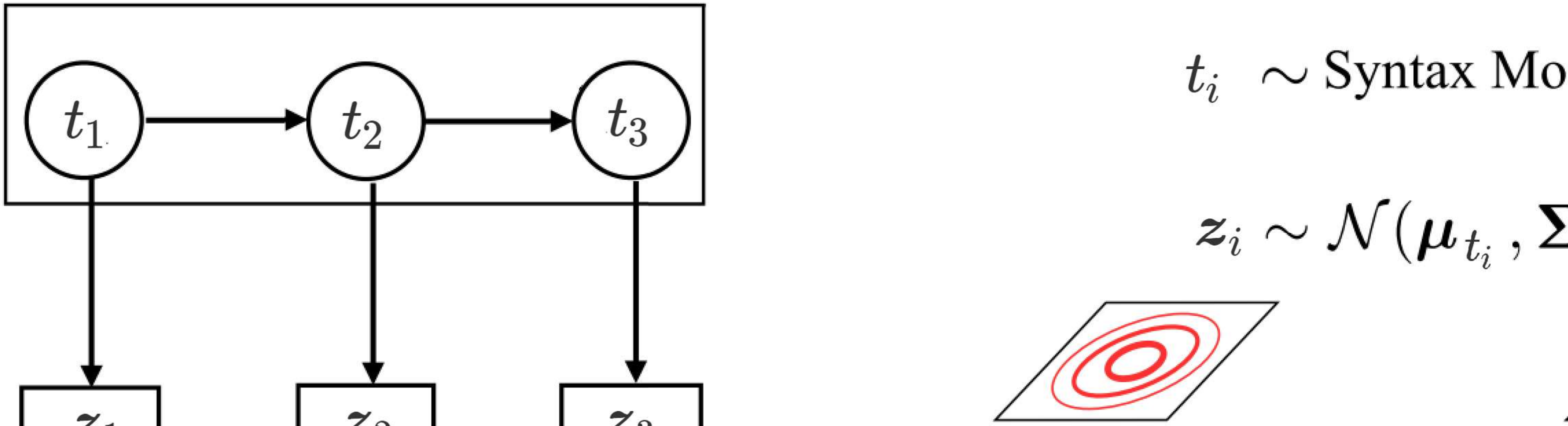
- Leveraging the word embeddings into the unsupervised HMM models;
- Learning an extra latent embedding layer for POS tagging;
- Using normalizing flows for inference and learning;
- Achieving a new state-of-the-art performance.

Introduction

Unsupervised POS induction aims to assign each word w_i of a sentence $\mathbf{w}_{<1,\cdots,n>}$ with a POS tag t_i without the help of labeled data for supervised crete POS tags $\mathbf{t}$ and the observed pre-trained word embeddings $\mathbf{e}$. They generate the latent embeddings using the same Gaussian HMM model, then transform it into the observed space through a normalizing flow $f(\cdot)$. The joint probability becomes

$$p(\mathbf{x}, \mathbf{t}) = p(\mathbf{e}, \mathbf{t}) = \sum_i p_\eta(t_i|t_{i-1}) \cdot p_\theta(\boldsymbol{z_i}|t_i) \cdot p_\phi(\boldsymbol{e_i}|\boldsymbol{z_i}),$$

