Modern Variational Methods for Semi-Supervised Text Classification

Tam Dang
Paul G. Allen School of Computer Science & Engineering
dangt7@cs.washington.edu

Abstract

In the advent of powerful pretrained feature extractors trained on massive amounts of data, modern architectures such as ELMO and BERT have been shown to significantly improve performance in nearly every language task that involves them. Despite what can be achieved through these modern pretraining methods, they fail to accommodate environments in which such methods are not feasible; when researchers are constrained either by the amount of relevant in-domain data, or by their computational resources. Modern architectures also rely heavily on the language modeling objective, which has thus made it the most widely studied objective for feature extraction from unstructured text as a consequence. In this thesis, I instead explore alternative learning methods for feature extraction using variational methods in the context of both the *low-resource* and *high-resource* settings. By introducing VAMPIRE,[1] a descendant of topic models, I adapt neural variational document modeling to modern pretraining methods in order to produce powerful feature extractors that are especially motivated when resources are limited. I also explore the notion of *joint* semi-supervision, the act of unifying both labeled and unlabeled data under a single objective. By studying the phenomena surrounding model pretraining and joint semi-supervision using the variational bound, while providing direct comparisons to strong baselines in both the low-resource and high-resource settings, I provide a comprehensive exposition of the modernization of semi-supervised variational methods using semi-supervised text classification as a case study.

[1] VAriational Methods for Pretraining in Resource-limited Environments.

1 Introduction

Motivated by the abundance of unlabeled data and the difficulty of acquiring labeled data, effective approaches for semi-supervised learning have long been sought after by the NLP community. Powerful semi-supervised models are of both pragmatic and theoretical interest, as such models would help alleviate the burden of seeking domain experts for data annotation, while providing insight as to how best discover and leverage underlying aspects of text in the face of limited labeled data.

Early work in semi-supervised learning for NLP emphasized *self-training*, the act of iteratively training a model on labeled instances and subsequently augmenting the labeled dataset with pseudo-labeled unlabeled instances in which the model had high confidence. Later work in *transfer learning*, the act of training a model unsupervised on an external corpus separate from the data for the target task with the intention of later adapting learned models or representations downstream, has shown to provide larger gains. Large and diverse external out-of-domain corpora coupled with powerful unsupervised models often lead to the largest gains, with datasets such as the 1 Billion Word Corpus (Chelba et al., 2013) being amongst the most commonly used pretraining corpora, and models such as ELMO (Peters et al., 2018) and BERT (Devlin et al., 2019) being amongst the most powerful modern architectures available for unsupervised feature extraction.

In this thesis, I instead explore the implications of model training in the *low-resource setting* (§2.4): environments in which unlabeled data may be plentiful but labeled data and computational resources are scarce. By introducing VAMPIRE as a novel unsupervised model, this thesis will argue that it is possible to learn semi-supervised models

effectively in the low-resource setting, and does so by observing phenomena surrounding the adaptation of variational document modeling to both modern pretraining methods and joint supervision (§3.4) in this environment.

VAMPIRE is an unsupervised model that takes a variational autoencoder (VAE) approach to document modeling (Kingma and Welling, 2014; Miao et al., 2015; Srivastava and Sutton, 2017). As a descendant of topic models (§2.1), VAMPIRE implements neural variational inference (§2.2) to discover a latent semantic space over documents in the form of mixtures of abstract *topics*. As a feature extractor, it combines variational methods with techniques inspired by modern language models (Peters et al., 2018) to produce deep representations useful for downstream tasks. By ignoring sequential information, and instead serving to encode and reconstruct unigram representations of documents, it avoids the added time complexity and difficulty involved in learning sequence-based VAEs (Bowman et al., 2016; Xu et al., 2017; Yang et al., 2017) and makes for lightweight and inexpensive feature extractors that still offer strong performance downstream in the face of limited labeled data.

The *interpretability* of VAMPIRE's learned topics, measured in terms of *topic coherence*, notably allows for an alternative measure of model quality in addition to model fit via held-out perplexity. Traditionally, perplexity has served as the canonical automatic measure of success for unsupervised text modeling in NLP. However, this thesis will argue the viability of automatic measures for topic coherence as alternatives to perplexity for model selection, and show that VAMPIRE, when optimized for topic coherence, often leads to stronger and more reliable downstream performance when compared against perplexity as a stopping criteria.

Lastly, this thesis explores two variations of VAMPIRE: each of which are trained *jointly* on both labeled and unlabeled data under a single objective. JOINT-VAMPIRE (§3.4), the first variant, will learn a topic model and a discriminative classifier simultaneously. STACKED-VAMPIRE, the second variant, will also learn a topic model and a discriminative classifier simultaneously, but will do so using pretrained VAMPIRE embeddings as input. By introducing VAMPIRE and its variations, and exploring several phenomena surrounding them in the low-resource setting (§2.4), this thesis contributes the following:

- I adapt variational document models to both modern pretraining methods and joint semi-supervision for semi-supervised text classification.
- I show that the variational lower bound can serve as powerful alternative objective for unsupervised feature extraction in NLP.
- I draw parallels between topic coherence, perplexity and downstream classification performance, and highlight the importance of appropriate stopping criteria for model selection.
- I discuss the strengths and pitfalls of the joint objective and deep generative document models.
- I demonstrate the effectiveness of VAMPIRE's methods for feature extraction against several, semi-supervised baselines in regards to required computational and data resources along with performance in downstream classification.

This thesis is an extension of work originally published as Gururangan et al. (2019), serving to formally introduce the jointly-trained variants of VAMPIRE while providing additional insight surrounding the learning of variational document models.

2 Background

2.1 Topic Models

Topic models are a family of methods that learn to represent a collection of documents via abstract *topics*, where each topic is a distinct distribution over the vocabulary. As a generative story, documents are assumed to have originated from a finite mixture of these topics. Topic models are then unsupervised models that learn low-dimensional representations of documents in the form of mixture proportions over latent topics.

Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a popular method for topic modeling, leverages Dirichlet-multinomial conjugacy for efficient inference via sampling or variational methods by assuming a Dirichlet prior for both topic mixture proportions and the topics themselves. Other

IMDB		YAHOO!	
Horror	**Classics**	**Food**	**Obstetrics**
giallo	dunne	cuisine	obstetrics
horror	cary	peruvian	vitro
gore	abbott	bengali	endometriosis
lugosi	musicals	cajun	fertility
zombie	astaire	potato	contraceptive
dracula	broadway	carne	pregnancy
bela	irene	idli	birth
cannibal	costello	pancake	ovarian
vampire	sinatra	tofu	menstrual
lucio	stooges	gumbo	prenatal

Table 1: Example topics learned by VAMPIRE in IMDB and YAHOO! datasets. Topic models learn a low-dimensional representation of documents in the form of mixtures over these topics; carving out a semantic space that can characterize documents by traits such as *horror*, *classics*, *food*, and even *obstetrics*. Traits such as these are subject to the domain in which the topic model is trained.

LDA-inspired models have since adopted alternative priors such as the logistic normal (Lafferty and Blei, 2006; Srivastava and Sutton, 2017).

The most likely words within a given topic are often viewed as representative of the latent aspect of the corpus captured by that topic. Because of this, topics are conventionally reported as lists of their most likely words. As can be seen in Table 1 and Table 2, the most highly weighted words in each topic are not an arbitrary group, but rather coherent collections of terms that conform well to human intuitions about the topical content of documents. It is, in part, this ability to learn interpretable topics that accounts for the popularity of topic models such as LDA in exploring text corpora.

One method for automatic evaluation of topic models is through model fit via held-out perplexity or likelihood. However, previous work has shown that perplexity does not necessarily correlate with what human experts would consider as meaningful topics (Chang et al., 2009). Therefore, to optimize for topic *coherence*, this thesis will focus on normalized pointwise mutual information (NPMI), which has been shown in practice to correlate well with with human judgements (Mimno et al., 2011; Lau et al., 2014). NPMI can be thought of as a measure of co-occurrence between words that also takes into account the rarity of words; it captures the intuition that related words should co-occur more commonly than chance and thus serves as an automatic measure of cohesiveness and specificity. NPMI is computed as

$$\text{NPMI}(v) = \sum_{j \neq i} \frac{\log \frac{P(v_i, v_j)}{P(v_i)P(v_j)}}{-\log P(v_i, v_j)} \quad (1)$$

where v is a list of the most frequent words in a given topic, and where each probability $P(v_i)$, $P(v_j)$, and $P(v_i, v_j)$ is computed using empirical frequencies in a held-out validation set. In practice, the list v for each topic is often limited to the top 10 or 20 most likely words of the topic. Later, I discuss how NPMI serves as an effective stopping criteria for model selection when pretraining VAMPIRE.

2.2 Neural Variational Inference

Latent variable modeling is widely applicable in NLP, and often entails the introduction of a continuous latent variable that explains the origins of the text. However, modeling text in this way with exact inference is impractical, as marginalization over the latent variable to produce the likelihood of the data results in an integral that is intractable to compute. Methods for learning this latent variable instead serve to *approximate* this integral.

Markov Chain Monte-Carlo methods such as Gibbs sampling can be used to approximate the integral. *Variational* methods for approximation however, involve introducing a parameteric *variational* distribution that is used to derive a lower bound for the marginal likelihood of the data. This is often referred to as the *variational lower bound*. A prior is also introduced for the latent variable. For certain models, it is possible to derive a closed-form solution or set of update equations for approximate Bayesian inference. For instance, LDA discussed in §2.1 introduces a variational distribution over topic mixture proportions with a Dirichlet prior.

Neural variational inference (NVI) implements variational inference by allowing variational distributions and priors to be parameterized by neural networks, thereby making the variational lower bound optimizable via gradient descent methods. This thesis focuses on models that perform NVI because such models can be seamlessly integrated into other neural models downstream, which allows for complex interaction between parameters of the NVI model and parameters of downstream models. In particular, a large part of this thesis will explore such interactions between topic mod-

els learned via NVI and downstream text classifiers.

2.3 Language Modeling

Language modeling (LM) is an objective that aims to, for every word in the corpus, learn its conditional probability given the complete history of words that precede it in order to produce probability distributions over the text. Early work in language modeling has involved approximating the posteriors using n-gram probabilities, but such methods have since been largely replaced by recurrent networks and transformers (Vaswani et al., 2017).

Per-word *perplexity*, the canonical measure of model fit for language models, measures a language model's ability to predict the next word at any point in the corpus.

Previous work in pretraining for NLP involving the training of language models, such as Peters et al. (2018), has found that for multi-layer language models, each layer encodes different information. Empirically, *deep* representations using an interpolation of several layers in favor of the top layer alone has shown large gains across a multitude of tasks (Peters et al., 2018). Prior work in language modeling has also shown that for deep recurrent networks, that representations built using the first layer tend to include the most transferable features (Liu et al., 2019). VAMPIRE draws inspiration from these insights; in later sections I demonstrate that internal states for deep representations are also applicable in the context of variational methods, and that of these internal states, that the first layer is the most useful downstream.

