Generative and Discriminative Text Classification with Recurrent Neural Networks

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom
DeepMind
{dyogatama,cdyer,lingwang,pblunsom}@google.com

Abstract

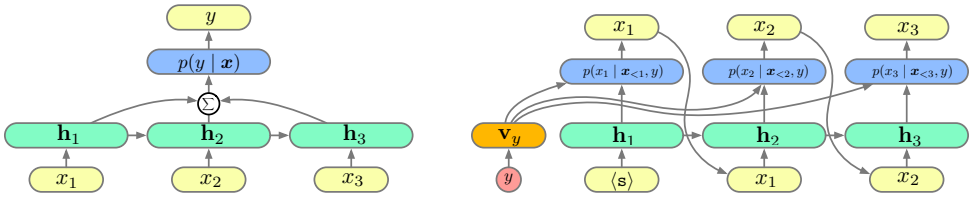
We empirically characterize the performance of discriminative and generative LSTM models for text classification. We find that although RNN-based generative models are more powerful than their bag-of-words ancestors (e.g., they account for conditional dependencies across words in a document), they have higher asymptotic error rates than discriminatively trained RNN models. However we also find that generative models approach their asymptotic error rate more rapidly than their discriminative counterparts—the same pattern that Ng & Jordan (2001) proved holds for linear classification models that make more naïve conditional independence assumptions. Building on this finding, we hypothesize that RNN-based generative classification models will be more robust to shifts in the data distribution. This hypothesis is confirmed in a series of experiments in zero-shot and continual learning settings that show that generative models substantially outperform discriminative models.

1 Introduction

Neural network models used in natural language processing applications are usually trained discriminatively. This strategy succeeds for many applications when training data is abundant and the data distribution is stable. Unfortunately, neural networks require a lot of training data, and they tend to generalize poorly when the data distribution shifts (e.g., new labels, new domains, new tasks). In this paper, we explore using generative models to obtain improvements in sample complexity and ability to adapt to shifting data distributions.

While neural networks are traditionally used as discriminative models (Ney, 1995; Rubinstein & Hastie, 1997), their flexibility makes them well suited to estimating class priors and class-conditional observation likelihoods. We focus on a simple NLP task—text classification—using discriminative and generative variant models based on a common neural network architecture (§2). These models use an LSTM (Hochreiter & Schmidhuber, 1997) to process documents as sequences of words. In the generative model, documents are generated word by word, conditioned on a learned class embedding; in the discriminative model the LSTM "reads" the document and uses its hidden representation to model the class posterior. In contrast to previous generative models for text classification, ours can model unbounded (conditional) dependencies among words in each document.

We demonstrate empirically that our discriminative model obtains a lower asymptotic error rate than its generative counterpart, but it approaches this rate more slowly (§3.4). This behavior is precisely the pattern that Ng & Jordan (2001) proved will hold in general for generative and discriminative *linear* models. Finding the same pattern with our models is somewhat surprising since our generative models are substantially more powerful than the linear models analyzed in that work (e.g., they model conditional dependencies among input features), and because their theoretical analysis relied heavily on linearity.



(a) Discriminative (b) Generative

Figure 1: Illustrations of our discriminative (left) and generative (right) LSTM models.

Encouraged by this result, we turn to learning problems where good sample complexity is crucial for success and explore whether generative models might be preferable to discriminative ones. We first consider the single-task continual learning setting in which the labels (classes) are introduced sequentially, and we can only learn from the newly introduced examples (§3.5). Discriminative models are known to suffer from catastrophic forgetting when learning sequentially from examples from a single class at a time, and specialized techniques are actively being developed to minimize this problem (Rusu et al., 2016; Kirkpatrick et al., 2017; Fernando et al., 2017). Generative models, on the other hand, are a more natural fit for this kind of setup since the maximization of the training objective for a new class can be decoupled from other classes more easily (e.g., parameters of a naïve Bayes classifier can be estimated independently for each class). In order to compare discriminative and generative models more fairly, we use a generative model that shares many parameters across classes and evaluate its performance in this setting.

Finally, we compare the performance of discriminative and generative LSTM language models for zero-shot learning, where we construct a semantic label space that is fixed during training based on an auxiliary task (§3.6). We investigate whether learning to map documents onto this semantic space (discriminative training) or learning to generate from points in the semantic space (generative training) is better. Here, we find substantial benefits for generative models.

