Latent-Variable Generative Models for Data-Efficient Text Classification

Xiaoan Ding[1] **Kevin Gimpel**[2]
[1]University of Chicago, Chicago, IL, 60637, USA
[2]Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
xiaoanding@uchicago.edu, kgimpel@ttic.edu

Abstract

Generative classifiers offer potential advantages over their discriminative counterparts, namely in the areas of data efficiency, robustness to data shift and adversarial examples, and zero-shot learning (Ng and Jordan, 2002; Yogatama et al., 2017; Lewis and Fan, 2019). In this paper, we improve generative text classifiers by introducing discrete latent variables into the generative story, and explore several graphical model configurations. We parameterize the distributions using standard neural architectures used in conditional language modeling and perform learning by directly maximizing the log marginal likelihood via gradient-based optimization, which avoids the need to do expectation-maximization. We empirically characterize the performance of our models on six text classification datasets. The choice of where to include the latent variable has a significant impact on performance, with the strongest results obtained when using the latent variable as an auxiliary conditioning variable in the generation of the textual input. This model consistently outperforms both the generative and discriminative classifiers in small-data settings. We analyze our model by using it for controlled generation, finding that the latent variable captures interpretable properties of the data, even with very small training sets.

1 Introduction

The most widely-used neural network classifiers are **discriminative**, that is, they are trained to explicitly favor the gold standard label over others. The alternative is to design classifiers that are **generative**; these follow a generative story that includes predicting the label and then the data conditioned on the label. Discriminative classifiers are preferred because they generally outperform their generative counterparts on standard benchmarks. These benchmarks typically assume large annotated training sets, little mismatch between training and test distributions, relatively clean data, and a lack of adversarial examples (Zue et al., 1990; Marcus et al., 1993; Deng et al., 2009; Lin et al., 2014).

However, when conditions are not ideal for discriminative classifiers, generative classifiers can actually perform better. Ng and Jordan (2002) showed theoretically that linear generative classifiers approach their asymptotic error rates more rapidly than discriminative ones. Based on this finding, Yogatama et al. (2017) empirically characterized the performance of RNN-based generative classifiers, showing advantages in sample complexity, zero-shot learning, and continual learning. Recent work in generative question answering models (Lewis and Fan, 2019) demonstrates better robustness to biased training data and adversarial testing data than state-of-the-art discriminative models.

In this paper, we focus on settings with small amounts of annotated data and improve generative text classifiers by introducing discrete latent variables into the generative story. Accordingly, the training objective is changed to log marginal likelihood of the data as we marginalize out the latent variables during learning. We parameterize the distributions with standard neural architectures used in conditional language models and include the latent variable by concatenating its embedding to the RNN hidden state before computing the softmax over words. While traditional latent variable learning in NLP uses the expectation-maximization (EM) algorithm (Dempster et al., 1977), we instead simply perform direct optimization of the log marginal likelihood using gradient-based methods. At inference time, we similarly marginalize out the latent variables while maximizing over the label.

We characterize the performance of our latent-variable generative classifiers on six text classification datasets introduced by Zhang et al. (2015). We observe that introducing latent variables leads to large and consistent performance gains in the small-data regime, though the benefits of adding latent variables reduce as the training set becomes larger.

To better understand the modeling space of latent variable classifiers, we explore several graphical model configurations. Our experimental results demonstrate the importance of including a direct dependency between the label and the input in the model. We study the relationship between the label, latent, and input variables in our strongest latent generative classifier, finding that the label and latent capture complementary information about the input. Some information about the textual input is encoded in the latent variable to help with generation.

We analyze our latent generative model by generating samples when controlling the label and latent variables. Even with small training data, the samples capture the salient characteristics of the label space while also conforming to the values of the latent variable, some of which we find to be interpretable. While discriminative classifiers excel at separating examples according to labels, generative classifiers offer certain advantages in practical settings that benefit from a richer understanding of the data-generating process.

2 Discriminative and Generative Text Classifiers

We begin by defining our baseline generative and discriminative text classifiers for document classification. Our models are essentially the same as those from Yogatama et al. (2017); we describe them in detail here because our latent-variable models will extend them.[1] Our classifiers are trained on datasets D of annotated documents. Each instance $\langle x, y\rangle \in D$ consists of a textual input $x = \{x_1, x_2, ..., x_T\}$, where T is the length of the document, and a label $y \in \mathcal{Y}$.

[1]The main difference between our baselines and the models in Yogatama et al. (2017) are: (1) their discriminative classifier uses an LSTM with "peephole connections"; (2) they evaluate a label-based generative classifier ("Independent LSTMs") that uses a separate LSTM for each label. They also evaluate the model we described here, which they call "Shared LSTMs". Their Independent and Shared LSTMs perform similarly across training set sizes.