2.4 Resource-limited Environments

In the low-resource setting, a researcher is constrained in terms of limited amounts of in-domain data, out-of-domain data, and computation. In some domains, labeled data can be acquired cheaply and easily, while others may require annotation by domain experts, which can severely limit availability. And while out-of-domain web-scale corpora are abundant for several languages like English, this is not true for all languages.

Computation as a resource describes the *feasibility* of computation, and several modern neural network architectures often require a level of computation that is beyond the reach of many researchers, especially those outside of STEM fields. For instance, to learn contextual embeddings, or to simply include them in a model downstream, can prove to be impractical without sufficient computational resources.

Despite the increasing availability of pretrained models for modern architectures such as ELMO or BERT, their applicability is limited not only by computational resources, but also by the inherent bias instilled in them from the data with which they were trained (Recasens et al., 2013; Bolukbasi et al., 2016; Zhao et al., 2019). The usage of such models is then less motivated for domains that can be particularly sensitive to bias. This encourages the existence of methods that can achieve the level of feature extraction as these pretrained models while using only the in-domain data.

It is in the face of limited labeled data and restricted computation that modern semi-supervised methods can be pushed to their limits. For this thesis, methods are tested in the low-resource setting because semi-supervised models are more powerful and motivated when they succeed in spite of limited labeled data. Given that most gains are made with models that are increasingly more infeasible to use from a computational standpoint, the ability for a model to use abundant unlabeled data quickly and efficiently to provide researchers constrained to low-resource settings with comparable models has become especially relevant.

3 Model

For this work, assume that we have a collection of N documents, $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, some of which may have categorical labels, $y_i \in \mathcal{Y}$. To better discuss the implications of semi-supervised learning, allow $\mathcal{D}_L = \{\boldsymbol{x}_i\}_{i=1}^{L}$ to represent the subset of documents in which a categorical label $y_i \in \mathcal{Y}$ is observed and $\mathcal{D}_{\mathcal{U}} = \{\boldsymbol{x}_i\}_{i=L+1}^{N}$ represent the subset of unlabeled documents.

3.1 Neural Variational Document Models

Neural variational document models (NVDMs) (Miao et al., 2015; Srivastava and Sutton, 2017) adapt the scalable variational methods introduced by Kingma and Welling (2014) to document modeling. More specifically, NVDMs are VAEs that perform approximate Bayesian inference by encoding documents to a continuous latent variable. Here, I describe the generic NVDM framework, and later discuss how additions to this framework gave rise to VAMPIRE in order produce NVDMs

that serve as strong feature extractors for text classification.

For a vocabulary V and document $\boldsymbol{x}_i$, let $\text{counts}(\boldsymbol{x}_i) \in \mathbb{R}^{|V|}$ be an input vector of word counts[2]; NVDMs are unsupervised, generative models that optimize a variational lower bound by encoding and reconstructing this input. Feedforward networks f_μ and f_σ generate parameters that parameterize the variational distribution over $\boldsymbol{z}_i$, which I assume to be approximately normal:

$$\boldsymbol{\mu}_i = f_\mu(\boldsymbol{x}_i) \tag{2}$$
$$\boldsymbol{\Sigma}_i = \text{diag}(f_\sigma(\boldsymbol{x}_i)) \tag{3}$$
$$q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) = \mathcal{N}(\boldsymbol{z}_i \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{4}$$

Using a Gaussian prior for latent document representations $p(\boldsymbol{z})$, a variational lower bound of the log-likelihood can be found for documents x_i (refer to the appendix §A for the full derivation),

$$\log(\boldsymbol{x}_i) \geq \mathcal{B}(\boldsymbol{x}_i) = \mathbb{E}_{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}\Big[\log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)\Big] - \text{KL}\Big[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \parallel p(\boldsymbol{z})\Big]. \tag{5}$$

The first RHS term of the bound serves as the *reconstruction loss*, ensuring that generated words are similar to the original document. The second RHS term is the KL-divergence, which encourages the variational approximation $q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$ to be similar to the normal prior, $p(\boldsymbol{z})$.

Using the *reparameterization trick* (Kingma and Welling, 2014), NVDMs avoid computation of the expectation by approximating it with a single sample,

$$\mathcal{B}(\boldsymbol{x}_i) \approx \log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i^s) - \text{KL}\Big[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \parallel p(\boldsymbol{z})\Big] \tag{6}$$
$$\boldsymbol{z}_i^s = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \cdot \boldsymbol{\varepsilon}^s \tag{7}$$

where $\boldsymbol{\varepsilon}^s \sim \mathcal{N}(0, \boldsymbol{I})$. $\boldsymbol{z}_i^s$ is then the output of a deterministic function of $\boldsymbol{x}_i$ and $\boldsymbol{\varepsilon}^s$ in which $\boldsymbol{\varepsilon}^s$ is an auxiliary variable, thereby making $\boldsymbol{z}_i^s$ differentiable with respect to the NVDM's parameters. This allows the variational lower bound to be optimized via gradient ascent methods.

$q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$ is a series of multilayer, feedforward networks and serves as the encoder, mapping documents $\boldsymbol{x}_i$ to a latent distribution parameterized by $\boldsymbol{\mu}_i$ and $\text{diag}(\boldsymbol{\sigma}_i)$:

$$\boldsymbol{c}_i = \text{counts}(\boldsymbol{x}_i) \tag{8}$$
$$\boldsymbol{h}_i = \text{MLP}(\boldsymbol{c}_i) \tag{9}$$
$$\boldsymbol{\mu}_i = f_\mu(\boldsymbol{x}_i) = \boldsymbol{W}_\mu \boldsymbol{h}_i + \boldsymbol{b}_\mu \tag{10}$$
$$\boldsymbol{\sigma}_i = f_\sigma(\boldsymbol{x_i}) = \exp(\boldsymbol{W}_\sigma \boldsymbol{h}_i + \boldsymbol{b}_\sigma) \tag{11}$$
$$\boldsymbol{z}_i^s = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \cdot \boldsymbol{\varepsilon}^s \tag{12}$$

To model the reconstruction $\log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)$, I assume each word is generated independently from an identical distribution (i.e., a unigram LM) and allow the decoder to produce this distribution. Let $\boldsymbol{B} \in \mathbb{R}^{K \times |V|}$ represent learnable word embeddings and $\boldsymbol{b} \in |\mathbb{V}|$ a learnable bias. The conditional probability of words is then modeled as a multinomial:

$$p(\boldsymbol{x}_{ij} \mid \boldsymbol{z}_i) \propto \exp(\boldsymbol{b} + \boldsymbol{B}^{\mathrm{T}} \boldsymbol{z}_i^s) \tag{13}$$
$$p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) = \prod_{j=1}^{V} p(\boldsymbol{x}_{ij} \mid \boldsymbol{z}_i^s)^{\boldsymbol{c}_{ij}} \tag{14}$$
$$\log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) = \sum_{j=1}^{V} \boldsymbol{c}_{ij} \cdot \log p(\boldsymbol{x}_{ij} \mid \boldsymbol{z}_i^s) \tag{15}$$

3.2 NVDMs as Topic Models

By interpreting $\boldsymbol{B} \in \mathbb{R}^{K \times |V|}$ as a series of row-wise positive and negative topic deviations for K distinct topics, where $\boldsymbol{z}_i^s$ serves as the unnormalized topic mixture proportions for a document $\boldsymbol{x}_i$, one could learn a topic model end-to-end via NVDMs as done in Miao et al. (2015). However, the work done by Srivastava and Sutton (2017) suggests that the NVDM framework alone is not enough to achieve comparable performance to LDA in terms of both perplexity and topic coherence.

Augmenting the NVDM framework to include explicit normalization of mixture proportions $\boldsymbol{z}_i^s$ not only allows reasoning about the document representation as a latent distribution over topics (Blei et al., 2003), but has also been shown to aid in the coherence of topics (Srivastava and Sutton, 2017):

$$\boldsymbol{\theta}_i = \text{softmax}(\boldsymbol{z}_i^s) \tag{16}$$
$$p(\boldsymbol{x}_{ij} \mid \boldsymbol{z}_i^s) \propto \exp(\boldsymbol{b} + \boldsymbol{B}^{\mathrm{T}} \boldsymbol{\theta}_i) \tag{17}$$

Allowing the learnable bias $\boldsymbol{b} \in \mathbb{R}^K$ to instead be a fixed background term containing the empirical log frequency of the vocabulary has also been shown to aid topic coherence (Card et al., 2018). Intuitively, this background frequency accounts for common words in the vocabulary as to promote variety in the learned topics.

[2] $\text{counts}(\boldsymbol{x}_i)$ for a document $\boldsymbol{x}_i$ is a vector representation of the *multiset* made up of the words found in $\boldsymbol{x}_i$. That is, the j^{th} component of $\text{counts}(\boldsymbol{x}_i)$ is the empirical word count of the j^{th} word in V occurring in $\boldsymbol{x}_i$.

3.3 VAMPIRE

VAMPIRE is a descendent of both topic models and NVDMs. In addition to interpreting word embeddings $\boldsymbol{B} \in \mathbb{R}^{K \times |V|}$ as K distinct row-wise topic deviations, it also leverages both NVDM augmentations discussed in §3.2: explicit normalization of mixture proportions and a fixed log frequency background term in place of a learnable bias.

As a semi-supervised, pretraining framework for text classification, VAMPIRE makes normalized topic proportions $\boldsymbol{\theta}_i$ from Eq. 16 available as supplemental features for downstream classifiers. VAMPIRE also draws inspiration from Peters et al. (2018), and provides a deep representation of the original bag-of-words representation via a weighted combination of layers in addition to $\boldsymbol{\theta}_i$. In particular, allow the MLP (multi-layer perceptron) in Eq. 9 to be n layers deep, and allow f_c to be an arbitrary probabilistic classifier. VAMPIRE embeddings then take the form of

$$\boldsymbol{r}_i = \lambda_0 \boldsymbol{\theta}_i + \sum_{k=1}^{n} \lambda_k \boldsymbol{h}_{ik} \tag{18}$$

$$p(y_i \mid \boldsymbol{x}_i) = f_c(y_i \mid [f_{s2v}(\boldsymbol{x_i}); \boldsymbol{r}_i]), \tag{19}$$

where $\boldsymbol{h}_{ik}$ is the k^{th} layer of the MLP during the encoding of $\boldsymbol{x}_i$, values $\{\lambda_0, ..., \lambda_n\}$ are softmax-normalized trainable parameters, and f_{s2v} denotes an arbitrary sequence-to-vector encoder that serves as an additional feature extractor for downstream classification. Note that by the construction of $\boldsymbol{r}_i$, VAMPIRE restricts the MLP to use the same hidden dimension for all layers, meaning that all $\boldsymbol{h}_{ik}$ are restricted to the same, fixed dimension. Models trained under the VAMPIRE can also be described as text-analogues of the M1 model described in Kingma et al. (2014).

