Variational Pretraining for Semi-supervised Text Classification

Suchin Gururangan[1] **Tam Dang**[2] **Dallas Card**[3] **Noah A. Smith**[1,2]
[1]Allen Institute for Artificial Intelligence, Seattle, WA, USA
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA
[3]Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA
suching@allenai.org {dangt7,nasmith}@cs.washington.edu dcard@cmu.edu

Abstract

We introduce VAMPIRE,[1] a lightweight pretraining framework for effective text classification when data and computing resources are limited. We pretrain a unigram document model as a variational autoencoder on in-domain, unlabeled data and use its internal states as features in a downstream classifier. Empirically, we show the relative strength of VAMPIRE against computationally expensive contextual embeddings and other popular semi-supervised baselines under low resource settings. We also find that fine-tuning to *in-domain* data is crucial to achieving decent performance from contextual embeddings when working with limited supervision. We accompany this paper with code to pretrain and use VAMPIRE embeddings in downstream tasks.

1 Introduction

An effective approach to semi-supervised learning has long been a goal for the NLP community, as unlabeled data tends to be plentiful compared to labeled data. Early work emphasized using unlabeled data drawn from the same distribution as the labeled data (Nigam et al., 2000), but larger and more reliable gains have been obtained by using contextual embeddings trained with a language modeling (LM) objective on massive amounts of text from domains such as Wikipedia or news (Peters et al., 2018a; Devlin et al., 2019; Radford et al., 2018; Howard and Ruder, 2018). The latter approaches play to the strengths of high-resource settings (e.g., access to web-scale corpora and powerful machines), but their computational and data requirements can make them less useful in resource-limited environments. In this paper, we instead focus on the low-resource setting (§2.1), and develop a lightweight approach to pretraining for semi-supervised text classification.

Our model, which we call VAMPIRE, combines a variational autoencoder (VAE) approach to document modeling (Kingma and Welling, 2013; Miao et al., 2016; Srivastava and Sutton, 2017) with insights from LM pretraining (Peters et al., 2018a). By operating on a bag-of-words representation, we avoid the time complexity and difficulty of training a sequence-to-sequence VAE (Bowman et al., 2016; Xu et al., 2017; Yang et al., 2017) while retaining the freedom to use a multi-layer encoder that can learn useful representations for downstream tasks. Because VAMPIRE ignores sequential information, it leads to models that are much cheaper to train, and offers strong performance when the amount of labeled data is small. Finally, because VAMPIRE is a descendant of topic models, we are able to explore model selection by topic *coherence*, rather than validation-set perplexity, which results in better downstream classification performance (§6.1).

In order to evaluate the effectiveness of our method, we experiment with four text classification datasets. We compare our approach to a traditional semi-supervised baseline (self-training), alternative representation learning techniques that have access to the in-domain data, and the full-scale alternative of using large language models trained on out-of-domain data, optionally fine-tuned to the task domain.

Our results demonstrate that effective semi-supervised learning is achievable for limited-resource settings, without the need for computationally demanding sequence-based models. While we observe that fine-tuning a pretrained BERT model to the domain provides the best results, this depends on the existence of such a model in the relevant language, as well as GPUs to fine-tune it. When this is not an option, our

[1]VAriational Methods for Pretraining In Resource-limited Environments

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5880–5894
Florence, Italy, July 28 - August 2, 2019.

model offers equivalent or superior performance to the alternatives with minimal computational requirements, especially when working with limited amounts of labeled data.

The major contributions of this paper are:

- We adapt variational document models to modern pretraining methods for semi-supervised text classification (§3), and highlight the importance of appropriate criteria for model selection (§3.2).
- We demonstrate experimentally that our method is an efficient and effective approach to semi-supervised text classification when data and computation are limited (§5).
- We confirm that fine-tuning is essential when using contextual embeddings for document classification, and provide a summary of practical advice for researchers wishing to use unlabeled data in semi-supervised text classification (§8).
- We release code to pretrain variational models on unlabeled data and use learned representations in downstream tasks.[2]

2 Background

2.1 Resource-limited Environments

In this paper, we are interested in the low-resource setting, which entails limited access to computation, labels, and out-of-domain data. Labeled data can be obtained cheaply for some tasks, but for others, labels may require expensive and time-consuming human annotations, possibly from domain experts, which will limit their availability.

While there is a huge amount of unlabeled text available for some languages, such as English, this scale of data is not available for all languages. *In-domain* data availability, of course, varies by domain. For many researchers, especially outside of STEM fields, *computation* may also be a scarce resource, such that training contextual embeddings from scratch, or even incorporating them into a model could be prohibitively expensive.

Moreover, even when such pretrained models are available, they inevitably come with potentially undesirable biases baked in, based on the data on which they were trained (Recasens et al., 2013; Bolukbasi et al., 2016; Zhao et al., 2019). Particularly for social science applications, it may be preferable to exclude such confounders by only working with in-domain or curated data.

Given these constraints and limitations, we seek an approach to semi-supervised learning that can leverage in-domain unlabeled data, achieve high accuracy with only a handful of labeled instances, and can run efficiently on a CPU.

2.2 Semi-supervised Learning

Many approaches to semi-supervised learning have been developed for NLP, including variants of bootstrapping (Charniak, 1997; Blum and Mitchell, 1998; Zhou and Li, 2005; McClosky et al., 2006), and representation learning using generative models or word vectors (Mikolov et al., 2013; Pennington et al., 2014). Contextualized embeddings have recently emerged as a powerful way to use out-of-domain data (Peters et al., 2018a; Radford, 2018), but training these large models requires a massive amount of appropriate data (typically on the order of hundreds of millions of words), and industry-scale computational resources (hundreds of hours on multiple GPUs).[3]

There have also been attempts to leverage VAEs for semi-supervised learning in NLP, mostly in the form of sequence-to-sequence models (Xu et al., 2017; Yang et al., 2017), which use sequence-based encoders and decoders (see §3). These papers report strong performance, but there are many open questions which necessitate further investigation. First, given the reported difficulty of training sequence-to-sequence VAEs (Bowman et al., 2016), it is questionable whether such an approach is useful in practice. Moreover, it is unclear if such complex models (which are expensive to train) are actually required for good performance on tasks such as text classification.