The discriminative classifier is trained to maximize the conditional probability of labels given documents: $\sum_{\langle x,y\rangle\in D} \log p(y \mid x)$. For our discriminative model, we encode a document x using an LSTM (Hochreiter and Schmidhuber, 1997), and use the average of the LSTM hidden states as the document representation. The classifier is built by adding a softmax layer on top of the LSTM state average to get a probability distribution over labels.

The generative classifier is trained to maximize the joint probability of documents and labels: $\sum_{\langle x,y\rangle\in D} \log p(x, y)$. The generative classifier uses the following factorization:

$$p(x, y) = p(x \mid y)p(y) \tag{1}$$

We parameterize $\log p(x \mid y)$ as a conditional LSTM language model using the standard sequential factorization:

$$\log p(x \mid y) = \sum_{t=1}^{T} \log p(x_t \mid x_{<t}, y) \tag{2}$$

We define a label embedding matrix $V_{\mathcal{Y}} \in \mathbb{R}^{d_1\times|\mathcal{Y}|}$. To predict the next word x_{t+1}, we concatenate the LSTM hidden state h_t with the label embedding v_y (a column of $V_{\mathcal{Y}}$), and feed it to a softmax layer to get the probability distribution over the vocabulary. More details about the factorization and parameterization are discussed in Section 3. The label prior $p(y)$ is acquired via maximum likelihood estimation and fixed during training of the remaining parameters.

At inference time, the prediction is made by maximizing $p(y \mid x)$ with respect to y for the discriminative classifier and maximizing $p(x \mid y)p(y)$ for the generative classifier.

3 Latent-Variable Generative Classifiers

We now introduce discrete latent variables into the standard generative classifier as shown in Figure 1. We refer to the latent-variable model as an auxiliary latent generative model, as we expect the latent variable to contain auxiliary information that can help with the generation of the input.

Following the graphical model structure in Figure 1(b), we factorize the joint probability $p(x, y, c)$ as follows:

$$p(x, y, c) = p_\Theta(x \mid c, y)p_\Phi(c)p_\Psi(y) \tag{3}$$

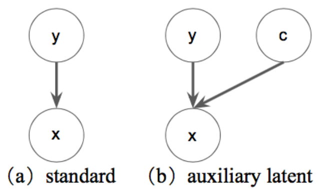


Figure 1: Graphical models of (a) standard generative classifier and (b) auxiliary latent generative classifier.

We parameterize $p_\Theta(x \mid c, y)$ as a conditional LSTM language model using the same factorization as above:

$$\log p_\Theta(x \mid c, y) = \sum_{t=1}^{T} \log p_\Theta(x_t \mid x_{<t}, c, y) \quad (4)$$

where Θ is the set of parameters of the language model. As in the generative classifier, we use a label embedding matrix $V_\mathcal{Y}$. In addition, we define a latent variable embedding matrix $V_\mathcal{C} \in \mathbb{R}^{d_2 \times |\mathcal{C}|}$ where $\mathcal{C}$ is the set of values for the discrete latent variable. Also like the generative classifier, we use an LSTM to predict each word with a softmax layer:

$$p_\Theta(x_t \mid x_{<t}, c, y) \propto \exp\{u_{x_t}^\top([h_t; v_y; v_c]) + b_{x_t}\} \quad (5)$$

where h_t is the hidden representation of $x_{<t}$ from the LSTM, v_y and v_c are the embeddings for the label and the latent variable, respectively, $[u; v]$ denotes vertical concatenation, u_{x_t} is the output word embedding, and b_{x_t} is a bias parameter.

The prior distribution of the latent variable is parameterized as follows:

$$p_\Phi(c) \propto \exp\{w_c^\top v_c + b_c\} \quad (6)$$

where Φ is the set of parameters for this distribution which includes the vector w_c and scalar b_c for each c.

As in the standard generative model, the label prior $p_\Psi(y)$ is acquired from the empirical label distribution in the training data and remains fixed during training.

Training. As is standard in latent-variable modeling, we train our models by maximizing the log marginal likelihood:

$$\max_{\Theta,\Phi,V_\mathcal{C},V_\mathcal{Y}} \sum_{\langle x,y\rangle \in \mathcal{D}} \log \sum_{c\in\mathcal{C}} p_\Theta(x \mid c, y) p_\Phi(c) p_\Psi(y) \quad (7)$$

In NLP, these sorts of optimization problems are traditionally solved with the EM algorithm. However, we instead directly optimize the above quantity using automatic differentiation. This is natural because we use softmax-transformed parameterizations; a more traditional parameterization would assign parameters directly to individual probabilities, which then requires constrained optimization.