3.4 JOINT-VAMPIRE

Under semi-supervised learning, we learn a model using $\mathcal{D}_L$ and either $\mathcal{D}_U$, a separate external corpus for transfer learning, or both. I define *joint* semi-supervised training as a form of learning that unites both $\mathcal{D}_L$ and $\mathcal{D}_U$ under a single objective as done in Kingma et al. (2014). Here, I will define the joint objective for document reconstruction and classification and show how VAMPIRE can be adapted to optimize it. The adaptation of VAMPIRE to the joint objective will be hereafter referred to as JOINT-VAMPIRE.

A model trained on the joint objective will learn to both discriminate documents and reconstruct them simultaneously. Later, I show that missing labels can be treated as a latent variable, enabling a single model to learn both latent document representation and latent document labels; optimizing for both reconstruction and discriminative classification simultaneously is then possible through variational inference.

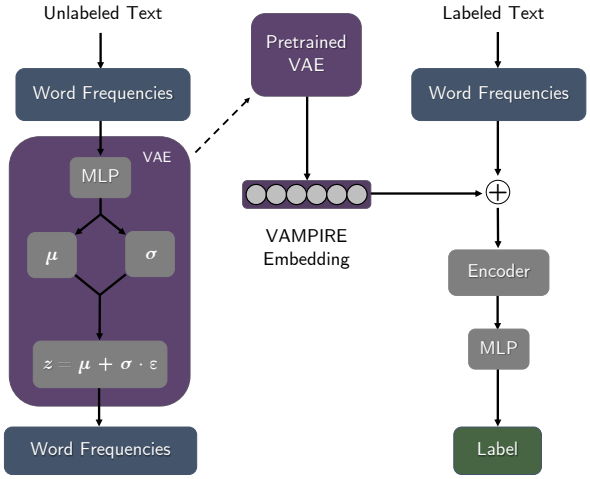

Figure 1: VAMPIRE involves pretraining a deep variational autoencoder (VAE; displayed on left) on unlabeled text. The VAE, which consists entirely of feed-forward networks, learns to reconstruct a word frequency representation of the unlabeled text with a logistic normal prior, parameterized by $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Downstream, the pretrained VAE's internal states are frozen and concatenated to task-specific word vectors to supplement a downstream classifier.

As a generative model, JOINT-VAMPIRE explains the origin of a document $\boldsymbol{x}_i$ via the following generative process as done in Kingma et al. (2014),

$$\begin{aligned} p(y) &= \text{Cat}(y \mid \pi) \\ p(\boldsymbol{z}) &= \mathcal{N}(\boldsymbol{z} \mid \mathbf{0}, \mathrm{I}) \\ p(\boldsymbol{x}_i \mid y_i, \boldsymbol{z}_i) &= f(\boldsymbol{x}_i \mid y_i, \boldsymbol{z}_i), \end{aligned} \tag{20}$$

with a categorical prior placed on labels $y \in \mathcal{Y}$ parameterized with probability π, and f being a likelihood function. Note that the conditional probability in Eq. 20 differs from the posterior in Eq. 15 by also conditioning on the categorical label y_i. More specifically, the joint objective will alter the original variational lower bound (5) by allowing $p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)$ and $q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$ to also condition on the label.

The joint objective can be described as a sum of two objective functions: one specific to unlabeled instances, and another specific to labeled instances. Following Kingma et al. (2014), I first

introduce a factored variational distribution over labels and latent representations $\boldsymbol{z}$ to the generative model described in Eq. 4:

$$q(y, \boldsymbol{z} \mid \boldsymbol{x}) = q(\boldsymbol{z} \mid y, \boldsymbol{x}) \cdot q(y \mid \boldsymbol{x}) \tag{21}$$

Allowing categorical labels to have a uniform prior, $p(y)$, and continuing to allow $\boldsymbol{z}$ to have a normal prior $p(\boldsymbol{z})$, this results in a new objective for labeled instances:

$$\begin{aligned} &\log p(\boldsymbol{x}_i,\ y_i) \geq \mathcal{B}_L(\boldsymbol{x}_i,\ y_i) \\ &= \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)}\Big[\log p(\boldsymbol{x}_i \mid y_i, \boldsymbol{z}_i) + \log p(y)\Big] \\ &- \mathrm{KL}\Big[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i) \parallel p(\boldsymbol{z})\Big] \end{aligned} \tag{22}$$

To be consistent with Kingma et al. (2014), the variational factor $q(y \mid \boldsymbol{x})$ will also serve as the discriminative classifier. In particular, $q(y \mid \boldsymbol{x})$ can be thought of as the downstream classifier that VAMPIRE is meant to supplement, which can be arbitrarily complex.

For the unlabeled objective, since the label is not observed, following Kingma and Welling (2014) I treat it as a discrete latent variable and take the expectation (refer to Appendix A for the full derivation),

$$\begin{aligned} &\log(\boldsymbol{x}_i) \geq \mathcal{B}_U(\boldsymbol{x}_i) \\ &= \Big[\sum_{y \in \mathcal{Y}} q(y \mid \boldsymbol{x}_i) \cdot \mathcal{B}(\boldsymbol{x}_i, y)\Big] + \Big[\mathcal{H}(q(y \mid \boldsymbol{x}_i))\Big], \end{aligned} \tag{23}$$

where $\mathcal{H}$ is Shannon entropy.

There are a multitude of ways to adapt VAMPIRE to the joint objective. For this work, I choose to produce the new posterior $q(\boldsymbol{z}_i \mid \boldsymbol{x}_i,\ y_i)$ by augmenting the input to VAMPIRE in Eq. 8 to instead be $[\text{counts}(\boldsymbol{x}_i); \vec{y}^{(i)}]$, where $\vec{y}^{(i)}$ is a one-hot representation of the label. As for the new posterior $p(\boldsymbol{x}_i \mid \boldsymbol{z}_i,\ y_i)$, reconstructed documents are augmented with label-specific topic deviations $\boldsymbol{C} \in \mathbb{R}^{|\mathcal{Y}| \times |V|}$,

$$p(\boldsymbol{x}_{ij} \mid \boldsymbol{z}_i^s,\ y_i) \propto \exp(\boldsymbol{b} + \boldsymbol{B}^{\mathrm{T}}\boldsymbol{\theta}_i + \boldsymbol{C}_{y_i}), \tag{24}$$

where $\boldsymbol{C}_{y_i}$ denotes the y_i^{th} row of C. The joint objective then takes the form of

$$J = \sum_{\boldsymbol{x}_i \in \mathcal{D}_L} \mathcal{B}_L(\boldsymbol{x}_i, y_i) + \sum_{\boldsymbol{x}_j \in \mathcal{D}_U} \mathcal{B}_U(\boldsymbol{x}_j). \tag{25}$$

Continuing to follow Kingma et al. (2014), the bounds also include an explicit classification loss weighted by a hyperparameter, α, to ensure that the classifier learns to discriminate directly from the labeled data. Choosing to model classification loss via cross entropy, the overall objective to be maximized thus becomes,

$$J^{\alpha} = J + \alpha \sum_{\boldsymbol{x}_i \in \mathcal{D}_L} \log p(y_i \mid \boldsymbol{x}_i) \tag{26}$$

which can be approximated via gradient descent methods using stratified minibatches containing both labeled and unlabeled instances. Models trained under this objective are best described as text-analogues of the M2 model described in Kingma et al. (2014).

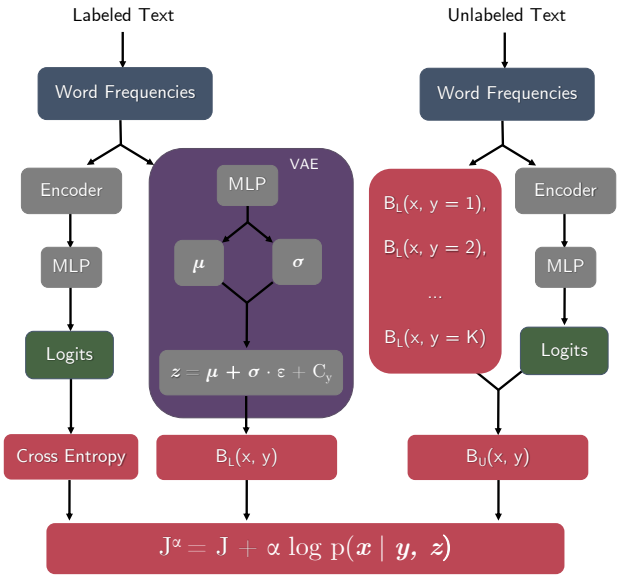

Figure 2: JOINT-VAMPIRE uses both labeled and unlabeled instances to train the joint objective. Only instances in which a label is observed contribute to the explicit cross entropy classification loss. For unlabeled instances, the labeled objective $\mathcal{B}_L(\boldsymbol{x}, y)$ for a document $\boldsymbol{x}$ is computed once for every categorical label in $\mathcal{Y}$. The expectation to compute $\mathcal{B}_U(x, y)$ is then the weighted average of all $\mathcal{B}_L(\boldsymbol{x}, y)$ where the weight for each is the corresponding classification logit from the encoding of $\boldsymbol{x}$, plus the Shannon entropy of the logits.

3.5 STACKED-VAMPIRE

Consider training an instance of VAMPIRE to convergence. We can then train JOINT-VAMPIRE using learned topic mixture proportions $\boldsymbol{z}_i$ as input in place of word frequencies. JOINT-VAMPIRE trained in this manner should then attempt to reconstruct $\boldsymbol{z}_i$ instead of word frequencies, thereby

introducing a second, *stacked* latent variable $\widetilde{\boldsymbol{z}}_i$ that is learned by the JOINT-VAMPIRE's VAE as it learns to encode and reconstruct $\boldsymbol{z}_i$. This results in a deep generative model (Kingma et al., 2014) that depicts the following joint probability,

$$p(\boldsymbol{x}, y, \boldsymbol{z}, \widetilde{\boldsymbol{z}}) = p(y)p(\widetilde{\boldsymbol{z}})p(\boldsymbol{z} \mid y, \widetilde{\boldsymbol{z}})p(\boldsymbol{x} \mid \boldsymbol{z}), \tag{27}$$

where priors $p(y)$ and $p(\widetilde{\boldsymbol{z}})$ are equivalent to priors $p(y)$ and $p(\boldsymbol{z})$ discussed in §3.4. I then define STACKED-VAMPIRE to be these very instances of JOINT-VAMPIREs that use topic mixture proportions $\boldsymbol{z}_i$ as both the target for reconstruction and as the input for predicting the categorical label of a document.

