Convolutional Poisson Gamma Belief Network

Chaojie Wang [1] **Bo Chen** [1] **Sucheng Xiao** [1] **Mingyuan Zhou** [2]

Abstract

For text analysis, one often resorts to a lossy representation that either completely ignores word order or embeds each word as a low-dimensional dense feature vector. In this paper, we propose convolutional Poisson factor analysis (CPFA) that directly operates on a lossless representation that processes the words in each document as a sequence of high-dimensional one-hot vectors. To boost its performance, we further propose the convolutional Poisson gamma belief network (CPGBN) that couples CPFA with the gamma belief network via a novel probabilistic pooling layer. CPFA forms words into phrases and captures very specific phrase-level topics, and CPGBN further builds a hierarchy of increasingly more general phrase-level topics. For efficient inference, we develop both a Gibbs sampler and a Weibull distribution based convolutional variational auto-encoder. Experimental results demonstrate that CPGBN can extract high-quality text latent representations that capture the word order information, and hence can be leveraged as a building block to enrich a wide variety of existing latent variable models that ignore word order.

1. Introduction

A central task in text analysis and language modeling is to effectively represent the documents to capture their underlying semantic structures. A basic idea is to represent the words appearing in a document with a sequence of one-hot vectors, where the vector dimension is the size of the vocabulary. This preserves all textual information but results in a collection of extremely large and sparse matrices for a text corpus. Given the memory and computation constraints, it is very challenging to directly model this lossless representation. Thus existing methods often resort to simplified lossy representations that either completely ignore word order (Blei et al., 2003), or embed the words into a lower dimensional feature space (Mikolov et al., 2013).

Ignoring word order, each document is simplified as a bag-of-words count vector, the vth element of which represents how many times the vth vocabulary term appears in that document. With a text corpus simplified as a term-document frequency count matrix, a wide array of latent variable models (LVMs) have been proposed for text analysis (Deerwester et al., 1990; Papadimitriou et al., 2000; Lee & Seung, 2001; Blei et al., 2003; Hinton & Salakhutdinov, 2009; Zhou et al., 2012). Extending “shallow” probabilistic topic models such as latent Dirichlet allocation (LDA) (Blei et al., 2003) and Poisson factor analysis (PFA) (Zhou et al., 2012), steady progress has been made in inferring multi-stochastic-layer deep latent representations for text analysis (Gan et al., 2015; Zhou et al., 2016; Ranganath et al., 2015; Zhang et al., 2018). Despite the progress, completely ignoring word order could still be particularly problematic on some common text-analysis tasks, such as spam detection and sentiment analysis (Pang et al., 2002; Tang et al., 2014).

To preserve word order, a common practice is to first convert each word in the vocabulary from a high-dimensional sparse one-hot vector into a low-dimensional dense word-embedding vector. The word-embedding vectors can be either trained as part of the learning (Kim, 2014; Kalchbrenner et al., 2014), or pre-trained by some other methods on an additional large corpus (Mikolov et al., 2013). Sequentially ordered word embedding vectors have been successfully combined with deep neural networks to address various problems in text analysis and language modeling. A typical combination method is to use the word-embedding layer as part of a recurrent neural network (RNN), especially long short-term memory (LSTM) and its variants (Hochreiter & Schmidhuber, 1997; Chung et al., 2014), achieving great success in numerous tasks that heavily rely on having high-quality sentence representation. Another popular combination method is to apply a convolutional neural network (CNN) (Lecun et al., 1998) directly to the embedding representation, treating the word embedding layer as an image input; it has been widely used in systems for entity search, sentence modeling, product feature mining, and so on (Xu

[1] National Laboratory of Radar Signal Processing, Collaborative Innovation Center of Information Sensing and Understanding, Xidian University, Xi’an, Shaanxi, China. [2] McCombs School of Business, The University of Texas at Austin, Austin, Texas 78712, USA. Correspondence to: Bo Chen <bchen@mail.xidian.edu.cn>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

& Sarikaya, 2013; Weston et al., 2014).

In this paper, we first propose convolutional PFA (CPFA) that directly models the documents, each of which is represented without information loss as a sequence of one-hot vectors. We then boot its performance by coupling it with the gamma belief network (GBN) of Zhou et al. (2016), a multi-stochastic-hidden layer deep generative model, via a novel probabilistic document-level pooling layer. We refer to the CPFA and GBN coupled model as convolutional Poisson GBN (CPGBN). To the best of our knowledge, CPGBN is the first unsupervised probabilistic convolutional model that infers multi-stochastic-layer latent variables for documents represented without information loss. Its hidden layers can be jointly trained with an upward-downward Gibbs sampler; this makes its inference different from greedy layer-wise training (Lee et al., 2009; Chen et al., 2013). In each Gibbs sampling iteration, the main computation is embarrassingly parallel and hence will be accelerated with Graphical Process Units (GPUs). We also develop a Weibull distribution based convolutional variational auto-encoder to provide amortized variational inference, which further accelerates both training and testing for large corpora. Exploiting the multi-layer structure of CPGBN, we further propose a supervised CPGBN (sCPGBN), which combines the representation power of CPGBN for topic modeling and the discriminative power of deep neural networks (NNs) under a principled probabilistic framework. We show that the proposed models achieve state-of-art results in a variety of text-analysis tasks.

2. Convolutional Models for Text Analysis

Below we introduce CPFA and then develop a probabilistic document-level pooling method to couple CPFA with GBN, which further serves as the decoder of a Weibull distribution based convolutional variational auto-encoder (VAE).

2.1. Convolutional Poisson Factor Analysis

Denote V as the vocabulary and let $D_j = (x_{j1}, ..., x_{jL_j})$ represent the L_j sequentially ordered words of the jth document, which can be represented as a sequence of one-hot vectors. For example, with vocabulary $V = \{$*"don't"*, *"hate"*, *"I"*, *"it"*, *"like"*$\}$, document $D_j = ($*"I"*, *"like"*, *"it"*$)$ can be represented as $\boldsymbol{X}_j = [\boldsymbol{x}_{j1}, \boldsymbol{x}_{j2}, \boldsymbol{x}_{j3}] \in \{0,1\}^{|V| \times L_j}$, where $\boldsymbol{x}_{j1} = (0,0,1,0,0)'$, $\boldsymbol{x}_{j2} = (0,0,0,0,1)'$, and $\boldsymbol{x}_{j3} = (0,0,0,1,0)'$ are one-hot column vectors. Let us denote $x_{jvl} = \boldsymbol{X}_j(v,l)$, which is one if and only if word l of document j matches term v of the vocabulary.

To exploit a rich set of tools developed for count data analysis (Zhou et al., 2012; 2016), we first link these sequential binary vectors to sequential count vectors via the Bernoulli-Poisson link (Zhou, 2015). More specifically, we link each x_{jvl} to a latent count as $x_{jvl} = \mathbf{1}(m_{jvk} > 0)$, where $m_{jvl} \in \mathbb{Z} := \{0, 1, \ldots\}$, and factorize the matrix $\boldsymbol{M}_j = \{m_{jvl}\}_{v,l} \in \mathbb{Z}^{|V| \times L_j}$ under the Poisson likelihood. Distinct from vanilla PFA (Zhou et al., 2012) where the columns of the matrix are treated as conditionally independent, here we introduce convolution into the hierarchical model to capture the sequential dependence between the columns. We construct the hierarchical model of CPFA as

$$\begin{aligned} &\boldsymbol{X}_j = \mathbf{1}(\boldsymbol{M}_j > 0),\ \boldsymbol{M}_j \sim \mathrm{Pois}(\textstyle\sum_{k=1}^{K} \boldsymbol{D}_k * \boldsymbol{w}_{jk}), \\ &\boldsymbol{w}_{jk} \sim \mathrm{Gam}(r_k, 1/c_j),\ \ \boldsymbol{D}_k(:) \sim \mathrm{Dir}(\eta \mathbf{1}_{|V|F}), \end{aligned} \tag{1}$$

where $*$ denotes a convolution operator, $\mathbb{R}_+ := \{x : x \geq 0\}$, $\boldsymbol{D}_k = (\boldsymbol{d}_{k1}, \ldots, \boldsymbol{d}_{kF}) \in \mathbb{R}_+^{|V| \times F}$ is the kth convolutional filter/factor/topic whose filter width is F, $\boldsymbol{d}_{kf} = (d_{k1f}, \ldots, d_{k|V|f})'$, and $\boldsymbol{D}_k(:) = (\boldsymbol{d}'_{k1}, \ldots, \boldsymbol{d}'_{kF})' \in \mathbb{R}_+^{|V|F}$; the latent count matrix $\boldsymbol{M}_j$ is factorized into the summation of K equal-sized latent count matrices, the Poisson rates of the kth of which are obtained by convolving $\boldsymbol{D}_k$ with its corresponding gamma distributed feature representation $\boldsymbol{w}_{jk} \in \mathbb{R}_+^{S_j}$, where $S_j := L_j - F + 1$. To complete the hierarchical model, we let $r_k \sim \mathrm{Gamma}(1/K, 1/c_0)$ and $c_j \sim \mathrm{Gamma}(e_0, 1/f_0)$. Note as in Zhou et al. (2016), we may consider K as the truncation level of a gamma process, which allows the number of needed factors to be inferred from the data as long as K is set sufficiently large.