Inference. The prediction is made by marginalizing out the latent variables as follows:

$$\hat{y} = \operatorname*{argmax}_{y\in\mathcal{Y}} \sum_{c\in\mathcal{C}} p_\Theta(x \mid c, y) p_\Phi(c) p_\Psi(y) \quad (8)$$

We experimented with other inference objectives and found similar results. More details can be found in Appendix C.

Differences with ensembles. Our latent-variable model resembles an ensemble of multiple generative classifiers, but there are two main differences. First, all parameters in the latent generative classifier are trained jointly, while a standard ensemble combines predictions from multiple, independently-trained models. Joint training leads to complementary information being captured by latent variable values, as shown in our analysis. Moreover, a standard ensemble will lead to far more parameters (10, 30, or 50 times as many in our experimental setup) since each generative classifier is a completely separate model. Our approach simply conditions on the embedding of the latent variable value and therefore does not add many parameters.

4 Experiments

4.1 Datasets

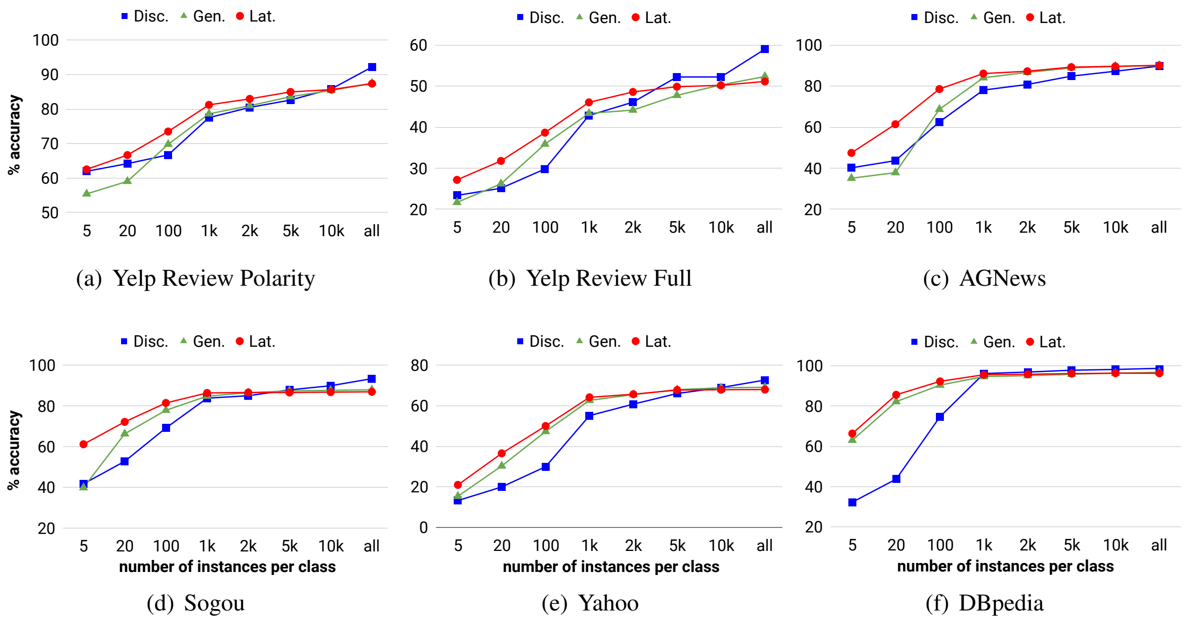
We present our results on six publicly available text classification datasets introduced by Zhang et al. (2015), which include news categorization, sentiment analysis, question/answer topic classification, and article ontology classification.[2]

To compare classifiers across training set sizes, we follow the setup of Yogatama et al. (2017) and construct multiple training sets by randomly sampling 5, 20, 100, 1k, 2k, 5k, and 10k instances per label from each dataset.

4.2 Training Details

In all experiments, the word embedding dimension and the LSTM hidden state dimension are set to

[2]A more detailed dataset description is in Appendix E.



(a) Yelp Review Polarity (b) Yelp Review Full (c) AGNews (d) Sogou (e) Yahoo (f) DBpedia

Figure 2: Comparison of classification accuracy of the discriminative (**Disc.**), standard generative (**Gen.**), and latent generative (**Lat.**) classifiers training across training set sizes.