To reconstruct latent document representations $\boldsymbol{z}$, recall that such representations are assumed to be approximately normal. Allow $\widetilde{\boldsymbol{z}}_s$ to be the single-sample approximation of $\widetilde{\boldsymbol{z}}$, computed using a VAE and reparameterization in same manner as $\boldsymbol{z}^s$. An approximate mean $\widetilde{\boldsymbol{\mu}}$ and variance $\widetilde{\boldsymbol{\sigma}}$ is then computed from the compression $\widetilde{\boldsymbol{z}}_s$ in order to maximize the likelihood of the Gaussian posterior $p(\boldsymbol{z} \mid y, \widetilde{\boldsymbol{z}})$,

$$\widetilde{\boldsymbol{h}} = \text{MLP}([\widetilde{\boldsymbol{z}}; \vec{y}]) \tag{28}$$

$$\widetilde{\boldsymbol{\mu}} = \widetilde{\boldsymbol{W}}_{\mu}\widetilde{\boldsymbol{h}} + \widetilde{\boldsymbol{b}}_{\mu} \tag{29}$$

$$\widetilde{\boldsymbol{\sigma}} = \exp(\widetilde{\boldsymbol{W}}_{\sigma}\widetilde{\boldsymbol{h}} + \widetilde{\boldsymbol{b}}_{\sigma}), \tag{30}$$

where $\widetilde{\boldsymbol{W}}_{\mu}$, $\widetilde{\boldsymbol{W}}_{\sigma}$, and the parameters of the MLP are learnable weights and both $\widetilde{\boldsymbol{b}}_{\mu}$ and $\widetilde{\boldsymbol{b}}_{\sigma}$ are learnable biases. The reconstruction loss is then the negative log-likelihood of the posterior $p(\boldsymbol{x} \mid y, \widetilde{\boldsymbol{z}})$. The log-likelihood is modeled as

$$\log p(\boldsymbol{z} \mid y, \widetilde{\boldsymbol{z}}) = \\ -\frac{1}{2}\left[\log|\widetilde{\Sigma}| + (\boldsymbol{z} - \widetilde{\boldsymbol{\mu}})^{\mathrm{T}}\widetilde{\Sigma}^{-1}(\boldsymbol{z} - \widetilde{\boldsymbol{\mu}}) + \log 2\pi\right] \tag{31}$$

where $\widetilde{\Sigma} = \text{diag}(\widetilde{\boldsymbol{\sigma}})$.

3.6 Model Selection

Since model pretraining on unlabeled data does not involve document labels, it is not clear how to best prime pretrained feature extractors for document classification. Modern architectures used in pretraining optimize for model fit, primarily through held-out perplexity. VAMPIRE, as a document model can also optimize for perplexity.

$\boldsymbol{\theta}_i$ however, is directly interpretable as a mixture over latent topics, thereby making topic coherence a possible alternative means for model selection. In this thesis, I explore topic coherence via NPMI and model fit through negative log-likelihood (NLL) as two distinct stopping criteria for pretraining, and observe which criterion leads to better classifiers.

3.7 Optimization

VAMPIRE and its variants are all optimized using Adam (Kingma and Ba, 2014). To prevent divergence during pretraining, a batchnorm layer is applied to the reconstruction of $\boldsymbol{x}$ (Ioffe and Szegedy, 2015).

Optimizing KL-divergence too early during pretraining can cause the variational distribution to *collapse* into the normal prior. To combat this, I implement KL-annealing (Bowman et al., 2016) by placing a scalar weight on the KL-divergence and gradually increasing this weight from 0 to 1 throughout pretraining.

4 Experimental Setup

For this thesis, four text classification tasks are used to measure the classification performance of VAMPIRE as the amount of labeled data varies from 200 to 10,000 to simulate the low-resource setting. One classification task is used in order to discuss phenomena surrounding JOINT-VAMPIRE and STACKED-VAMPIRE.

The low-resource setting defined in §2.4 does not define in-domain unlabeled data as a scarce resource. Therefore, I assume that an abundant amount of in-domain unlabeled data is available for each task, ranging from 75,000 to 125,000 unlabeled examples that comes from the union of the examples made available as a result of throttling the labeled data, and any additional unlabeled data already provided by the corpus. Given the high level of variance in performance that can arise from severely limiting the amount of labeled data, each experiment of VAMPIRE is run with five random seeds, and the mean performance on test averaged across seeds is reported.

To explore the utility of VAMPIRE when computational resources are limited, model size, time to convergence, and ease of use with the CPU in addition to raw classification performance and topical coherence will also be observed.

Both JOINT-VAMPIRE and STACKED-

Dataset	Label Type	Classes	Documents
AG	topic	4	127600
HATESPEECH	hatespeech	4	99996
IMDB	sentiment	2	100000
YAHOO!	topic	15	150015

Table 2: Summary statistics of the datasets used throughout experimentation.

VAMPIRE are trained via stratified minibatches containing both labeled and unlabeled instance. Originally, minibatches sampled labeled and unlabeled instances indifferently; in later sections, I discuss why naïve sampling of mixed minibatches can lead to severe imbalances labeled to unlabeled instances, which ultimately leads to learning either a degenerate topic model or classifier when optimizing the joint objective.

4.1 Datasets and Preprocessing

Performance on classifiers supplemented with VAMPIRE is tested on a variety of label types across four datasets: AG News (Zhang et al., 2015), IMDB (Maas et al., 2011), the YAHOO! Answers datasets (Chang et al., 2008), as well as a dataset of tweets labeled in terms of four HATESPEECH categories (Founta et al., 2018). In the case study surrounding JOINT-VAMPIRE and STACKED-VAMPIRE, only IMDB is used. Label types and summary statistics for each dataset is included in Table 2.

For the four tasks, the official test set is used if the corpus provides one. Otherwise, a random, stratified sample of 25,000 labeled documents serves as the test set. In either case, a sample of 5,000 instances from the remaining labeled documents is used as a validation set. Each document used throughout training is truncated to 400 words and is then tokenized via spaCy.[3] For all model pretraining, vocabularies are restricted to the 30,000 most frequent words in the corpus, after the exclusion of stopwords,[4] words containing digits or punctuation, and words containing fewer than three characters. Vocabularies are left unrestricted when training classification downstream.

4.2 VAMPIRE Architecture

A two-stage constrained random hyperparameter search is performed on each task for VAMPIRE. Search space, sampling bounds and final assignments for each hyperparameter chosen for VAMPIREs and classifiers used in reported test performance are outlined in Table 7.

STACKED-VAMPIRE and JOINT-VAMPIRE optimize the joint objective, thereby requiring only a single-stage constrained random search over a similar search space. In addition to VAMPIRE's hyperparameters, the two joint models also search over α: the explicit weighting of classification loss. In particular, the search for α consists of choosing from values 1, 20, 50, and 100.

4.3 Downstream Classifiers

For all experiments, I chose to use a Deep Averaging Network (DAN) architecture (Iyyer et al., 2015) to serve as the baseline sequence-to-vector encoder, $f_{s2v}(\boldsymbol{x})$ described in §3.3. Preliminary baseline results from training on subsamples of 200 instances and 10,000 instances from both IMDB andAG are outlined in Figure 7. DANs with one-layer MLPs and moderate dropout consistently outperform more expressive models, such as convolutional or recurrent networks, on validation data with less hyperparameter tuning and variance in performance; this was especially true when labeled data is severely restricted.

Learnable embeddings corresponding to each token are summed and passed as features to a multi-layer perceptron that serves as the classifier. This is often referred to as a *Bag-of-Embeddings* encoder.

$$p(y_i \mid \boldsymbol{x}_i) = \text{MLP}\left(\frac{1}{|\boldsymbol{x}_i|}\sum_{j=1}^{|\boldsymbol{x}_i|} E(\boldsymbol{x}_i)_j\right) \quad (32)$$

where $E(\boldsymbol{x})$ converts a sequence of tokens to a sequence of embeddings that can be initialized arbitrarily. In practice, such embeddings are often initialized with off-the-shelf GLOVE embeddings (Pennington et al., 2014), or contextual embeddings. For this work, embeddings are randomly initialized to allow direct comparison to GLOVE and contextual embeddings.

To evaluate VAMPIRE as a feature extractor for classification, VAMPIRE embeddings will supplement the downstream MLP classifier as additional features via vector concatenation, i.e.,

$$p(y_i \mid \boldsymbol{x}_i) = \text{MLP}\left(\left[\boldsymbol{r}_i; \frac{1}{|\boldsymbol{x}_i|}\sum_{j=1}^{|\boldsymbol{x}_i|} E(\boldsymbol{x}_i)_j\right]\right) \quad (33)$$

[3] https://spacy.io/

[4] http://snowball.tartarus.org/algorithms/english/stop.txt

Case studies on JOINT-VAMPIRE and STACKED-VAMPIRE will also use DANs as the baseline sequence-to-vector encoder of choice. STACKED-VAMPIRE will continue to supplement the MLP with learned VAMPIRE embeddings. However experimentation involving JOINT-VAMPIRE will provide only on the features produced by the DAN to the downstream MLP classifier in order to observe whether learning topic mixtures and classification in tandem is beneficial.

5 Resources and Baselines

I compare VAMPIRE against baselines from two differing environments: the *high-resource* setting and the *low-resource* setting.

The High-resource Setting

In the high-resource setting, I assume plentiful computational resources and massive amounts of out-of-domain data are available in addition to the in-domain data for the target task. Modern approaches in this environment for NLP involve intensive GPU pretraining of large feature extractors, with optimization of the language modeling objective being the canonical approach to learning. For this thesis, I evaluate the performance of two powerful architectures that exemplify these modern methods of the high-resource setting:

I. A Transformer-based ELMO (Peters et al., 2018). As a transformer-variant of ELMO, this baseline learns deep representations using the internal states of a bi-directional transformer, and simulates LM pretraining by masking future tokens when training the forward direction, and previous tokens when training the backward direction.

II. BERT (Devlin et al., 2019) is a bidirectional transformer that, unlike the transformer-variant of ELMO, learns deep representations taking into account the *entire* surrounding context without masking.

More specifically, I use two approaches to evaluate both ELMO and BERT as feature extractors for downstream classification: **(a)** through the use of their pretrained *frozen* embeddings, and **(b)** through embeddings resulting from semi-supervised *fine-tuning* on the unlabeled and labeled in-domain data. I fine-tune ELMO and BERT using their original objectives on the unlabeled data, and a classification objective on the labeled data using a single softmax classification layer. ELMO in particular, is fine-tuned to the labeled data by using an average over LM hidden states as features to the classification layer. Following (Howard and Ruder, 2018), I apply slanted triangular learning rates and gradual unfreezing. To fine-tune BERT, I feed hidden states corresponding to the `[CLS]` tokens of each instance to the classification layer. I use AllenNLP[5] to fine-tune ELMO and Pytorch-pretrained-BERT[6] to fine-tune BERT.

Notably, the high-resource setting enables pretraining of intensive models from a computational standpoint in addition to the abundance of out-of-domain data. This thesis then also evaluates performance of intensive models pretrained on the in-domain data without out-of-domain data in order to directly observe the advantages to be gained from plentiful computational resource alone. More specifically, contextual word embeddings resulting from a Transformer-based ELMO pretrained on the in-domain data serve as the third baseline of the high-resource setting. Pretrained in-domain ELMO embeddings are frozen and concatenated to randomly initialized word embeddings in order to produce a direct comparison VAMPIRE, differing only in model architecture and computational resource.

The Low-resource Setting

In the low-resource setting, I assume that unlabeled data is plentiful, but labeled data and computational resources come at a premium. For some baselines, I also assume the unavailability of massive amounts of out-of-domain data, which often occurs for tasks that are beyond commonly supported languages such as English. In this environment, researchers are constrained to lightweight approaches that must run efficiently on the CPU, while making the most out of the limited amount of available labeled data. This thesis considers four baselines for the low-resource setting:

I. A purely supervised model using 50-dimensional randomly initialized word embeddings with no access to unlabeled data .