Here, we instead base our framework on neural document models (Miao et al., 2016; Srivastava and Sutton, 2017; Card et al., 2018), which offer both faster training and an explicit interpretation in the form of *topics*, and explore their utility in the semi-supervised setting.

3 Model

In this work, we assume that we have L documents, $\mathcal{D}_L = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{L}$, with observed cat-

[2] http://github.com/allenai/vampire

[3] For example, ULMFIT was trained on 100 million words, and BERT used 3.3 billion. While many pretrained models have been made available, they are unlikely to cover every application, especially for rare languages.

egorical labels $y \in \mathcal{Y}$. We also assume access to a larger set of U documents drawn from the same distribution, but for which the labels are unobserved, i.e, $\mathcal{D}_U = \{\boldsymbol{x}_i\}_{i=L+1}^{U+L}$. Our primary goal is to learn a probabilistic classifier, $p(y \mid \boldsymbol{x})$.

Our approach heavily borrows from past work on VAEs (Kingma and Welling, 2013; Miao et al., 2016; Srivastava and Sutton, 2017), which we adapt to semi-supervised text classification (see Figure 1). We do so by pretraining the document model on unlabeled data (§3.1), and then using learned representations in a downstream classifier (§3.3). The downstream classifier makes use of multiple internal states of the pretrained document model, as in Peters et al. (2018b). We also explore how to best do model selection in a way that benefits the downstream task (§3.2).

3.1 Unsupervised Pretraining

In order to learn useful representations, we initially ignore labels, and assume each document is generated from a latent variable, $\boldsymbol{z}$. The functions learned in estimating this model then provide representations which are used as features in supervised learning.

Using a variational autoencoder for approximate Bayesian inference, we simultaneously learn an *encoder*, which maps from the observed text to an approximate posterior $q(\boldsymbol{z} \mid \boldsymbol{x})$, and a *decoder*, which reconstructs the text from the latent representation. In practice, we instantiate both the encoder and decoder as neural networks and assume that the encoder maps to a normally distributed posterior, i.e., for document i,

$$q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) = \mathcal{N}\left(\boldsymbol{z}_i \mid f_\mu(\boldsymbol{x}_i), \text{diag}(f_\sigma(\boldsymbol{x}_i))\right) \quad (1)$$

$$\boldsymbol{x}_i \sim p(\boldsymbol{x}_i \mid f_d(\boldsymbol{z}_i)). \quad (2)$$

Using standard principles of variational inference, we derive a variational bound on the marginal log-likelihood of the observed data,

$$\log p(\boldsymbol{x}_i) \geq \mathcal{B}(\boldsymbol{x}_i) = \mathbb{E}_{q(\boldsymbol{z}_i \mid \boldsymbol{x}_i)}[\log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)] - \text{KL}[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \parallel p(\boldsymbol{z})]. \quad (3)$$

Intuitively, the first term in the bound can be thought of as a *reconstruction loss*, ensuring that generated words are similar to the original document. The second term, the *KL divergence*, encourages the variational approximation to be close to the assumed prior, $p(\boldsymbol{z})$, which we take to be a spherical normal distribution.

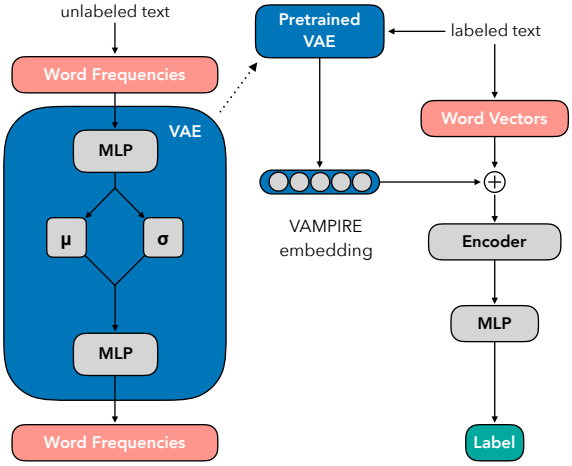

Figure 1: VAMPIRE involves pretraining a deep variational autoencoder (VAE; displayed on left) on unlabeled text. The VAE, which consists entirely of feedforward networks, learns to reconstruct a word frequency representation of the unlabeled text with a logistic normal prior, parameterized by $\boldsymbol{\mu}$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

Using the *reparameterization trick* (Kingma and Welling, 2013; Rezende et al., 2014), we replace the expectation with a single-sample approximation,[4] i.e.,

$$\mathcal{B}(\boldsymbol{x}_i) \approx \log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i^{(s)}) - \text{KL}[q(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \parallel p(\boldsymbol{z})] \quad (4)$$

$$\boldsymbol{z}_i^{(s)} = f_\mu(\boldsymbol{x}_i) + f_\sigma(\boldsymbol{x}_i) \cdot \boldsymbol{\varepsilon}^{(s)}, \quad (5)$$

where $\boldsymbol{\varepsilon}^{(s)} \sim \mathcal{N}(0, \mathbf{I})$ is sampled from an independent normal. All parameters can then be optimized simultaneously by performing stochastic gradient ascent on the variational bound.

