SGVB Topic Modeling

by Otto Fabius

5619858

Supervised by Max Welling

A master's thesis for MSc in Artificial Intelligence

Track: Learning Systems

UNIVERSITEIT VAN AMSTERDAM

University of Amsterdam

the Netherlands

May 17th 2017

Abstract

Latent Dirichlet Allocation (LDA), and more recently Deep Exponential Families (DEF), are effective methods for performing inference on bag-of-words data with a generative model. However, the dependencies in their respective graphical Models are all linear, limiting the power of such a model. Recent developments in efficient large-scale variational inference have made it possible to train generative models with non-linear dependencies such as neural networks on large amounts of data. In this thesis we investigate if such methods can be applied to bag-of-words data effectively.

Specifically, we develop and test how to best use Stochastic Gradient Variational Bayes for bag-of-words topic modeling. We define a Topic Variational Autoencoder, which closely resembles a typical SGVB approach, and develop additional methods to complement or improve upon this approach. These methods include the use of stick-breaking priors on the latent variables, and the use of dimensionality reduction methods on large vocabulary sizes. In our experiments on the resulting models, we first characterize the behavior of a Topic VAE during optimization. Next, we experiment with different architectures and examine its performance when trained to different amounts of data. Furthermore, we test the efficacy and feasibility of batch normalization, stick-breaking priors, and dimensionality reduction techniques. For all these experiments we use the small KOS blog post dataset and a large collection of New York Times articles. Lastly, we compare our results to LDA and DEF, showing that an approach like ours is a very appealing alternative to current methods.

Acknowledgements

First and foremost I would like to thank my supervisor Max Welling for his guidance and for his positive, inspiring attitude throughout my thesis. I would also like to thank Patrick Putzky and Thomas Kipf for many fruitful discussions.

I would also like to thank my fellow students, in particular Joost van Amersfoort, who helped me learn so much during my Master's in AI. My appreciation also goes out to Dieuwke Hupkes, who inspired me to take on this Master's Programme.

Lastly, I would like to thank my family and Yvonne Koper for supporting me in everything I do.

List of Symbols

We use lowercase bold-faced symbols to denote vectors, and uppercase bold-faced symbols to denote matrices.

V	Number of unique words in a dataset.
N_d	Number of documents in a dataset.
$\mathbf{x}$	Data point
$\hat{\mathbf{x}}$	Data point normalized to have unit length
θ	Model parameters
ϕ	Parameters of approximate posterior model
z	Stochastic latent variable
μ	Mean of stochastic latent variable
σ^2	Variance of stochastic latent variable
$\mathcal{L}$	Lower Bound
$\tilde{\mathcal{L}}$	Lower Bound estimate
$\tilde{\mathcal{L}_w}$	Per-word lower bound estimate
D_{KL}	Kullback-Leibler Divergence
$\mathbf{X}^{fr}$	Frequent words in dataset
$\mathbf{X}^{if}$	Infrequent words in dataset
$\mathbf{X}^{ld}$	Lower dimensional representation of $\mathbf{X}^{if}$
F	Matrix of node feature vectors

We use index i to indicate a data point, index k to indicate a feature, and index j to indicate a latent variable. In all applications in this thesis, our data points are documents represented by count vectors. Therefore, for example, $\hat{\mathbf{x}}_{ik}$ is the relative frequency of word k in document i.

List of Abbreviations

DEF	Deep Exponential Families
GCN	Graph Convolutional Network
LDA	Latent Dirichlet Allocation
SGVB	Stochastic Gradient Variational Bayes
VAE	Variational Autoencoder
SVD	Singular Value Decomposition
NVI	Neural Variational Inference

Contents

Chapter 1

Introduction

SGVB [18] [30] is a recent, popular method for efficient, large scale unsupervised learning. While its use was demonstrated on images in the original work [18] [30], it can be applied to different types of data and can even be extended to modeling sequential data [7] [9]. However, it has not been applied successfully to bag-of-words data. One of the most popular models for bag-of-words topic modeling has been Latent Dirichlet Allocation (LDA) [5]. The graphical model of LDA closely resembles that of a typical model trained with SGVB, but an SGVB approach has several potential benefits over LDA. This work details the efforts to investigate an SGVB approach to bag-of-words topic modeling.

1.1 Contents

In the Chapter (2) we describe in more detail the area of Topic Modelling. We also detail on a few successfull approaches, and motivate in more detail the choice for SGVB as a method in topic modeling. We also explain the background material on which our methods depend. In Chapter 3 we detail the models used in our experiments. These Experiments are reported in Chapter 4, along with their results and the interpretation thereof. Lastly, we will summarize this work and briefly discuss our results as a whole in the conclusion, and end with some recommendations for future work on this topic. The work also includes an Appendix detailing our theoretical efforts to combine Graph Convolutional Networks [19] with our SGVB approach.

1.2 Implementation

All models were implemented in Theano and run on a 16-core Google Cloud Compute instance. [1] The code and documentation will be made freely available on Github. [2]

[1]https://cloud.google.com/compute/

[2]https://github.com/OttoFabius/sgvb˙topic

Chapter 2

Background

In this background section we will introduce the relevant concepts and literature for this thesis. First we will briefly introduce the area Bag-of-Words Topic Modeling in section 2.1. Then we will briefly discuss the concept of Variational Inference, which can be applied to the various models discussed in this Chapter. Next, we will explain two generative topic models, LDA (Section 2.3) and DEF (Section 2.4). After this, we discuss SGVB [18] [30] and its relation to LDA in Section 2.5. Lastly, we will detail the Stick-Breaking Autoencoder [24], a variation of the VAE described in [18], in Section 2.6.

2.1 Bag-of-words Topic Modeling

A Bag-of-Words representation of a document is a simplified vector representation as the multiset of its words. Thus, word-order and grammar are not considered in Bag-of-Words representations, but the simple vector representation allows for fast computation and use of off-the-shelf machine learning algorithms. Bag-of-Words representations have been a popular choice for document retrieval (e.g.[21]) and document classification (e.g. [22]), and has been used in the context of Computer Vision by considering all the features in an image independent of location (e.g. [10]).

The application of the Bag-of-Words representation we are interested in is that of topic modeling. A topic model is a statistical model for discovering the hidden structure of a collection of documents. This hidden structure is often referred to as topics, implying the word probabilities in a document depend on the topic. In this work we will therefore sometimes refer to the hidden structure (or: latent representation) of a document as the topics. Although this intuitive way if interpretation is common practice in the literature (e.g. [5]), we must point out that the use of the word topics does not imply that the structure found with a topic model necessarily corresponds to what humans would judge to be topics. Evidence of this was provided by Chang et al. [6], who conducted large scale user studies that showed popular probability based evaluation metrics do not always agree with semantically meaningful topics. However, in this work we do not have the means to perform user analyses on the topics found and we stick to much more commonly used practical measures which we will discuss in Section 4.1.3).

2.2 Variational Inference

Variational Inference is a method often applied to approximation the posterior distribution $p(\mathbf{Z}|\mathbf{X})$ of latent variables $\mathbf{Z}$ given observed variables $\mathbf{X}$. In most cases there is no analytical solution to the true posterior. Therefore, Variational Inference approximates the true posterior $p(\mathbf{Z}|\mathbf{X})$ with the variational distribution $q(\mathbf{Z})$, and optimizes $q(\mathbf{Z})$ to be as close as possible to $p(\mathbf{Z}|\mathbf{X})$.

Following [4], we start by decomposing the marginal log likelihood $\ln p(\mathbf{X})$:

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \mathrm{KL}(q||p) \tag{2.1}$$

Where

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \mathrm{d}\mathbf{Z} \tag{2.2}$$

$$\mathrm{KL}(q||p) = -\int q(\mathbf{Z}) \ln \frac{p(\mathbf{X}|\mathbf{Z})}{q(\mathbf{Z})} \mathrm{d}\mathbf{Z} \tag{2.3}$$

Here, $\mathrm{KL}(q||p)$ is a measure of how closely $q(\mathbf{Z})$ approximates $p(\mathbf{Z}|\mathbf{X})$. Since it is always positive or zero, $\mathcal{L}(q)$ is a lower bound on the model evidence $\ln p(\mathbf{X})$, and is therefore often called Evidence Lower Bound, or ELBO. In this work we will frequently refer to the ELBO as the lower bound.

2.3 LDA

The first popular *generative* probabilistic bag-of-words topic model is LDA [5], which assumes the following generative process:

1. Choose $N \sim \text{Poisson}(\xi)$
2. Choose $\theta \sim \text{Dir}(\alpha)$
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$
   (b) Choose a word $w_n \sim p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n

In this model, the Dirichlet distribution is assumed to have a known, fixed dimensionality, corresponding to the number of topics.
To optimize model we need to perform inference on the posterior distribution of the topics $\mathbf{z}$ and its multinomial parameters θ given the document. Several methods have been developed for this, including Gibbs sampling methods [12] [27], Expectation Propagation [23] and Variational Inference methods [5] [33]. This thesis connects only to variational inference

methods and a discussion of these methods is outside the scope of this thesis. . In Section 2.4.1 we discuss method used to perform (variational) inference in DEF, which can be used for inference in LDA too.

2.4 Deep Exponential Families

Deep Exponential Families (DEF) [29] is a class of hierarchical generative topic models. Essentially, it is a generalization of LDA to arbitrary probability distributions from the exponential family, which can be organized hierarchically in the generative process. Ranganath et al. [29] also detail their specific inference method used, based on Black Box Variational Inference [28]. The combination results in a more powerful model than "basic" LDA, achieving state-of-the-art perplexity scores on two bag-of-words datasets. [29]. This shows that deeper, hierarchical models can yield better results. Our models in Chapter 3 can be fully understood without this background Section 2.4.

Exponential Families are a large class of probability distributions that take the following general form [29]:

$$p(x) = h(x)\exp(n^{\mathrm{T}})T(X) - a(\eta) \tag{2.4}$$

where h is the base measure, η are the natural parameters, T are the sufficient statistics, and a is the log normalizer. Specific distributions are therefore defined by the choice of h and T, and include for example the Normal, Beta, and Poisson distribution [29]. The generative process of a DEF model is defined by a chain of such distributions, where the natural parameters of each distribution are given by draws from the distribution in the previous layer. The distribution of the top layer is defined by hyperparameter η. This way each document $\mathbf{x_i}$ has a hierarchical latent representation which consists of one vector $z_{i,l}$ for each of the L layers.