[5] https://allennlp.org/elmo

[6] https://github.com/huggingface/pytorch-pretrained-BERT

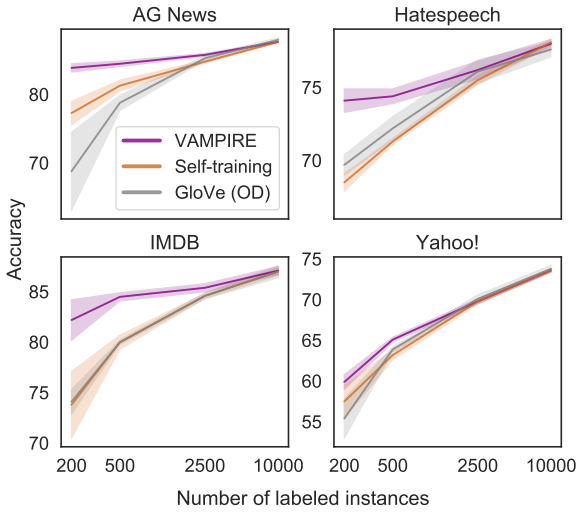

Figure 3: Learning curves for all datasets in the low-resource setting, showing the mean (line) and one standard deviation (bands) over five runs for VAMPIRE, self-training, and 840B-token GLOVE embeddings. Full results are in Table 3.

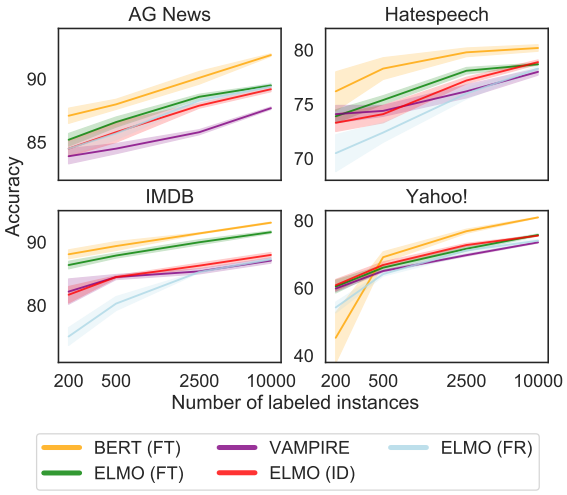

Figure 4: Comparison between high-resource methods and VAMPIRE on four datasets; ELMO performance benefits greatly from training on (ID), or fine-tuning (FT) to, the in-domain data (as does BERT; full results in Appendix 5). Key: FT (fine-tuned), FR (frozen), ID (in-domain).

II. A semi-supervised model trained on the labeled data using 300-dimensional GLOVE-embeddings, which were pretrained on 840 billion words.[7]

III. The same model as II. with the exception that GLOVE-embeddings are pretrained on only in-domain data.

IV. Self training using both unlabeled and labeled in-domain data.

Self training is achieved by iteratively training I., predicting labels on all unlabeled instances, and augmenting the training set with all unlabeled instances whose label is predicted with high confidence. This is repeated this up to five times using the model with highest validation accuracy. On each iteration, the threshold for a given label is equal to the 90th percentile of predicted probabilities for validation instances with the corresponding label.

6 Results

Early experimentation showed that the classification performance of JOINT-VAMPIRE equated roughly to random guessing, and that STACKED-VAMPIRE, while providing strong classification performance, could not compete with VAMPIRE. For these reasons, I report classification performance only for VAMPIRE, and defer discussion of the shortcomings of JOINT-VAMPIRE and STACKED-VAMPIRE to §7.

In the **low-resource** setting, VAMPIRE achieves the highest classification accuracy of all low-resource methods considered. Differences between VAMPIRE and baseline models in this environment are especially pronounced when labeled data is severely restricted. Table 3 shows downstream classification performance for all low-resource methods on each dataset. Figure 3 displays the learning curves that result from varying the amount of labeled data.

In the **high-resource** setting, fine-tuned pretrained BERT outperforms all other high-resource methods downstream. Notably, for both ELMO and BERT, classification performance suffers dramatically when only pretrained frozen embeddings are used when compared to their fine-tuned counterparts. These discrepancies, displayed in Figure 4, are especially pronounced in HATESPEECH and IMDB, where ELMO models trained only on in-domain data offered superior performance to that of the frozen ELMO embeddings pretrained on out-of-domain data. These differences however, are less pronounced in YAHOO! and AG.

These results demonstrate the importance of in-domain data to strong performance downstream.

[7] http://nlp.stanford.edu/projects/glove/

Dataset	Model	200	500	2500	10000
IMDB	Baseline	68.5 (7.8)	79.0 (0.4)	84.4 (0.1)	87.1 (0.3)
	Self-training	73.8 (3.3)	80.0 (0.7)	84.6 (0.2)	87.0 (0.4)
	GLOVE (ID)	74.5 (0.8)	79.5 (0.4)	84.7 (0.2)	87.1 (0.4)
	GLOVE (OD)	74.1 (1.2)	80.0 (0.2)	84.6 (0.3)	87.0 (0.6)
	VAMPIRE	**81.1** (0.9)	**84.0** (0.7)	**85.7** (1.4)	**87.7** (0.4)
AG	Baseline	68.8 (2.0)	77.3 (1.0)	84.4 (0.1)	87.5 (0.2)
	Self-training	77.3 (1.7)	81.3 (0.8)	84.8 (0.2)	87.7 (0.1)
	GLOVE (ID)	70.4 (1.2)	78.0 (1.0)	84.1 (0.3)	87.1 (0.2)
	GLOVE (OD)	68.8 (5.7)	78.8 (1.1)	85.3 (0.3)	**88.0** (0.3)
	VAMPIRE	**83.9** (0.5)	**84.8** (0.2)	**86.1** (0.1)	87.9 (0.2)
YAHOO!	Baseline	54.5 (2.8)	63.0 (0.5)	69.5 (0.3)	73.6 (0.2)
	Self-training	57.5 (2.0)	63.2 (0.6)	69.8 (0.3)	73.6 (0.2)
	GLOVE (ID)	55.2 (2.3)	63.5 (0.3)	69.7 (0.3)	73.5 (0.3)
	GLOVE (OD)	55.4 (2.4)	63.9 (0.3)	**70.1** (0.5)	**73.8** (0.4)
	VAMPIRE	**59.2** (1.6)	**65.0** (0.4)	69.7 (0.4)	73.7 (0.3)
HATESPEECH	Baseline	67.7 (1.8)	71.3 (0.2)	75.6 (0.4)	77.8 (0.2)
	Self-training	68.5 (0.6)	71.3 (0.2)	75.5 (0.3)	78.1 (0.2)
	GLOVE (ID)	69.7 (1.2)	71.9 (0.5)	76.0 (0.3)	**78.3** (0.2)
	GLOVE (OD)	69.7 (0.7)	72.2 (0.8)	76.1 (0.8)	77.6 (0.5)
	VAMPIRE	**74.6** (0.6)	**75.2** (0.6)	**76.9** (0.3)	**78.3** (0.2)

Table 3: Test accuracies in the low-resource setting on four text classification datasets under varying levels of labeled training data (200, 500, 2500, and 10000 documents). Each score is reported as an average over five seeds, with the highest mean result in each setting shown in bold followed by standard deviation in parenthesis.

Models pretrained on unlabeled, out-of-domain data, in both the high-resource and low-resource settings, have been proven to provide gains over purely supervised models alone. However, it is now also clear that semi-supervised methods can be significantly improved upon by including in-domain data, whether it be through fine-tuning a pretrained language model in the high-resource setting, or for training VAMPIRE in the low-resource setting. Previous work has also shown this to be true for other tasks, such as natural language inference and and question answering (Yogatama et al., 2019).

7 Analysis

7.1 NPMI vs. NLL

To measure the effectiveness of topic coherence as an alternative means of model selection, I pretrain 200 instances of VAMPIRE on IMDB; 100 models selected via NPMI, and 100 models selected via NLL, using the validation set for both criteria. Figure 5 depicts the relationships between NPMI, NLL, and downstream classification accuracy. In Figure 5A, NPMI and NLL are shown to be negatively correlated (ρ = -0.72), suggesting that models that more closely fit the data tend to have more coherent topics.

I then use the pretrained instances of VAMPIRE to train 200 downstream classifiers with identical hyperparameters in order to observe which stopping criteria leads to better classification performance. Figures 5B and 5C suggest that better-trained VAMPIRE instances, according to *either* criterion, leads to better classification downstream, with $\rho = 0.55$ and $\rho =$

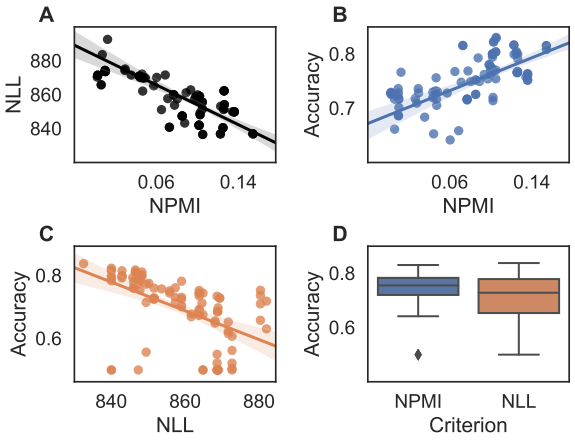

Figure 5: Comparing NPMI and NLL as early stopping criteria for VAMPIRE model selection. NPMI and NLL are correlated measures of model fit, but NPMI-selected VAMPIRE models have lower variance on downstream classification performance with 200 labeled documents of IMDB. Accuracy is reported on the validation data. See Section 7.1 for more details.

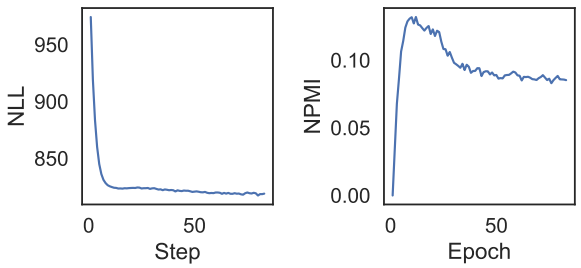

Figure 6: An example learning curve when training VAMPIRE on the IMDB dataset. If trained for too long, we observe many cases in which NPMI (higher is better) degrades while NLL (lower is better) continues to decrease. To avoid selecting a model that has poor topic coherence, we recommend performing model selection with NPMI rather than NLL.

-0.53 for NPMI and NLL, respectively. However, models selected using NLL were shown to have higher variance in accuracy downstream. Figure 5 suggests that although optimal models from either criterion are closely related in performance, NPMI models proved to be more reliable. NLL models were also shown to have less coherent topics after converging in addition to the high variance in classification performance. Because of this, all reported metrics from Table 3 use instances of VAMPIRE selected using NPMI.