2 Models

Inputs to a text classification system are a document $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$, where T is its length in words, and it will predict a label $y \in \mathcal{Y}$. We compare discriminative and generative text classification models. Discriminative models are trained to distinguish the correct label among possible choices. Given a collection of labeled documents $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, these models are trained to maximize the conditional probability of the labels given the documents: $\sum_{i=1}^N \log p(y_i \mid \boldsymbol{x}_i)$. Generative models, on the other hand, are trained to maximize the joint probability of labels and documents under the following factorization: $\sum_{i=1}^N \log p(\boldsymbol{x}_i, y_i) = \sum_{i=1}^N \log p(\boldsymbol{x}_i \mid y_i) p(y_i)$. When predictions are made, Bayes' rule is used to compute $p(y \mid \boldsymbol{x})$.

In both models, we represent a word x by a D-dimensional embedding $\mathbf{x} \in \mathbb{R}^D$. Figure 1 shows an illustration of our models, and we describe them in details in the following section.

2.1 Discriminative Model

Our discriminative model uses LSTM with "peephole" connections to encode a document and build a classifier on top of the encoder by using the average of the LSTM hidden representations as the document representation.

Specifically, given an input word embedding $\mathbf{x}_t$, we compute its hidden representation $\mathbf{h}_t \in \mathbb{R}^E$ with LSTM as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{x}_t; \mathbf{h}_{t-1}; \mathbf{c}_{t-1}] + \mathbf{b}_i) \qquad \mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{x}_t; \mathbf{h}_{t-1}; \mathbf{c}_{t-1}] + \mathbf{b}_f)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c[\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{b}_c) \qquad \mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{x}_t; \mathbf{h}_{t-1}; \mathbf{c}_t] + \mathbf{b}_o)$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$$

where $[\mathbf{u}; \mathbf{v}]$ denotes vector concatenation. We then add a softmax layer on top of this LSTM, so the probability of predicting a label $y \in \mathcal{Y}$ is: $p(y \mid \boldsymbol{x}) \propto \exp((\frac{1}{T}\sum_{t=0}^T \mathbf{h}_t^\top)\mathbf{v}_y + b_y)$, where $\mathbf{V} \in \mathbb{R}^{E \times |\mathcal{Y}|}$ is the softmax parameters and $\mathbf{b} \in \mathbb{R}^{|\mathcal{Y}|}$ is the bias. We use a simple average of LSTM hidden representations since in our preliminary experiments it works better than using the last hidden state $\mathbf{h}_T$, and it is computationally much cheaper for long documents (hundreds of words) than attention-based models. Importantly, this model is trained discriminatively to maximize the conditional probability of the label given the document: $p(y \mid \boldsymbol{x}; \mathbf{W}, \mathbf{V})$.

2.2 Generative Models

Our generative model is a class-based language model, shown in Figure 1. Here, we similarly compute hidden representation $\mathbf{h}_t$ with LSTM. Additionally, we also have a label embedding matrix $\mathbf{V} \in \mathbb{R}^{E \times |\mathcal{Y}|}$. We use the chain rule to factorize the probability $p(\boldsymbol{x} \mid y)$ into a sequential prediction: $p(\boldsymbol{x} \mid y) = \prod_{t=1}^{T} p(x_t \mid \boldsymbol{x}_{<t}, y)$. To predict the word x_t, we concatenate the LSTM's hidden representation $\boldsymbol{x}_{<t}$ (called $\mathbf{h}_t$) with the label embedding $\mathbf{v}_y$ and add a softmax layer over vocabulary with parameters $\mathbf{U}$ and class-specific bias parameters $\mathbf{b}_y$: $p(x_t \mid \boldsymbol{x}_{<t}, y) \propto \exp(\mathbf{u}_{x_t}^{\top}[\mathbf{h}_t; \mathbf{v}_y] + b_{y,x_t})$. We designate this model "Shared LSTM", since it shares some parameters across classes (i.e., the word embedding matrix, LSTM parameters $\mathbf{W}$, and softmax parameters $\mathbf{U}$). This model's novelty owes to the fact that there is a single conditional model that shares parameters whose behavior is modulated by the given label embedding, whereas in traditional generative classification models, each label has an independent LM associated with it, such as the generative n-gram language classification models in Peng & Schuurmans (2003).