100. All LSTMs use one layer and are unidirectional. The label dimensionality of all generative classifiers is set to 100. We adopt the same parameter settings as Yogatama et al. (2017) to ensure the results are comparable. For the latent-variable generative classifiers, we choose 10 or 30 latent variable values with embeddings of dimensionality 10, 50, or 100.

For optimization, we use Adam (Kingma and Ba, 2015) with learning rate 0.001. We do early stopping by evaluating the classification accuracy on the development set.

Due to memory limitations and computational costs, we truncate the length of the input sequences to 80 tokens before adding <s> and </s> to indicate the start and end of the document. Though truncation decreases the performance of the models, all models use the same truncated inputs, so the comparison is still fair.[3]

4.3 Baselines

To confirm we have built strong baselines, we first compare our implementation of the generative and discriminative classifiers to prior work. Our results in Appendix A show that our baselines are comparable to those of Yogatama et al. (2017).

[3]In other experiments, we compared performance with different truncation limits across training set sizes, finding the trends to be consistent with those presented here.

5 Results

5.1 Data Efficiency

Figure 2 shows results for the discriminative, generative, and latent generative classifiers in terms of data efficiency. Data efficiency is measured by comparing the accuracies of the classifiers when trained across varying sizes of training sets. Numerical comparisons on two datasets are shown in Table 1.

With small training sets, the latent generative classifier consistently outperforms both the generative and discriminative classifiers. When the generative classifier is better than the discriminative one, as in DBpedia, the latent classifier resembles the generative classifier. When the discriminative classifier is better, as in Yelp Polarity, the latent classifier patterns after the discriminative classifier. However, when the number of training examples is in the range of approximately 5,000 to 10,000 per class, the discriminative classifier tends to perform best.

With small training sets, the generative classifier outperforms the discriminative one in most cases except the very smallest training sets. For example, in the Yelp Review Polarity dataset, the first two points are from classifiers trained with only 10 and 40 instances in total. The other case in which generative classifiers underperform is when training over large training sets, which

	Δ(Lat., Gen.)		Δ(Lat., Disc.)	
	AGNews	**DBpedia**	**AGNews**	**DBpedia**
5	+12.3	+3.3	+7.2	+34.0
20	+23.5	+3.3	+17.7	+41.8
100	+9.8	+1.8	+16.0	+17.5
1k	+2.0	+0.9	+8.0	+0.0
all	+0.1	-0.4	+0.3	-2.4

Table 1: **Δ(Lat., Gen.)**: change in accuracy when moving from generative to latent generative classifier; **Δ(Lat., Disc.)**: change in accuracy when moving from discriminative to latent generative classifier. The first column shows the number of training instances per class.

	5	**20**	**100**	**1k**	**all**
Yelp P	+6.4	+8.2	+6.6	+8.6	+3.4
Yelp F	+4.7	+7.7	+11.3	+9.0	+11.8
AGNews	+13.4	+19.9	+27.9	+1.8	+0.4
Sogou	+22.3	+24.0	+13.9	+3.3	+2.8
Yahoo	+8.4	+17.7	+22.5	+11.1	+3.6
DBpedia	+44.6	+44.6	+28.4	+8.8	+2.3

Table 2: Changes in accuracy when adding a directed edge from the label to the input, i.e., the improvement in accuracy when moving from the middle to the hierarchical latent generative classifier. Each column shows a different number of training instances per class.

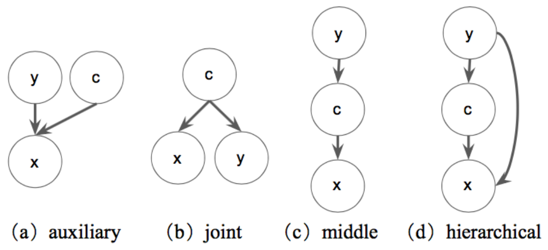


Figure 3: Graphical models of (a) auxiliary, (b) joint, (c) middle, and (d) hierarchical latent generative classifiers.

agrees with the theoretical and empirical findings in prior work (Yogatama et al., 2017; Ng and Jordan, 2002).

5.2 Effect of Graphical Model Structure

There are multiple choices to factorize the joint probability of the variables x, y, and c, which correspond to different graphical models. Here we consider other graphical model structures, namely those shown in Figure 3. We refer to the model in Figure 3(b) as the "joint" latent generative classifier since it uses the latent variable to jointly generate x and y. We refer to the model in Figure 3(c) as the "middle" latent generative classifier as the latent variable separates the textual input from the label. We use similar parameterizations for these models as for the auxiliary latent classifier, with conditional language models to generate x where the embedding of the latent variable is concatenated to the hidden state as in Section 3.