Each layer $l \geq 1$ also contains a set of learnable weights $\mathbf{W}_{l-1}$ shared across data, that provides a different linear weighting on each layer of latent variables $\mathbf{z}_l$.

2.4.1 Variational Inference in DEF

We will very briefly touch upon the inference method used in [29]. One way of restricting $q(\mathbf{Z})$ is by making simplifying assumptions about the distribution. Mean field variational inference [4] assumes $q(\mathbf{Z})$ factorizes into several disjoint groups, without making any other assumptions about the form of $q(\mathbf{Z})$. As explained in 2.2, variational inference equates to maximizing the lower bound (see equation 2.2). For DEF, this can be written as [29]:

$$\mathcal{L}(q) = \mathbb{E}_{q(z,W)}[\ln p(x,z,W) - \ln q(z,W)] \tag{2.5}$$

They assume it factorizes over all layers of latent variables for each datapoint $\mathbf{z_l}$ and shared latent variables $\mathbf{W_l}$, which leads to an approximate posterior of the general form:

$$q(z,W) = q(\mathbf{W}_0 \prod_{l=1}^{L} q(\mathbf{W}_l) \prod_{n=1}^{N} q(\mathbf{z}_n, l) \tag{2.6}$$

Of course, specific choices of which distributions to use further define this approximate posterior.

Inserting the expression of the approximate posterior in Equation 2.5 is therefore the expression that needs to be optimized, which is done by means of backpropagation of noisy, stochastic but unbiased gradients as in Black Box Variational Inference [28]. It does require somewhat tedious mathematics for each choice of DEF model. For further details we refer to the original paper [29].

Their approach leads to good perplexity scores and allows for flexible model construction, but the high variance leads to long training times. Along with the large amount of model choices that can be made, hyperparameter optimization is therefore not the most convenient, especially on large datasets [29].

2.5 Stochastic Gradient Variational Bayes

Stochastic Gradient Variational Bayes concerns the problem scenario of a generative model $p_\theta(\mathbf{X}|\mathbf{Z})$ parametrized by θ, and an approximation to the true posterior $q_\phi(\mathbf{Z}|\mathbf{X})$, parametrized by ϕ. From a coding perspective, $q_\phi(\mathbf{Z}|\mathbf{X})$ is often referred to as the *encoder*, as it produces a code $\mathbf{Z}$ of $\mathbf{X}$. Similarly, $p_\theta(\mathbf{X}|\mathbf{Z})$ is referred to as the *decoder*, and in this work we will frequently use this terminology. SGVB uses a different approach to variational Inference than DEF (Section 2.4.1).

2.5.1 Variational Inference with Parametrized Distributions

As explained in Section 2.4.1, one way of restricting $q(\mathbf{Z})$ is by making simplifying assumptions about the posterior. A different approach to restricting $q(\mathbf{Z})$ is by parameterizing the approximation by ϕ, noted $q_\phi(\mathbf{Z})$. This way, one can optimize ϕ by performing stochastic gradient ascent on the lower bound in equation 2.2. However, obtaining gradients of this lower bound is not straightforward as it often decomposes into terms which are not all tractable to compute.

In SGVB, the encoder and the decoder are optimized jointly, i.e. the parameters θ and ϕ are learned in the same process. Note that the approximation $q(\mathbf{Z}|\mathbf{X})$ now depends on the observed variable $\mathbf{X}$. SGVB obtains stochastic samples of the lower bound in Equation 2.2 by applying a reparametrization trick to it.

2.5.2 Lower Bound Estimator

Before we detail the reparametrization, we must first rewrite the lower bound, following [18]. We start from the form in equation 2.2, but now have an approximate posterior dependent on $\mathbf{X}$ and parametrization of encoder and decoder by θ and ϕ, respectively. The dependency of $q(\mathbf{Z}|\mathbf{X})$ on $\mathbf{X}$ necessitates writing this in terms of individual data points $\mathbf{x}$:

$$\mathcal{L}(\theta, \phi, \mathbf{x}_i) = \int q_\phi(\mathbf{z}|\mathbf{x}_i) \ln \frac{p_\theta(\mathbf{x}_i, \mathbf{z})}{q_\phi(\mathbf{z})} \mathrm{d}\mathbf{z} \tag{2.7}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[-\ln q_\phi(\mathbf{z}|\mathbf{x}_i) + \ln p_\theta(\mathbf{x_i}, \mathbf{z})] \tag{2.8}$$

$$= -\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\ln p_\theta(\mathbf{x_i}|\mathbf{z})] \tag{2.9}$$

Using this estimate, the lower bound w.r.t. ϕ and θ is optimized by error backpropagation. However, the gradients w.r.t. ϕ and θ have no closed form solution as we need to integrate over the latent variables. Rather than use a Monte Carlo estimate for these gradients (as done in [26]), [18] use a practical estimator for the lower bound and its derivatives that exhibits much lower variance.

2.5.3 Reparametrization

To obtain l samples from $q_\phi(\mathbf{z}|\mathbf{x}_i)$, [18] sample from a (differentiable) transformation as follows:

$$\hat{\mathbf{z}^{(l)}} \sim g_\phi(\epsilon, x) \tag{2.10}$$

Auxiliary variable ϵ is sampled from some matching distribution $p(\epsilon)$ s.t. samples $\hat{\mathbf{z}}$ are distributed according to $q_\phi(\mathbf{z}|\mathbf{x}_i)$. Although this is not possible for any parametrization of $q_\phi(\mathbf{z}|\mathbf{x}_i)$, it is straightforward for any location-scale family and any distribution with a tractable inverse CDF.

For example, let us choose to parametrize $q_\phi(\mathbf{z}|\mathbf{x}_i)$ as $N(\mu, \sigma^2\mathbf{I})$,a Multivariate Normal distribution wit independent variables. It is a location-scale distribution, so we can choose $g_\phi = \mu(x_i) + \epsilon \cdot \sigma(x_i)^2$ with $\epsilon \sim N(0, I)$.

With this reparametrization, the obtained samples of the gradient of Equation 2.9 w.r.t ϕ and θ allow us to efficiently optimize these with stochastic backpropagation.

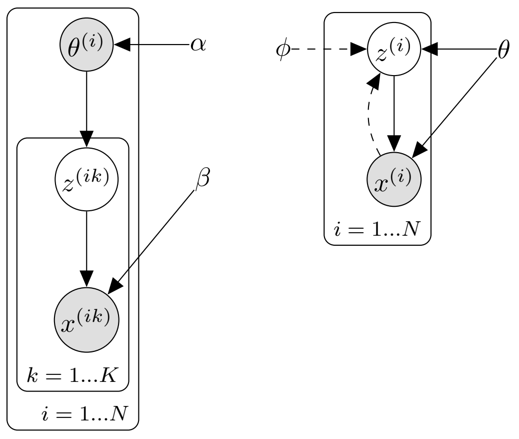


Figure 2.1: Graphical Model of LDA (left) and VAE Topic Model (left)

2.5.4 Relation VAE to LDA

The graphical model of the VAE used to illustrate the use of SGVB in [18] is closely related to LDA [5] (Section 2.3). As can be seen in Figure 2.5.4, both are generative models where the data is generated by some process from a latent representation of data point $\mathbf{x}_i$. The generative process is governed by parameters, which are to be optimized through inference. However, the parametrization of LDA is clearly defined, where a VAE allows any parametrization as long as it is differentiable. Of course, an obvious choice for such a powerful non-linear parametrization is a (deep) neural network. This also allows for efficient, scalable Variational Inference with a lot more flexibility of the approximation to the posterior. This makes SGVB a promising approach to Topic Modeling.

A key difference in the graphical models is the extra plate for each word in LDA. This means that where in LDA each word in a document has its own latent representation (topic), the latent representation is shared across all features in a data point in a VAE. This is further elaborated on in the description of our approach, see Section 3.1.

2.6 Stick-Breaking VAE

One restriction of SGVB in general is the need for a distribution over the approximate posterior that either belongs to the location-scale family, or has a tractable inverse CDF (see

Section 2.5.3. Therefore, e.g. Beta distributed latent variables can not be used in the SGVB framework.

Nalisnyck & Smith extend SGVB to Stick-breaking priors by using the Kumaraswami Distribution as the approximate posterior. Their motivation is that using Beta distributed latent variables leads to non-centered, more discriminative latent representations. They show such an approach is competitive to independent Normally distributed latent variables on the MNIST and Frey Faces datasets. LDA has a Dirichlet prior on the latent variables [5], as do several effective DEF models in [29]. This seems to be a good modeling choice for the sparse bag-of-words documents, possibly because they would often fall into only a few of many possible supposed topics [32]. Therefore, a Stick-Breaking VAE isprice promising for topic modeling purposes. The rest of this Section 2.6 is a description of the relevant part of Nalisnyck & Smyth [24].

2.6.1 The Stick-Breaking Process

Nalisnick & Smith [24] use Stick-Breaking priors to the approximate posterior. The name Stick-Breaking is an analogy to sequentially breaking off k parts π_k proportionate to v_k of a length-1 stick, which represents the total probability mass of the distribution. Mathematically this is defined as:

$$\pi_k = v_1 \text{ iff k=1}$$
$$\pi_k = v_1 \prod_{j<k} (1 - v_j) \text{ iff } k \neq 1$$

Where $v_k \sim \text{Beta}(\alpha, \beta)$. The Dirichlet process is characterized by sampling $v_k \sim \text{Beta}(\alpha_0, 1)$, which [24] use in their model. The stick-breaking process is infinite-dimensional, but can be truncated by setting $v_K = 1$. Choosing such as prior in SGVB, however, does not facilitate reparametrization of $q_\phi(\mathbf{z}|\mathbf{x}_i)$ (see Section 2.5.3). Therefore, the Nalisnick & Smith [24] use the Kumaraswami Distribution for $q_\phi(\mathbf{z}|\mathbf{x}_i)$.

2.6.2 The Kumaraswami Distribution

The Kumaraswami distribution is defined by:

$$\text{Kumaraswami}(x; a, b) = abx^{a-1}(1-x)^{b-1} \tag{2.11}$$

It closely resembles the Beta distribution and for $a = 1$, $b = 1$, or both, they are equivalent. Samples can be drawn from the Kumaraswami distribution by means of the (tractable) inverse CDF [24]:

$$x \sim (1 - u^{\frac{1}{b}})^{\frac{1}{a}} \text{ with } u \sim \text{Uniform}(0, 1) \tag{2.12}$$

They also compute the KL Divergence between the Stick-Breaking prior $p(\mathbf{z})$ and the approximate (Kumaraswami) posterior $q_\phi(\mathbf{z}|\mathbf{x}_i)$ (see Section 2.5.2) analytically (see Section 3.2 for the exact expression).