In addition to the above model, we also experiment with a class-based generative language model where there is no shared component among classes (i.e., every class has its own word embedding, LSTM, and softmax parameters). One benefit of this approach is that training can be parallelized across classes, although the resulting model has larger number of parameters. We denote this model by "Independent LSTMs".

Note that the underlying LSTM of both our generative models is similar to the discriminative model, except that it is trained to maximize the joint probability $p(y, \boldsymbol{x}; \mathbf{W}, \mathbf{V}, \mathbf{U}) = p(\boldsymbol{x} \mid y; \mathbf{W}, \mathbf{V}, \mathbf{U})p(y)$. In terms of the number of parameters, these generative models have extra parameters $\mathbf{U}$ that are needed to predict words (we can view the label embedding matrix as a substitute for the softmax parameter in the discriminative case). For prediction, we compute $\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} p(\boldsymbol{x} \mid y; \mathbf{W}, \mathbf{V}, \mathbf{U})p(y)$ using the empirical relative frequency estimate of $p(y)$.

3 Experiments

3.1 Datasets

We use publicly available datasets from Zhang et al. (2015) to evaluate our models (`http://goo.gl/JyCnZq`). They are standard text classification datasets that include news classification, sentiment analysis, Wikipedia article classification, and questions and answers categorization. Table 1 shows descriptive statistics of datasets used in our experiments. For each dataset, we randomly hold 5,000 examples from the original training set to be used as our development set.

Table 1: Descriptive statistics of datasets used in our experiments.

Name	#Train	#Dev	# Test	# Classes
AGNews	115,000	5,000	7,600	4
Sogou	445,000	5,000	60,000	5
Yelp	645,000	5,000	50,000	5
Yelp Binary	555,000	5,000	7,600	2
DBPedia	555,000	5,000	70,000	14
Yahoo	1,395,000	5,000	60,000	10

3.2 Baselines

In addition to our generative vs. discriminative models in §2 and baselines from previous work on these datasets, we also compare with the following generative models:

Naïve Bayes classifier. A simple count-based unigram language model that uses naïve Bayes assumption to factorize $p(\boldsymbol{x} \mid y) = \prod_{t=1}^{T} p(x_t \mid y)$.

Kneser–Ney Bayes classifier. A more sophisticated count-based language model that uses trigrams and Kneser–Ney smoothing $p(\boldsymbol{x} \mid y) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, x_{t-2}, y)$. Similar to the naïve Bayes classifier, we construct one language model per class and predict by computing: $\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} p(\boldsymbol{x} \mid y)p(y)$.

Naïve Bayes neural network. Last, we also design a naïve Bayes baseline where $p(x_t \mid y)$ is modeled by a feedforward neural network (in our case, we use a two layer neural network). This

Table 2: Summary of results on the full datasets.

Models	AGNews	Sogou	Yelp Bin	Yelp Full	DBPed	Yahoo
Naïve Bayes	90.0	86.3	86.0	51.4	96.0	68.7
Kneser–Ney Bayes	89.3	94.6	81.8	41.7	95.4	69.3
MLP Naïve Bayes	89.9	76.1	73.6	40.4	87.2	60.6
Discriminative LSTM	92.1	94.9	92.6	59.6	98.7	73.7
Generative LSTM–independent comp.	90.7	93.5	90.0	51.9	94.8	70.5
Generative LSTM–shared comp.	90.6	90.3	88.2	52.7	95.4	69.3
bag of words (Zhang et al., 2015)	88.8	92.9	92.2	58.0	96.6	68.9
fastText (Joulin et al., 2016)	92.5	96.8	95.7	63.9	98.6	72.3
char-CNN (Zhang et al., 2015)	87.2	95.1	94.7	62.0	98.3	71.2
char-CRNN (Xiao & Cho, 2016)	91.4	95.2	94.5	61.8	98.6	71.7
very deep CNN (Conneau et al., 2016)	91.3	96.8	95.7	64.7	98.7	73.4

is an extension of the naïve Bayes baseline, where we replace the class-conditional count-based unigram language model with a class-conditional vector-based unigram language model.