We can interpret $d_{kvf} := \boldsymbol{D}_k(v, f)$ as the probability that the vth term in the vocabulary appears at the fth temporal location for the kth latent topic, and expect each $\boldsymbol{D}_k$ to extract both global cooccurrence patterns, such as common topics, and local temporal structures, such as common n-gram phrases, where $n \leq F$, from the text corpus. Note the convolution layers of CPFA convert text regions of size F (*e.g.*,"am so happy" with $F = 3$) to feature vectors, directly learning the embedding of text regions without going through a separate learning for word embedding. Thus CPFA provides a potential solution for distinguishing polysemous words according to their neighboring words. The length of the representation weight vector $\boldsymbol{w}_{jk}$ in our model is $S_j = L_j - F + 1$, which varies with the document length L_j. This differs CPFA from traditional convolutional models with a fixed feature map size (Zhang et al., 2017; Miao et al., 2018; Min et al., 2019), which requires either heuristic cropping or zero-padding.

2.2. Convolutional Poisson Gamma Belief Network

There has been significant recent interest in inferring multi-stochastic-layer deep latent representations for text analysis in an unsupervised manner (Gan et al., 2015; Zhou et al., 2016; Ranganath et al., 2015; Wang et al., 2018; Zhang et al., 2018), where word order is ignored. The key intuition behind these models, such as GBN (Zhou et al., 2016), is

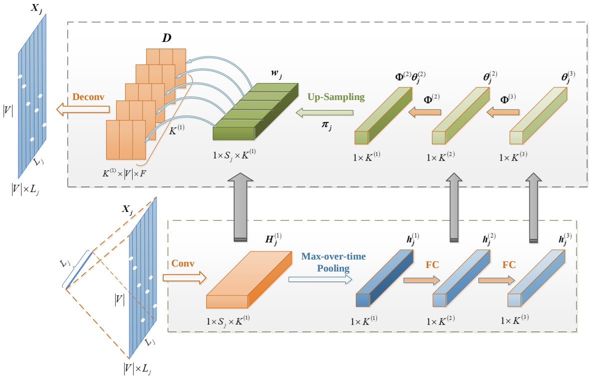

Figure 1. The proposed CPGBN (upper part) and its corresponding convolutional variational inference network (lower part).

that words frequently co-occurred in the same document can form specific word-level topics in shallow layers; as the depth of the network increases, frequently co-occurred topics can form more general ones. Here, we propose a model to preserve word order, without losing the nice hierarchical topical interpretation provided by a deep topic model. The intuition is that by preserving word order, words can first form short phrases; frequently co-occurred short phrases can then be combined to form specific phrase-level topics; and these specific phrase-level topics can form increasingly more general phrase-level topics when moving towards deeper layers.

As in Fig. 1, we couple CPFA in (1) with GBN to construct CPGBN, whose generative model with T hidden layers, from top to bottom, is expressed as

$$\begin{aligned}
&\boldsymbol{\theta}_j^{(T)} \sim \mathrm{Gam}(\boldsymbol{r}, 1/c_j^{(T+1)}),\\
&\dots,\\
&\boldsymbol{\theta}_j^{(t)} \sim \mathrm{Gam}(\boldsymbol{\Phi}^{(t+1)}\boldsymbol{\theta}_j^{(t+1)}, 1/c_j^{(t+1)}),\\
&\dots,\\
&\boldsymbol{\theta}_j^{(1)} \sim \mathrm{Gam}(\boldsymbol{\Phi}^{(2)}\boldsymbol{\theta}_j^{(2)}, 1/c_j^{(2)}),\\
&\boldsymbol{w}_{jk} = \boldsymbol{\pi}_{jk}\theta_{jk}^{(1)},\ \boldsymbol{\pi}_{jk} \sim \mathrm{Dir}\big(\boldsymbol{\Phi}_{k:}^{(2)}\boldsymbol{\theta}_j^{(2)}/S_j\boldsymbol{1}_{S_j}\big),\\
&\boldsymbol{M}_j \sim \mathrm{Pois}\big(\textstyle\sum_{k=1}^{K^{(1)}} \boldsymbol{D}_k * \boldsymbol{w}_{jk}\big),
\end{aligned} \tag{2}$$

where $\boldsymbol{\Phi}_{k:}$ is the kth row of $\boldsymbol{\Phi}$ and superscripts indicate layers. Note CPGBN first factorizes the latent count matrix $\boldsymbol{M}_j \in \mathbb{Z}^{|V|\times L_j}$ under the Poisson likelihood into the summation of $K^{(1)}$ convolutions, the kth of which is between $\boldsymbol{D}_k \in \mathbb{R}_+^{|V|\times F}$ and weight vector $\boldsymbol{w}_{jk} \in \mathbb{R}_+^{S_j}$. Using the relationship between the gamma and Dirichlet distributions (*e.g.*, Lemma IV.3 of Zhou & Carin (2012)), $\boldsymbol{w}_{jk} = (w_{jk1}, \dots, w_{jkS_j})' = (\theta_{jk}^{(1)}\pi_{jk1}, \dots, \theta_{jk}^{(1)}\pi_{jkS_j})' \in \mathbb{R}^{S_j}$ in (2) can be equivalently generated as

$$w_{jks} \sim \mathrm{Gam}\big(\boldsymbol{\Phi}_{k:}^{(2)}\boldsymbol{\theta}_j^{(2)}/S_j, 1/c_j^{(2)}\big),\ s = 1, \dots, S_j, \tag{3}$$

which could be seen as a specific probabilistic document-level pooling algorithm on the gamma shape parameter. For $t \in \{1, ..., T-1\}$, the shape parameters of the gamma distributed hidden units $\boldsymbol{\theta}_j^{(t)} \in \mathbb{R}_+^{K^{(t)}}$ are factorized into the product of the connection weight matrix $\boldsymbol{\Phi}^{(t+1)} \in \mathbb{R}_+^{K^{(t)}\times K^{(t+1)}}$ and hidden units $\boldsymbol{\theta}_j^{(t+1)} \in \mathbb{R}_+^{K^{(t+1)}}$ of layer $t+1$; the top layer's hidden units $\boldsymbol{\theta}_j^{(T)}$ share the same $\boldsymbol{r} \in \mathbb{R}_+^{K^{(T)}}$ as their gamma shape parameters; and $c_j^{(t+1)}$ are gamma scale parameters. For scale identifiability and ease of inference, the columns of $\boldsymbol{D}_k$ and $\boldsymbol{\Phi}^{(t+1)} \in \mathbb{R}_+^{K^{(t)}\times K^{(t+1)}}$ are restricted to have unit L_1 norm. To complete the hierarchical model, we let $\boldsymbol{D}_k(:) \sim \mathrm{Dir}(\eta^{(1)}\boldsymbol{1}_{|V|F})$, $\boldsymbol{\phi}_k^{(t)} \sim \mathrm{Dir}(\eta^{(t)}\boldsymbol{1}_{K^{(t)}})$, $r_k \sim \mathrm{Gam}(1/K^{(T)}, 1)$, and $c_j^{(t+1)} \sim \mathrm{Gam}(e_0, 1/f_0)$.

Examining (3) shows CPGBN provides a probabilistic document-level pooling layer, which summarizes the content coefficients $\boldsymbol{w}_{jk}$ across all word positions into $\theta_{jk}^{(1)} = \sum_{s=1}^{S_j} w_{jks}$; the hierarchical structure after $\theta_{jk}^{(1)}$ can be flexibly modified according to the deep models (not restricted to GBN) to be combined with. The proposed pooling layer can be trained jointly with all the other layers, making it distinct from a usual one that often cuts off the message passing from deeper layers (Lee et al., 2009; Chen et al., 2013). We note using pooling on the first hidden layer is related to shallow text CNNs that use document-level pooling directly after a single convolutional layer (Kim, 2014; Johnson & Zhang, 2015a), which often contributes to improved efficiency (Boureau et al., 2010; Wang et al., 2010).