Figure 4 shows the comparison of the standard and the three latent generative classifiers on Yelp Review Polarity, AGNews, and DBpedia.[4] We observe that the auxiliary model consistently performs best, while the other two latent generative classifiers do not consistently improve over the standard generative classifier. On DBpedia, we see surprisingly poor performance when adding latent variables suboptimally. This suggests that the choice of where to include latent variables has a significant impact on performance.

Dependency between label and input variable. We observe that the most prominent difference between the auxiliary and the other two latent-variable models is that the label variable y is directly linked to the input variable x in the auxiliary model, which is also the case in the standard generative model.

In order to verify the importance of this direct dependency between the label and input, we create a new latent-variable model by adding a directed edge between y and x to the middle latent generative model. We refer to this model as the "hierarchical" latent generative classifier, which is shown in Figure 3(d). The results in Table 2 show the performance gains after adding this edge, which are all positive and sometimes very large. The resulting hierarchical model is very close in performance to the auxiliary model, which is unsurprising because these two models differ only in the presence of the edge from y to c.

5.3 Effect of Latent Variables

We conduct a comparison to demonstrate that the performance gains are due to the latent-variable structure instead of an increased number of parameters when adding the latent variables.[5]

For the latent generative classifier, we choose 10 latent variable values with embeddings of di-

[4]Similar trends are observed for all datasets, so we only show three for brevity.

[5]The results in the preceding sections use the models with configurations tuned on the development sets. We follow the practice of Yogatama et al. (2017) and fix label dimensionality to 100, as described in Section 4.2. The only tuned hyperparameters are the number of latent variable values and the dimensions of their embeddings.

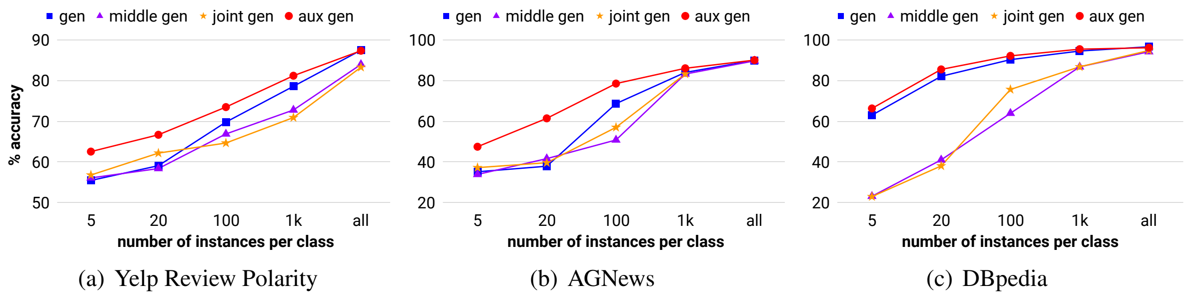


Figure 4: Comparison of generative classifier (**gen**) and latent generative classifiers (**middle gen, joint gen, aux gen**).

		5	20	100	1k	all
Yelp P	**Gen.**	55.4	59.0	69.8	78.6	87.5
	Gen. PC	55.4	58.4	69.5	78.5	87.5
	Lat.	62.5	66.6	73.5	81.2	87.3
	Lat. PC	62.2	66.2	73.0	80.8	87.1
AGNews	**Gen.**	35.1	37.8	68.7	84.0	90.0
	Gen. PC	33.7	37.8	68.4	83.2	89.8
	Lat.	47.4	61.4	78.5	86.1	90.1
	Lat. PC	47.2	61.4	78.3	84.5	90.1

Table 3: Accuracy comparison of standard generative (**Gen.**) and latent (**Lat.**) classifiers under earlier experimental configurations and parameter-comparison configurations (**PC**). When controlling for the number of parameters, the latent classifier still outperforms the standard generative classifier, which indicates the performance gains are due to the latent variables instead of an increased number of parameters.

	AGNews		**Sogou**	
	Direct	**EM**	**Direct**	**EM**
5	47.4 (62)	47.4 (62)	61.1 (144)	61.2 (147)
20	61.4 (72)	61.1 (69)	72.1 (30)	72.2 (29)
100	78.5 (10)	78.5 (10)	81.4 (11)	81.4 (11)
1k	86.1 (18)	86.1 (18)	86.4 (9)	85.9 (9)
all	90.1 (7)	90.0 (5)	86.9 (0)	86.9 (0)

Table 4: Comparison of the classification accuracy and convergence speed of the classifiers trained with direct optimization (**Direct**) of the log marginal likelihood and the EM algorithm (**EM**). The numbers inside the parentheses are the numbers of epochs required to reach the classification accuracies listed outside the parentheses.

mensionality 10, and a label dimensionality of 100 (**Lat. PC** in Table 3). For the standard generative classifier, we choose a label dimensionality of 110 (**Gen. PC** in Table 3). So, the numbers of parameters are comparable, since we ensure the same number of parameters in the "output" word embeddings in the softmax layer of the language model, which is the decision that most strongly affects the number of parameters.