3.3 Implementation Details

In all our experiments, we set the word embedding dimension D and the LSTM hidden dimension E to 100.[1] For the generative model, the dimension of the class embedding is also set to 100. We train our model using AdaGrad (Duchi et al., 2012) and tune the learning rate on development sets. We also use the development sets to decide when to stop training based on classification accuracy as the evaluation metric.

3.4 Sample Complexity and Asymptotic Errors

Ng & Jordan (2001) theoretically and empirically show that generative linear models reach their (higher) asymptotic error faster than discriminative models (naïve Bayes classifier vs. logistic regression). While it is difficult to derive the theoretical properties of expressive recurrent neural network models such as ours, we empirically evaluate the performance of these models.

Table 2 summarizes our results on the full datasets, along with results from previous work on these datasets. Our discriminative LSTM model is competitive with other discriminative models based on logistic regression (Zhang et al., 2015; Joulin et al., 2016) or convolutional neural networks (Zhang et al., 2015; Xiao & Cho, 2016; Conneau et al., 2016). All of the generative models have lower classification accuracies. These results agree with Ng & Jordan (2001) that discriminative models have lower asymptotic errors than generative models.

Comparing various generative models, we can see that the generative LSTM models are generally better than baseline generative models with stronger independence assumptions (i.e., naïve Bayes, Kneser–Ney Bayes, and naïve Bayes neural network). Our results suggest that LSTM is an effective method to capture dependencies among words in a document. We also compare the two generative LSTM models: shared LSTM and independent LSTMs. The results are roughly similar.

Next, we evaluate our models with varying training size. For each of our six datasets, we randomly choose 5, 20, 100, and 1000 examples *per class*. We train the models on these smaller datasets and report results in Figure 2. Our results show that the generative shared LSTM model outperforms the discriminative model in almost all cases in the small-data regime on all datasets except one (AG News). Among generative models, the generative LSTM model still achieves better classification accuracies compared to naïve Bayes and Kneser–Ney Bayes models, even in the small-data regime. While it is difficult to analyze the theoretical sample complexity of deep recurrent models, we see this collection of results as an empirical support that generative nonlinear models have lower sample complexity than their discriminative counterparts.

3.5 Continual learning

Our next set of experiments investigate properties of discriminative and generative LSTM models to adapt to data distribution shifts. An example of data distribution shift is when new classes are introduced to the models. In the real-world setting, being able to detect emergence of a new class

[1] We also experimented with setting both D and E to 50 and 300, they resulted in comparable performance.