2.7 Neural Variational Inference for Topic Models

A very recent method applying SGVB to inference in Topic Models is Neural Variational Inference (NVI) for Topic Models [32] (NVI is used as a synonym for SGVB). In short, they use NVI to perform inference in LDA. Instead of using a Kumaraswami posterior, they use a Laplace approximation to the Dirichlet prior. On a small dataset, their perplexity scores are unimpressive, but they achieve high topic coherence scores for some model choices. Notably, Batch Normalization together with a high learning rate was paramount in achieving these high topic coherence scores. Although their method is related to our methods, the details of [32] are not needed for understanding the rest of the current work and we will not discuss there further.

Chapter 3

SGVB Topic Models

In this Chapter, we present in detail the models and methods we use in our experiments.

3.1 Topic VAE

In this section we describe our initial approach for topic modeling with SGVB. It closely resembles the VAE model used in the experiments by Kingma and Welling[18], so we call this a Topic VAE. We will describe the application-specific choices made within the general SGVB framework as described in section 2.5, and derive the objective function used for optimization.

3.1.1 Model Specifics

Within the general SGVB framework described in the previous chapter, we specify the following characteristics of our Topic VAE:

1. The representation of our documents $\mathbf{X}$
2. The encoder $q(\mathbf{z}|\mathbf{x})$
3. The prior over the latent variables, $p(z)$
4. A differentiable way of sampling $\mathbf{z}$ from $p(\mathbf{z}|\mathbf{x})$
5. The decoder $p(\mathbf{x}|\mathbf{z})$

1: The representation of our documents $\mathbf{X}$

The Bag-of-Words representation of the documents $\mathbf{X}$ is normalized such that each document $\mathbf{x}_i$ is represented by a unit vector $\hat{\mathbf{x}_i} = \frac{\mathbf{x}_i}{\sum_{k=1}^{V} x_{ik}}$. Although normalizing data is standard practice in neural network approaches (e.g. [3]), typically each feature is normalized independently to have zero mean and unit variance. In our approach, however, all features (word counts) of a data point (document) are normalized s.t. they represent word probabilities. Note that this representation no longer contains information on the length of documents, so this approach assumes the latent representations are independent of document length.

2: The encoder $q(\mathbf{z}|\mathbf{x})$

The encoder $q(\mathbf{z}|\mathbf{x})$ is a fully connected neural network with one or more hidden layers with ReLU activation functions. The input is $\tilde{\mathbf{x}}$ and the output is the mean and log standard deviation $\{\boldsymbol{\mu}, \log \boldsymbol{\sigma}^2\}$ of the Multivariate Gaussian $N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$. For one hidden layer this would be the following function:

$$\mathbf{h_{e1}} = \text{ReLU}(\hat{\mathbf{x}}\mathbf{W_{e1}} + \mathbf{b}) \tag{3.1}$$

$$\boldsymbol{\mu} = \mathbf{h_{e1}}\mathbf{W}_{\mu} \tag{3.2}$$

$$\log \boldsymbol{\sigma}^2 = \mathbf{h_{e1}}\mathbf{W}_{\sigma} \tag{3.3}$$

Our encoder therefore only differs from the encoder used in the experiments in Kingma & Welling [18] in the use of the ReLU function and, in some cases, multiple hidden layers. The ReLU function is slightly faster and preliminary experiments did not show any difference in performance between using ReLU, sigmoid, or tanh nonlinearities.

3: The prior over the latent variables, $p(z)$

We also use the same prior over latent variables, $p(\mathbf{z}) = N(0, \mathbf{I})$. Although the SGVB framework does not necessarily restrict $p(\mathbf{z})$ to this choice, it is common practice in SGVB approaches [18] [30] [8].

4: A differentiable way of sampling $\mathbf{z}$ from $p(\mathbf{z}|\mathbf{x})$.

We use and the same (differentiable) sampling method as used in [18]: transformation function $g_\phi(\boldsymbol{\epsilon}, \mathbf{x}) = \boldsymbol{\mu} + \boldsymbol{\sigma}^2 \odot \boldsymbol{\epsilon}$, and sampling function $p(\epsilon) = N(0, \mathbf{I})$. See Section 2.5.3 for an explanation of this reparametrization.

5: The decoder $p(\mathbf{x}|\mathbf{z})$

The decoder $p(\mathbf{x}|\mathbf{z})$ is also a neural network with as input (a sample from) latent representation $\mathbf{z}$ and as output the probabilities of a Multinomial distribution, with a ReLU activation function used in each hidden layer. With one hidden layer, this is specified by:

$$\mathbf{h_{d1}} = \text{ReLU}(\mathbf{z}\mathbf{W_{d1}} + \mathbf{b_{d1}}) \tag{3.4}$$

$$p(\mathbf{x}|\mathbf{z}) = \text{softmax}(\mathbf{h_{d1}}\mathbf{W_{d2}} + \mathbf{b_{d2}}) \tag{3.5}$$

Where the softmax$(\mathbf{x}) = \dfrac{e^{\mathbf{x}}}{\sum_{k=1}^{K} e^{x_k}}$ In section 2.5.4, we noted that LDA has a latent representation $\mathbf{z_{ik}}$) for each document $\mathbf{x_{ik}}$. In our approach we condition $\mathbf{z}_i$ on the whole document and therefore have document-wide latent representations. While this is a fundamental difference between our approaches, one might obtain different samples from $q_\phi(\mathbf{z}|\mathbf{x}_i)$ to have a different representation of each word in a document, although they would still be drawn from the same distribution. In this work we refrain from this for practical and computational reasons and draw one document-wide sample.

3.1.2 Objective Function

The general form of the SGVB estimator is:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x_i}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p_\theta(\mathbf{z})) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}_i|\mathbf{z}_i^{(l)}) \tag{3.6}$$

And because we consistently only use one sample from sampling function $p(\boldsymbol{\epsilon})$ per data point, this simplifies to:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x_i}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p(_\theta(\mathbf{z})) + \log p_\theta(\mathbf{x}_i|\mathbf{z}_i) \tag{3.7}$$

Because we use Gaussian latent variables with diagonal covariance, we can integrate the KL Divergence analytically as done in Kingma and Welling [18]. Adding the expression for the Multinomial likelihood $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$, this results in the following expression for the lower bound:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x_i}) = -\frac{1}{2}\sum_{j=1}^{J}\{1 + \log \sigma^2_{\phi,ij} - \mu^2_{\phi,ij} - \sigma^2_{\phi,ij}\} + \sum_{k=1}^{K} x_{ik} \log(y_{ik}) \tag{3.8}$$

Notably, this is the lower bound per *document.* In (bag-of-words) topic modeling, likelihood measures such as perplexity are usually per-word measures. To obtain a per-word lower bound, we must divide the total lower bound for a set of evaluated documents by the number of words in that set:

$$\tilde{\mathcal{L}}_w(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{X}) = \frac{1}{\sum\limits_{i=1}^{N}\sum\limits_{k=1}^{K} \mathbf{X}_{ik}} \sum_{i=1}^{N} \tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x_i}) \tag{3.9}$$

We cannot compare this lower bound estimate in 3.9 nor the one in 3.8 directly to different models or evaluation measures. Therefore, the lower bound estimate is merely for model optimization and comparison between Topic VAE models. Although 3.8 and 3.9 are functionally equivalent, a per-word lower bound is independent of average document size and more relatable to other per-word measures in topic modeling, so we use this measure for reporting experimental results.

3.2 Stick-Breaking Topic VAE

Now that we have described in detail our basic approach to Topic modeling with SGVB, in this section we will describe an alternative to using a Gaussian latent variables; latent variables with Stick-Breaking priors. Where Section 3.1 essentially detailed how to use a VAE for topic modeling, this section details how to use a stick-breaking VAE for Topic Modeling (see [24]), and we therefore call this a Stick-Breaking Topic VAE.

As in section 3.1, we will first describe the model specifics and then explain the resulting objective function.

3.2.1 Model Specifics

Once again, for a fully specified model, we need to define:

1. The representation of our documents $\mathbf{X}$
2. The encoder $q(\mathbf{z}|\mathbf{x})$.
3. $p(\mathbf{z})$, a prior over the latent variables.
4. A differentiable way of sampling $\mathbf{z}$ from $p(\mathbf{z}|\mathbf{x})$. As detailed in 2.6.2, we can do this by sampling from the inverse CDF of the Kumaraswami distribution: $\mathbf{x} = (1 - \boldsymbol{\epsilon}^{\frac{1}{\mathbf{b}}})^{\frac{1}{\mathbf{a}}}$, where $\boldsymbol{\epsilon} \sim \text{Uniform}(0, 1)$
5. The decoder $p(\mathbf{x}|\mathbf{z})$. This remains unchanged te decoder described in section 3.1.

While $\tilde{\mathbf{X}}$ and $p(\mathbf{x}|\mathbf{z})$ are equivalent to Section 3.1, point 2-4 concern the latent variables and are therefore different. Point by point, the Stick-Breaking Topic VAE is defined by:

1. Our representation $\tilde{\mathbf{X}}$ is the same as in for the Topic VAE discussed in 3.1.

2. The encoder $q(z|x)$ now encodes the parameters a, b of a Kumaraswami distributions instead of μ and σ^2 of univariate Gaussians (see 3.2 and 3.3)
3. The prior $p(\mathbf{z})$ is now defined by the stick-breaking process described in 2.6.1. Note that in order to do this, we much choose a Dirichlet prior α_0 (see Section 2.6.1).
4. A differentiable way of sampling $\mathbf{z}$ from $p(\mathbf{z}|\mathbf{x})$ is sampling from the inverse CDF of the Kumaraswami distribution: $\mathbf{x} = (1 - \boldsymbol{\epsilon}^{\frac{1}{\mathbf{b}}})^{\frac{1}{\mathbf{a}}}$, where $\boldsymbol{\epsilon} \sim \text{Uniform}(0, 1)$ (see also 2.6.2)
5. The decoder $p(\mathbf{x}|\mathbf{z})$ remains unchanged compared to the decoder with a softmax output described in section 3.1.