Table 3 shows the results with different configurations, including the choices mentioned above as well as the results from earlier configurations mentioned in the paper. We observe that the latent generative classifiers still perform better in terms of data efficiency, which shows that the latent-variable structure accounts for the performance gains.

5.4 Learning via Expectation-Maximization

The results reported before are evaluated on the classifiers trained by directly maximizing the log marginal likelihood via gradient-based optimization. In addition, we train our latent generative classifiers with the EM algorithm (Salakhutdinov et al., 2003). More training details can be found in Appendix B.

To speed convergence, we use a mini-batch version of EM, updating the parameters after each mini-batch. Our results in Table 4 show that the direct approach and the EM algorithm have similar performance in terms of classification accuracy and convergence speed in optimizing the parameters of our latent models. Similar trends appear for the other datasets.

6 Analysis

6.1 Interpretation of Latent Variables

We take the strongest latent-variable model, the auxiliary latent generative classifier, and analyze the relationship among the latent, input, and label variables. We use the AGNews dataset, which contains 4 categories: world, sports, business, and sci/tech. The classifier we analyze has 10 values for the latent variable and is trained on a training set containing 1k instances per class.

We first investigate the relationship between the latent variable and the label by counting co-occurrences. For each instance in the development set, we calculate the posterior probability distribution over the latent variable, and pick the value with the highest probability as the preferred latent variable value for that instance. This is reasonable since in our trained model, the posterior distribution over latent variable values is peaked. Then we categorize the data by their preferred latent variable values and count the gold standard labels in each group. We observe that the labels are nearly uniformly distributed in each latent variable value, suggesting that the latent variables are not obviously being used to encode information about the label.

Thus, we hypothesize there should be information other than that pertaining to the label that causes the data to cluster into different latent variable values. We study the differences of the input texts among the 10 clusters by counting frequent words, manually scanning through instances, and looking for high-level similarities and differences. We report our manual labeling for the latent variable values in Table 5.

For example, value 1 is mostly associated with future and progressive tenses; the words "will", "next", and "new" appear frequently. Value 2 tends to contain past and perfect verb tenses (the phrases "has been" and "have been" appear frequently). Value 3 contains region names like "VANCOUVER", "LONDON", and "New Brunswick", while value 7 contains country-oriented terms like "Indian", "Russian", "North Korea", and "Ireland". Our choice of only 10 latent variable values causes them to capture the coarse-grained patterns we observe here. It is possible that more fine-grained differences would appear with a larger number of values.

6.2 Generation with Latent Variables

Another advantage of generative models is that they can be used to generate data in order to better understand what they have learned, especially in seeking to understand latent variables. We use our auxiliary latent generative classifier to generate multiple samples by setting the latent variable and the label. Instead of the soft mixture of discrete latent variable values that is used in classification (since we marginalize over the latent variable at test time), here we choose a single latent variable value when generating a textual sample.

To increase generation diversity, we use temperature-based sampling when choosing the next word, where higher temperature leads to higher variety and more noise. We set the temperature to 0.6. Note that the latent-variable model here is trained on only 4000 instances (1k for each label) from AGNews, so the generated samples do suffer from the small size of data used in training the language model. Table 6 shows some generated examples. We observe that different combinations of the latent variable and label lead to generations that comport with both the labels and our interpretations of the latent variable values.

We speculate that the reason our generative classifiers perform well in the data-efficient setting is that they are better able to understand the data via language modeling rather than directly optimizing the classification objective. Our generated samples testify to the ability of generative classifiers to model the underlying data distribution even with only 4000 instances.

7 Related Work

Supervised Generative Models. Generative models have traditionally been used in supervised settings for many NLP tasks, including naive Bayes and other models for text classification (Maron, 1961; Yogatama et al., 2017), Markov models for sequence labeling (Church, 1988; Bikel et al., 1999; Brants, 2000; Zhou and Su, 2002), and probabilistic models for parsing (Magerrnan and Marcus, 1991; Black et al., 1993; Eisner, 1996; Collins, 1997; Dyer et al., 2016). Recent work in generative models for question answering (Lewis and Fan, 2019) learns to generate questions instead of directly penalizing prediction errors, which encourages the model to better understand the input data. Our work is directly inspired by that of Yogatama et al. (2017), who build RNN-based generative text classifiers and show scenarios where they can be empirically useful.