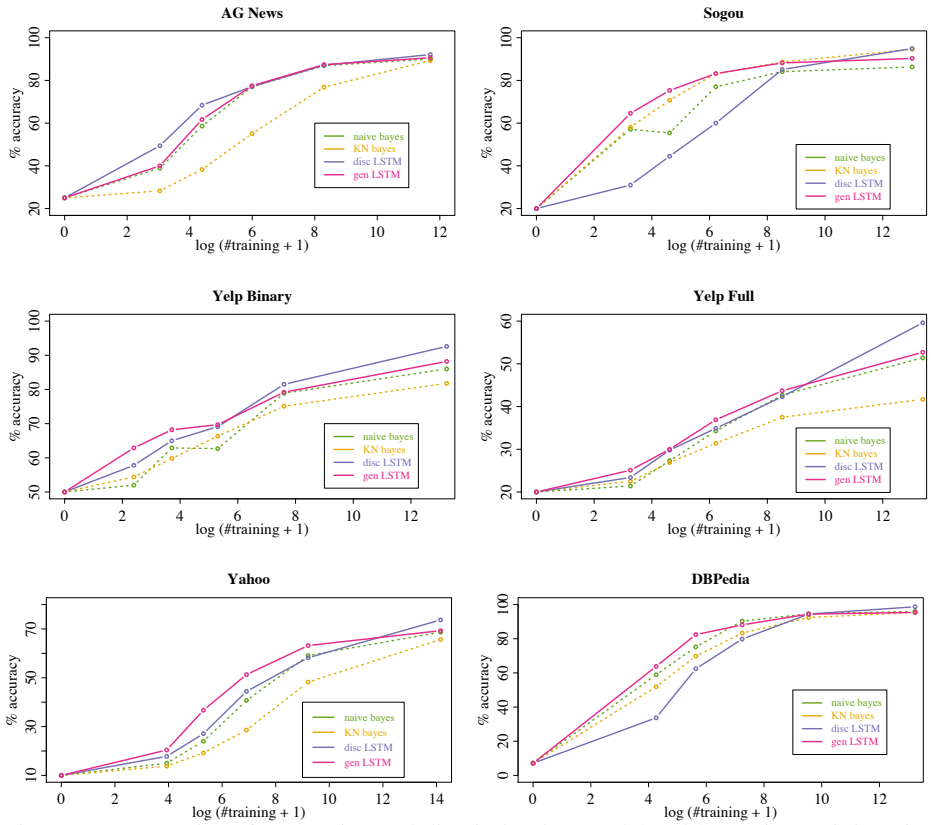


Figure 2: Accuracies of generative and discriminative models with varying training size.

and train on examples from a new class that shows up at a later time without having to retrain the model on the entire dataset is extremely attractive, especially in cases where we have a very large dataset and model. We focus on how well these models learn from new classes in this (sub)section and discuss detection of data distribution shifts in more details in §4.

Setup. We consider the setup where models are presented with examples from each class sequentially (single-task continual learning).[2] Here, each model has to learn information from newly introduced examples to be able to correctly classify documents into this new class, but they cannot train from previously seen classes while doing so. Table 3 summarizes our results and we discuss them in details in the followings.

Discriminative LSTM. We first investigate the performance of the LSTM discriminative model. Discriminative models are known to suffer from catastrophic forgetting—where the models overtrain on the newly introduced class and fail to retain useful information from previous classes—in this setup. In these experiments, every time we see examples from a new class, we update all parameters of the model with information the new examples. However, even after extensive tuning of learning rate and freezing some components of the network,[3] we were unable to avoid catastrophic forgetting. We observe that since the model is trained to *discriminate* among possible classes, when it only sees examples from a single class for tens or hundreds of iterations, it adjusts its parameters to always predict the new class. Of course, in theory, there might be an oracle learning rate that would prevent catastrophic forgetting while still acquiring enough knowledge about the newly introduced classes to update the model. However, we find that in practice it is very difficult to discover these learning rate values, especially for a reasonably large LSTM model that takes high-dimensional input such as a long news article. Even for the same learning rate value, the development set performance varies widely across multiple training runs. Note that since this model is trained discriminatively,

[2]Here, we focus on a single-task continual learning setup, although the idea can be generalized to a multitask setting as well.

[3]For example, we experiment with fixing the word embedding matrix and train all other components and fixing the LSTM language model component after seeing the first class and only train the softmax parameters.

Datasets	Shared-Gen	Ind.-Gen	Disc.
AG News	90.2	90.7	40.5
Yelp Full	51.4	52.7	20.0
Yelp Binary	86.4	90.0	57.2
DBPedia	95.7	94.8	8.3
Yahoo	68.5	70.5	10.0

Table 3: Continual learning results. Shared and Ind.-Gen are the generative shared and independent models respectively (see text for details).

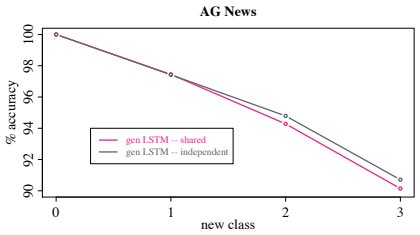


Figure 3: Classification accuracies on the AG News dataset for the generative LSTM models as we introduce class 0, 1, 2, and 3.

it is also not trivial make use of information from unlabeled data to pretrain some components of the model, except for the word embedding matrix. A promising method to prevent catastrophic forgetting in discriminative models is elastic weight consolidation (Kirkpatrick et al., 2017). However, the method requires computing a Fisher information matrix, and it is not clear how to compute it efficiently for complex models such as LSTM on GPUs.