Prior work has shown that when training neural topic models, NPMI and NLL improve together during the first few epochs, but NPMI is left behind late into training, and often suffers dramatically as NLL reaches convergence (Ding et al., 2018). The work in this thesis confirms these findings in that training VAMPIRE to convergence via NLL also leads to NPMI-deteriorating. I depict this explicitly in Figure 6.

7.2 Latent Topics Learned by VAMPIRE

Qualitative analysis of the latent topics learned by VAMPIRE further demonstrates its ability as an unsupervised feature extractor. Each $\boldsymbol{\theta}_i$ produced by VAMPIRE as an intermediate representation of the final embedding can be explicitly interpreted as topic mixture proportions, thereby making the topics a representative summary of what was learned from pretraining. Topics from IMDB and YAHOO! are shown in Table 1 and Appendix 2.

Model	Parameters	Time
VAMPIRE (GPU)	3.8M	10 min
VAMPIRE (CPU)	3.8M	20 min
ELMO (GPU)	159.2M	12 hr 35 min

Table 4: VAMPIRE is substantially more compact than Transformer-based ELMO but is still competitive under low-resource settings. Here, I report computational requirements for pretraining VAMPIRE and ELMO on in-domain unlabeled text from the IMDB dataset. A GeForce GTX 1080 Ti GPU is used for all GPU results. Results for training VAMPIRE are also reported using a 2.60GHz Intel Xeon CPU. VAMPIRE uses about 743MB of memory on a GPU, while ELMO requires roughly 8.5GB.

7.3 Learnable Scalar Weights for Deep Representations

Recall from Eq. 18 that VAMPIRE embeddings are a weighted average between the normalized topic mixture proportion vector $\boldsymbol{\theta}_i$ and internal states of the n-layer MLP applied to the empirical word counts using averaging weights $\lambda_0, \lambda_1, ...\lambda_n$. Since each weight λ_i is learned when training the classifier downstream, I investigate which layers in particular are most amenable to learning a discriminative classifier by observing these weights throughout training.

In particular, the first layer of the MLP, $\boldsymbol{h}_{i1}$ along with the normalized topic proportions $\boldsymbol{\theta}_i$ were consistently shown to be of high importance to the downstream classifier with intermediate layers being severely downweighted. However, it was also observed that a multi-layer MLP in the VAE leads to larger gains in classification performance downstream. Because of this, I continue to use multi-layer MLPs in the VAE, but to improve learning, I heavily upweight averaging weights corresponding to $\boldsymbol{\theta}_i$ and $\boldsymbol{h}_{i1}$ during initialization.

7.4 Computational Requirements of VAMPIRE

Through its simplicity, VAMPIRE achieves a significantly more compact representation compared to modern architectures used in the high-resource setting. Table 4 depicts the computational requirements for training VAMPIRE and ELMO on IMDB using a GPU. To further illustrate the compactness of VAMPIRE, benchmarks for training VAMPIRE on IMDB with a CPU in addition to its GPU performance is also reported.

I find that it is possible to train instances

of VAMPIRE orders of magnitude faster than a Transformer-based ELMO using significantly fewer parameters while still being competitive when both architectures are trained under low-resource settings. VAMPIRE is especially motivated in the low-resource setting given that it can be trained efficiently on the CPU.

7.5 Insights from Joint Training

Early experimentation with JOINT-VAMPIRE and STACKED-VAMPIRE deemed the joint objective unfit to compete against baselines from both the low-resource and high-resource settings as an effective method for semi-supervised learning. Here, I discuss the shortcomings of the joint objective, deep generative document models, and several phenomena surrounding the optimization of these unique models.

Imbalances Discovered in JOINT-VAMPIRE

Initially, JOINT-VAMPIRE was trained using stratified minibatches sampled randomly without replacement from a *combined* training set of both labeled and unlabeled instances. More specifically, combined training sets were built from identical subsamples of labeled data for IMDB outlined in Table 3 for direct comparison to VAMPIRE. With this naïve sampling scheme came the issue of labeled instances being severely outnumbered by unlabeled instances, resulting in the majority of minibatches containing roughly 1-2 labeled instances on average. Because of this, classifiers trained with the joint objective under this scheme have a tendency to overfit given that so few examples contribute to each update.

In an attempt to combat this, rather than sampling labeled and unlabeled instances purely at random from a combined dataset, I instead used stratified sampling to collect a fixed number of labeled instances from a restricted set of labeled data with replacement, while sampling the remaining portion of the minibatch as unlabeled instances from an unrestricted set of unlabeled data without replacement. With this new sampling scheme, the ratio of labeled to unlabeled instances was more balanced, while adhering to the limitations that come with the low-resource setting. However, despite intensive random hyperparameter search over 100 different configurations of JOINT-VAMPIRE, the joint objective still failed to learn a model that could discriminate documents beyond random guessing.

Balancing Two Distinct Objectives

The shortcomings of JOINT-VAMPIRE could also be attributed to its sensitivity to α, the weight of the classification objective in Eq. 26. For low values of α on the order of 1 to 50, the joint objective learns to ignore the classification objective, and instead prioritizes reconstruction. Similarly, for high values of α upwards of 100-200, the model quickly disregards the reconstruction objective early in training in favor of learning the discriminative classifier. To my knowledge, no compromise in α exists to prevent models from reprioritizing to a single objective.

Experimentation with the joint objective was done in the hopes that gradients computed for the classifier from its role in the reconstruction of x (Eq. 23) would aid in its learning to discriminate. It is through this objective that the classifier is given a glimpse as to how documents are constructed, which may hold clues regarding the origin of document labels. However, given that neither an optimal ratio of labeled to unlabeled data nor an optimal value of α could be found through intensive hyperparameter search, it is unclear if models can jointly learn to reconstruct and discriminate documents when working with limited amounts of labeled data.

Label-specific Topics from JOINT-VAMPIRE

A fortunate aspect of JOINT-VAMPIRE is its ability to to learn label-specific topic deviations $\boldsymbol{C} \in \mathbb{R}^{|\mathcal{Y}| \times |V|}$ (Eq. 24). Labeled instances in IMDB are annotated with 0 for negative examples and 1 for positive examples. When trained on IMDB in the low resource setting, JOINT-VAMPIRE correctly upweights tokens with negative connotation such as 'garbage', 'boring', 'trash' in the portion of $\boldsymbol{C}$ corresponding to the negative label, $\boldsymbol{C}_0$. It also upweights words with positive connotation such as 'inspiring', 'oscar', and 'award-winning' to the topic deviations corresponding to the positive label, $\boldsymbol{C}_1$. Although the joint objective failed to learn an effective classifier, the ability to learn label-specific topics is promising, and future work should consider this when re-visiting document classification that use NVDMs downstream.

The Shortcomings of STACKED-VAMPIRE

Recall that STACKED-VAMPIRE, as a deep generative model, can be best thought of as an instance of JOINT-VAMPIRE that aims to encode and recon-

struct VAMPIRE representations while simultaneously learning a discriminative classifier. In particular, unnormalized topic proportion mixtures z_i provided by VAMPIRE are now the target of reconstruction, thereby making STACKED-VAMPIRE a deep generative model that learns a latent variable that is *stacked* onto that learned by VAMPIRE. Because of this, the latent space learned by STACKED-VAMPIRE cannot be directly interpreted as topics, making NLL the only means of model selection. This trade-off was made in an attempt to unburden the joint objective from having to learning topics outright, to observe whether complex interactions between NVI and classification components become beneficial beginning from this more advantageous initialization.

Using pretrained instances of VAMPIRE trained with 64 topics, initial experimentation with STACKED-VAMPIRE halved the latent space to 32 dimensions. To further improve classification performance, pretrained VAMPIRE embeddings were also provided to the classification component of STACKED-VAMPIRE as supplemental features. Similar to JOINT-VAMPIRE, I perform random hyperparameter search over 100 configurations STACKED-VAMPIRE, including searches over the classification loss α as well as labeled to unlabeled data ratios per minibatches.

As a result, STACKED-VAMPIRE produces large gains compared to the joint objective alone. However, it still fails to outperform the pipelined approach of pretraining VAMPIRE and training the downstream classifier separately, often coming to within sub-percent differences in classification performance across several subsamples of IMDB. In other words, STACKED-VAMPIRE performance tends to be either similar or 1% below that of VAMPIRE. Stacked models that perform similarly to VAMPIRE often had identical hyperparameters for the downstream classifiers, and similar batch sizes for labeled instances.

8 Conclusion

The revival of semi-supervised NLP through powerful, pretrained models has sparked advancement for nearly every language task. Models like ELMO and BERT re-defined what is possible in NLP, and have since served as the strongest motivators for the utility of unlabeled data. Despite what they achieve, they come at a significant cost, and they neglect consideration of the low-resource setting and alternative learning methods through their reliance on abundant data and the language modeling objective. In this work, I addressed this by exploring alternative learning objectives using variational methods, with thorough consideration of the low-resource and high-resource settings, using semi-supervised text classification as a case study.

By introducing VAMPIRE as a lightweight, pretraining framework for this task, I show that variational methods can produce competitive unsupervised feature extractors in settings where ELMO and BERT are infeasible to train. Through VAMPIRE, I adapt variational document modeling to modern approaches in transfer learning, and demonstrate that optimizing for topic coherence in place of model fit reliably results in features useful for strong classification performance downstream.

This thesis also observed phenomena surrounding *joint* semi-supervision. For now, in regards to document modeling and classification, it remains inconclusive as to how best approach balancing the two warring objectives. Results from JOINT-VAMPIRE demonstrated that there is no mutual benefit from complex interaction between topics and a discriminative classifier through joint training, and further results from STACKED-VAMPIRE show this to be true even for joint models that begin with more advantageous initialization in the form of pretrained VAMPIRE representations. Taking the results of VAMPIRE, JOINT-VAMPIRE and STACKED-VAMPIRE together, I find that downstream classifiers benefit much more greatly from using the learned document representations downstream rather than being involved in reconstruction throughout training.

Acknowledgments

To begin, I want to acknowlegdge my incredible VAMPIRE co-authors. Thanks to Noah Smith, for making this research possible as my advisor, and for the invitation to join his ever-growing cohort of ARK undergrads. Before I joined the ARK, he took the time to meet with me to discuss in length, the levels of influence that engineers and researchers had on the world. His perspective made me reflect about the kind of work I wanted to do, and I left his office convinced that I wanted to do it all; I wanted my work to be incredible and cutting edge, but at the same time grounded in pragmatism and practicality. With the culmination of my research ending with VAMPIRE, I could not be more grateful to him for my time in the ARK, and for helping me discover the kind impact I want to make in NLP and computer science.

Thanks to Dallas Card, my designated ARK PhD student. Looking back now, and recalling the fact that I didn't even know what the word "latent" was during our first meeting, (a word I have used dozens of times in this thesis), it is incredible how accepting he was of me as his mentee, and how far he has taken me since. He pushed me to be great and to always do good science, and happily explained complex concepts as well as the basics when I needed it. I don't think he knows how great of a mentor he really is, and I hope he continues to mentor up-and-coming NLP practitioners as a postdoc at Stanford.