A powerful way of encoding and decoding text is to use sequence models. That is, $f_\mu(\boldsymbol{x})$ and $f_\sigma(\boldsymbol{x})$ would map from a sequence of tokens to a pair of vectors, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, and $f_d(\boldsymbol{z})$ would similarly decode from $\boldsymbol{z}$ to a sequence of tokens, using recurrent, convolutional, or attention-based networks. Some authors have adopted this approach (Bowman et al., 2016; Xu et al., 2017; Yang et al., 2017), but as discussed above (§2.2), it has a number of disadvantages.

In this paper, we adopt a more lightweight and directly interpretable approach, and work with word frequencies instead of word sequences. Using the same basic structure as Miao et al. (2016)

[4] We leave experimentation with multi-sample approximation (e.g., importance sampling) to future work.

but employing a softmax in the decoder, we encode $f_\mu(\boldsymbol{x})$ and $f_\sigma(\boldsymbol{x})$ with multi-layer feed forward neural networks operating on an input vector of word counts, $\boldsymbol{c}_i$:

$$\boldsymbol{c}_i = \text{counts}(\boldsymbol{x}_i) \tag{6}$$
$$\boldsymbol{h}_i = \text{MLP}(\boldsymbol{c}_i) \tag{7}$$
$$\boldsymbol{\mu}_i = f_\mu(\boldsymbol{x}_i) = \boldsymbol{W}_\mu \boldsymbol{h}_i + \boldsymbol{b}_\mu \tag{8}$$
$$\boldsymbol{\sigma}_i = f_\sigma(\boldsymbol{x_i}) = \exp(\boldsymbol{W}_\sigma \boldsymbol{h}_i + \boldsymbol{b}_\sigma) \tag{9}$$
$$\boldsymbol{z}_i^{(s)} = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \cdot \boldsymbol{\varepsilon}^{(s)}. \tag{10}$$

For a decoder, we use the following form, which reconstructs the input in terms of topics (coherent distributions over the vocabulary):

$$\boldsymbol{\theta}_i = \text{softmax}(\boldsymbol{z}_i^{(s)}) \tag{11}$$
$$\boldsymbol{\eta}_i = \text{softmax}(\boldsymbol{b} + \boldsymbol{B}\boldsymbol{\theta}_i) \tag{12}$$
$$\log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i^{(s)}) = \sum_{j=1}^{V} \boldsymbol{c}_{ij} \cdot \log \boldsymbol{\eta}_{ij}, \tag{13}$$

where j ranges over the vocabulary.

By placing a softmax on $\boldsymbol{z}$, we can interpret $\boldsymbol{\theta}$ as a distribution over latent topics, as in a topic model (Blei et al., 2003), and $\boldsymbol{B}$ as representing positive and negative topical deviations from a background $\boldsymbol{b}$. This form (essentially a unigram LM) allows for much more efficient inference on $\boldsymbol{z}$, compared to sequence-based encoders and decoders.

3.2 Model Selection via Topic Coherence

Because our pretraining ignores document labels, it is not obvious that optimizing it to convergence will produce the best representations for downstream classification. When pretraining using a LM objective, models are typically trained until model fit stops improving (i.e., perplexity on validation data). In our case, however, $\boldsymbol{\theta}_i$ has a natural interpretation as the distribution (for document i) over the latent "topics" learned by the model ($\boldsymbol{B}$). As such, an alternative is to use the quality of the topics as a criterion for early stopping.

It has repeatedly been observed that different types of topic models offer a trade-off between perplexity and topic quality (Chang et al., 2009; Srivastava and Sutton, 2017). Several methods for automatically evaluating topic *coherence* have been proposed (Newman et al., 2010; Mimno et al., 2011), such as normalized pointwise mutual information (NPMI), which Lau et al. (2014) found to be among the most strongly correlated with human judgement. As such, we consider using either log likelihood or NPMI as a stopping criteria for VAMPIRE pretraining (§6.1), and evaluate them in terms of which leads to the better downstream classifier.

NPMI measures the probability that two words collocate in an external corpus (in our case, the validation data). For each topic t in $\boldsymbol{B}$, we collect the top ten most probable words and compute NPMI between all pairs:

$$\text{NPMI}(t) = \sum_{i,j \leq 10;\, j \neq i} \frac{\log \frac{P(t_i, t_j)}{P(t_i)P(t_j)}}{-\log P(t_i, t_j)} \tag{14}$$

We then arrive at a global NPMI for $\boldsymbol{B}$ by averaging the NPMIs across all topics. We evaluate NPMI at the end of each epoch during pretraining, and stop training when NPMI has stopped increasing for a pre-defined number of epochs.

3.3 Using a Pretrained VAE for Text Classification

Kingma et al. (2014) proposed using the latent variable of an unsupervised VAE as features in a downstream model for classifying images. However, work on pretraining for NLP, such as Peters et al. (2018a), found that LMs encode different information in different layers, each of which may be more or less useful for certain tasks. Here, for an n-layer MLP encoder on word counts $\boldsymbol{c}_i$, we build on that idea, and use as representations a weighted sum over $\boldsymbol{\theta}_i$ and the internal states of the MLP, $\boldsymbol{h}_i^{(k)}$, with weights to be learned by the downstream classifier.[5]

That is, for any sequence-to-vector encoder, $f_{s2v}(\boldsymbol{x})$, we propose to augment the vector representations for each document by concatenating them with a weighted combination of the internal states of our variational encoder (Peters et al., 2018a). We can then train a supervised classifier on the weighted combination,

$$\boldsymbol{r}_i = \lambda_0 \boldsymbol{\theta}_i + \sum_{k=1}^{n} \lambda_k \boldsymbol{h}_i^{(k)} \tag{15}$$
$$p(y_i \mid \boldsymbol{x}_i) = f_c([\boldsymbol{r}_i; f_{s2v}(\boldsymbol{x_i})]), \tag{16}$$

where f_c is a neural classifier and $\{\lambda_0, \ldots, \lambda_n\}$ are softmax-normalized trainable parameters.