Generative LSTM. Next, we consider the generative independent LSTMs model. Parameter estimations of some generative models, such as the naïve Bayes and this independent LSTMs, can be naturally decoupled across classes. As a result, these models can easily incorporate information from newly introduced examples from a new class. Every time a new class is introduced, we simply learn a new model of $p(\boldsymbol{x} \mid y_{\text{new}}; \mathbf{W}_{y_{\text{new}}}, \mathbf{v}_{y_{\text{new}}}, \mathbf{U}_{y_{\text{new}}})$ and $\forall y \in \mathcal{Y}$, update $p(y)$. One possible drawback of this approach is that the size of the model grows with the number of classes.

Last, we experiment with the generative shared LSTM model. Our training procedure for this model is as follows. We first train the LSTM language model part on a large amount of unlabeled data.[4] When training the language model on unlabeled data, we remove the class-specific bias component $\mathbf{b}_y$ and set the class embedding $\mathbf{v}_y$ to be a random vector $\tilde{\mathbf{y}}$ with a bounded norm $\|\tilde{\mathbf{v}}_y\|_2 \leq 1$. After we pretrain the shared components, we freeze their parameters and tune the class embedding $\mathbf{v}_y$ as well as the class-specific softmax bias $\mathbf{b}_y$ on the labeled training data. The benefit of this training—compared to having a separate LSTM model for each class—is that it is faster to train (in the presence of examples from a new class). Given a new class y, we only need to learn two vectors: $\mathbf{v}_y$ and $\mathbf{b}_y$. We can see from results in Table 3 and Figure 3 that the generative shared LSTM model trained with this procedure approaches the performance of its equivalent model that can see examples from all classes, and performs competitively with the generative independent LSTMs model.

3.6 Zero-shot learning

Our last set of experiments compare the performance of discriminative and generative LSTM language models for zero-shot learning, where the label embedding space is fixed based on an auxiliary task. Humans can acquire new concepts and learn relations among these concepts from an external task, and use this knowledge effectively across multiple tasks. In these experiments, we use datasets where the class labels are semantically meaningful concepts (e.g., `science`, `sports`, `business`—instead of star ratings from 1 to 5).

Setup. We remove labels of documents from one of the classes, but we provide the models with knowledge about the classes from external sources in the form of class embeddings $\mathbf{V}$. For example, when labels are words (e.g. `science`, `sports`, `business`, etc.), we construct a semantic space by learning the label embeddings $\mathbf{v}_y$ using standard word embedding techniques for all $y \in \mathcal{Y}$. In order to do this, we use pretrained GloVe word embedding vectors (Pennington et al., 2014).[5] In cases where the class labels consist of more than one words (e.g., society and culture), we choose one word from the labels (e.g., society). At test time, we see examples from all classes and evaluate precision and recall of the hidden class, as well as the overall accuracy on all classes.

Discriminative LSTM. The model learns from the labeled data to place documents in the semantic space, such that embeddings of documents are close to embeddings of their respective labels. In practice, we fix the softmax parameters $\mathbf{V}$ and learn an embedding of the document to maximize $\exp((\frac{1}{T}\sum_{t=0}^{T}\mathbf{h}_t^\top)\mathbf{v}_y)$, where $\mathbf{h}_t$ is the LSTM hidden state for word t in the document. In our experiments, this model never predicts the hidden class (zero precision and recall). Our results show

[4] In practice, we train on the corresponding training dataset without using document labels.

[5] `http://nlp.stanford.edu/projects/glove/`

Table 4: Zero shot learning results on four datasets. Hidden class indicates the class that is not included in the training data. We show precision and recall on test data for the hidden class, as well as accuracy for examples from all classes.