Thanks to Suchin Gururangan, for being a better partner during the engineering of VAMPIRE than I could have ever hoped for. Because of him, I now know that to train a good topic model, one need only look for Batman and dog orthopedics.

This thesis would not have been possible without the people that helped me discover my calling in CSE. Thanks to Dan Grossman; I couldn't have succeeded as a petitioner in his offering of CSE 341 without his encouragement and his "fatherly" sense of humor. Thanks to Maggie Morris, for being the first person in the CSE department to believe in me. Lastly, and begrudgingly, I thank my sister, Natalie, for being right about CSE being the best fit for me. Special thanks to Dallas Card, Suchin Gururangan, Noah Smith, and Joseph Zhong for graciously accepting to revise this thesis. Thank you to the reading committee: Richard Anderson, Jenifer Hiigli, and Noah Smith, for their careful evaluation of this work.

Finally, thanks to my family. To my Mom and Dad, thank you for the incredible sacrifices you have made for me, so that I could pursue my education to my fullest extent. To my sisters, Natalie and Quyên, thank you for your encouragement, and for the unwavering faith you have in me.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. In *Proceedings of NeurIPS*.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. In *Proceedings of NeurIPS*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of CoNLL*.

Dallas Card, Chenhao Tan, and Noah A. Smith. 2018. Neural models for documents with metadata. In *Proceedings of ACL*.

Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of NeurIPS*.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *Proceedings of AAAI*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Ran Ding, Ramesh Nallapati, and Bing Xiang. 2018. Coherence-aware neural topic modeling. In *Proceedings of EMNLP*.

Antigoni Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of Twitter abusive behavior. In *Proceedings of AAAI*.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. 2019. Variational pretraining for semi-supervised text classification. In *Proceedings of ACL*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of ACL*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings ICML*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. *CoRR*, abs/1312.6114.

John D. Lafferty and David M. Blei. 2006. Correlated topic models. In *Proceedings of NeurIPS*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of EACL*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural variational inference for text processing. *CoRR*, abs/1511.06038.

David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*.

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

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of ACL*.

Akash Srivastava and Charles A. Sutton. 2017. Autoencoding variational inference for topic models. In *Proceedings of ICLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NeurIPS*.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *Proceedings of AAAI*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of ICML*.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of NeurIPS*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Ryan Cotterell, Vicente Ordonez, and Kai-Wei Chang. 2019. Gender bias in contextualized word embeddings. In *Proceedings of NAACL*.

A Appendices

The Variational Lower Bound

Recall from §2.2 that modeling documents $\boldsymbol{x}_i$ directly via a latent variable $\boldsymbol{z}_i$ using variational inference results in integration that is infeasible to compute. Below, I derive the variational lower bound: an approximation of the otherwise intractable integral used widely in NVI.

Suppose we would like to model the likelihood of a document, $p(\boldsymbol{x}_i)$. Through the introduction of a latent variable $\boldsymbol{z}_i$, we may do so by computing the marginal likelihood,

$$p(\boldsymbol{x_i}) = \int_{\boldsymbol{z}_i} p(\boldsymbol{x}_i, \boldsymbol{z}_i)\, d\boldsymbol{z}_i \tag{34}$$

Note that the logarithm as a function is monotonically increasing; maximizing the logarithm of an objective is then equivalent to directly maximizing the objective,

$$\log p(\boldsymbol{x_i}) = \log \int_{\boldsymbol{z}_i} p(\boldsymbol{x}_i, \boldsymbol{z}_i)\, d\boldsymbol{z}_i. \tag{35}$$

Allow $q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$ be the *variational distribution* that models the latent variable. We change nothing when multiplying the objective by $\frac{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}$,

$$\log p(\boldsymbol{x_i}) = \log \int_{\boldsymbol{z}_i} \frac{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)} \cdot p(\boldsymbol{x}_i, \boldsymbol{z}_i)\, d\boldsymbol{z}_i. \tag{36}$$

However, through this, it is now possible to take the expectation with respect to the variational distribution,

$$\log p(\boldsymbol{x_i}) = \log \mathbb{E}_{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}\left[\frac{p(\boldsymbol{x}_i, \boldsymbol{z}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)}\right] \tag{37}$$

By *Jensen's Inequality*, for any convex function f and random variable X, we find that $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$. Since the natural log is concave, we leverage Jensen's Inequality to push it into the expectation:

$$\begin{aligned}
&\log p(\boldsymbol{x_i}) = \log \mathbb{E}_{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}\left[\frac{p(\boldsymbol{x}_i, \boldsymbol{z}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}\left[\log \frac{p(\boldsymbol{x}_i, \boldsymbol{z}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)}\right] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}\Big[\log p(\boldsymbol{x}_i, \boldsymbol{z}_i) - \log q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)\Big] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}\Big[\log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) + \log p(\boldsymbol{z}_i) \\
&\qquad\qquad - \log q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)\Big] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i|\boldsymbol{x}_i)}\left[\log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) + \log \frac{p(\boldsymbol{z}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)}\right]
\end{aligned} \tag{38}$$

Allowing $p(\boldsymbol{z})$ to serve as the prior for the latent variable, we find that the final term in the expectation is the definition of *KL-divergence*, the relative entropy between two distributions measured by their log ratio in expectation. The variational lower bound $\mathcal{B}(\boldsymbol{x}_i)$ is then as follows:

$$\begin{aligned}
\log(\boldsymbol{x}_i) \geq \mathcal{B}(\boldsymbol{x}_i) = \mathbb{E}_{q(z_i|\boldsymbol{x}_i)}\Big[\log p(\boldsymbol{x}_i \mid z_i)\Big] \\
-\mathrm{KL}\Big[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \parallel p(\boldsymbol{z})\Big].
\end{aligned} \tag{39}$$

Model parameters can optimize the above objective using gradient descent methods. There exists a closed form solution for the KL-divergence, $\mathrm{KL}\Big[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \parallel p(\boldsymbol{z})\Big]$, when the prior and posterior are assumed to be normal (Kingma and Welling, 2014),

$$\begin{aligned}
&-\mathrm{KL}\Big[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \parallel p(\boldsymbol{z})\Big] \\
&= \int_{\boldsymbol{z}_i} q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \cdot \frac{\log p(\boldsymbol{z}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)}\, d\boldsymbol{z}_i \\
&= \int_{\boldsymbol{z}_i} q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \cdot (\log p(\boldsymbol{z}_i) - q(\boldsymbol{z}_i \mid \boldsymbol{x}_i))\, d\boldsymbol{z}_i \\
&= \sum_{i=1}^{K} 1 + \log(\boldsymbol{\sigma}_i^2) - \boldsymbol{\mu}_i^2 - \boldsymbol{\sigma}_i^2
\end{aligned} \tag{40}$$

where K is the number of topics. VAMPIRE and each of its variants assume a normal prior for topic mixture proportions and computes the KL-diverence using Eq. 40.

The Joint Variational Lower Bound

The joint equivalent of Eq. 39 follows naturally from introducing categorical labels $y \in \mathcal{Y}$ into the variational lower bound. By assuming each document has a label y_i, and that y_i and $\boldsymbol{z}_i$ are unconditionally independent, I define the joint variational lower bound as an approximation of $p(\boldsymbol{x}_i, y_i)$,

$$
\begin{aligned}
&\log p(\boldsymbol{x}_i, y_i) = \log \int_{\boldsymbol{z}_i} p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i)\, d\boldsymbol{z}_i \\
&= \log \int_{\boldsymbol{z}_i} q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i) \frac{p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)}\, d\boldsymbol{z}_i \\
&= \log \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \left[\frac{p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \right] \\
&\geq \mathcal{B}_L(\boldsymbol{x}_i, y_i) = \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \left[\log \frac{p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i)}{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \right] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \Big[\log p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i) - \log q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i) \Big] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \Big[\log p(\boldsymbol{x}_i \mid y_i, \boldsymbol{z}_i) + \log p(y_i) \\
&\qquad + \log p(\boldsymbol{z}_i) - \log q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i) \Big] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \Big[\log p(\boldsymbol{x}_i \mid y_i, \boldsymbol{z}_i) + \log p(y_i) \Big] \\
&\qquad - \mathrm{KL} \Big[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i) \parallel p(\boldsymbol{z}_i) \Big]
\end{aligned}
\tag{41}
$$

The KL-divergence is still computed via Eq. 40.

I define $\mathcal{B}_U(\boldsymbol{x}_i)$ to be the joint variational bound when the label y_i is not observed. When the categorical label is missing, it is treated as another latent variable. The expectation is then taken to compute the marginal likelihood of $\boldsymbol{x}_i$,

$$
\begin{aligned}
&\log p(\boldsymbol{x}_i) = \log \int_{y_i} \int_{\boldsymbol{z}_i} p(\boldsymbol{x}_i, y, \boldsymbol{z}_i)\, d\boldsymbol{z}_i\, dy_i \\
&= \log \int_{y_i} \int_{\boldsymbol{z}_i} q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i) \frac{p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i)}{q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i)}\, d\boldsymbol{z}_i\, dy_i \\
&= \log \mathbb{E}_{q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i)} \left[\frac{p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i)}{q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i)} \right] \\
&\geq \mathbb{E}_{q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i)} \left[\log \frac{p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i)}{q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i)} \right] \\
&= \mathbb{E}_{q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i)} \Big[\log p(\boldsymbol{x}_i, y_i, \boldsymbol{z}_i) - \log q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i) \Big] \\
&= \mathbb{E}_{q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i)} \Big[\log p(\boldsymbol{x}_i \mid y_i, \boldsymbol{z}_i) + \log p(y_i) \\
&\qquad + \log p(\boldsymbol{z}_i) - \log q(y_i, \boldsymbol{z}_i \mid \boldsymbol{x}_i) \Big] \\
&= \sum_{y_i \in \mathcal{Y}} q(y_i \mid \boldsymbol{x}_i) \cdot \Bigg(\mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \Big[\log p(\boldsymbol{x}_i \mid y_i, \boldsymbol{z}_i) \\
&\qquad + \log p(y_i) + \log p(\boldsymbol{z}_i) - \log q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i) \Big] \\
&\qquad - \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \Big[\log q(y_i \mid \boldsymbol{x}_i) \Big] \Bigg) \\
&= \sum_{y_i \in \mathcal{Y}} q(y_i \mid \boldsymbol{x}_i) \cdot \Big(\mathcal{B}(\boldsymbol{x}_i, y_i) - \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i, y_i)} \Big[\log q(y_i \mid \boldsymbol{x}_i) \Big] \Big) \\
&= \sum_{y_i \in \mathcal{Y}} q(y_i \mid \boldsymbol{x}_i) \cdot \big(\mathcal{B}(\boldsymbol{x}_i, y_i) - \log q(y_i \mid \boldsymbol{x}_i) \big) \\
&= \sum_{y_i \in \mathcal{Y}} q(y_i \mid \boldsymbol{x}_i) \cdot \mathcal{B}(\boldsymbol{x}_i, y_i) - q(y_i \mid \boldsymbol{x}_i) \log q(y_i \mid \boldsymbol{x}_i) \\
&= \sum_{y_i \in \mathcal{Y}} q(y_i \mid \boldsymbol{x}_i) \cdot \mathcal{B}(\boldsymbol{x}_i, y_i) - \sum_{y_i \in \mathcal{Y}} q(y_i \mid \boldsymbol{x}_i) \log q(y_i \mid \boldsymbol{x}_i) \\
&= \Big[\sum_{y_i \in \mathcal{Y}} q(y_i \mid \boldsymbol{x}_i) \cdot \mathcal{B}(\boldsymbol{x}_i, y_i) \Big] + \Big[\mathcal{H}(q(y_i \mid \boldsymbol{x}_i)) \Big]
\end{aligned}
\tag{42}
$$

where $\mathcal{H}$ is Shannon entropy.