2.3. Convolutional Inference Network for CPGBN

To make our model both scalable to big corpora in training and fast in out-of-sample prediction, below we introduce a convolutional inference network, which will be used in hybrid MCMC/variational inference described in Section 3.2. Note the usual strategy of autoencoding variational inference is to construct an inference network to map the observations directly to their latent representations, and optimize the encoder and decoder by minimizing the negative evidence lower bound (ELBO) as $L_g = \sum_{j=1}^{J} L_g(\boldsymbol{X}_j)$, where

$$\begin{aligned}
L_g(\boldsymbol{X}_j) = &\textstyle\sum_{t=2}^{T} \mathbb{E}_Q\Big[\ln \frac{q(\boldsymbol{\theta}_j^{(t)}\,|\,-)}{p(\boldsymbol{\theta}_j^{(t)}\,|\,\boldsymbol{\Phi}^{(t+1)},\boldsymbol{\theta}_j^{(t+1)})}\Big]\\
&+ \textstyle\sum_{k=1}^{K^{(1)}} \sum_{s=1}^{S_j} \mathbb{E}_Q\Big[\ln \frac{q(w_{jks}\,|\,-)}{p(w_{jks}\,|\,\boldsymbol{\Phi}^{(2)},\boldsymbol{\theta}_j^{(2)})}\Big]\\
&- \mathbb{E}_Q[\ln p(\boldsymbol{X}_j \,|\, \{\boldsymbol{D}_k, \boldsymbol{w}_{jk}\}_{1,K^{(1)}})];
\end{aligned} \tag{4}$$

following Zhang et al. (2018), we use the Weibull distribution to approximate the gamma distributed conditional posterior of $\boldsymbol{\theta}_j^{(t)}$, as it is reparameterizable, resembles the gamma distribution, and the Kullback–Leibler (KL) divergence from the gamma to Weibull distributions is analytic; as in Fig. 1, we construct the autoencoding variational distribution as $Q = q(\boldsymbol{w}_{jk} \,|\, -) \prod_{t=2}^{T} q(\boldsymbol{\theta}_j^{(t)} \,|\, -)$, where

$$\begin{aligned} q(\boldsymbol{w}_{jk} \,|\, -) &= \text{Weibull}(\boldsymbol{\Sigma}_{jk}^{(1)} + \boldsymbol{\Phi}_{k:}^{(2)} \boldsymbol{\theta}_j^{(2)}, \boldsymbol{\Lambda}_{jk}^{(1)}), \\ q(\boldsymbol{\theta}_j^{(t)} \,|\, -) &= \text{Weibull}(\boldsymbol{\sigma}_j^{(t)} + \boldsymbol{\Phi}^{(t+1)} \boldsymbol{\theta}_j^{(t+1)}, \boldsymbol{\lambda}_j^{(t)}). \end{aligned} \tag{5}$$

The parameters $\boldsymbol{\Sigma}_j^{(1)}$, $\boldsymbol{\Lambda}_j^{(1)} \in \mathbb{R}^{K^{(1)} \times S_j}$ of $\boldsymbol{w}_j = (\boldsymbol{w}_{j1}, \ldots, \boldsymbol{w}_{jK^{(1)}})' \in \mathbb{R}^{K^{(1)} \times S_j}$ are deterministically transformed from the observation $\boldsymbol{X}_j$ using CNNs specified as

$$\begin{aligned} \boldsymbol{H}_j^{(1)} &= \text{relu}(\boldsymbol{C}_1^{(1)} * \boldsymbol{X}_j + \boldsymbol{b}_1^{(1)}), \\ \boldsymbol{\Sigma}_j^{(1)} &= \exp(\boldsymbol{C}_2^{(1)} * \text{pad}(\boldsymbol{H}_j^{(1)}) + \boldsymbol{b}_2^{(1)}), \\ \boldsymbol{\Lambda}_j^{(1)} &= \exp(\boldsymbol{C}_3^{(1)} * \text{pad}(\boldsymbol{H}_j^{(1)}) + \boldsymbol{b}_3^{(1)}), \end{aligned}$$

where $\boldsymbol{b}_1^{(1)}, \boldsymbol{b}_2^{(1)}, \boldsymbol{b}_3^{(1)} \in \mathbb{R}^{K^{(1)}}$, $\boldsymbol{C}_1^{(1)} \in \mathbb{R}^{K^{(1)} \times |V| \times F}$, $\boldsymbol{C}_2^{(1)}, \boldsymbol{C}_3^{(1)} \in \mathbb{R}^{K^{(1)} \times K^{(1)} \times F}$, $\boldsymbol{H}_j^{(1)} \in \mathbb{R}^{K^{(1)} \times S_j}$, and $\text{pad}(\boldsymbol{H}_j^{(1)}) \in \mathbb{R}^{K^{(1)} \times L_j}$ is obtained with zero-padding; the parameters $\boldsymbol{\sigma}_j^{(t)}$ and $\boldsymbol{\lambda}_j^{(t)}$ are transformed from $\boldsymbol{h}_j^{(1)} = \text{pool}(\boldsymbol{H}_j^{(1)})$ specified as

$$\begin{aligned} \boldsymbol{h}_j^{(t)} &= \text{relu}(\boldsymbol{U}_1^{(t)} \boldsymbol{h}_j^{(t-1)} + \boldsymbol{b}_1^{(t)}), \\ \boldsymbol{\sigma}_j^{(t)} &= \exp(\boldsymbol{U}_2^{(t)} \boldsymbol{h}_j^{(t)} + \boldsymbol{b}_2^{(t)}), \\ \boldsymbol{\lambda}_j^{(t)} &= \exp(\boldsymbol{U}_3^{(t)} \boldsymbol{h}_j^{(t)} + \boldsymbol{b}_3^{(t)}), \end{aligned}$$

where $\boldsymbol{b}_1^{(t)}, \boldsymbol{b}_2^{(t)}, \boldsymbol{b}_3^{(t)}, \boldsymbol{h}_j^{(t)} \in \mathbb{R}^{K^{(t)}}$, $\boldsymbol{U}_1^{(t)} \in \mathbb{R}^{K^{(t)} \times K^{(t-1)}}$, and $\boldsymbol{U}_2^{(t)}, \boldsymbol{U}_3^{(t)} \in \mathbb{R}^{K^{(t)} \times K^{(t)}}$ for $t \in \{2, ..., T\}$.

Further we develop sCPGBN, a supervised generalization of CPGBN, for text categorization tasks: by adding a softmax classifier on the concatenation of $\{\boldsymbol{\theta}_j^{(t)}\}_{1,T}$, the loss function of the entire framework is modified as

$$L = L_g + \xi L_c,$$

where L_c denotes the cross-entropy loss and ξ is used to balance generation and discrimination (Higgins et al., 2017).

3. Inference

Below we describe the key inference equations for CPFA shown in (1), a single hidden-layer version of CPGBN shown in (2), and provide more details in the Appendix. How the inference of CPFA, including Gibbs sampling and hybrid MCMC/autoencoding variational inference, is generalized to that of CPGBN is similar to how the inference of PFA is generalized to that of PGBN, as described in detail in Zhou et al. (2016) and Zhang et al. (2018) and omitted here for brevity.

3.1. Gibbs Sampling

Directly dealing with the whole matrix by expanding the convolution operation with Toeplitz conversion (Bojanczyk et al., 1995) provides a straightforward solution for the inference of convolutional models, which transforms each observation matrix $\boldsymbol{M}_j$ into a vector, on which the inference methods for sparse factor analysis (Carvalho et al., 2008; James et al., 2010) could then be applied. However, considering the sparsity of the document matrix consisting of one-hot vectors, directly processing these matrices without considering sparsity will bring unnecessary burden in computation and storage. Instead, we apply data augmentation under the Poisson likelihood (Zhou et al., 2012; 2016) to upward propagate latent count matrices $\boldsymbol{M}_j$ as

$$(\boldsymbol{M}_{j1}, ..., \boldsymbol{M}_{jK} \,|\, -) \sim \text{Multi}(\boldsymbol{M}_j; \boldsymbol{\zeta}_{j1}, ..., \boldsymbol{\zeta}_{jK}),$$

where $\boldsymbol{\zeta}_{jk} = (\boldsymbol{D}_k * \boldsymbol{w}_{jk}) / (\sum_{k=1}^{K} \boldsymbol{D}_k * \boldsymbol{w}_{jk})$. Note we only need to focus on nonzero elements of $\boldsymbol{M}_{jk} \in \mathbb{Z}^{|V| \times L_j}$. We rewrite the likelihood function by expanding the convolution operation along the dimension of $\boldsymbol{w}_{jk}$ as

$$m_{jkvl} \sim \text{Pois}(\textstyle\sum_{s=1}^{S_j} w_{jks} d_{kv(l-s+1)}),$$

where $d_{kv(l-s+1)} := 0$ if $l - s + 1 \notin \{1, 2, \ldots, F\}$. Thus each nonzero element m_{jkvl} could be augmented as

$$(\boldsymbol{m}_{jkvl} \,|\, m_{jkvl}) \sim \text{Multi}(m_{jkvl}; \delta_{jkvl1}, ..., \delta_{jkvlS_j}), \tag{6}$$

where $\delta_{jkvls} = w_{jks} d_{kv(l-s+1)} / \sum_{s=1}^{S_j} w_{jks} d_{kv(l-s+1)}$ and $\boldsymbol{m}_{jkvl} \in \mathbb{Z}^{S_j}$. We can now decouple $\boldsymbol{D}_k * \boldsymbol{w}_{jk}$ in (1) by marginalizing out $\boldsymbol{D}_k$, leading to

$$\boldsymbol{m}_{jk\cdot\cdot} \sim \text{Pois}(\boldsymbol{w}_{jk}).$$

where the symbol "$\cdot$" denotes summing over the corresponding index and hence $\boldsymbol{m}_{jk\cdot\cdot} = \sum_{v=1}^{|V|} \sum_{l=1}^{L_j} \boldsymbol{m}_{jkvl}$. Using the gamma-Poisson conjugacy, we have

$$(\boldsymbol{w}_{jk} \,|\, -) \sim \text{Gam}(m_{jk\cdot\cdot} + r_k, 1/(1 + c_j^{(2)})).$$