Dataset	Hidden Class	Prec.	Recall	Acc.
AG News	world	94.4	77.8	87.8
	sports	95.7	83.3	85.5
	business	84.9	60.1	83.9
	science and tech	92.0	54.3	83.2
Sogou	sports	95.0	80.5	87.4
	finance	24.2	0.7	73.1
	entertainment	90.2	78.8	86.6
	automobile	42.9	0.7	72.1
	science and tech	99.7	58.7	85.6
Yahoo	society and culture	42.8	7.9	64.9
	science and math	48.3	9.8	63.2
	health	26.3	0.4	61.8
	education and reference	23.5	3.8	65.2
	computers and internet	45.4	3	60.8
	sports	52.9	52.9	64.6
	business and finance	43.6	17.3	66.2
	entertainment and music	44.9	2.3	63.2
	family and relationships	8.3	0.05	62.5
	politics and government	48.6	10.4	62.1
DBPedia	company	98.9	46.6	93.3
	educational institution	99.2	49.5	92.8
	artist	88.3	4.3	90.3
	athlete	96.5	90.1	94.6
	office holder	0	0	89.1
	mean of transportation	96.5	74.3	94.2
	building	99.9	37.7	92.1
	natural place	98.9	88.2	95.4
	village	99.9	68.1	93.8
	animal	99.7	68.1	93.8
	plant	99.2	76.9	94.3
	album	0.03	0.001	88.8
	film	99.4	73.3	94.5
	written work	93.8	26.5	91.3

that while discriminative training of an expressive model such as LSTM on high dimensional text data produces a reliable predictor of seen classes, the resulting model overfits to the seen classes.

Generative LSTM. The model learns to generate from points in the label semantic space using the labeled documents. The model may infer how to generate a document about `politics` without ever having seen such an example in the training data. Similar to the discriminative case, we fix $\mathbf{V}$, which in this case plays the role of class embeddings, and train other parts of the model on all training data except examples from the hidden class (we use the generative shared LSTM since naturally the generative independent LSTMs cannot be used in this experiment). We observe on the development set that this model is able to predict examples from the unseen class with high precision, but very low recall ($\approx 1\%$). We design a self-training algorithm that add predicted hidden class examples from the development set to the training set and allow the model to train on these predicted examples. We show the results in Table 4. We can see that for most hidden classes, the generative model achieves good performance. For example, on the AG News dataset, the model performs reasonably well for any of the hidden classes. For a more difficult dataset where the overall accuracy is not very high such as Yahoo, the precision of the hidden class is lower, and as a result the recall also suffers. Nonetheless, the model is still able to achieve reasonable overall accuracy in some cases (recall that the accuracy of this model trained on the full dataset without any hidden class is 69.3).

Of course, if we include predicted hidden class examples to the training set of a discriminative model, it can also achieve good performance on all classes. However, the main point is that the discriminative LSTM model *never* predicts the hidden classes without any training data.

Two-class zero-shot learning. We also perform experiments with the generative LSTM model when we hide two classes on the AG News dataset and show the results in Table 5. In this case, the model does not perform as well since the precision of predicting hidden classes drops significantly, introducing too much noise in the training data. However, the model is still able to learn some useful

Table 5: Zero-shot learning results with two hidden class on the AG News dataset. We show P0 (P1) and R0 (R1) that indicate precision and recall for hidden class one (two), as well as overall accuracy.

Classes	P0	R0	P1	R1	Acc.
world+sports	43.2	3.4	54.7	90.2	67.2
world+business	55.4	75.6	25.9	2.2	67.6
world+science/tech	40.5	5.7	38.7	47.8	61.1
sports+business	62.3	80.7	48.3	6.6	67.6
sports+science/tech	66.2	85.5	66.8	6.7	67.6
business+science/tech	43.6	62.1	59.0	1.9	63.3

information since the overall accuracy is still higher than 50% (the accuracy of models that are only trained on two classes without any zero-shot learning of the hidden classes).

4 Discussion

Computational complexity. In terms of training and inference time, discriminative models are much faster. For example, for our smallest (biggest) dataset that contains 115,000 (1,395,000) training examples, it takes approximately two hours (two days) to get good generative models, whereas training the discriminative models only takes approximately 20 minutes (6 hours). The main drawback of generative models in NLP applications is in the softmax computation, since it has to be done over an entire vocabulary set. In many cases, such as in our experiments, the size of the vocabulary is in the order of hundreds of thousands. There are approximate methods to speed up this softmax computation, such as via hierarchical softmax (Morin & Bengio, 2005), noise contrastive estimation (Mnih & Teh, 2012), sampled softmax (Jean et al., 2015), or one-vs-each approximation (Titsias, 2017). However, even with these approximations, discriminative models are still much faster.