Text Classification. Traditionally, linear classifiers (McCallum and Nigam, 1998; Joachims, 1998; Fan et al., 2008) have been used for text classification. Recent work has scaled up text classification to larger datasets with models based on logistic regression (Joulin et al., 2017), convolutional neural networks (Kim, 2014; Zhang et al., 2015; Conneau et al., 2017), and recurrent neural networks (Xiao and Cho, 2016; Yogatama et al.,

id	description	examples
1	future/progressive tenses	... Commission **is likely to** follow opinion in the U.S. on the merger suit ... ... to increase computer software exports **is beginning to** show results ...
2	past/perfect tense	A screensaver targeting spam-related websites appears to **have been** too successful . Universal **has signed** a handful of artists to a digital-only record label . ..
3	region names, locations	**Newcastle** manager Bobby Robson ... relieved of his duties ... **Newcastle** announced ... **ABUJA** ... its militias **in Darfur** before they would sign ...
4	mixture	
5	abbreviations	**St.** Louis advanced to the **N.L.** championship series for the third time in five years ... **UAL (UALAQ.OB : OTC BB** - news - research) ... (**UAIRQ.OB : OTC BB** ...
6	numbers, money-related	...challenge larger rivals in the fast-growing **2.1 billion-a-year** sleep aid market .... to a **$ 25,000 prize** , and more importantly , into the history books ...
7	dates	... an Egyptian diplomat said **on Friday**, and the abduction of ... **earlier this month** . ... expected **Monday or Tuesday** , ... doctors and nurses off for the **holiday weekend** ...
8	country-oriented terms	**Rwandan** President ... in the **Democratic Republic of the Congo** after ... Pope John Paul II issued a new appeal for peace **in Iraq and the Middle East** ...
9	mixure	
10	symbols, links	... **A HREF = ” http : / / www.reuters.co.uk / financeQuoteLookup.jhtml** ... **& lt ;** strong **& gt ;** Analysis **& lt ; /** strong **& gt ;** Contracting out the blame ...

Table 5: Latent variable values (“id”), our manually-defined descriptions, and examples of instances associated to them. Boldface is used to highlight cues to our labeling. We use the term “mixture” when we did not find clear signals to interpret the latent variable value.

Latent variable id = 3: region names, locations	
world	**BEIJING** (Reuters) - **Oklahoma** supporters unemployment claims that he plans to trying to restore access next season ’s truce by ruling , saying a major parliament .
sport	The **Dallas** Cowboys today continued advantage today with **Miami** and the Hurricanes had to get the big rotation for the first time this year .
business	**Las Vegas** took one more high-stepping kick across the pond as casino operator Caesars Entertainment Inc .
sci/tech	**SAN FRANCISCO** - Sun Microsystems on Monday will surely offer the deal to sell up pioneer members into two years and archiving .
Latent variable id = 6: numbers, money-related	
world	An Israeli helicopter gunship fired a missile among **$ 5** million in to Prime Minister Ariel Sharon on the streets of U.S. warming may not be short-lived .
sport	On Wednesday it would win the disgruntled one of the season opener in a # 36 ; **8.75 billion** of World Cup final day for second .
business	Reuters - U.S. drug company Biogen Idec is considering an all-share bid of more than **8.5 billion euros** (# 36 ; **10.6 billion**) for Irish peer Elan , a newspaper reported on Sunday .
sci/tech	The JVC Everio GZ-MC100 (**$ 1199.95**) and GZ-MC200 (**$ 1299.95**) will use **4GB** Microdrive cards , which are removable hard drives measuring **1.5** inches square , but will also lost vital ” fans to recently over .
Latent variable id = 10: symbols, links	
world	A German court is set to hear all its secular oil , and western Kerik in Fallujah . **& lt ; A HREF = ” http : / / www.investor.reuters.com / FullQuote.aspx ? ticker = Agency target = Army** ...
sport	White Sox to an overpowering 49-0 victory over The world championship game . **& lt ; br & gt ; & lt ; br & gt ;** Comcast SportsNet
business	NEW YORK (Reuters) - U.S. stocks climbed on Monday , with a steep decline in commodity prices and lower crude oil dented shares of Alcoa Inc . **& lt ; A HREF = ” http : / / www.investor.reuters.com / FullQuote.aspx ? ticker = GDT.N target = / stocks / quickinfo / fullquote ” & gt ; & lt ; / A & gt ;**.
sci/tech	Spyware problems introduced a radio frequency code Thursday . **& lt ; FONT face = ” verdana , MS Sans Serif , arial , helvetica ” size = ” -2 ” color = ” # 666666 ” & gt ; & lt ; B & gt ; -washingtonpost.com & lt ; / B & gt ; & lt**

Table 6: Generated examples by controlling the latent variables and labels (world, sport, business, sci/tech) with our latent classifier trained on a small subset of the AGNews dataset.