3.2.2 Objective Function

The objective function for our Stick-Breaking Topic VAE once again consists of two parts: the KL divergence $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p(_\theta(\mathbf{z}))$ and the reconstruction error $p_\theta(\mathbf{x_i}|\mathbf{z_i})$ (see equation 3.7). As $p_\theta(\mathbf{x_i}|\mathbf{z_i})$ is once again modeled as a Multinomial, that part remains unchanged: $p_\theta(\mathbf{x_i}|\mathbf{z_i}) = \sum_{k=1}^{K} x_{ik} \log(y_{ik})$. $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i)||p(_\theta(\mathbf{z}))$ is now the KL divergence between the Dirichlet stick-breaking prior distribution and the Kumaraswami posterior distribution. Following Nalisnick & Smyth [24]:

$$D_{KL}(q(\boldsymbol{\beta}_\mathbf{i}|\mathbf{x}_i)||p(\boldsymbol{\beta}_i;\alpha_0)) = \sum_{k=1}^{K} \mathbb{E}_q[\log q(v_{i,k})] - \mathbb{E}_q \log p(v_{i,k})] \tag{3.10}$$

We also truncate the infinite-dimensional distribution (stick-breaking process) by setting $\pi_{i,K}$ such that $\sum_{k=1}^{K} \pi_{i,k} = 1$, in which case the KL divergence 3.10 can be written as:

$$\sum_{k=1}^{K} \mathbb{E}_q[\log q(v_{i,k})] - \mathbb{E}_q \log p(v_{i,k})] = \frac{a_0 - \alpha}{a_0}(-e - \Psi(b_\phi - \frac{1}{b_\phi})) + \log(a_\phi b_\phi) + \log B(\alpha, \beta) \tag{3.11}$$

$$-\frac{b_\phi - 1}{b_\phi} + (\beta - 1) b_\phi \sum_{m=1}^{\infty} \frac{1}{m + a_\phi b_\phi} B(\frac{m}{a_\phi}, b_\phi) \tag{3.12}$$

where e is Eulers constant, $\Psi(\cdot)$ is the Digamma function, and $B(\cdot)$ is the Beta function. The infinite sum, which originates from a Taylor approximation, is approximated by the leading ten terms as in Nalisnick & Smith [24]. The Digamma function Ψ in 3.11 is approximated with a second order Taylor approximation as higher orders lead to more numerical instability around $b_\phi = 0$.
For a full derivation of the KL Divergence between the Kumaraswami distribution and the Dirichlet distribution, see Nalisnick and Smyth (2016) [24].

3.3 Dealing with large vocabulary sizes

One problem of using a VAE approach to topic modeling is the large dimensionality of large corpora. Calculating $p(\mathbf{x}|\mathbf{z})$ dominates the computational cost of each forward pass, and this scales linearly with the vocabulary size. Another reason large vocabulary sizes are troublesome is that very infrequent words get assigned extremely low probabilities and few examples, which respectively lead to numerical instability and overfitting. Typically, only words that occur more often than some threshold in the dataset are used in Topic Modeling. For our approach, this threshold is relatively high for large datasets (see 4.1.1). This makes training time and model evaluation a lot faster, and optimization easier. Because modeling only the more frequent words has many advantages in our approach, we investigate how to include information on the discarded words in our VAE approach in an efficient, scalable manner.

We project the infrequent words linearly onto a smaller subspace, retaining as much information as possible, and use this as additional information in our encoder. We select from dataset $\mathbf{X}$ a subset of frequent words $\mathbf{X}_{fr}$ to use in the VAE approach described in section 3.1. Instead of discarding the infrequent words $\mathbf{X}^{if}$, we construct a lower dimensional representation of $\mathbf{X}^{if}$, which we will call $\mathbf{X}^{ld}$. This representation is concatenated with the first hidden layer $\mathbf{h_{e1}}$ (equation 3.1 in chapter 3.1). This way, the second layer of the encoder becomes:

$$\mathbf{h_{e2}} = \mathrm{ReLU}(\begin{pmatrix}\mathbf{h_{e1}} & \mathbf{x}^{ld}\end{pmatrix}\mathbf{W_{e2}} + \mathbf{b_{e2}}) \tag{3.13}$$

This way, the last rows of $\mathbf{W_{e2}}$ multiply only with $\mathbf{x}^{ld}$, yet each hidden unit in $\mathbf{h_{e2}}$ depends on both $\mathbf{x}^{ld}$ and $\mathbf{h_{e1}}$ (and thus also on $\mathbf{x}^{fr}$). Optionally, a hidden layer with learnable parameters can be applied separately to transform $\mathbf{x}^{ld}$ before the concatenation in equation 3.13. The lowercase $\mathbf{h_{e1}}$ and $\mathbf{x}^{ld}$ means this is formulated for one document, but a formulation in terms of minibatches is exactly the same.
It is important to note that we do not attempt to model projected data $\mathbf{X}^{ld}$ explicitly in $p(x|z)$ as its probability distribution has now changed. It therefore merely provides the encoder with additional information. In this work, we investigate two ways of achieving such a lower dimensional representation.

3.3.1 Random Projection

One effective way of achieving such a lower dimensional representation of sparse data is by means of a Random Projection, a method frequently used for dimensionality in the context of text (see e.g. [2] for a comparison to other dimensionality reduction methods). The method is based on the Johnson-Lindenstrauss lemma [11], which states that a linear mapping of a large set of high-dimensional vectors to a much lower dimensionality can approximately preserve distances between vectors. A linear mapping of N sparse vectors of dimension D onto a K-dimensional space is done by multiplication of the projection matrix $\mathbf{R}_{D\text{x}K}$:

$$\mathbf{X}^{ld} = \mathbf{X}_{N\text{x}K} = \mathbf{R}_{D\text{x}K}\mathbf{X}_{N\text{x}D} \tag{3.14}$$

Where $\mathbf{X}_{N\text{x}D}$ is our bag-of-words data with N documents and vocabulary size D.

For the Johnson-Lendenstrauss lemma to hold, the matrix R must be orthogonal. However, a results by Hecht-Nielsen [14] states that vectors in high-dimensional spaces are very likely to be close to orthogonal. Often Gaussian distributed random matrices are used, and Bingham and Mannila [2] show experimentally that a high-dimensional random matrix with Gaussian distributed values is close to orthogonal.
In this work we shall therefore restrict ourselves to a matrix $\mathbf{R}_{D\text{x}K}$ with entries drawn from $N(0, \sigma^2)$. For efficient optimization, it is best that the entries in $\mathbf{X}^{ld}$ are in approximately the same order of magnitude as H_{e1} (see equation 3.1). Therefore we choose $\sigma = \frac{0.5}{\sqrt{\bar{n}}}$, where $\bar{n}$ is the average document length in the dataset.

3.3.2 Singular Value Decomposition

The second method of obtaining a lower dimensional representation of infrequent words $\mathbf{X}^{if}$ is by SVD:

$$\mathbf{X}^{if} = \mathbf{U\Sigma V} \tag{3.15}$$

We then use K left-singular vectors with the highest singular values as $\mathbf{X}^{ld}$. We then once again use each row of $\mathbf{X}^{ld}$ as in equation 3.13.
Bingham & Mannila [2] performed experiments on dimensionality reduction techniques for text processing. Notably, they calculated the distances of random document pairs before and after dimensionality reduction. Reducing dimensionality with SVD lead to lower differences,

but Random Projections also preserved these distances quite well. Documents were normalized to have unit length as in our applications, and the vocabulary size in their experiments was 5000 which is comparable to our vocabulary size (see Section 4.1.1 on datasets). Performing SVD is slower than using a Random Projection, although this is negligible for our application. In Section 4.2.6 we experiment with both RP and SVD as methods to obtain $\mathbf{X}^{(ld)}$ in the encoder.

Chapter 4

Experiments and Results

4.1 General

Before we describe our specific experiments, we will describe the datasets used, the general optimization method used in our experiments, and the evaluation metrics reported.

4.1.1 Datasets

For all methods, we ran experiments on the KOS and New York Times datasets, freely available at UCI[1]. Stopwords and words that occur less than ten times in the whole dataset were removed for both datasets. After this, the KOS dataset contains 3430 blog post entries from the "Daily Kos" outlet, and has a vocabulary size of 6906. The dataset was split into 3330 training documents and 130 test documents. The NY times dataset consists of 300,000 articles from the New York Times and has a vocabulary size of 102,660 words. For the NY Times dataset, we only use words that occur over 3,000 times in the dataset, which are 5319 unique words. For this dataset, a test set of 1,000 documents was used.

4.1.2 General Optimization Method

For all experiments we use the AdaM optimization method [17]) with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. Unless otherwise stated, we used learning rate $\alpha = 0.01$ for the KOS

[1] https://archive.ics.uci.edu/ml/datasets/Bag+of+Words

dataset and $\alpha = 0.003$ for NY Times dataset. We use minibatch size of 50 for the KOS dataset and 200 for NY Times.

Weights for the first layer were initialized with random samples from $N(0, \mathrm{I})$. initial weights in consecutive layers were drawn from $N(0, \frac{1}{n})$, where n is the number of input units to the corresponding layer. Biases were initialized at 0.

Lastly, the KL divergence is weighted during training, with the weight linearly increasing from 0 to 1 during the first 100 epochs of training. This is done to avoid local optima dominated by the KL divergence, referred to as nt collapsing (internal communication with M. Welling).

4.1.3 Evaluation

Comparison between different Topic VAE models is done by calculating the per-word lower bound estimate $\tilde{\mathcal{L}}_w$ (see equation 3.9) on the test set. However, we can not compare our models to other methods using this metric.

Evaluating topic models is often done by calculating the perplexity of held out words on the test set (e.g. [5, 25, 29]). In this work, held-out perplexity is calculated for the test set as in Newman & Welling [25] by using half the words, randomly sampled, in each document for inference (i.e. calculating $p(\mathbf{z}|\mathbf{x})$). Let $\mathbf{x}_i^{(s)}$ be the half of document $\mathbf{x}_i$ used for inference, and $\mathbf{x}_i^{(u)}$ be the other half of the document, used for evaluation. The average per-word perplexity of the unseen words $\mathbf{X}^{(\mathbf{u})}$ in the test documents $\mathbf{X}$ under the word-probabilities $p(\mathbf{x}^{(u)}|\mathbf{z}^{(s)})$, where $\mathbf{z}^{(s)} \sim p(\mathbf{z}|\mathbf{x}^{(s)})$, is then given by:

$$\text{perplexity} = \frac{1}{\sum\limits_{i=1}^{N}\sum\limits_{k=1}^{K} x_{ik}} \sum_{i=1}^{N}\sum_{k=1}^{K} \log p(x_k|\mathbf{z}_i^{(s)}) x_{ik}^{(u)} \tag{4.1}$$

As this perplexity estimate depends also on the sampled noise ϵ in the reparametrization, it is stochastic. Whenever we estimate the perplexity of a model in this work, we will evaluate every document ten times with independent samples ϵ to reduce the variance of this estimate to an acceptable level.