Similarly, we can expand the convolution along the other direction as $m_{jkvl} \sim \text{Pois}(\sum_{f=1}^{F} d_{kvf} w_{jk(l-f+1)})$, where $w_{jk(l-f+1)} := 0$ if $l - f + 1 \notin \{1, 2, \ldots, S_j\}$, and obtain $(\boldsymbol{d}_{jkvl} \,|\, m_{jkvl}) \sim \text{Multi}(m_{jkvl}; \xi_{jkvl1}, \ldots, \xi_{jkvlF})$, where $\xi_{jkvlf} = d_{kvf} w_{jk(l-f+1)} / \sum_{f=1}^{F} d_{kvf} w_{jk(l-f+1)}$ and $\boldsymbol{d}_{jkvl} \in \mathbb{Z}^F$. Further applying the relationship between the Poisson and multinomial distributions, we have

$$((\boldsymbol{d}'_{jk1\cdot}, \ldots, \boldsymbol{d}'_{jkV\cdot})' \,|\, m_{jk\cdot\cdot}) \sim \text{Multi}(m_{jk\cdot\cdot}; \boldsymbol{D}_k(:)).$$

With the Dirichlet-multinomial conjugacy, we have

$$(\boldsymbol{D}_k(:) \,|\, -) \sim \text{Dir}((\boldsymbol{d}'_{\cdot k1\cdot}, \ldots, \boldsymbol{d}'_{\cdot kV\cdot})' + \eta \boldsymbol{1}_{|V|F}).$$

Exploiting the properties of the Poisson and multinomial distributions helps CPFA fully take advantages of the sparsity of the one-hot vectors, making its complexity comparable to a regular bag-of-words topic model that uses Gibbs sampling for inference. Note as the multinomial related samplings inside each iteration are embarrassingly parallel, they are accelerated with GPUs in our experiments.

Algorithm 1 Hybrid stochastic-gradient MCMC and autoencoding variational inference for CPGBN

```
Set mini-batch size $m$ and number of dictionaries $K$;
Initialize encoder parameter $\boldsymbol{\Omega}$ and model parameter $\{\boldsymbol{D}_k\}_{1,K}$;
for $iter = 1, 2, \cdots$ do
    Randomly select a mini-batch of $m$ documents to form a subset $\boldsymbol{X} = \{\boldsymbol{X}_j\}_{1,m}$;
    for $j = 1, 2, \cdots$ do
        Draw random noise $\boldsymbol{\epsilon}_j$ from uniform distribution;
        Calculate $\nabla_{\boldsymbol{\Omega}} L(\boldsymbol{\Omega}, \boldsymbol{D}; \boldsymbol{x}_j, \boldsymbol{\epsilon}_j)$ according to (8) and update $\boldsymbol{\Omega}$;
    end for
    Sample $\{\boldsymbol{w}_j\}_{1,m}$ from (5) given $\boldsymbol{\Omega}$;
    Parallely process each positive point in $\boldsymbol{X}$ to obtain $(\boldsymbol{d}'_{\cdot k1\cdot}, \ldots, \boldsymbol{d}'_{\cdot k|V|\cdot})'$ according to (6);
    Update $\{\boldsymbol{D}_k\}_{1,K}$ according to (7)
end for
```

3.2. Hybrid MCMC/Variational Inference

While having closed-form update equations, the Gibbs sampler requires processing all documents in each iteration and hence has limited scalability. Fortunately, there have been several related research on scalable inference for discrete LVMs (Ma et al., 2015; Patterson & Teh, 2013). Specifically, TLASGR-MCMC of Cong et al., 2017, which uses an elegant simplex constraint and increases the sampling efficiency via the use of the Fisher information matrix (FIM), with adaptive step-sizes for the topics of different layers, can be naturally extended to our model. The efficient TLASGR-MCMC update of $\boldsymbol{D}_k$ in CPFA can be described as

$$\begin{aligned}\boldsymbol{D}_k^{(new)}(:) = \Big\{&\boldsymbol{D}_k(:) + \tfrac{\varepsilon_i}{M_k}[(\rho(\boldsymbol{d}'_{\sim k1\cdot}, ..., \boldsymbol{d}'_{\sim k|V|\cdot})' + \eta) \\ &-(\rho \boldsymbol{d}_{\sim k\cdot\cdot} + \eta\,|V|\,F)\boldsymbol{D}_k(:)] + N\big(0, \tfrac{2\varepsilon_i}{M_k}\mathrm{diag}(\boldsymbol{D}_k(:))\big)\Big\}_{\angle}, \end{aligned} \tag{7}$$

where i denotes the number of mini-batches processed so far; the symbol $\sim$ in the subscript denotes summing over the data in a mini-batch; and the definitions of ρ, ε_i, $\{\cdot\}_{\angle}$, and M_k are analogous to these in Cong et al. (2017) and omitted here for brevity.

Similar to Zhang et al. (2018), combining TLASGR-MCMC and the convolutional inference network described in Section 2.3, we can construct a hybrid stochastic-gradient-MCMC/autoencoding variational inference for CPFA. More specifically, in mini-batch based each iteration, we draw a random sample of the CPFA global parameters $\boldsymbol{D} = \{\boldsymbol{D}_k\}_{1,K}$ via TLASGR-MCMC; given the sampled global parameters, we optimize the parameters of the convolutional inference network, denoted as $\boldsymbol{\Omega}$, using the $-$ELBO in (4), which for CPFA is simplified as

$$\begin{aligned}L_g = -\textstyle\sum_{j=1}^{J} \mathbb{E}_{q(\boldsymbol{w}_j \,|\, \boldsymbol{X}_j)}[\ln p(\boldsymbol{X}_j \,|\, \{\boldsymbol{D}_k, \boldsymbol{w}_{jk}\}_{1,K})] \\ +\textstyle\sum_{j=1}^{J} \mathbb{E}_{q(\boldsymbol{w}_j \,|\, \boldsymbol{x}_j)}[\ln \tfrac{q(\boldsymbol{w}_j \,|\, \boldsymbol{x}_j)}{p(\boldsymbol{w}_j)}].\end{aligned} \tag{8}$$

We describe the proposed hybrid stochastic-gradient-MCMC/autoencoding variational inference algorithm in Algorithm 1, which is implemented in TensorFlow (Abadi et al., 2016), combined with pyCUDA (Klockner et al., 2012) for more efficient computation.

4. Related Work

With the bag-of-words representation that ignores the word order information, a diverse set of deep topic models have been proposed to infer a multilayer data representation in an unsupervised manner. A main mechanism of them is to connect adjacent layers by specific factorization, which usually boosts the performance (Gan et al., 2015; Zhou et al., 2016; Zhang et al., 2018). However, limited by the bag-of-words representation, they usually perform poorly on sentiment analysis tasks, which heavily rely on the word order information (Xu & Sarikaya, 2013; Weston et al., 2014). In this paper, the proposed CPGBN could be seen as a novel convolutional extension, which not only clearly remedies the loss of word order, but also inherits various virtues of deep topic models.

Benefiting from the advance of word-embedding methods, CNN-based architectures have been leveraged as encoders for various natural language processing tasks (Kim, 2014; Kalchbrenner et al., 2014). They in general directly apply to the word embedding layer a single convolution layer, which, given a convolution filter window of size n, essentially acts as a detector of typical n-grams. More complex deep neural networks taking CNNs as the their encoder and RNNs as decoder have also been studied for text generation (Zhang et al., 2016; Semeniuta et al., 2017). However, for unsupervised sentence modeling, language decoders other than RNNs are less well studied; it was not until recently that Zhang et al. (2017) have proposed a simple yet powerful, purely convolutional framework for unsupervisedly learning sentence representations, which is the first to force the encoded latent representation to capture the information from the entire sentence via a multi-layer CNN specification. But there still exists a limitation in requiring an additional large corpus for training word embeddings, and it is also difficult to visualize and explain the semantic meanings learned by black-box deep networks.