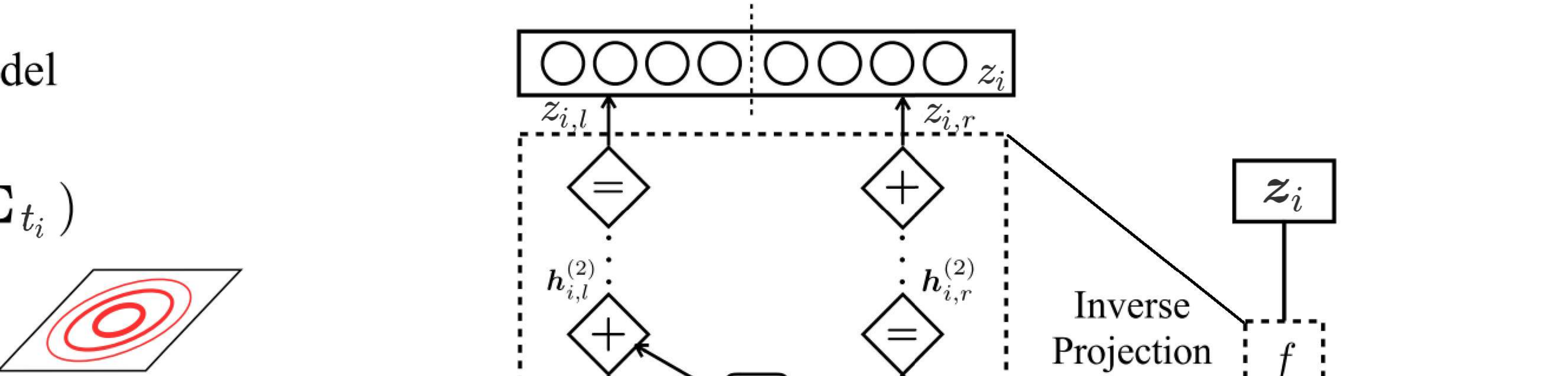
where $p_\phi(\boldsymbol{e_i}|\boldsymbol{z_i}) = \delta(\boldsymbol{e_i} - f(\boldsymbol{z_i}))$ is a Dirac delta function to force the latent variable $\boldsymbol{z_i}$ to be mapped exactly the same as the observed embedding $\boldsymbol{e_i}$.


t_1
t_2
t_3
$t_i \sim$ Syntax Mo
$\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_{t_i}, \boldsymbol{\Sigma}$
z_1
z_2
z_3

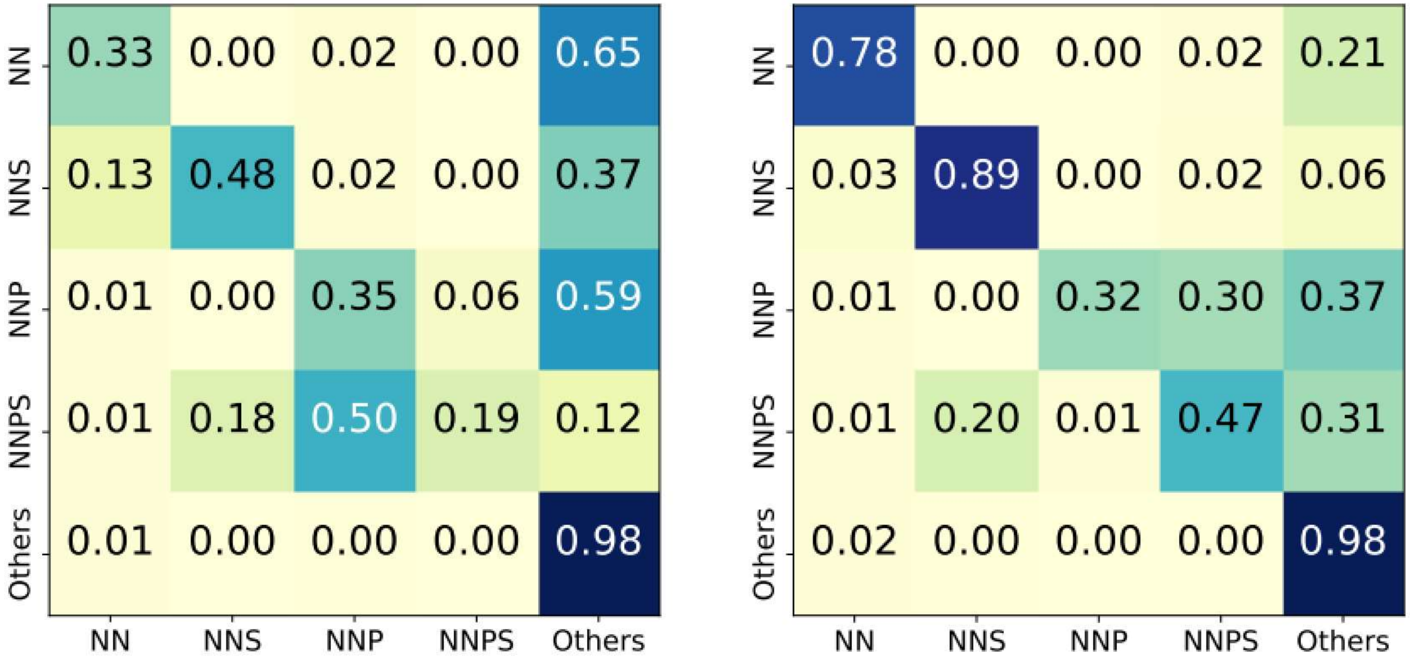
alytical expression of the marginal emission dist.