4.2 Experiments and Results

4.2.1 Optimization and characteristics

Here we describe experiments on both datasets described in 4.1.1 with a Topic VAE (3.1) to characterize behavior during optimization and to inspect initial results.

Since we have already defined our model (3.1) and optimization method (4.1.2), we need only to define the architecture, i.e. the number of hidden layers and hidden units in both the encoder and the decoder. Overfitting in de decoder is something to keep in mind, so we roughly follow this guideline proposed by Hinton [15] to prevent overfitting: "(...) estimate how many bits it would take to describe each data-vector if you were using a good model. (...) Then multiply that estimate by the number of training cases and use a number of parameters that is about an order of magnitude smaller." This means we can estimate the number of hidden units in the decoder according to $n_h \sim \frac{N*b}{V \cdot 10}$, where N is number of training cases and b is the number of bits needed. As the perplexity is in the order of $2 \cdot 10^3$ (i.e. in [29]), the number of bits is approximately $\log_2(2 \cdot 10^3) = 11$ for either dataset. For the KOS dataset we have $n_h \sim= \frac{3330 \cdot 11}{6906 \cdot 10} = 0.53$. Therefore, we chose to use a linear decoder (i.e. no hidden layer) and 8 latent variables. For the encoder, we use 200 hidden units. Note that even then we use more parameters than ideal and we might be prone to overfitting in the decoder.

We use 100.000 randomly selected documents for training from the NY Times dataset, which means we can use in the order of $n_h \sim= \frac{100.000 \cdot 11}{5319 \cdot 10} = 20.7$ hidden units. We use 50 hidden units in the decoder, which might just be enough to overfit, and 400 hidden units in the encoder. We use 32 latent variables for the NY Times dataset.

We evaluate and report the per-word lower bound $\tilde{\mathcal{L}}_w$ on both the train and test set during

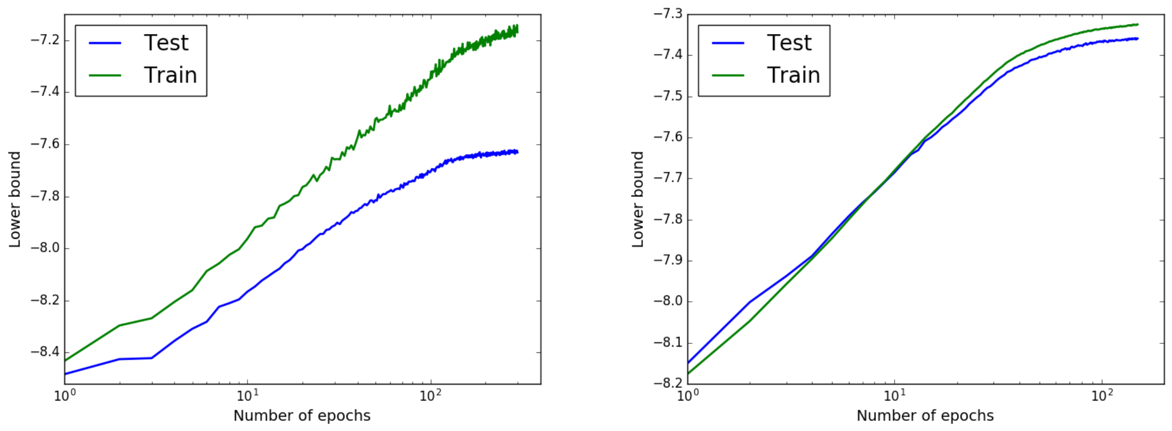


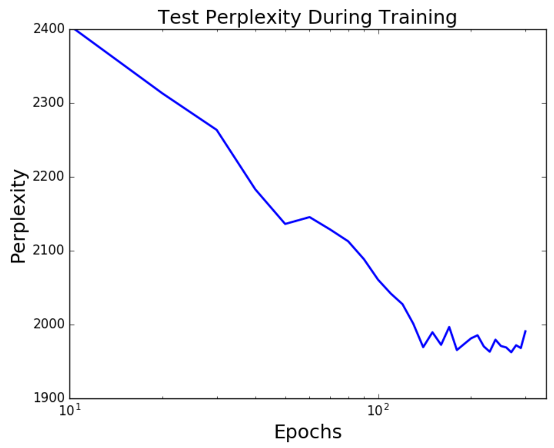
Figure 4.1: Test and train lower bound evaluated during training. Left: KOS dataset, right: NY Times dataset.

training, see Figure 4.1. The lower bound on the test set does not decrease as the models converge, so they do not overfit on the train set. Notably, the train and test lower bound diverge much more on the KOS dataset than on the NY Times dataset, indicating the model trained on KOS learns much more that is specific to the training set. Because the training set of KOS is so small compared to the NY Times dataset, the model trained on KOS learns more that is specific to the train set and does not generalize as well as the model trained on NY Times.

The 50% held-out test set perplexity for both models is shown in Figure 4.2. Notably, the perplexity does not improve after 50 (NY Times) and 100 (KOS) epochs, where the test lower bounds in Figure 4.1 start to converge but still show significant improvements after this point. This observation relates to the results of our comparison to other methods (see 4.2.8), and will be discussed further in that section.

4.2.2 Hyperparameter optimization

Next, we train Topic VAE models with different encoder and decoder structures and compare results. We use 8 and 32 latent variables for the KOS and NY Times datasets, respectively. We use the datasets as specified in 4.1.1, but only use 100,000 NY Times documents for training due to limited computational resources. Many other hyperparameters are the same



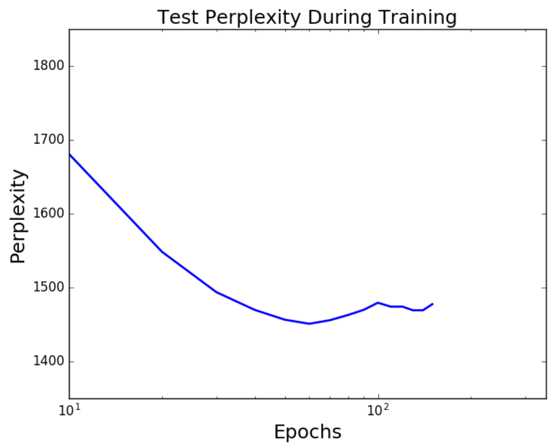


Figure 4.2: Perplexity evaluated during training. Left: KOS dataset, right: NY Times dataset.

for all models compared in this experiment and are chosen based on preliminary experiments. These include the optimization method and details, initialization of parameters, gradual increase of KL Divergence (see 4.1.2), but also the use of ReLUs and strength of prior on $p(\mathbf{z})$ (i.e., the variance σ^2 of the Gaussian $N(0, \mathbf{I})$). We do not vary these because we did not note a significant influence of these choices on model performance in these preliminary experiments.

Figure 4.3 and 4.4 detail the lower bound of both the train and test sets of the used datasets for converged models with different encoder/decoder structures. We also report the the 50% held-out perplexity on the test set.

The results on the KOS dataset are dominated by the tendency of the decoder to overfit. The number of parameters in the decoder is dominated by the last layer, which contains (n_h x V) parameters. For the KOS dataset the number of documents N (3300 in the train set) is small compared to the output space V (6906). For the NY Times dataset, with $N = 100,000$ documents used for training and $V = 5319$, this does not become a problem within the used range of architectures.

4.2.3 Varying the training set size

One question regarding our approach is how well this is suited for different amounts of available data. We assume it to be applicable to, and effective for, large amounts of training

Figure 4.3: Hyperparameter Optimization results on KOS. The best scores are shown in red.

Encoder	Decoder	LB Train	LB Test	Perplexity
200	-	-7.142	-7.622	-1990
200	20	-7.062	-7.740	2272
200	50	-6.905	-7.850	2554
200-100	-	-7.146	-7.630	1992
200-100	20	-7.262	-7.682	2133
200-100	50	-6.941	-7.789	2409
200-100	10-10	-7.217	-7.737	2259
200-100	20-20	-7.121	-7.872	2667
200-100	50-50	-7.024	-7.961	2955

Figure 4.4: Hyperparameter Optimization results on NY Times. The best scores are shown in red.

Encoder	Decoder	LB Train	LB Test	Perplexity
400	-	-7.354	-7.379	1492
400	50	-7.325	-7.358	1478
1000	-	-7.341	-7.379	1513
400-200	-	-7.345	-7.375	1493
400-200	100	-7.284	-7.327	1429
1000-600-300	200	-7.212	-7.308	1415
1000-600-300	500	-7.180	-7.297	1384
1000-600-300	200-500	-7.161	-7.324	1448

data. Therefore we want to know if and how much a model improves, in general, when trained on more data. It is also useful to gain some insight in how prone a model is to overfitting with different amounts of data.

To that end, we train a model with a relatively high capacity that was shown to perform well in the results of the hyperparameter optimization experiments in section 4.2.2. We use training set sizes between 5,000 and 300,000 documents and test the models on the same test set of 1,000 documents. We compare effectiveness of converged models by reporting the test lower bound and 50% held-out test set perplexity. Due to the large difference in training set size, a different number of epochs is used for each model. The results are shown in Figure 4.2.3.

Convergence took around 100 epochs for all dataset sizes. The linear increase of the KL divergence during the first 100 epochs (see section 4.1.2) prevented earlier convergence for larger amounts of data. Inspection of the lower bound on the test set revealed that overfitting occurred when using 5,000 and 10,000 training documents, but not when using 20,000 or more documents. This indicates that for larger amounts of data, a more powerful model (i.e.