[5] We also experimented with the joint training and combined approaches discussed in Kingma et al. (2014), but found that neither of these reliably improved performance over our pretraining approach.

3.4 Optimization

In all cases, we optimize models using Adam (Kingma and Ba, 2014). In order to prevent divergence during pretraining, we make use of a batch-norm layer on the reconstruction of $\boldsymbol{x}$ (Ioffe and Szegedy, 2015). We also use KL-annealing (Bowman et al., 2016), placing a scalar weight on the KL divergence term in Eq. (3), which we gradually increase from zero to one. Because our model consists entirely of feedforward neural networks, it is easily parallelized, and can run efficiently on either CPUs or GPUs.

4 Experimental Setup

We evaluate the performance of our approach on four text classification tasks, as we vary the amount of labeled data, from 200 to 10,000 instances. In all cases, we assume the existence of about 75,000 to 125,000 unlabeled in-domain examples, which come from the union of the unused training data and any additional unlabeled data provided by the corpus. Because we are working with a small amount of labeled data, we run each experiment with five random seeds, each with a different sample of labeled training instances, and report the mean performance on test data.

4.1 Datasets and Preprocessing

We experiment with text classification datasets that span a variety of label types. The datasets we use are the familiar AG News (Zhang et al., 2015), IMDB (Maas et al., 2011), and 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). Summary statistics are presented in Table 1. In all cases, we either use the official test set, or take a random stratified sample of 25,000 documents as a test set. We also sample 5,000 instances as a validation set.

We tokenize documents with spaCy, and use up to 400 tokens for sequence encoding ($f_{s2v}(\boldsymbol{x})$). For VAMPIRE pretraining, we restrict the vocabulary to the 30,000 most common words in the dataset, after excluding tokens shorter than three characters, those with digits or punctuation, and stopwords.[6] We leave the vocabulary for downstream classification unrestricted.

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

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

Table 1: Datasets used in our experiments.

4.2 VAMPIRE Architecture

In order to find reasonable hyperparameters for VAMPIRE, we utilize a random search strategy for pretraining. For each dataset, we take the model with the best NPMI for use in the downstream classifiers. We detail sampling bounds and final assignments for each hyperparameter in Table 5 in Appendix A.1.

4.3 Downstream Classifiers

For all experiments we make use of the Deep Averaging Network (DAN) architecture (Iyyer et al., 2015) as our baseline sequence-to-vector encoder, $f_{s2v}(\boldsymbol{x})$. That is, embeddings corresponding to each token are summed and passed through a multi-layer perceptron.

$$p(y_i \mid \boldsymbol{x}_i) = \text{MLP}\left(\frac{1}{|\boldsymbol{x}_i|}\sum_{j=1}^{|\boldsymbol{x}_i|} E(\boldsymbol{x}_i)_j\right), \quad (17)$$

where $E(\boldsymbol{x})$ converts a sequence of tokens to a sequence of vectors, using randomly initialized vectors, off-the-shelf GLOVE embeddings (Pennington et al., 2014), or contextual embeddings.

To incorporate the document representations learned by VAMPIRE in a downstream classifier, we concatenate them with the average of randomly initialized trainable embeddings, i.e.,

$$p(y_i \mid \boldsymbol{x}_i) = \text{MLP}\left(\left[\boldsymbol{r}_i; \frac{1}{|\boldsymbol{x}_i|}\sum_{j=1}^{|\boldsymbol{x}_i|} E(\boldsymbol{x}_i)_j\right]\right). \quad (18)$$

Preliminary experiments found that DANs with one-layer MLPs and moderate dropout provide more reliable performance on validation data than more expressive models, such as CNNs or LSTMs, with less hyperparameter tuning, especially when working with few labeled instances (details in Appendix A.2).

4.4 Resources and Baselines

In these experiments, we consider baselines for both low-resource and high-resource settings, where the high-resource baselines have access to

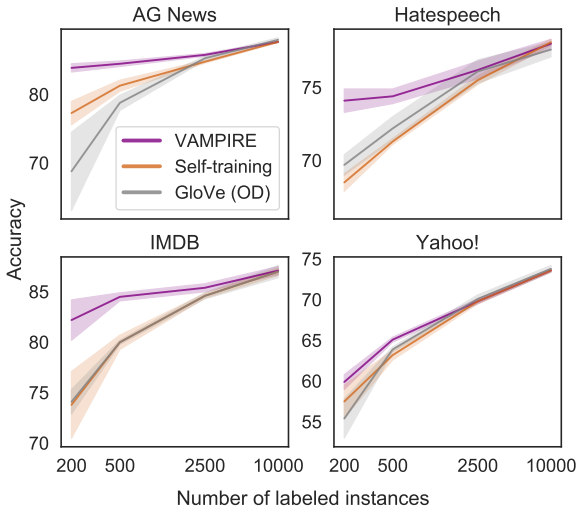

Figure 2: Learning curves for all datasets in the low-resource setting, showing the mean (line) and one standard deviation (bands) over five runs for VAMPIRE, self-training, and 840B-token GLOVE embeddings. Full results are in Table 2.

greater computational resources and a either massive amount of unlabeled data or a pretrained model, such as ELMO or BERT.[7]

Low resource In the low-resource setting we assume that computational resources are at a premium, so we are limited to lightweight approaches such as VAMPIRE, which can run efficiently on a CPU. As baselines, we consider **a)** a purely supervised model, with randomly initialized 50-dimensional embeddings and no access to unlabeled data; **b)** the same model initialized with 300-dimensional GLOVE vectors, pretrained on 840 billion words;[8] **c)** 300-dimensional GLOVE vectors trained on only in-domain data; and **d)** self-training, which has access to the in-domain unlabeled data. For self-training, we iterate over training a model, predicting labels on all unlabeled instances, and adding to the training set all unlabeled instances whose label is predicted with high confidence, repeating this up to five times and 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.