For text categorization, the bi-grams (or a combination of bi-grams and unigrams) are confirmed to provide more discriminative power than unigrams (Tan et al., 2002; Glorot et al., 2011). Motivated by this observation, Johnson & Zhang (2015a) tackle document categorization tasks by directly applying shallow CNNs, with filter width three, on one-hot encoding document matrices, outperforming both traditional n-grams and word-embedding based methods without the aid of additional training data. In addition, the shallow CNN serves as an important building block in many other supervised applications to help achieve sate-of-art results (Johnson & Zhang, 2015b; 2017).

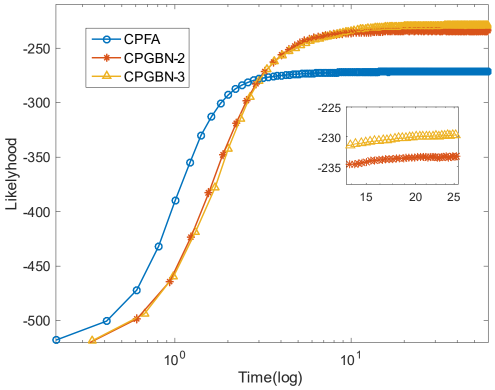

Figure 2. Point likelihood of CPGBNs on TREC as a function of time with various structural settings.

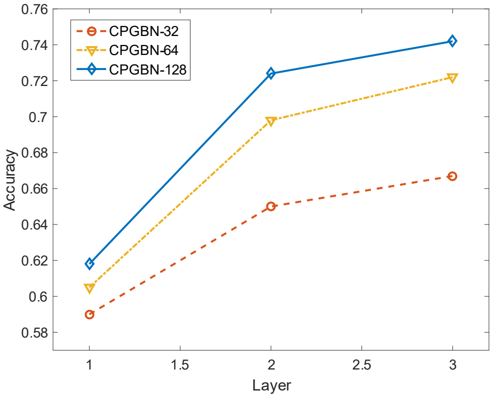

Figure 3. Classification accuracy (%) of the CPGBNs on TREC as a function of the depth with various structural settings.

5. Experimental Results

5.1. Datasets and Preprocessing

We test the proposed CPGBN and its supervised extension (sCPGBN) on various benchmarks, including:

- **MR**: Movie reviews with one sentence per review, where the task is to classify a review as being positive or negative (Pang & Lee, 2005).

- **TREC**: TREC question dataset, where the task is to classify a question into one of six question types (whether the question is about abbreviation, entity, description, human, location, or numeric) (Li & Roth, 2002).

- **SUBJ**: Subjectivity dataset, where the task is to classify a sentence as being subjective or objective (Pang & Lee, 2004).

- **ELEC**: ELEC dataset (Mcauley & Leskovec, 2013) consists of electronic product reviews, which is part of a large Amazon review dataset.

- **IMDB**: IMDB dataset (Maas et al., 2011) is a benchmark dataset for sentiment analysis, where the task is to determine whether a movie review is positive or negative.

Table 1. Summary statistics for the datasets after tokenization (C: Number of target classes. L: Average sentence length. N: Dataset size. V: Vocabulary size. V_{pre}: Number of words present in the set of pre-trained word vectors. $Test$: Test set size, where CV means 10-fold cross validation).

Data	MR	TREC	SUBJ	ELEC	IMDB
C	2	6	2	2	2
L	20	10	23	123	266
N	10662	5952	10000	50000	50000
V	20277	8678	22636	52248	95212
V_{pre}	20000	8000	20000	30000	30000
$Test$	CV	500	CV	25000	25000

We follow the steps listed in Johnson & Zhang (2015a) to tokenize the text, where emojis such as ":-)" are treated as tokens and all the characters are converted to lower case. We then select the top V_{pre} most frequent words to construct the vocabulary, without dropping stopwords; we map the words not included in the vocabulary to a same special token to keep all sentences structurally intact. The summary statistics of all benchmark datasets are listed in Table 1.

5.2. Inference Efficiency

In this section we show the results of the proposed CPGBN on TREC. First, to demonstrate the advantages of increasing the depth of the network, we construct three networks of different depths: with $(K = 32)$ for $T = 1$, $(K_1 = 32, K_2 = 16)$ for $T = 2$, and $(K_1 = 32, K_2 = 16, K_3 = 8)$ for $T = 3$. Under the same configuration of filter width $F = 3$ and the same hyperparameter setting, where $e_0 = f_0 = 0.1$ and $\eta^{(t)} = 0.05$, the networks are trained with the proposed Gibbs sampler. The trace plots of model likelihoods are shown in Fig. 2. It is worth noting that increasing the network depth in general improves the quality of data fitting, but as the complexity of the model increases, the model tends to converge more slowly in time.

Considering that the data fitting and generation ability is not necessarily strongly correlated the performance on specific tasks, we evaluate the proposed models on document classification. Using the same experimental settings as mentioned above, we investigate how the classification accuracy is impacted by the network structure. On each network, we apply the Gibbs sampler to collect 200 MCMC samples after 500 burn-ins to estimate the posterior mean of the feature usage weight vector $\boldsymbol{w}_j$, for every document in both the training and testing sets. A linear support vector machine (SVM) (Cortes & Vapnik, 1995) is taken as the classifier on the first hidden layer, denoted as $\boldsymbol{\theta}_j^{(1)}$ in (2), to make a fair comparison, where each result listed in Table 2 is the

Table 2. Comparison of classification accuracy on unsupervisedly extracted feature vectors and average training time (seconds per Gibbs sampling iteration across all documents) on three different datasets.

Model	Size	Accuracy			Time		
		MR	TREC	SUBJ	MR	TREC	SUBJ
LDA	200	54.4±0.8	45.5±1.9	68.2±1.3	3.93	0.92	3.81
DocNADE	200	54.2±0.8	62.0±0.6	72.9±1.2	-	-	-
DPFA	200	55.2±1.2	51.4±0.9	74.5±1.9	6.61	1.88	6.53
DPFA	200-100	55.4±0.9	52.0±0.6	74.4±1.5	6.74	1.92	6.62
DPFA	200-100-50	56.1±0.9	62.0±0.6	78.5±1.4	6.92	1.95	6.80
PGBN	200	56.3±0.6	66.7±1.8	76.2±0.9	3.97	1.01	3.56
PGBN	200-100	56.7±0.8	67.3±1.7	77.3±1.3	5.09	1.72	4.39
PGBN	200-100-50	57.0±0.5	67.9±1.5	78.3±1.2	5.67	1.87	4.91
WHAI	200	55.6±0.8	60.4±1.9	75.4±1.5	-	-	-
WHAI	200-100	56.2±1.0	63.5±1.8	76.0±1.4	-	-	-
WHAI	200-100-50	56.4±0.6	65.6±1.7	76.5±1.1	-	-	-
CPGBN	200	61.5±0.8	68.4±0.8	77.4±0.8	3.58	0.98	3.53
CPGBN	200-100	62.4±0.7	73.4±0.8	81.2±0.8	8.19	1.99	6.56
CPGBN	200-100-50	**63.6**±0.8	**74.4**±0.6	**81.5**±0.6	10.44	2.59	7.87

Table 3. Example phrases learned from TREC by CPGBN.

Kernel Index	Visualized Topic			Visualized Phrase
	1st Column	2nd Column	3rd Column	
192th Kernel	**how** cocktail stadium run	do many much long	you years miles degrees	**how** do you, **how** many years, **how** much degrees
80th Kernel	microsoft virtual answers.com softball	e-mail email ip brothers	address addresses floods score	microsoft e-mail address, microsoft email address, virtual ip address
177th Kernel	**who** willy bar hydrogen	created wrote fired are	maria angela snoopy caesar	**who** created snoopy, **who** fired caesar, **who** wrote angela
47th Kernel	dist all-time wheel saltpepter	**how** stock 1976 westview	far high tall exchange	dist **how** far, dist **how** high , dist **how** tall

average accuracy of five independent runs. Fig. 3 shows a clear trend of improvement in classification accuracy, by increasing the network depth given a limited first-layer width, or by increasing the hidden-layer width given a fixed depth.

5.3. Unsupervised Models

In our second set of experiments, we evaluate the performance of different unsupervised algorithms on MR, TREC, and SUBJ datasets by comparing the discriminative ability of their unsupervisely extracted latent features. We consider LDA (Blei et al., 2003) and its deep extensions, including DPFA (Gan et al., 2015) and PGBN (Zhou et al., 2016), which are trained with batch Gibbs sampling. We also consider WHAI (Zhang et al., 2018) and DocNADE (Lauly et al., 2017) that are trained with stochastic gradient descent.