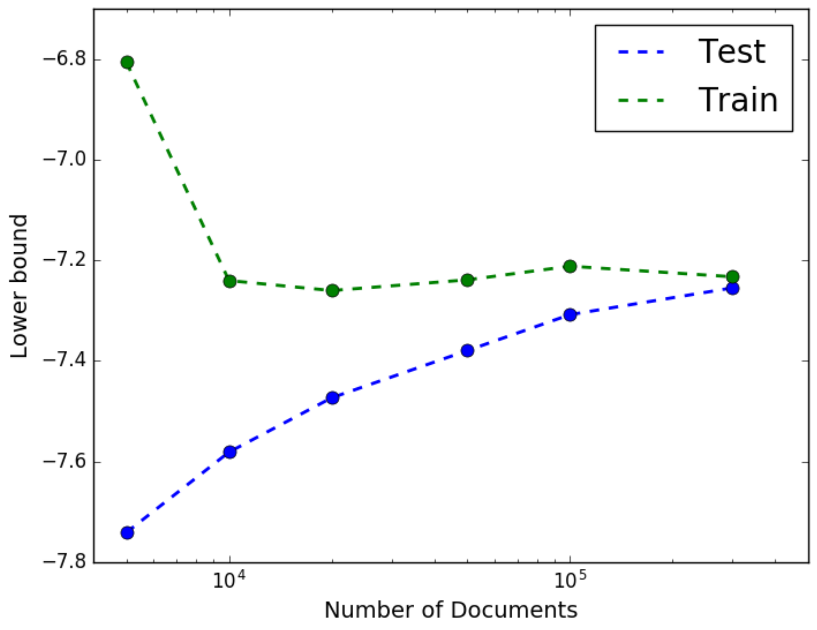


Figure 4.5: $\tilde{\mathcal{L}}_w$ evaluated on the train and test set after convergence, for a the model trained on an increasing number of documents.

with more parameters) would yield better results. This is also indicated by the fact that the lower bound on the train set does not improve further when using more documents, but the lower bound on the test set does. In Table 4.4, it can indeed be seen that 500 hidden units in the decoder leads to better results than the 200 units used in this experiment, when using 100,000 training documents.

4.2.4 Batch normalization

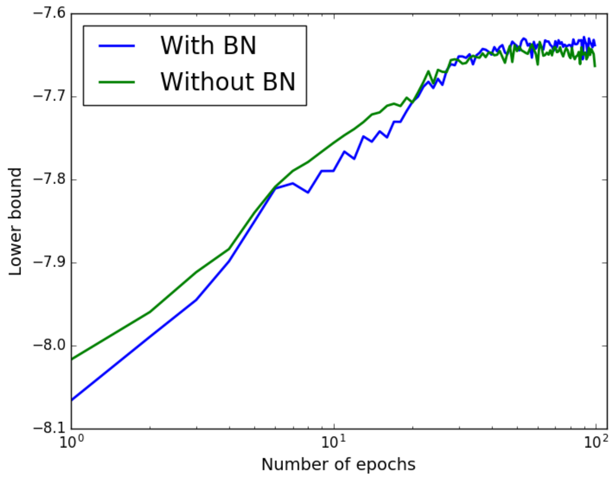
As detailed in 2.7, Srivastava & Sutton [32] found that Batch Normalization ([32]), together with a relatively high learning rate and momentum, yielded significantly better results for some model choices they made.

We therefore train a model on each dataset with batch normalization and compare the convergence and model performance to the same model trained without batch normalization. We already use high momentum in our experiments and a learning rate as high as possible without leading to numerical instability (see Section 4.1.2). Srivastrava & Sutton [32]

found that batch normalization allows for a higher learning rate, so we will once again use the highest learning rate that does not lead to numerical instability. In our model, we can use batch normalization on each hidden layer activation. Therefore, we choose architectures with multiple hidden layers, trained without batch normalization in section 4.2.2: For the NY Times dataset we use two hidden layers of 400 and 200 hidden units in the encoder and 100 units in the hidden layer of the decoder. For the KOS dataset we used hidden layers of 200 and 100 hidden units in the encoder, but no hidden layer in the decoder. Once again we only use the random subset of 100,000 documents for training on the NY Times dataset.
Training the models for a few epochs with different learning rates α showed that the highest α that does not lead to numerical instability is $\alpha = 0.003$ on the NY Times dataset and as high as $\alpha = 0.05$ on the KOS dataset. This was the case both with and without batch normalization. Figure 4.6 shows that convergence was not faster with batch normalization. Furthermore, Figure 4.7 shows model performance was not significantly different with and without batch normalization. The small differences present are so small they can be attributed to the variance due to (stochastic) optimization and the variance present in the estimate $\tilde{\mathcal{L}}_w$ and the perplexity.
Srivastrava & Sutton [32] attribute the need for Batch Normalization to the problem referred to as "component collapsing". In short, VAEs often have local minima very close to the initial prior on the posterior, such that components are "turned off". In the case of [32], all inferred topics were identical when this happened. Component collapsing was likely avoided in all our experiments due to the low KLD weight used initially (see Section 4.1.2).

4.2.5 Stickbreaking Topic VAE

We trained Stickbreaking VAE's (see 2.6) on both the KOS and NY datasets with an architecture that performed well with Gaussian latent variables. For the NY Times dataset we used 400 and 200 hidden units in the encoder and 100 units in the decoder and 32 latent variables, i.e. we truncated the stick-breaking process at the 32nd latent variable. For KOS we used 400 and 200 hidden units in the encoder and 100 units in the decoder, and 8 latent variables. Op-



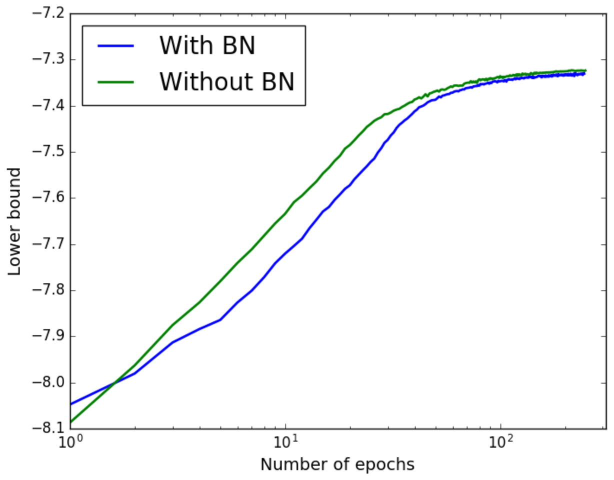
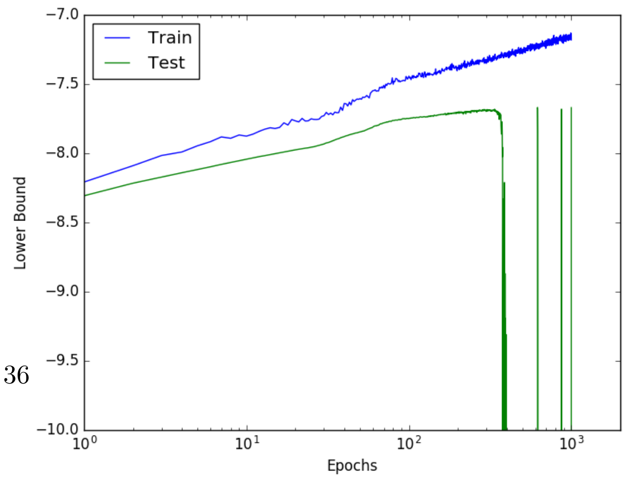


Figure 4.6: text

Dataset	Batch Normalization	Test lower bound	Perplexity
KOS	No	-7.648	1995
KOS	Yes	-7.638	1977
NY Times	No	-7.327	1429
NY Times	Yes	-7.331	1449

Figure 4.7: Estimates of the lower bound and 50% held-out test perplexity after convergence with and without batch normalization, on both datasets.

timization of a Stick-Breaking VAE proved to very problematic due to numerical instability. We took the exact same measures to prevent numerical instability in Stick-Breaking VAE's taken by Nalisnick & Smyth [24]: adding small numbers where division by 0 or taking the log of 0 might otherwise occur, and sampling noise ϵ not from $U(0,1)$ but from $U(0.01,0.99)$. The the learning rate α was lowered to ten times lower than in our previous experiments, and batch normalization was used on all hidden layers. We tested over a wide range of concentration parameter α_0. Despite this, instability was always problematic before convergence. Figure 4.8 shows a relatively successful attempt at training a model on the KOS dataset, with only some instability in the evaluation of $\tilde{\mathcal{L}}_w$ on the test set. More often then not, the instability would make training to convergence (or close to it) im-



Dataset	α_0	Test lb	perplexity
KOS	0.03	N/A	N/A
KOS	0.1	N/A	N/A
KOS	0.3	-7.9*	2551*
KOS	1	-7.673	2085
KOS	3	-7.670	2058
NY Times	0.03	N/A	N/A
NY Times	0.1	N/A	N/A
NY Times	0.3	-7.503*	1789
NY Times	1	-7.357	1553
NY Times	3	N/A	N/A

Figure 4.9: Test lowerbound and test set perplexity for various values of α_0. An asterisk (*) denotes the model could not be trained to convergence, in which case the best performance achieved is reported. N/A denotes the model could not be trained anywhere near convergence.

possible. Table 4.9 shows the model performance for various values of shape parameter α_0 of the prior on the stick-breaking weights $\beta(1, \alpha_0)$.

Nalisnick & Smyth [24] also found that optimization of a Stick-breaking VAE is harder than a VAE with Gaussian latent variables because of the necessity to back-propagate errors through the stick-breaking process. As each stick length (i.e. latent variable z_j) depends on all previous stick-lengths, this results in coupled gradients [24]. That they were able to train models successfully is likely because their data consisted of small images (MNIST and Frey Faces[2]), which are lower dimensional, much less sparse, and include much more regularity than bag-of-words data. We were only able to obtain results with an α_0 relatively close to 1. It seems that a very smooth prior distribution $\text{Beta}(1, \alpha_0)$ on the residual fractions v (see section 2.6) helps, as $\alpha_0 = 1$ yields a uniform prior distribution $\text{Beta}(1, 1) = U(0, 1)$.

4.2.6 Large vocabulary sizes

Here we outline how we tested the benefit of of using the infrequent words in the encoder with either a random projection (see 3.3.1) or SVD (see 3.3.2). We use the same number of

[2]Available at http://www.cs.nyu.edu/ roweis/data.html

dataset	method	hidden layer	test $\tilde{\mathcal{L}}_w$	perplexity
KOS	None	no	-6.639	744
KOS	RP	no	-6.645	735
KOS	SVD	no	-6.645	742
KOS	RP	yes	-6.642	730
KOS	SVD	yes	-6.639	739
NY Times	None	no	-7.308	1415
NY Times	RP	no	-7.349	1470
NY Times	SVD	no	-7.334	1441
NY Times	RP	yes	-7.310	1401
NY Times	SVD	yes	-7.306	1394

Figure 4.10: Results for different methods of using X^{if} in the encoder.

hidden layers and hidden units as in the previous section 4.2.5.