[7] As discussed above, we consider these models to be representative of the high-resource setting, both because they were computationally intensive to train, and because they were made possible by the huge amount of English text that is available online.

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

High resource In the *high-resource* setting, we assume access to plentiful computational resources and massive amounts of out-of-domain data, which may be indirectly accessed through pretrained models. Specifically, we evaluate the performance of a Transformer-based ELMO (Peters et al., 2018b) and BERT, both (a) off-the-shelf with frozen embeddings and (b) after semi-supervised fine-tuning to both unlabeled and labeled in-domain data. To perform semi-supervised fine-tuning, we first use ELMO and BERT's original objectives to fine-tune to the unlabeled data. To fine-tune ELMO to the labeled data, we average over the LM states and add a softmax classification layer. We obtain the best results applying slanted triangular learning rates and gradual unfreezing (Howard and Ruder, 2018) to this fine-tuning step. To fine-tune BERT to labeled data, we feed the hidden state corresponding to the `[CLS]` token of each instance to a softmax classification layer. We use AllenNLP[9] to fine-tune ELMO, and Pytorch-pretrained-BERT[10] to fine-tune BERT.

We also experiment with ELMO trained only on in-domain data as an example of high-resource LM pretraining methods, such as Dai and Le (2015), when there is no out-of-domain data available. Specifically, we generate contextual word representations with a Transformer-based ELMO. During downstream classification, the resulting vectors are frozen and concatenated to randomly initialized word vectors prior to the summation in Eq. (17).

5 Results

In the **low-resource** setting, we find that VAMPIRE achieves the highest accuracy of all low-resource methods we consider, especially when the amount of labeled data is small. Table 2 shows the performance of all low-resource models on all datasets as we vary the amount of labeled data, and a subset of these are also shown in Figure 2 for easy comparison.

In the **high-resource** setting, we find, not surprisingly, that fine-tuning the pretrained BERT model to in-domain data provides the best performance. For both BERT and ELMO, we find that using frozen off-the-shelf vectors results

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

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

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	**82.2** (2.0)	**84.5** (0.4)	**85.4** (0.4)	**87.1** (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** (0.6)	**84.5** (0.4)	**85.8** (0.2)	87.7 (0.1)
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	**59.9** (0.9)	**65.1** (0.3)	69.8 (0.3)	73.6 (0.2)
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	**74.1** (0.8)	**74.4** (0.5)	**76.2** (0.6)	78.0 (0.3)

Table 2: 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 standard deviation in parentheses, and the highest mean result in each setting shown in bold.

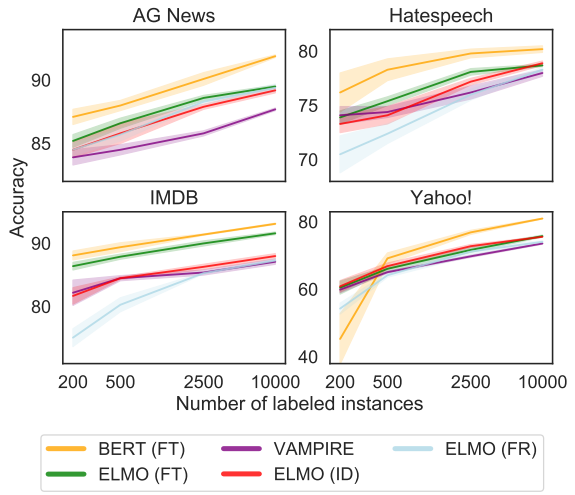

Figure 3: High-resource methods (plus VAMPIRE) on four datasets; ELMO performance benefits greatly from training on (ID), or fine-tuning (FT) to, the in-domain data (as does BERT; full results in Appendix B). Key: FT (fine-tuned), FR (frozen), ID (in-domain).

in surprisingly poor performance, compared to fine-tuning to the task domain, especially for HATESPEECH and IMDB.[11] For these two datasets, an ELMO model trained *only* on in-domain data offers far superior performance to frozen off-the-shelf ELMO (see Figure 3). This difference is smaller, however, for YAHOO! and AG. (Please see Appendix B for full results).

These results taken together demonstrate that although pretraining on massive amounts of web text offers large improvements over purely supervised models, access to unlabeled *in-domain* data is critical, either for fine-tuning a pretrained language model in the high-resource setting, or for training VAMPIRE in the low-resource setting. Similar findings have been reported by Yogatama et al. (2019) for tasks such as natural language inference and question answering.

6 Analysis

6.1 NPMI versus NLL as Stopping Criteria

To analyze the effectiveness of different stopping criterion in VAMPIRE, we pretrain 200 VAMPIRE models on IMDB: 100 selected via NPMI, and 100 selected via negative log likelihood (NLL) on validation data. Interestingly, we observe that VAMPIRE NPMI and NLL values are negatively correlated ($\rho = -0.72$; Figure 4A), suggesting that upon convergence, trained models that better fit the data also tend to have more coherent topics. We then train 200 downstream classifiers with the same hyperparameters, on a fixed 200 document random subset of the IMDB dataset, uniformly sampling over the NPMI- and NLL-selected VAMPIRE models as additional features. In Figure 4B and Fig-

[11] See also Howard and Ruder (2018).