To make a fair comparison, we let CPGBNs to have the same hidden layer widths as the other methods, and set the filter width as 3 for the convolutional layer. Listed in Table 2 are the results of various algorithms, where the means and error bars are obtained from five independent runs, using the code provided by the original authors. For all batch learning algorithms, we also report in Table 2 their average run time for an epoch ($i.e.$, processing all training documents once). Clearly, given the same generative network structure, CPGBN performs the best in terms of classification accuracy, which can be attributed to its ability to utilize the word order information. The performance of CPGBN has a clear trend of improvement as the generative network becomes deeper, which is also observed on other deep generative models including DPFA, PGBN, and WHAI. In terms of running time, the shallow LDA could be the most efficient model compared to these more sophisticated ones, while CPGBN of a single layer achieves a comparable effectiveness thanks to its efficient use of GPU for parallelizing its computation inside each iteration. Note all running times are reported based on a Nvidia GTX 1080Ti GPU.

In addition to quantitative evaluations, we have also visually inspected the inferred convolutional kernels of CPGBN, which is distinct from many existing convolutional models that build nonlinearity via "black-box" neural networks. As shown in Table 3, we list several convolutional kernel elements of filter width 3 learned from TREC, using a single-hidden-layer CPGBN of size 200. We exhibit the top 4 most probable words in each column of the corresponding kernel element. It's particularly interesting to note that the words in different columns can be combined into a variety of interpretable phrases with similar semantics. CPGBN explicitly take the word order information into consideration to extract phrases, which are then combined into a hierarchy of phrase-level topics, helping clearly improve the quality of unsupervisedly extracted features. Take the 177th convolutional kernel for example, the top word of its 1st topic is "who," its 2nd topic is a verb topic: "created, wrote, fired, are," while its 3rd topic is a noun topic: "maria/angela/snoopy/caesar." These word-level topics can be combined to construct phrases such as "who, created/ wrote/ fired/are, maria/angela/snoopy/caesar," resulting in a phrase-level topic about "human," one of the six types of questions in TREC. Note these shallow phrase-level topics will become more general in a deeper layer of CPGBN. We provide two example phrase-level topic hierarchies in the Appendix to enhance interpretability.

5.4. Supervised Models

Table 4 lists the comparison of various supervised algorithms on three common benchmarks, including SUBJ, ELEC, and IMDB. The results listed there are either quoted from published papers, or reproduced with the code provided by the original authors. We consider bag-of-words representation based supervised topic models, including sAVITM (Srivastava & Sutton, 2017), MedLDA (Zhu et al., 2014), and sWHAI (Zhang et al., 2018). We also consider three types of bag-of-n-gram models (Johnson & Zhang, 2015a), where $n \in \{1, 2, 3\}$, and word embedding based methods, indicated with suffix "-wv," including SVM-wv (Zhang & Wallace, 2017) and RNN-wv and LSTM-wv (Johnson & Zhang, 2016). In addition, we consider several related CNN based methods, including three different variants of Text CNN (Kim, 2014)—CNN-rand, CNN-static, and CNN-non-static—and CNN-one-hot (Johnson & Zhang, 2015a) that is based on one-hot encoding.

We construct three different sCPGBNs with $T \in \{1, 2, 3\}$, as described in Section 2.3. As shown in Table 4, the word embedding based methods generally outperform the methods based on bag-of-words, which is not surprising as the latter completely ignore word order. Among all bag-of-words representation based methods, sWHAI performs the best and even achieves comparable performance to some word-embedding based methods, which illustrates the benefits of having multi-stochastic-layer latent representations.

Table 4. Comparison of classification accuracy on supervised feature extraction tasks on three different datasets.

Model	SUBJ	ELEC	IMDB
sAVITM (Srivastava & Sutton, 2017)	85.7	83.7	84.9
MedLDA (Zhu et al., 2014)	86.5	84.6	85.7
sWHAI-layer1 (Zhang et al., 2018)	90.6	86.8	87.2
sWHAI-layer2 (Zhang et al., 2018)	91.7	87.5	88.0
sWHAI-layer3 (Zhang et al., 2018)	92.0	87.8	88.2
SVM-unigrams (Tan et al., 2002)	88.5	86.3	87.7
SVM-bigrams (Tan et al., 2002)	89.4	87.2	88.2
SVM-trigrams (Tan et al., 2002)	89.7	87.4	88.5
SVM-wv (Zhang & Wallace, 2017)	90.1	85.9	86.5
RNN-wv (Johnson & Zhang, 2016)	88.9	87.5	88.3
LSTM-wv (Johnson & Zhang, 2016)	89.8	88.3	89.0
CNN-rand (Kim, 2014)	89.6	86.8	86.3
CNN-static (Kim, 2014)	93.0	87.8	88.9
CNN-non-static (Kim, 2014)	93.4	88.6	89.5
CNN-one-hot (Johnson & Zhang, 2015a)	91.1	91.3	91.6
sCPGBN-layer1	93.4±0.1	91.6±0.3	91.8±0.3
sCPGBN-layer2	93.7±0.1	92.0±0.2	92.4±0.2
sCPGBN-layer3	**93.8**±0.1	**92.2**±0.2	**92.6**±0.2

As for n-grams based models, although they achieve comparable performance to word-embedding based methods, we find via experiments that both their performance and computation are sensitive to the vocabulary size. Among the CNN related algorithms, CNN-one-hot tends to have a better performance on classifying longer texts than Text CNN does, which agrees with the observations of Zhang & Wallace (2017); possible explanation for this phenomenon is that CNN-one-hot is prone to overfitting on short documents. Moving beyond CNN-one-hot, sCPGBN could help capture the underlying high-order statistics to alleviate overfitting, as commonly observed in deep generative models (DGMs) (Li et al., 2015), and improves its performance by increasing its number of stochastic hidden layers.

6. Conclusion

We propose convolutional Poisson factor analysis (CPFA), a hierarchical Bayesian model that represents each word in a document as a one-hot vector, and captures the word order information by performing convolution on sequentially ordered one-hot word vectors. By developing a principled document-level stochastic pooling layer, we further couple CPFA with a multi-stochastic-layer deep topic model to construct convolutional Poisson gamma belief network (CPGBN). We develop a Gibbs sampler to jointly train all the layers of CPGBN. For more scalable training and fast testing, we further introduce a mini-batch based stochastic inference algorithm that combines both stochastic-gradient MCMC and a Weibull distribution based convolutional variational auto-encoder. In addition, we provide a supervised extension of CPGBN. Example results on both unsupervised and supervised feature extraction tasks show CPGBN combines the virtues of both convolutional operations and deep topic models, providing not only state-of-the-art classification performance, but also highly interpretable phrase-level deep latent representations.

Acknowledgements

B. Chen acknowledges the support of the Program for Young Thousand Talent by Chinese Central Government, the 111 Project (No. B18039), NSFC (61771361), NSFC for Distinguished Young Scholars (61525105), and the Innovation Fund of Xidian University. M. Zhou acknowledges the support of Award IIS-1812699 from the U.S. National Science Foundation and the McCombs Research Excellence Grant.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent Dirichlet allocation. *JMLR*, 3:993–1022, 2003.

Bojanczyk, A. W., Brent, R. P., De Hoog, F. R., and Sweet, D. R. On the stability of the bareiss and related toeplitz factorization algorithms. *SIAM Journal on Matrix Analysis and Applications*, 16(1):40–57, 1995.

Boureau, Y., Ponce, J., and Lecun, Y. A theoretical analysis of feature pooling in visual recognition. In *ICML*, pp. 111–118, 2010.

Carvalho, C. M., Chang, J. T., Lucas, J. E., Nevins, J. R., Wang, Q., and West, M. High-dimensional sparse factor modeling: Applications in gene expression genomics. *J. Amer. Statist. Assoc.*, 103(484):1438–1456, 2008.

Chen, B., Polatkan, G., Sapiro, G., Blei, D. M., Dunson, D. B., and Carin, L. Deep learning with hierarchical convolutional factor analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1887–1901, 2013.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Cong, Y., Chen, B., Liu, H., and Zhou, M. Deep latent Dirichlet allocation with topic-layer-adaptive stochastic gradient Riemannian MCMC. In *ICML*, pp. 864–873, 2017.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. Indexing by latent semantic analysis. *J. Amer. Soc. Inf. Sci.*, 1990.

Gan, Z., Chen, C., Henao, R., Carlson, D. E., and Carin, L. Scalable deep Poisson factor analysis for topic modeling. In *ICML*, pp. 1823–1832, 2015.

Glorot, X., Bordes, A., and Bengio, Y. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pp. 513–520, 2011.

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, volume 3, 2017.

Hinton, G. E. and Salakhutdinov, R. Replicated softmax: An undirected topic model. In *NIPS*, pp. 1607–1614, 2009.

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

James, G. M., Sabatti, C., Zhou, N., and Zhu, J. Sparse regulatory networks. *AOAS*, 4(2):663–686, 2010.

Johnson, R. and Zhang, T. Effective use of word order for text categorization with convolutional neural networks. *NAACL*, pp. 103–112, 2015a.

Johnson, R. and Zhang, T. Semi-supervised convolutional neural networks for text categorization via region embedding. In *NIPS*, pp. 919–927, 2015b.