Dataset	Model	200	500	2500	10000
IMDB	ELMo (FR)	75.1 (1.4)	80.3 (1.1)	85.3 (0.1)	87.3 (0.3)
	BERT (FR)	81.5 (1.0)	83.9 (0.4)	86.8 (0.3)	88.2 (0.3)
	ELMo (ID)	81.7 (1.3)	84.5 (0.2)	86.3 (0.4)	88.0 (0.4)
	VAMPIRE	82.2 (2.0)	84.5 (0.4)	85.4 (0.4)	87.1 (0.4)
	ELMo (FT)	86.4 (0.6)	87.9 (0.4)	90.0 (0.4)	91.6 (0.2)
	BERT (FT)	**88.1** (0.7)	**89.4** (0.7)	**91.4** (0.1)	**93.1** (0.1)
AG	ELMo (FR)	84.5 (0.5)	85.7 (0.5)	88.3 (0.2)	89.4 (0.3)
	BERT (FR)	84.6 (1.1)	85.7 (0.7)	88.0 (0.4)	89.0 (0.3)
	ELMo (ID)	84.5 (0.6)	85.8 (0.8)	87.9 (0.2)	89.2 (0.2)
	VAMPIRE	83.9 (0.6)	84.5 (0.4)	85.8 (0.2)	87.7 (0.1)
	ELMo (FT)	85.2 (0.5)	86.6 (0.4)	88.6 (0.2)	89.5 (0.1)
	BERT (FT)	**87.1** (0.6)	**88.0** (0.4)	**90.1** (0.5)	**91.9** (0.1)
YAHOO!	ELMo (FR)	54.3 (1.6)	64.2 (0.6)	71.2 (1.3)	74.1 (0.3)
	BERT (FR)	57.0 (1.3)	64.2 (0.5)	70.0 (0.3)	73.8 (0.2)
	ELMo (ID)	**60.9** (1.7)	66.9 (0.9)	72.8 (0.5)	75.6 (0.1)
	VAMPIRE	59.9 (0.9)	65.1 (0.3)	69.8 (0.3)	73.6 (0.2)
	ELMo (FT)	60.5 (1.9)	66.1 (0.7)	71.7 (0.7)	75.8 (0.3)
	BERT (FT)	45.3 (7.5)	**69.2** (1.6)	**76.9** (0.6)	**81.0** (0.1)
HATESPEECH	ELMo (FR)	70.5 (1.7)	72.4 (0.9)	76.0 (0.5)	78.3 (0.2)
	BERT (FR)	75.1 (0.6)	76.3 (0.3)	77.8 (0.4)	79.0 (0.2)
	ELMo (ID)	73.3 (0.8)	74.1 (0.8)	77.2 (0.3)	78.9 (0.2)
	VAMPIRE	74.1 (0.8)	74.4 (0.5)	76.2 (0.6)	78.0 (0.3)
	ELMo (FT)	73.9 (0.6)	75.4 (0.4)	78.1 (0.3)	78.7 (0.1)
	BERT (FT)	**76.2** (1.8)	**78.3** (1.0)	**79.8** (0.4)	**80.2** (0.3)

Table 5: Results in the high-resources setting.

YAHOO!

Canine Care	Networking	Multiplayer Gaming	Harry Potter
training	wireless	multiplayer	dumbledore
obedience	homepna	massively	longbottom
schutzhund	network	rifle	hogwarts
housebreaking	verizon	cheating	malfoy
iliotibial	phone	quake	weasley
crate	blackberry	warcraft	rubeus
ligament	lan	runescape	philosopher
orthopedic	telephone	socom	albus
fracture	bluetooth	fortress	hufflepuffs
gait	broadband	duel	trelawney

Nutrition	Baseball	Sexuality	Religion
nutritional	baseball	homophobia	islam
obesity	sox	heterosexuality	jesus
weight	yankees	orientation	isaiah
bodybuilding	rodriguez	transsexuality	semitism
anorexia	gehrig	cultures	christian
diet	cardinals	transgender	baptist
malnutrition	astros	polyamory	jewish
nervosa	babe	gay	prophet
gastric	hitter	feminism	commandments
watchers	sosa	societal	god

Table 6: Example topics learned by VAMPIRE in the YAHOO! dataset.

Computing Infrastructure	GeForce GTX 1080 GPU
Number of search trials	Per dataset: 60 pretraining trials, 100 classifier trials
Search strategy	uniform sampling
Model implementation	http://github.com/allenai/vampire

Hyperparameter	**Search space**	IMDB	AG	YAHOO!	HATESPEECH
number of epochs	50	50	50	50	50
patience	5	5	5	5	5
batch size	64	64	64	64	64
KL divergence annealing	*choice*[sigmoid, linear, constant]	linear	linear	linear	sigmoid
KL annealing sigmoid weight 1	0.25	N/A	N/A	N/A	0.25
KL annealing sigmoid weight 2	15	N/A	N/A	N/A	15
KL annealing linear scaling	1000	1000	1000	1000	N/A
VAMPIRE hidden dimension	*uniform-integer*[32, 128]	72	107	121	94
Number of encoder layers	*choice*[1, 2, 3]	2	3	3	2
Encoder activation	*choice*[relu, tanh, softplus]	softplus	softplus	tanh	softplus
Mean projection layers	1	1	1	1	1
Mean projection activation	linear	linear	linear	linear	linear
Log variance projection layers	1	1	1	1	1
Log variance projection activation	linear	linear	linear	linear	linear
Number of decoder layers	1	1	1	1	1
Decoder activation	linear	linear	linear	linear	linear
apply batchnorm	True	True	True	True	True
z-dropout	*random-uniform*[0, 0.5]	0.1	0.29	0.32	0.37
learning rate optimizer	Adam	Adam	Adam	Adam	Adam
learning rate	*loguniform-float*[1e-4, 1e-2]	0.00045	0.0026	0.00050	0.00016
update background frequency	*choice*[True, False]	False	False	False	False
vocabulary size	30000	30000	30000	30000	30000

Dataset	VAMPIRE **NPMI**	**Best Validation Accuracy**
IMDB	0.092	0.846
AG	0.202	0.856
YAHOO!	0.428	0.624
HATESPEECH	0.059	0.753

Table 7: VAMPIRE search space, best assignments, and associated performance on the four datasets we consider in this work. This search strategy is two-staged. First we pretrain VAMPIRE, sampling its hyperparameters with these bounds. Then we train fixed downstream classifiers on random 200-document subsets of each dataset, uniformly sampling over the pretrained VAMPIRE models as additional features. We select the VAMPIRE hyperparameters that produce the best downstream validation accuracy for each dataset.

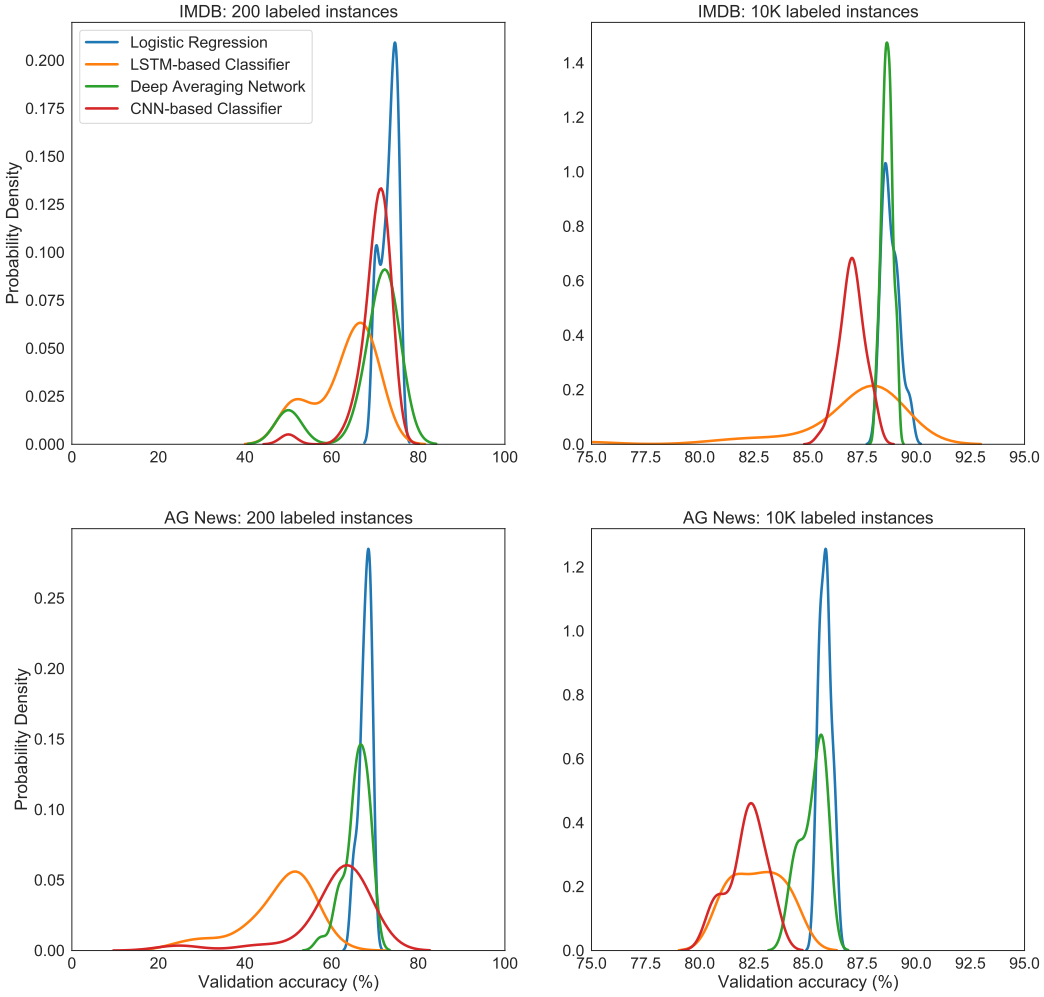

Figure 7: Probability densities of supervised classification accuracy in low-resource (200 labeled instances; left) and high-resource (10K labeled instances; right) settings for IMDB and AG datasets using randomly initialized trainable embeddings. Each search consists of 300 trials over 5 seeds and varying hyperparameters. We experiment with four different classifiers: Logistic Regression, LSTM-based classifier, Deep Averaging Network, and a CNN-based Classifier. We choose to use the Deep Averaging Network for all classifier baselines, due to its reliability, expressiveness, and low-maintenance.