$$p(\boldsymbol{e}_i|t_i) = p_\theta(f^{-1}(\boldsymbol{e}_i)|t_i) \left|\det \frac{\partial f^{-1}}{\partial \boldsymbol{e}_i}\right|.$$

It means, by reversely projecting the observed embeddings $\mathbf{e}$ into the latent manifold $\mathbf{z}$ via f^{-1}, the proposed model is identical to the original Gaussian HMM with an extra determinant term as regularization. However, the latent embeddings would be more appropriate for separating the POS tags.



System	**M-1**	**VM**
Discrete HMM	62.7	53.8
Gaussian HMM	75.4	68.5
This work	80.8	74.1



	NN	NNS	NNP	NNPS	Others
NN	0.33	0.00	0.02	0.00	0.65
NNS	0.13	0.48	0.02	0.00	0.37
NNP	0.01	0.00	0.35	0.06	0.59
NNPS	0.01	0.18	0.50	0.19	0.12
Others	0.01	0.00	0.00	0.00	0.98

	NN	NNS	NNP	NNPS	Others
NN	0.78	0.00	0.00	0.02	0.21
NNS	0.03	0.89	0.00	0.02	0.06
NNP	0.01	0.00	0.32	0.30	0.37
NNPS	0.01	0.20	0.01	0.47	0.31
Others	0.02	0.00	0.00	0.00	0.98

Figure 4: Normalized confusion matrix of Gaussian (left) and this work (right).

t_i without the help of labeled data for supervised training. Traditional research adopted the hidden Markov model (HMM) as the generative model for this task, where the output tokens are discrete variables and therefore the syntactic similarities between tokens are lost.

To depict such similarities, Gaussian HMM [3] tries to leveraging the word embeddings onto the HMM model by inheriting the hidden Markov chain of the POS tags but changing the emission outputs from discrete tokens to continuous embeddings $\mathbf{e}$. To make it works, the emission probability is replaced by a multivariate Gaussian distribution.

However, the widely-used skip-gram word embeddings depict more on semantic similarities between words, making the embeddings of different syntactic