Johnson, R. and Zhang, T. Supervised and semi-supervised text categorization using LSTM for region embeddings. *ICML*, pp. 526–534, 2016.

Johnson, R. and Zhang, T. Deep pyramid convolutional neural networks for text categorization. In *ACL*, pp. 562–570, 2017.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. A convolutional neural network for modelling sentences. In *ACL*, pp. 655–665, 2014.

Kim, Y. Convolutional neural networks for sentence classification. In *EMNLP*, pp. 1746–1751, 2014.

Klockner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., and Fasih, A. R. Pycuda and pyopencl: A scripting-based approach to gpu run-time code generation. *Parallel computing*, 38(3):157–174, 2012.

Lauly, S., Zheng, Y., Allauzen, A., and Larochelle, H. Document neural autoregressive distribution estimation. *JMLR*, 18(113):1–24, 2017.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, D. D. and Seung, H. S. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

Lee, H., Pham, P. T., Largman, Y., and Ng, A. Y. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, pp. 1096–1104, 2009.

Li, C., Zhu, J., Shi, T., and Zhang, B. Max-margin deep generative models. In *NIPS*, pp. 1837–1845, 2015.

Li, X. and Roth, D. Learning question classifiers. In *International Conference on Computational Linguistics*, pp. 1–7, 2002.

Ma, Y. A., Chen, T., and Fox, E. B. A complete recipe for stochastic gradient mcmc. In *NIPS*, pp. 2917–2925, 2015.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *ACL*, pp. 142–150, 2011.

Mcauley, J. and Leskovec, J. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM RecSys*, pp. 165–172, 2013.

Miao, X., Zhen, X., Liu, X., Deng, C., Athitsos, V., and Huang, H. Direct shape regression networks for end-to-end face alignment. In *CVPR*, pp. 5040–5049, 2018.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *NIPS*, pp. 3111–3119, 2013.

Min, S., Chen, X., Zha, Z., Wu, F., and Zhang, Y. A two-stream mutual attention network for semi-supervised biomedical segmentation with noisy labels. *AAAI*, 2019.

Pang, B. and Lee, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, pp. 271–278, 2004.

Pang, B. and Lee, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pp. 115–124, 2005.

Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up? sentiment classification using machine learning techniques. In *ACL*, pp. 79–86, 2002.

Papadimitriou, C., Raghavan, P., Tamaki, H., and Vempala, S. Latent semantic indexing: A probabilistic analysis. *J. Computer and System Sci.*, 2000.

Patterson, S. and Teh, Y. W. Stochastic gradient riemannian langevin dynamics on the probability simplex. In *NIPS*, pp. 3102–3110, 2013.

Ranganath, R., Tang, L., Charlin, L., and Blei, D. Deep exponential families. In *AISTATS*, pp. 762–771, 2015.

Semeniuta, S., Severyn, A., and Barth, E. A hybrid convolutional variational autoencoder for text generation. *EMNLP*, pp. 627–637, 2017.

Srivastava, A. and Sutton, C. A. Autoencoding variational inference for topic models. *ICLR*, 2017.

Tan, C., Wang, Y., and Lee, C. The use of bigrams to enhance text categorization. *Information Processing and Management*, 38(4):529–546, 2002.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pp. 1555–1565, 2014.

Wang, C., Chen, B., and Zhou, M. Multimodal Poisson gamma belief network. In *AAAI*, 2018.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T. S., and Gong, Y. Locality-constrained linear coding for image classification. In *CVPR*, pp. 3360–3367, 2010.

Weston, J., Chopra, S., and Adams, K. Tagspace: Semantic embeddings from hashtags. In *EMNLP*, pp. 1822–1827, 2014.

Xu, P. and Sarikaya, R. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 78–83, 2013.

Zhang, H., Chen, B., Guo, D., and Zhou, M. WHAI: Weibull hybrid autoencoding inference for deep topic modeling. *ICLR*, 2018.

Zhang, Y. and Wallace, B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *IJCNLP*, pp. 253–263, 2017.

Zhang, Y., Gan, Z., and Carin, L. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, volume 21, 2016.

Zhang, Y., Shen, D., Wang, G., Gan, Z., Henao, R., and Carin, L. Deconvolutional paragraph representation learning. In *NIPS*, pp. 4169–4179, 2017.

Zhou, M. Infinite edge partition models for overlapping community detection and link prediction. In *AISTATS*, pp. 1135–1143, 2015.

Zhou, M. and Carin, L. Negative binomial process count and mixture modeling. *arXiv preprint arXiv:1209.3442v1*, 2012.

Zhou, M., Hannah, L., Dunson, D., and Carin, L. Beta-negative binomial process and Poisson factor analysis. In *AISTATS*, pp. 1462–1471, 2012.

Zhou, M., Cong, Y., and Chen, B. Augmentable gamma belief networks. *JMLR*, 17(1):5656–5699, 2016.

Zhu, J., Chen, N., Perkins, H., and Zhang, B. Gibbs max-margin topic models with data augmentation. *JMLR*, 15 (1):1073–1110, 2014.

A. Inference for CPGBN

Here we describe the derivation in detail for convolutional Poisson gamma belief network (CPGBN) with T hidden layers, expressed as

$$\begin{aligned}
&\boldsymbol{\theta}_j^{(T)} \sim \text{Gam}(\boldsymbol{r}, 1/c_j^{(T+1)}),\\
&...,\\
&\boldsymbol{\theta}_j^{(t)} \sim \text{Gam}(\boldsymbol{\Phi}^{(t+1)}\boldsymbol{\theta}_j^{(t+1)}, 1/c_j^{(t+1)}),\\
&...,\\
&\boldsymbol{\theta}_j^{(1)} \sim \text{Gam}(\boldsymbol{\Phi}^{(2)}\boldsymbol{\theta}_j^{(2)}, 1/c_j^{(2)}),\\
&\boldsymbol{w}_{jk} = \boldsymbol{\pi}_{jk}\theta_{jk}^{(1)},\ \boldsymbol{\pi}_{jk} \sim \text{Dir}\big(\boldsymbol{\Phi}_{k:}^{(2)}\boldsymbol{\theta}_j^{(2)}/S_j \mathbf{1}_{S_j}\big),\\
&\boldsymbol{M}_j \sim \text{Pois}\big(\textstyle\sum_{k=1}^{K^{(1)}} \boldsymbol{D}_k * \boldsymbol{w}_{jk}\big).
\end{aligned} \tag{9}$$

Note using the relationship between the gamma and Dirichlet distributions ($e.g.$, Lemma IV.3 of Zhou & Carin (2012)), the elements of $\boldsymbol{w}_{jk}$ in the first hidden layer can be equivalently generated as

$$w_{jks} \sim \text{Gam}\big(\boldsymbol{\Phi}_{k:}^{(2)}\boldsymbol{\theta}_j^{(2)}/S_j, 1/c_j^{(2)}\big),\ s = 1, \ldots, S_j. \tag{10}$$

Note the random variable $\theta_{jk}^{(1)}$, which pools the random weights of all words in document j, follows

$$\theta_{jk}^{(1)} = \textstyle\sum_{s=1}^{S_j} w_{jks} \sim \text{Gam}(\boldsymbol{\Phi}_{k:}^{(2)}\boldsymbol{\theta}_j^{(2)}, 1/c_j^{(2)}). \tag{11}$$

As described in Section 3.1, we have

$$\boldsymbol{m}_{jk\cdot\cdot} \sim \text{Pois}(\boldsymbol{w}_{jk}), \tag{12}$$

$$((\boldsymbol{d}'_{jk1\cdot}, \ldots, \boldsymbol{d}'_{jkV\cdot})' \,|\, m_{jk\cdot\cdot}) \sim \text{Multi}(m_{jk\cdot\cdot}; \boldsymbol{D}_k(:)) \tag{13}$$

leading to the following conditional posteriors:

$$\begin{aligned}
(\boldsymbol{w}_{jk} \,|\, -) &\sim \text{Gam}(\boldsymbol{m}_{jk\cdot\cdot} + r_k, 1/(1 + c_j^{(2)})).\\
(\boldsymbol{D}_k(:) \,|\, -) &\sim \text{Dir}((\boldsymbol{d}'_{\cdot k1\cdot}, \ldots, \boldsymbol{d}'_{\cdot kV\cdot})' + \eta\mathbf{1}_{|V|F}).
\end{aligned} \tag{14}$$