So far, we used vocabulary sizes of 6906 (KOS) and 5303 (NY Times), which in case of the KOS dataset was the full vocabulary. Therefore, for this experiment we will use only words that occur 50 times or more in the KOS dataset. This way, we will have 1930 frequent words X^{fr} and 4976 infrequent words X^{if}. The NY Times dataset contains the 5319 frequent words (see 4.1.1), and 97341 infrequent words which were discarded in all previous experiments. These infrequent words will be represented by either a random projection of these words or the largest singular values, as described in 3.3.1 and 3.3.2, respectively. We use dimensionality of the representation of the projection of words $K = 200$ in all experiments. We use an encoder with a large capacity: three hidden layers of 1,000, 600 and 300 hidden units, respectively. For the decoder we use one hidden layer with 200 units. Table 4.10 compares both methods on both datasets to the baseline where X^{if} is not used. Hardly any improvements over the baseline are realized: Using a hidden layer yields comparable results and not using a hidden layer seems to even have a slight adverse effect on performance compared to the baseline for both datasets. It seems that the information in X^{if} is not very useful to learn a better inference model $p(z|x)$. Another reason for the poor results might be that, although the capacity of the encoder is very high, the smaller capacity of the decoder is the limiting factor of the models and a better encoding $p(z|x)$ would therefore not lead to a better reconstruction $p(x|z)$.

Architecture*	Latent Variables	Infrequent words	Batch Normalization	Test lower bound	Perplexity
S	Gaussian	Unused	no	-7.320	1429
	Gaussian	Unused	yes	-7.331	1449
	Gaussian	Random Projection	no	-7.380	1500
	Gaussian	SVD	no	-7.372	1487
	stick-breaking	Unused	yes	-7.363	1523
L	Gaussian	Unused	no	-7.308	1415
	Gaussian	Unused	yes	-7.310	1420
	Gaussian	Random Projection	no	-7.310	1401
	Gaussian	SVD	no	-7.306	1394
	stick-breaking	Unused	yes	N/A	N/A

Figure 4.11: Overview of model performance of different approaches used for two different architectures. The smaller architecture, denoted with S, has 400 and 200 units in the hidden layers of the encoder and 100 hidden units in the decoder. The larger one, denoted with L, has has 1,000, 600 and 300 units in the hidden layers of the encoder and 200 hidden units in the decoder.

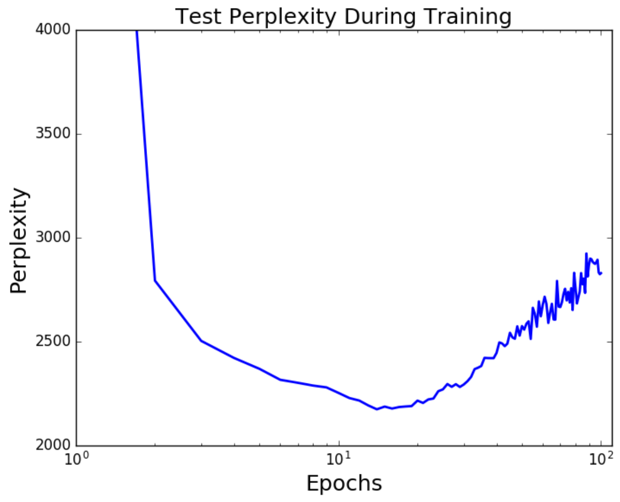
4.2.7 Overview of investigated methods

As described in 4.2.7, we compare the results of the different methods used in this thesis on the NY Times dataset: Gaussian (4.2.1) and stick-sreaking (4.2.5) latent variables, Random Projections or SVD on infrequent words (4.2.6), and the use of Batch Normalization (4.2.4). Figure 4.11 compares the test lower bound and 50% held-out test perplexity for the different methods on two architectures used in the hyperparameter optimization section (section 4.2.2, Table 4.4).

In general, the basic approach without Random Projections or SVD on X^{if} and with Gaussian latent variables was hardly improved upon. Batch normalization did not lead to improvements in convergence speed or model performance, and stick-breaking priors on the latent variables proved very hard to optimize.

4.2.8 Comparison to Other Methods

In this section we compare the Topic VAE method to the results on LDA ([5] and section 2.3) and Deep Exponential Families ([29] and section 2.4) reported in [29]. LDA is still a popular model and Deep Exponential Families is the state-of-the-art in terms of reported held-out test perplexity. Ranganath et al. [29] trained models on the 8,000 most frequent words of 165,000 NY Times documents. In both LDA and DEF, (variational) inference was performed with the method described in 2.4.1. They report the 10% held-out test set per-



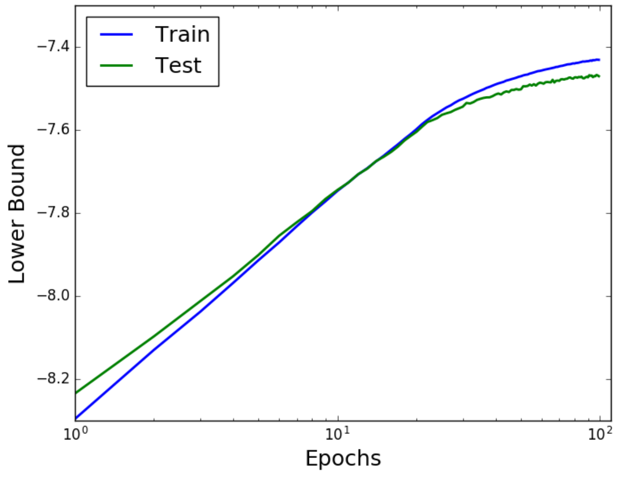


Figure 4.12: 10% held-out test set perplexity and $\tilde{\mathcal{L}}_w$ evaluated during training.

plexity on a test set of 1,000 of the same documents. We do the same with a Topic VAE model with Gaussian latent variables (Section 3.1). We use the same architecture referred to as "Large" in Table 4.11.

Model	Perplexity
LDA	2717
DEF (worst)	2653
DEF (best)	2251
VAE Topic	2174

Figure 4.13: 10% held-out perplexity on 1,000 held out NY Times documents. Lower is better. [28]

An important finding is that during training the perplexity starts getting worse even though the test lower bound has not finished improving. This is shown in Figure 4.12. This observation connects to the earlier observation that in previous experiments, the 50% held-out test perplexity stopped improving before the model had converged (see Figure 4.2). This indicates that held out test perplexity, especially when few words are used for inference, is a bad measure of model performance at least in some cases. To the best of our knowledge, this problem has not been described or investigated in the literature.

To compare our model to the other methods, we therefore use early stopping, i.e. we report the best 10% held-out test perplexity achieved during training, even though results get worse later on during convergence. As can be seen in Figure 4.2.8, the Topic VAE scores better than the other methods.

Chapter 5

Conclusion

In this work we investigated how to best apply SGVB top Topic Modeling. We first ran experiments with a typical VAE approach with a Multinomial reconstruction error. With these experiments, we investigated which architectures work well, how well the model performs with increasing amounts of data, and if the model easily overfits. Next, we developed two possible improvements: A Stick-Breaking VAE and the use of infrequent words in the encoder. We ran experiments with these methods and compared the results to our initial model. We also tested whether batch normalization is a useful inclusion in the developed models. Lastly, we compared our model to LDA and DEF by evaluating the 10 % held-out perplexity on the NY Times dataset.

In our experiments with the Topic VAE, we found that the decoder easily overfits on the small KOS dataset. The decoder did not overfit on the larger collection of NY Times documents; for that dataset, the best results were achieved with a more powerful decoder. For both datasets, additional hidden layers in the encoder did not lead to large improvements. We further determined that batch normalization did not benefit optimization or model performance within the performed experiments.

The performance of the Topic VAE improved when trained on increasing amounts of data, as is to be expected. This supports the claim that a Topic VAE is a useful model choice for large datasets.

The presented Stick-Breaking Topic VAE did not yield significant improvements over the Topic VAE. The Stick-Breaking VAE could not be trained to convergence due to unsolved numerical stability issues and even performance similar to the Topic VAE was not achieved. As for the additional use of infrequent words in the encoder of a Topic VAE, this addition only lead to small improvements in performance. We would therefore not recommend using the infrequent words in a dataset at all in practice.

While it proved hard to improve over the basic Topic VAE, it outperformed LDA and the state-of-the-art DEF. It was also shown that 10% held-out test perplexity used to compare methods is not a good measure of model performance in our case, as perplexity decreased during training while the test lower bound decreased. We therefore used early stopping for our method. It is unclear if LDA and DEF would also achieve better 10% held-out test perplexity scores with early stopping. It would therefore be good to also compare methods on e.g. 50% held-out test perplexity, which was unfortunately not feasible within this work.

Based on this work, a Topic VAE is an excellent model for bag-of-words topic modeling on large collections of text: It is straightforward to train, scales well with large amounts of documents, and achieves state-of-the-art results.

5.1 Future Work

Up until now, topic models have generally been binary latent variable models [5], [29]. The presented work, as well as other recent work [32], is a step in the direction of effective generative modeling of bag-of-words data with deep neural networks. NVI for Topic Models as in [32] should definitely be tried in combination with (deep) neural networks. This would allow for a Dirichlet prior on latent variables, which proved hard to do with the Stick-Breaking Topic VAE of Section 3.2.

The current work was compared to other methods on only one dataset, and with an evaluation metric that proved to be deceptive under at least some circumstances. Therefore, a more rigorous evaluation should be conducted to verify the results obtained here, using metrics such as 50% held-out perplexity or topic coherence (such as in [32]).

Further development of such models should include the use of dropout: In our work we found that overfitting can become a problem in the decoder and using dropout to combat this problem would allow for deeper, more powerful structures. Simpler decoders limit the complexity of the inference performed: if the latent representation of documents is a highly non-linear transformation of the documents, a simple decoder will not be able to reconstruct the original document from the latent representation. Therefore, more powerful decoders would allow for more powerful inference architectures.

Our current models all have one latent representation of documents. However, different architectures of (hierarchical) latent variables are possible, comparable to the hierarchical approach of DEF. Recent work on Hierarchical Variational Auto-Encoders has already been done in other application areas [31] [1] [36].

Our work has also shown the described models perform well on large amounts of data. With an efficient implementation, more computational resources, and more effective and efficient (hierarchical) architectures, it is hard to predict what might be inferred from documents based solely upon their bag-of-words representations.

Bibliography

[1] Philip Bachman, *An architecture for deep, hierarchical generative models*, Advances in Neural Information Processing Systems, 2016, pp. 4826–4834.

[2] Ella Bingham and Heikki Mannila, *Random projection in dimensionality reduction: applications to image and text data*, Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 245–250.

[3] Christopher M Bishop, *Neural networks for pattern recognition*, Oxford university press, 1995.