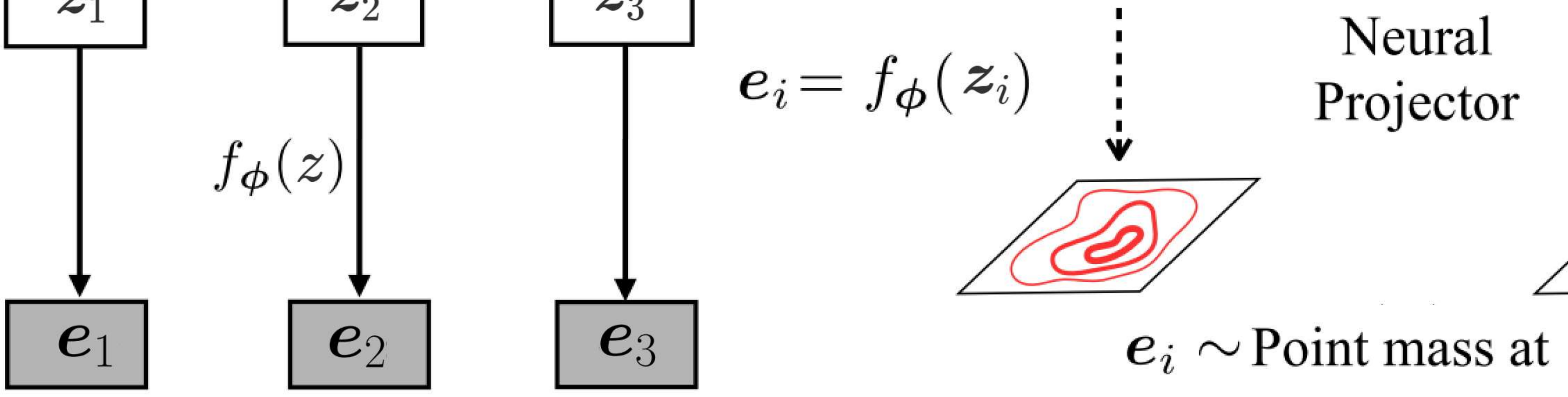
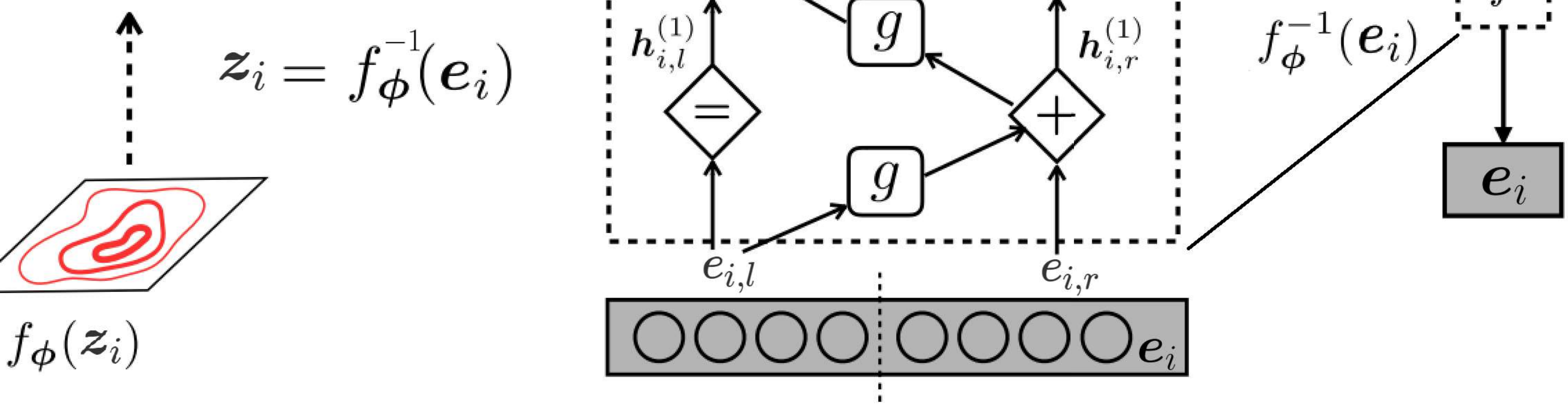