Since $\boldsymbol{w}_{jk} = \boldsymbol{\pi}_{jk}\theta_{jk}^{(1)}$, from (12) we have

$$m_{jk\cdot\cdot s} \sim \text{Pois}(\pi_{jks}\theta_{jk}^{(1)}), \tag{15}$$

$$m_{jk\cdot\cdot\cdot} \sim \text{Pois}(\theta_{jk}^{(1)}). \tag{16}$$

Since $\sum_{s=1}^{S_j} \pi_{jks} = 1$ by construction, we have

$$(\boldsymbol{m}_{jk\cdot\cdot} \,|\, m_{jk\cdot\cdot\cdot}) \sim \text{Multi}(m_{jk\cdot\cdot\cdot}; \boldsymbol{\pi}_{jk}), \tag{17}$$

and hence the following conditional posteriors:

$$(\theta_{jk}^{(1)} \,|\, -) \sim \text{Gam}(m_{jk\cdot\cdot\cdot} + \boldsymbol{\Phi}_{k:}^{(2)}\boldsymbol{\theta}_j^{(2)}, 1/(1 + c_j^{(2)})), \tag{18}$$

$$(\boldsymbol{\pi}_{jk}|-) \sim \text{Dir}(\boldsymbol{m}_{jk\cdot\cdot}/m_{jk\cdot\cdot\cdot} + \boldsymbol{\Phi}_{k:}^{(2)}\boldsymbol{\theta}_j^{(2)}/S_j), \tag{19}$$

$$(c_j^{(2)}|-) \sim \text{Gam}\big(\textstyle\sum_{k=1}^{K^{(1)}} \boldsymbol{\Phi}_{k:}^{(2)}\boldsymbol{\theta}_j^{(2)} + a_0, 1/(\sum_{k=1}^{K^{(1)}} \theta_{jk}^{(1)} + b_0)\big). \tag{20}$$

The derivation for the parameters of layer $t \in \{2, ..., T\}$ is the same as that of gamma belief network (GBN) (Zhou et al., 2016), omitted here for brevity.

B. Sensitivity to Filter Width

To investigate the effect of the filter width of the convolutional kernel, we have evaluated the performance of CPFA ($i.e.$, CPGBN with a single hidden layer) on the SUBJ dataset with a variety of filter widths (unsupervised feature extraction + linear SVM for classification). We use the same CPFA code but vary its setting of the filter width. Averaging over five independent runs, the accuracy for filter wdith 1, 2, 3, 4, 5, 6, and 7 are 74.9 ± 0.9, 77.3 ± 0.4, 77.5 ± 0.5, 77.8 ± 0.4, 77.6 ± 0.5, 78.0 ± 0.4, and 77.5 ± 0.4, respectively. Note when the filter width reduces to 1, CPFA reduces to PFA ($i.e.$, no convolution). These results suggest the performance of CPFA has low sensitivity to the filter width. While setting the filter width as three may not be the optimal choice, it is a common practice for existing text CNNs (Kim, 2014; Johnson & Zhang, 2015a).

C. Hierarchical Visualization

Distinct from word-level topics learned by traditional topic models (Deerwester et al., 1990; Papadimitriou et al., 2000; Lee & Seung, 2001; Blei et al., 2003; Hinton & Salakhutdinov, 2009; Zhou et al., 2012), we propose novel phrase-level topics preserving word order as shown in Table. 3, where each phrase-level topic is often combined with several frequently co-occurred short phrases. To explore the connections between phrase-level topics of different layers learned by CPGBN, we follow Zhou et al. (2016) to construct trees to understand the general and specific aspects of the corpus. More specifically, we construct trees learned from TREC dataset, with the network structure set as $[K^{(1)}, K^{(2)}, K^{(3)}] = [200, 100, 50]$. We pick a node at the top layer as the root of a tree and grow the tree downward by drawing a line from node k at layer t to the top M relevant nodes k' at layer $t-1$.

As shown in Fig. 5, we select the top 3 relevant nodes at the second layer linked to the selected root node, and the top 2 relevant nodes at the third layer linked to the selected nodes at the second layer. Considering the TREC corpus only consists of questions (questions about abbreviation, entity, description, human, location, or numeric), most of the topics learned by CPGBN are focused on short phrases on asking specific questions, as shown in Table. 3. Following the branches of the tree in Fig. 5, the root node covers very general question types on "how many, how long, what, when, why," and it is clear that the topics become more and more specific when moving along the tree from the top to bottom, where the shallow topics of the first layer tend to focus on a single question type, $e.g.$, the $183th$ bottom-layer node queries "how many" and the $88th$ one queries "how long."

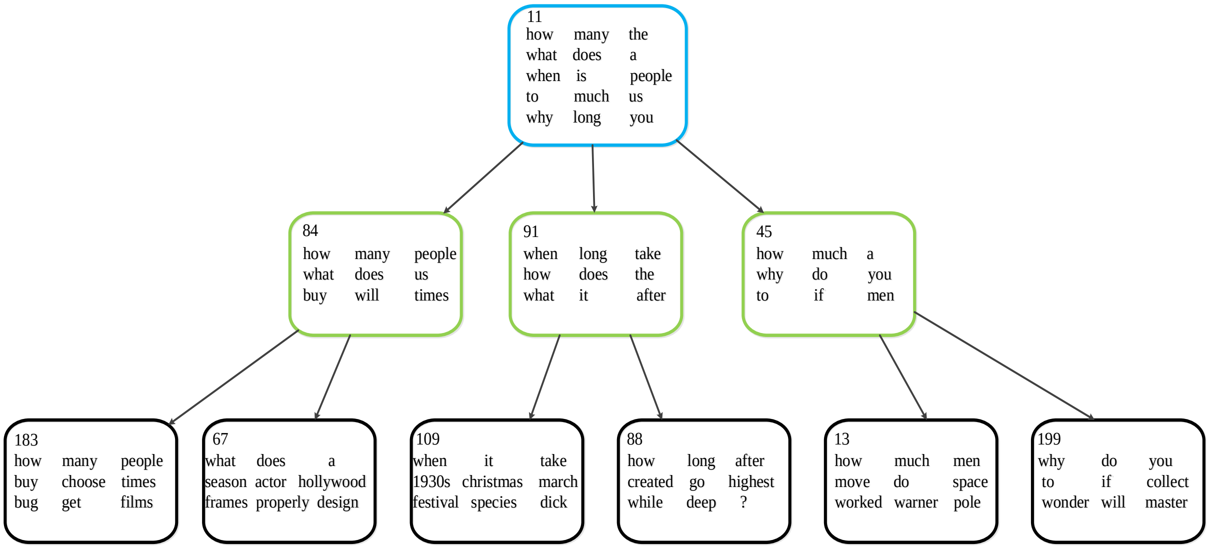

Figure 4. The $[6, 3, 1]$ phrase-level tree that includes all the lower-layer nodes (directly or indirectly) linked to the $11th$ node of the top layer, taken from the full $[200, 100, 50]$ network inferred by CPGBN on TREC dataset.

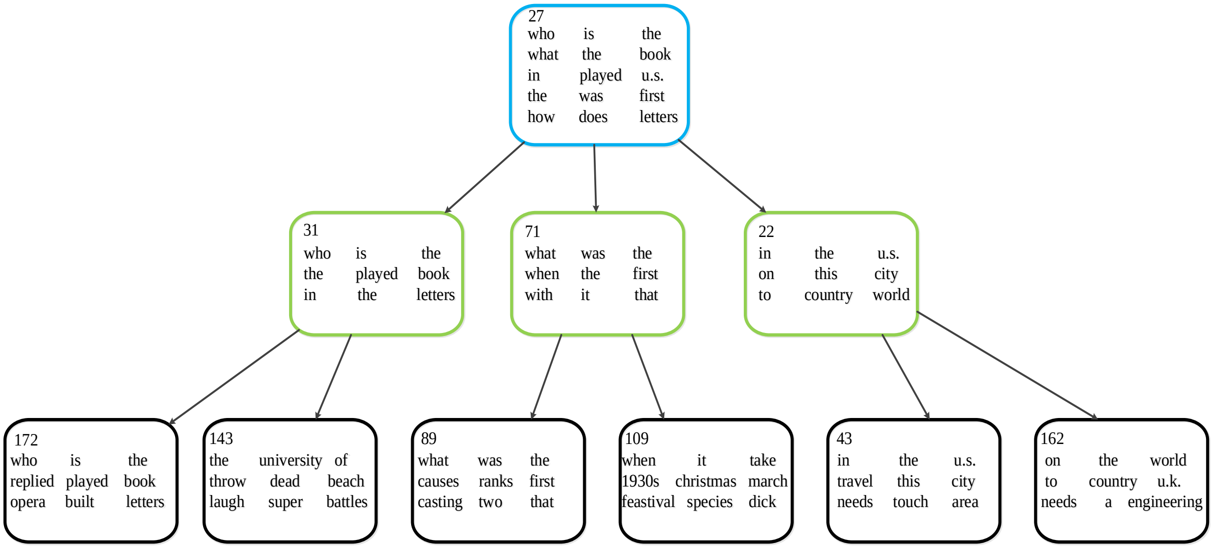

Figure 5. The $[6, 3, 1]$ phrase-level tree that include all the lower-layer nodes (directly or indirectly) linked to the $27th$ node of the top layer, taken from the full $[200, 100, 50]$ network inferred by CPGBN on TREC dataset.