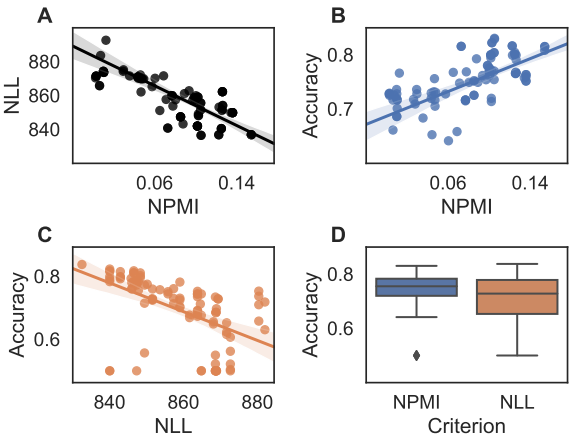

Figure 4: Comparing NPMI and NLL as early stopping criteria for VAMPIRE model selection. NPMI and NLL are correlated measures of model fit, but NPMI-selected VAMPIRE models have lower variance on downtream classification performance with 200 labeled documents of IMDB. Accuracy is reported on the validation data. See §6.1 for more details.

ure 4C, we observe that better pretrained VAMPIRE models (according to either criterion) tend to produce better downstream performance. (ρ = 0.55 and ρ = –0.53, for NPMI and NLL respectively).

However, we also observe higher variance in accuracy among the VAMPIRE models obtained using NLL as a stopping criterion (Figure 4D). Such models selected via NLL have poor topic coherence and downstream performance. As such, doing model selection using NPMI is the preferred alternative, and all VAMPIRE results in Table 2 are based on pretrained models selected using this criterion.

The experiments in Ding et al. (2018) provide some insight into this behaviour. They find that when training neural topic models, model fit and NPMI initially tend to improve on each epoch. At some point, however, perplexity continues to improve, while NPMI starts to drop, sometimes dramatically. We also observe this phenomenon when training VAMPIRE (see Appendix C). Using NPMI as a stopping criterion, as we propose to do, helps to avoid degenerate models that result from training too long.

In some preliminary experiments, we also observe cases where NPMI is artificially high because of redundancy in topics. Applying batchnorm to the reconstruction markedly improves diversity of collocating words across topics, which has also been noted by Srivastava and Sutton (2017). Future work may explore assigning a word diversity regularizer to the NPMI metric, so as to encourage models that have both stronger coherence and word diversity across topics.

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 3: Example topics learned by VAMPIRE in IMDB and YAHOO! datasets. See Appendix D for more examples.

6.2 Learned Latent Topics

In addition to being lightweight, one advantage of VAMPIRE is that it produces document representations that can be explicitly interpreted in terms of topics. Although the input we feed into the downstream classifier combines this representation with internal states of the encoder, the topical interpretation helps to summarize what the pretraining has learned. Examples of topics learned by VAMPIRE are provided in Table 3 and Appendix D.

6.3 Learned Scalar Layer Weights

Since the scalar weight parameters in $\boldsymbol{r}_i$ are trainable, we are able to investigate which layers of the pretrained VAE the classifier tends to prefer. We consistently find that the model tends to upweight the first layer of the VAE encoder, $\boldsymbol{h}^{(1)}$, and $\boldsymbol{\theta}$, and downweight the other layers of the encoder. To improve learning, especially under low resource settings, we initialize the scalar weights applied to the first encoder layer and $\boldsymbol{\theta}$ with high values and downweighted the intermediate layers, which increases validation performance. However, we also have observed that using a multi-layer encoder in VAMPIRE leads to larger gains downstream.

6.4 Computational Requirements

An appealing aspect of VAMPIRE is its compactness. Table 4 shows the computational requirements involved in training VAMPIRE on a single GPU or CPU, compared to training an ELMO model from scratch on the same data on

Model	Parameters	Time
VAMPIRE (GPU)	3.8M	7 min
VAMPIRE (CPU)	3.8M	22 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, we display the computational requirements for pretraining VAMPIRE and ELMO on in-domain unlabeled text from the IMDB dataset. We report results on training VAMPIRE (with hyperparameters listed in Appendix A.1) and ELMO (with its default configuration) on a GeForce GTX 1080 Ti GPU, and VAMPIRE on a 2.60GHz Intel Xeon CPU. VAMPIRE uses about 750MB of memory on a GPU, while ELMO requires about 8.5GB.

a GPU. It is possible to train VAMPIRE orders of magnitude faster than ELMO, even without expensive hardware, making it especially suitable for obtaining fast results when resources are limited.

7 Related Work

In addition to references given throughout, many others have explored ways of enhancing performance when working with limited amounts of labeled data. Early work on speech recognition demonstrated the importance of pretraining and fine-tuning deep models in the semi-supervised setting (Yu et al., 2010). Chang et al. (2008) considered "dataless" classification, where the names of the categories provide the only supervision. Miyato et al. (2016) showed that adversarial pretraining can offer large gains, effectively augmenting the amount of data available. A long line of work in *active learning* similarly tries to maximize performance when obtaining labels is costly (Settles, 2012). Xie et al. (2019) describe novel data augmentation techniques leveraging back translation and tf-idf word replacement. All of these approaches could be productively combined with the methods proposed in this paper.

8 Recommendations

Based on our findings in this paper, we offer the following practical advice to those who wish to do effective semi-supervised text classification.