Figure 2: Depict of

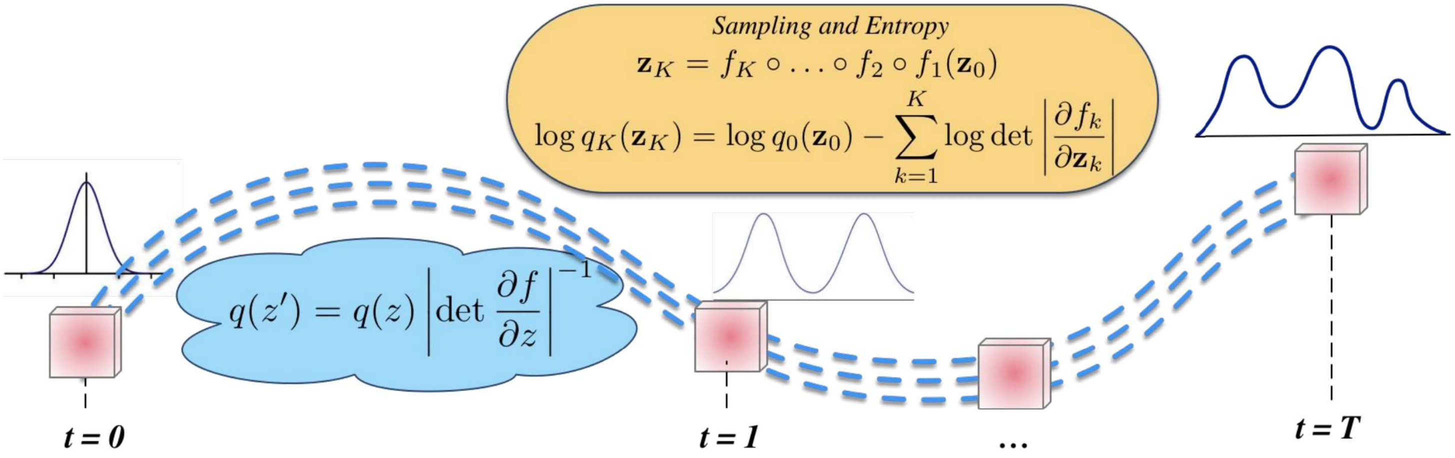
Normaliz

Normalizing flows [1] are a kind of generative models. The basic idea is to transform the latent variable $\mathbf{z}$ as $\mathbf{x}$ through a complex invertible function $f(\cdot)$. Following the *change of variable rule*, the probability of the $\mathbf{x}$ can be expressed using the original



the proposed model

ing Flows


Sampling and Entropy
$\mathbf{z}_K = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{z}_0)$
$\log q_K(\mathbf{z}_K) = \log q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log \det \left| \frac{\partial f_k}{\partial \mathbf{z}_k} \right|$
$q(z') = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1}$
t = 0
t = 1
...
t = T

Conclusion

This work depicts a novel generative approach to leverage continuous word representations for unsupervised POS induction. By adding a latent embedding layer between the discrete input and the continuous output and model the generative process with normalizing flows, this approach achieves a new state-of-the-art result and provides an example for using generative models on NLP tasks.

References

[1] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE:

words, making the embeddings of different syntactic parts overlapped and therefore is not ideal for the POS induction task.