2017), the latter of which is most closely-related to our models.

Latent-variable Models. Latent variables have been widely used in both generative and discriminative models to learn rich structure from data (Petrov and Klein, 2007, 2008; Blunsom et al., 2008; Yu and Joachims, 2009; Morency et al., 2008). Recent work in neural networks has shown that introducing latent variables leads to higher representational capacity (Kingma and Welling, 2014; Chung et al., 2015; Burda et al., 2016; Ji et al., 2016). However, unlike variational autoencoders (Kingma and Ba, 2015) and related work that use continuous latent variables, our model is more similar to recent efforts that combine neural architectures with discrete latent variables and end-to-end training (Ji et al., 2016; Kim et al., 2017; Kong et al., 2017; Chen and Gimpel, 2018; Wiseman et al., 2018, *inter alia*).

8 Discussion and Future Work

An alternative solution to the small-data setting is to use language representations pretrained on large, unannotated datasets (Mikolov et al., 2013; Pennington et al., 2014; Devlin et al., 2019). In other experiments not reported here, we found that using pretrained word embeddings leads to larger performance improvements for the discriminative classifiers than the generative ones.

Future work will explore the performance of latent generative classifiers in other challenging experimental conditions, including testing robustness to data shift and adversarial examples as well as zero-shot learning. Another thread of future work is to explore the performance of discriminative models with latent variables, and investigate combining pretrained representations with both generative and discriminative classifiers.

9 Conclusion

We focused in this paper on improving the data efficiency of generative text classifiers by introducing discrete latent variables into the generative story. Our experimental results demonstrate that, with small annotated training data, latent generative classifiers have larger and more stable performance gains over discriminative classifiers than their standard generative counterparts. Analysis reveals interpretable latent variable values and generated samples, even with very small training sets.

Acknowledgments

We would like to thank Lingyu Gao, Qingming Tang, and Lifu Tu for helpful discussions, Michael Maire and Janos Simon for their useful feedback, the anonymous reviewers for their comments that improved this paper, and Google for a faculty research award to K. Gimpel that partially supported this research.

References

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine learning*, 34(1-3):211–231.

Ezra Black, Fred Jelinek, John Lafrerty, David M. Magerman, Robert Mercer, and Salim Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 31–37, Columbus, Ohio, USA. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-08: HLT*, pages 200–208, Columbus, Ohio. Association for Computational Linguistics.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Sixth Applied Natural Language Processing Conference*, pages 224–231, Seattle, Washington, USA. Association for Computational Linguistics.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Mingda Chen and Kevin Gimpel. 2018. Smaller text classifiers with discriminative cluster embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 739–745, New Orleans, Louisiana. Association for Computational Linguistics.

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

Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, Texas, USA. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, Valencia, Spain. Association for Computational Linguistics.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 332–342, San Diego, California. Association for Computational Linguistics.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Structured attention networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2017. Segmental recurrent neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Mike Lewis and Angela Fan. 2019. Generative question answering: Learning to answer the whole question. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

David M. Magerrnan and Mitchell P. Marcus. 1991. Pearl: A probabilistic chart parser. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. E. Maron. 1961. Automatic indexing: An experimental inquiry. *J. ACM*, 8(3):404–417.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, volume 752, pages 41–48.

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*, pages 3111–3119.

Louis-Philippe Morency, Xu Sun, Daisuke Okanoharay, and Jun'ichi Tsujii. 2008. Modeling Latent-Dynamic in Shallow Parsing: A Latent

Conditional Model with Improved Inference. In *The 22nd International Conference on Computational Linguistics (COLING 2008)*, Manchester, UK.

Andrew Y. Ng and Michael I. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in neural information processing systems*, pages 841–848.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2008. Discriminative log-linear grammars with latent variables. In *Advances in neural information processing systems*, pages 1153–1160.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. 2003. Relationship between gradient and EM steps in latent variable models.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium. Association for Computational Linguistics.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *ICML*, volume 2, page 5.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 473–480, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Victor Zue, Stephanie Seneff, and James Glass. 1990. Speech database development at MIT: TIMIT and beyond. *Speech communication*, 9(4):351–356.