- When resources are unlimited, the best results can currently be obtained by using a pretrained model such as BERT, but fine-tuning to in-domain data is critically important (see also Howard and Ruder, 2018).
- When computational resources and annotations are limited, but there is plentiful unlabeled data, VAMPIRE offers large gains over other low-resource approaches.
- Training a language model such as ELMO on only in-domain data offers comparable or somewhat better performance to VAMPIRE, but may be prohibitively expensive, unless working with GPUs.
- Alternatively, resources can be invested in getting more annotations; with sufficient labeled data (tens of thousands of instances), the advantages offered by additional unlabeled data become negligible. Of course, other NLP tasks may involve different trade-offs between data, speed, and accuracy.

9 Conclusions

The emergence of models like ELMO and BERT has revived semi-supervised NLP, demonstrating that pretraining large models on massive amounts of data can provide representations that are beneficial for a wide range of NLP tasks. In this paper, we confirm that these models are useful for text classification when the number of labeled instances is small, but demonstrate that fine-tuning to in-domain data is also of critical importance. In settings where BERT cannot easily be used, either due to computational limitations, or because an appropriate pretrained model in the relevant language does not exist, VAMPIRE offers a competitive lightweight alternative for pretraining from unlabeled data in the low-resource setting. When working with limited amounts of labeled data, we achieve superior performance to baselines such as self-training, or using word vectors pretrained on out-of-domain data, and approach the performance of ELMO trained only on in-domain data at a fraction of the computational cost.

Acknowledgments

We thank the members of the AllenNLP and ARK teams for useful comments and discussions. We also thank the anonymous reviewers for their insightful feedback. Computations on beaker.org were supported in part by credits from Google Cloud.

References

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

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*.

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 *Advances in Neural Information Processing Systems*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *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 *Advances in Neural Information Processing Systems*.

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

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings AAAI*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*.

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*.

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, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *CoRR*, abs/1312.6114.

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*.

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*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings NAACL*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of ICML*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

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

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2016. Virtual adversarial training for semi-supervised text classification. *CoRR*, abs/1605.07725.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Proceedings of NAACL*.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3).

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. 2018a. Deep contextualized word representations. In *Proceedings of NAACL*.

Matthew E. Peters, Mark Neumann, Luke S. Zettlemoyer, and Wen tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of EMNLP*.

Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *CoRR*, abs/1811.01088.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2018. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

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

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of ICML*.

Burr Settles. 2012. *Active Learning*. Morgan & Claypool.

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

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.

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

Dong Yu, Li Deng, and George E. Dahl. 2010. Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*.

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*.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17:1529–1541.

A Hyperparameter Search

In this section, we describe the hyperparameter search we used to choose model configurations, and include plots illustrating the range of validation performance observed in each setting.

A.1 VAMPIRE Search

For the results presented in the paper, we varied the hyperparameters of VAMPIRE across a number of different dimensions, outlined in Table 5.

A.2 Classifier Search

To choose a baseline classifier for which we experiment with all pretrained models, we performed a mix of manual tuning and random search over four basic classifiers: CNN, LSTM, Bag-of-Embeddings (i.e., Deep Averaging Networks), and Logistic Regression.

Figure 6 shows the distribution of validation accuracies using 200 and 10,000 labeled instances, respectively, for different classifiers on the IMDB and AG datasets. Under the low-resource setting, we observe that logistic regression and DAN based classifiers tend to lead to more reliable validation accuracies. With enough compute, CNN-based classifiers tend to produce marginally higher validation accuracies, but the probability is mostly centered below those of the logistic regression and DAN classifiers. LSTM-based classifiers tend to have extremely high variance under the low-resource setting. For this work, we choose to experiment with the DAN classifier, which comes with the richness of vector-based representations, along with the reliability that comes with having very few hyperparameters to tune.

B Results in the High Resource Setting

Table 6 shows the results of all high-resource methods (along with VAMPIRE) on all datasets, as we vary the amount of labeled data. As can be seen, training ELMO *only* on in-domain data results in similar or better performance to using an off-the-shelf ELMO or BERT model, *without* fine-tuning it to in-domain data.

Except for one case in which it fails badly (YAHOO! with 200 labeled instances), fine-tuning BERT to the target domain achieves the best performance in every setting. Though we performed a substantial hyperparameter search under this regime, we attribute the failure of fine-tuning BERT under this setting to potential hyperparameter decisions which could be improved with further tuning. Other work has suggest that random initializations have a significant effect on the failure cases of BERT, pointing to the brittleness of fine-tuning (Phang et al., 2018).

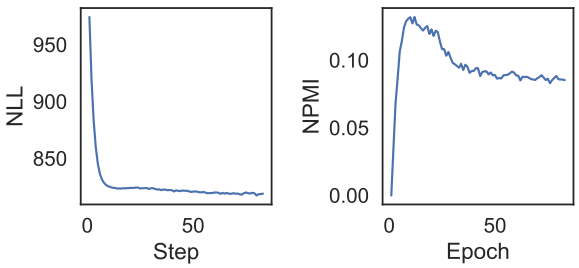

Figure 5: An example learning curve when training VAMPIRE on the IMDB dataset. If trained for too long, we observe many cases in which NPMI (higher is better) degrades while NLL (lower is better) continues to decrease. To avoid selecting a model that has poor topic coherence, we recommend performing model selection with NPMI rather than NLL.

The performance gap between fine-tuned ELMO and frozen ELMO in AG News corpus is much smaller than that of the other datasets, perhaps because the ELMO model we used was pretrained on the Billion Words Corpus, which is a news crawl. This dataset is also an example where frozen ELMO tends to out-perform using VAMPIRE. We attribute the strength of frozen, pretrained ELMO under this setting as further evidence of the importance of in-domain data for effective semi-supervised text classification.

C Further Details on NPMI vs. NLL as Stopping Criteria

In the main paper, we note that we have observed cases in which training VAMPIRE for too long results in NPMI degradation, while NLL continues to improve. In Figure 5, we display example learning curves that point to this phenomenon.