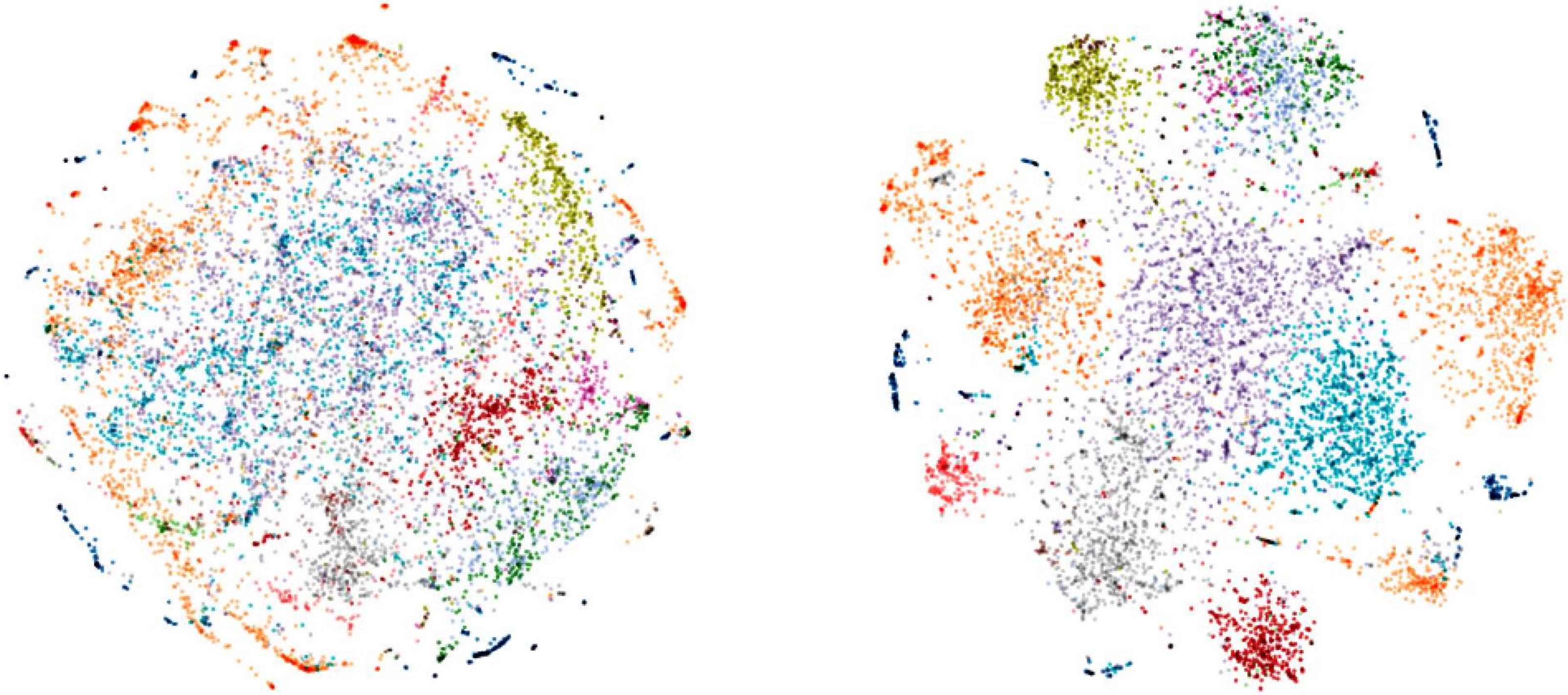


Figure 1: The General v.s. latent word embeddings

$p_\eta(\mathbf{z})$ with the Jacobian determinant of $f(\cdot)$:

$$p_\lambda(\mathbf{x}) = p_\eta(\mathbf{z})|\det \frac{\partial f^{-1}}{\partial \mathbf{x}}| = p_\eta(\mathbf{z})|\det \frac{\partial f}{\partial \mathbf{z}}|^{-1}. \quad (1)$$

By successively applying the transformations we obtain an arbitrary complex transformation $f = f_K \circ \cdots \circ f_1$, where

$$\log p_\lambda(\mathbf{x}) = \log p_\eta(\mathbf{z}) - \sum_k \log|\det \frac{\partial f_k}{\partial z_k}|. \quad (2)$$

While it is straight-forward to construct such a function using neural networks, computing the Jacobian determinant of such an arbitrary function is not trivial (need the time complexity of $O(n^3)$).

Figure 3: Normalizing flows

To make the computation of the Jacobian determinant tractable, the authors split $\boldsymbol{z}$ as two parts $\{\boldsymbol{z}_1, \boldsymbol{z}_2\}$, and design the mapping as

$$\boldsymbol{e}_1 = \boldsymbol{z}_1, \boldsymbol{e}_2 \qquad = \boldsymbol{z}_2 + g(\boldsymbol{z}_1).$$

$g(\cdot)$ is an arbitrary function. But the Jacobian determinant of this mapping will be always triangular and the corresponding determinant can be obtained in $O(n)$. The flexibility of this transformation can be further enhanced by staking multiple layers.

Non-linear Independent Components Estimation. In *ICLR, workshop*, volume 1, pages 1–13, 2015.

[2] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Unsupervised Learning of Syntactic Structure with Invertible Neural Projections. In *EMNLP*, 2018.

[3] Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. Unsupervised POS Induction with Word Embeddings. In *HLT: NAACL*, pages 1311–1316, 2015.

Acknowledgements

Thanks to all the GPP staff, lectures, and peer reviewers for their efforts.