[4] Christopher M. Bishop, *Pattern recognition and machine learning*, Cambridge university press, 2006.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan, *Latent dirichlet allocation*, Journal of machine Learning research **3** (2003), no. Jan, 993–1022.

[6] Jonathan Chang, Jordan L Boyd-Graber, Sean Gerrish, Chong Wang, and David M Blei, *Reading tea leaves: How humans interpret topic models.*, Nips, vol. 31, 2009, pp. 1–9.

[7] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio, *A recurrent latent variable model for sequential data*, Advances in neural information processing systems, 2015, pp. 2980–2988.

[8] Carl Doersch, *Tutorial on variational autoencoders*, arXiv preprint arXiv:1606.05908 (2016).

[9] Otto Fabius and Joost R van Amersfoort, *Variational recurrent auto-encoders*, arXiv preprint arXiv:1412.6581 (2014).

[10] Li Fei-Fei and Pietro Perona, *A bayesian hierarchical model for learning natural scene categories*, Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 2, IEEE, 2005, pp. 524–531.

[11] Peter Frankl and Hiroshi Maehara, *The johnson-lindenstrauss lemma and the sphericity of some graphs*, Journal of Combinatorial Theory, Series B **44** (1988), no. 3, 355–362.

[12] Thomas L Griffiths and Mark Steyvers, *Finding scientific topics*, Proceedings of the National academy of Sciences **101** (2004), no. suppl 1, 5228–5235.

[13] W Keith Hastings, *Monte carlo sampling methods using markov chains and their applications*, Biometrika **57** (1970), no. 1, 97–109.

[14] Robert Hecht-Nielsen, *Context vectors: general purpose approximate meaning representations self-organized from raw data*, Computational intelligence: Imitating life (1994), 43–56.

[15] GE Hinton, *Neural networks: Tricks of the trade*, Springer, Berlin, Heidelberg (2012), 599–619.

[16] Sergey Ioffe and Christian Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, arXiv preprint arXiv:1502.03167 (2015).

[17] Diederik Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, ICLR Conference Paper (2015).

[18] Diederik P Kingma and Max Welling, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114 (2013).

[19] Thomas N Kipf and Max Welling, *Semi-supervised classification with graph convolutional networks*, arXiv preprint arXiv:1609.02907 (2016).

[20] Hayato Kobayashi, *Perplexity on reduced corpora.*, ACL (1), 2014, pp. 797–806.

[21] Thomas K Landauer and Michael L Littman, *Fully automatic cross-language document retrieval using latent semantic indexing*, (1990).

[22] Yonghong Li and Anil K Jain, *Classification of text documents*, Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on, vol. 2, IEEE, 1998, pp. 1295–1297.

[23] Thomas Minka and John Lafferty, *Expectation-propagation for the generative aspect model*, Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 2002, pp. 352–359.

[24] Eric Nalisnick and Padhraic Smyth, *Deep generative models with stick-breaking priors*, arXiv preprint arXiv:1605.06197 (2016).

[25] David Newman, Padhraic Smyth, Max Welling, and Arthur U Asuncion, *Distributed inference for latent dirichlet allocation*, Advances in neural information processing systems, 2007, pp. 1081–1088.

[26] John Paisley, David Blei, and Michael Jordan, *Variational bayesian inference with stochastic search*, arXiv preprint arXiv:1206.6430 (2012).

[27] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling, *Fast collapsed gibbs sampling for latent dirichlet allocation*, Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 569–577.

[28] Rajesh Ranganath, Sean Gerrish, and David Blei, *Black box variational inference*, Artificial Intelligence and Statistics, 2014, pp. 814–822.

[29] Rajesh Ranganath, Linpeng Tang, Laurent Charlin, and David M Blei, *Deep exponential families.*, AISTATS, 2015.

[30] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra, *Stochastic backpropagation and approximate inference in deep generative models*, arXiv preprint arXiv:1401.4082 (2014).

[31] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther, *Ladder variational autoencoders*, Advances in Neural Information Processing Systems, 2016, pp. 3738–3746.

[32] Akash Srivastava and Charles Sutton, *Neural variational inference for topic models*, Under review as a conference paper at ICLR 2017 (2017).

[33] Yee Whye Teh, David Newman, and Max Welling, *A collapsed variational bayesian inference algorithm for latent dirichlet allocation*, NIPS, vol. 6, 2006, pp. 1378–1385.

[34] Hanna M Wallach, David M Mimno, and Andrew McCallum, *Rethinking lda: Why priors matter*, Advances in neural information processing systems, 2009, pp. 1973–1981.

[35] Eric P Xing, Michael I Jordan, and Stuart Russell, *A generalized mean field algorithm for variational inference in exponential families*, Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2002, pp. 583–591.

[36] Shengjia Zhao, Jiaming Song, and Stefano Ermon, *Learning hierarchical features from generative models*, arXiv preprint arXiv:1702.08396 (2017).

Appendix A: Graph Convolutional Networks for Topic VAEs

Bag-of-words data can be seen as a bipartite graph between document nodes and word nodes. From this perspective, it makes sense to look into incorporating the work by Kipf & Welling [19] on Graph Convolutional Networks in our approach. Therefore, as part of this thesis, efforts were made to combine GCNs and Topic VAEs. Ultimately, these efforts did not lead to feasible model adjustments or improvements and no experiments were conducted. However, in this Appendix we will detail these efforts after briefly introducing GCNs. As this appendix is not integral to our work, we will keep the background to the bare minimum.

5.2 Graph Convolutional Networks

Recent work by Kipf & Welling [19] details a method to perform fast approximate convolutions on graphs with neural networks, thereby encoding a graph structure. This method is successfully applied to (sparse) citation networks and knowledge graphs in a semi-supervised learning setting. Their approach is applicable to any weighted graph, where our graph of documents connected to words is bipartite. Therefore, we would require only one Graph Convolution (with a forward- and backward pass) to connect all nodes in bag-of-words graph. Their propagation rule for one layer in a Graph Convolutional Network (GCN) is derived from a first-order approximation of localized spectral filters on graphs. It is given by:

$$H = \text{ReLU}(\bar{D}^{-\frac{1}{2}}\bar{A}\bar{D}^{-\frac{1}{2}}FW) \tag{5.1}$$

Where ReLU can be an other nonlinearity of choice. $\bar{A}$ is the (symmetric) adjacency matrix A with added self connections I:$\bar{A} = A + I$, and $\bar{D}_{ii} = \sum_j \bar{A}_{ij}$. Further, W is a trainable weight matrix similar to those in our earlier approaches.

5.3 Combining GCNs with Topic VAEs

We now investigate how to apply GCNs to the encoder of a Topic VAE. In our application, $\bar{A}$ in Equation 5.1 would be of dimension $(N_d + V)$ x $(N_d + V)$, with both the upper left corner and the bottom right corner an identity matrix. We can rewrite 5.1 as:

$$H = \text{ReLU}(X'W) \tag{5.2}$$

Where $X' = \bar{D}^{-\frac{1}{2}}\bar{A}\bar{D}^{-\frac{1}{2}}F$. A straightforward way of using Graph Convolutions is to incorporate one Graph Convolution in our encoder as in 5.1.
In this case, F would logically reduce to I since we have no node features.
Let us rewrite $X'W$, leaving out the self-connections **I** in **A**:

$$X' = \bar{D}^{-\frac{1}{2}} \begin{pmatrix} 0 & G \\ G^T & 0 \end{pmatrix} \bar{D}^{-\frac{1}{2}}W = \begin{pmatrix} \bar{G}W_1 \\ \bar{G^T}W_2 \end{pmatrix} \tag{5.3}$$

Where $\bar{G}_{ij} = \frac{G_{ij}}{\sqrt{\sum_i G_{ij} \sum_j G_{ij}}}$

We now have multiplications with two weight matrices W_1 and W_2, one that has parameters for each N_d documents and one that has parameters for each word in V. Having parameters for each document is not scalable to large datasets, especially since a batched approach to $\bar{G^T} W_2$ is impossible.

One way of encoding some document-level information is to use the covariance matrix $\bar{G^T}\bar{G}$ in stead of $\bar{G^T}$. Now, the number of parameters in our first layer would scale with $(2 \cdot V)$ instead of $(V + N)$. 5.3 now becomes:

$$\begin{pmatrix} \bar{G}W_1 \\ \bar{G^T}\bar{G}W_c \end{pmatrix} \tag{5.4}$$

Using a minibatch approach to $\bar{G^T}\bar{G}W_2$ requires calculating $\bar{G^T}\bar{G_{batch}}W_c$ for each batch, which is of complexity $O(V \text{ x } V \text{ x } h)$, where h is the number of hidden units. Even for a large batch size, this is much more expensive than calculating the sparse multiplication $\bar{G}_{batch}W_1$, which is of $O(\text{nonzero}(\bar{G}_{batch}) \text{ x } h \text{ x batchsize})$.

Using the full covariance matrix for each minibatch allows for computing $\bar{G^T}\bar{G}$ only once. However, writing out the first layer for one row $\mathbf{g}$ of $\bar{G}$ shows us this approach does not combine information in $\mathbf{g}$ and $\bar{G}_T\bar{G}$:

$$h_1 = \text{ReLU}(\begin{pmatrix} \mathbf{g} \\ \bar{G^T}\bar{G} \end{pmatrix} W_1 + b_1) \tag{5.5}$$

$$h_1 = \text{ReLU}(\begin{pmatrix} \mathbf{g}W_{1a} \\ \bar{G^T}\bar{G}W_{1b} \end{pmatrix} + b_1) \tag{5.6}$$

Leaving out $\bar{G^T}W_2$ in 5.3 leaves us with the original first layer of our encoder, except with a different TF-IDF-like normalization of our data.

While this normalization could be used for a VAE approach, this begs the question how to model $p(\mathbf{x}|\mathbf{z})$. While we chose to model $p(\mathbf{x}|\mathbf{z})$ as a Multinomial (see Chapter 3.1 and equation 3.8), the normalized data is no longer Multinomially distributed and there is no obvious choice for modeling $p(\mathbf{x}|\mathbf{z})$.

In this Appendix we have investigated the use of a GCN for incorporation in SGVB Topic modeling. We found that it is not possible to incorporate this without adding parameters to the model for each document. This is detrimental to scalability, which is one of the main advantages of our general approach. Furthermore, we considered augmenting the data pre-training, but this invalidates the Multinomial distribution of the data. Therefore, we concluded that GCNs are not useful in combination with SGVB Topic modeling, and did not perform any experiments on this.