D Additional Learned Topics

In Table 7 we display some additional topics learned by VAMPIRE on the YAHOO! dataset.

Computing Infrastructure	GeForce GTX 1080 GPU
Number of search trials	60 trials per dataset
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	constant
KL annealing sigmoid weight 1	0.25	N/A	N/A	N/A	N/A
KL annealing sigmoid weight 2	15	N/A	N/A	N/A	N/A
KL annealing linear scaling	1000	1000	1000	1000	N/A
VAMPIRE hidden dimension	*uniform-integer*[32, 128]	80	81	118	125
Number of encoder layers	*choice*[1, 2, 3]	2	2	3	3
Encoder activation	*choice*[relu, tanh, softplus]	tanh	relu	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
z-dropout	*random-uniform*[0, 0.5]	0.47	0.49	0.41	0.45
learning rate optimizer	Adam	Adam	Adam	Adam	Adam
learning rate	*loguniform-float*[1e-4, 1e-2]	0.00081	0.00021	0.00024	0.0040
update background frequency	*choice*[True, False]	False	False	False	False
vocabulary size	30000	30000	30000	30000	30000

Dataset	VAMPIRE **NPMI**
IMDB	0.131
AG	0.224
YAHOO!	0.475
HATESPEECH	0.139

Table 5: VAMPIRE search space, best assignments, and associated performance on the four datasets we consider in this work.

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 6: 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 7: Example topics learned by VAMPIRE in the YAHOO! dataset.

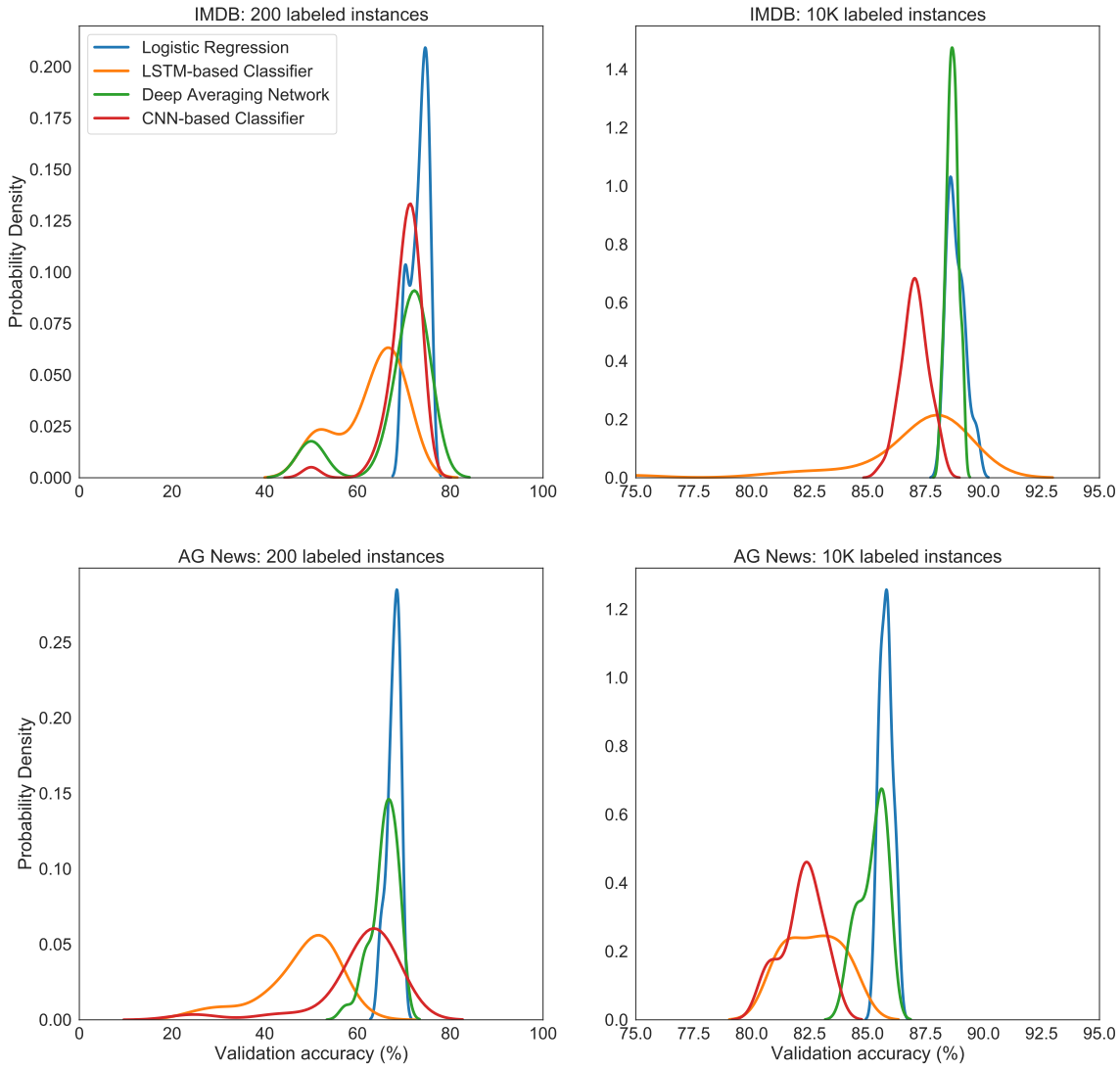

Figure 6: Probability densities of supervised classification accuracy in low-resource (200 labeled instances; left) and high-resource (10K labeled instances; right) settings for IMDB and AG datasets using randomly initialized trainable embeddings. Each search consists of 300 trials over 5 seeds and varying 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.