Data likelihood. In generative models, we can easily compute the probability of a document by marginalizing over classes: $p(\boldsymbol{x}) = \sum_{y \in \mathcal{Y}} p(\boldsymbol{x} \mid y)p(y)$. Since there is no explicit model of $p(\boldsymbol{x})$ in discriminative models, obtaining it would require a separate language model training. We explore whether we can use $p(\boldsymbol{x})$ as an indicator of the presence of a new (unknown) class. We use the AG News corpus and train the generative LSTM model on examples from only 3 labels (recall that there are 4 labels in this corpus). We compute $p(\boldsymbol{x})$ for all documents in the test set and show the results in Figure 4. We can see that examples from the class where there is no training data (i.e., class 0 and class 1 in the top and bottom figures, respectively) tend to have lower marginal likelihoods than examples from classes observed in the training data. In practice, we can use this observation to see whether there is data distribution shifts and we need to update parameters of our models.

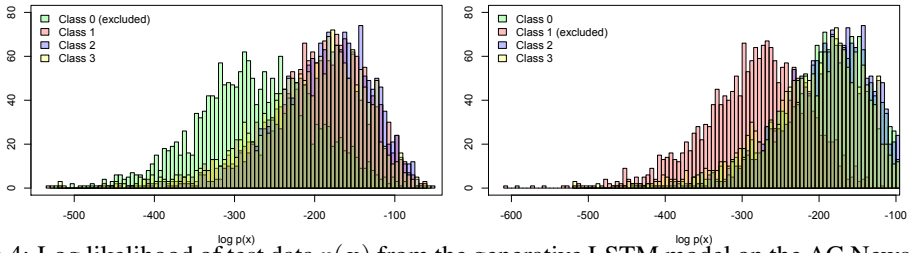


Figure 4: Log likelihood of test data $p(\boldsymbol{x})$ from the generative LSTM model on the AG News dataset when training data only includes three classes. In the top plot, we exclude training examples from class 0, whereas in the bottom plot we exclude training examples class 1. See text for details.

5 Conclusion

We have compared discriminative and generative LSTM-based text classification models in terms of sample complexity and asymptotic error rates. We showed that generative models are better than their discriminative counterparts in small-data regime, empirically extending the (theoretical) results of Ng & Jordan (2001) from linear to nonlinear models. Formal characterization of the generalization behavior of complex neural networks is difficult, with findings from convex problems failing to account for empirical facts about generalization Zhang et al. (2017). As such, this result is remarkable for being one domain in which generalization behavior of simpler models transfers to more complex models.

We also investigated their properties in the continual and zero-shot settings. Our collection of results showed that generative models are more suitable in these settings and they were able to obtain comparable performance to generative models trained on the full datasets in the standard setting.

References

Conneau, Alexis, Schwenk, Holger, Barrault, Loic, and Lecun, Yann. Very deep convolutional networks for text classification. *arXiv preprint*, 2016.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Resesarch*, 12:2121–2159, 2012.

Fernando, Chrisantha, Banarse, Dylan, Blundell, Charles, Zwols, Yori, Ha, David, Rusu, Andrei A., Pritzel, Alexander, and Wierstra, Daan. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint*, 2017.

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

Jean, Sebastian, Cho, Kyunghyun, Memisevic, Roland, and Bengio, Yoshua. On using very large target vocabulary for neural machine translation. In *Proc. of ACL-IJCNLP*, 2015.

Joulin, Armand, Grave, Edouard, Bojanowski, Piotr, and Mikolov, Tomas. Bag of tricks for efficient text classification. *arXiv preprint*, 2016.

Kirkpatrick, James, Pascanu, Razvan, Rabinowitz, Neil, Veness, Joel, Desjardins, Guillaume, Rusu, Andrei A., Milan, Kieran, Quan, John, Ramalhoa, Tiago, Grabska-Barwinska, Agnieszka, Hassabis, Demis, Clopath, Claudia, Kumaran, Dharshan, and Hadsell, Raia. Overcoming catastrophic forgetting in neural networks. *arXiv preprint*, 2017.

Mnih, Andriy and Teh, Yee Whye. A fast and simple algorithm for training neural probabilistic language models. In *Proc. of ICML*, 2012.

Morin, Frederic and Bengio, Yoshua. Hierarchical probabilistic neural network language model. In *Proc. of AISTATS*, 2005.

Ney, Hermann. On the probabilistic interpretation of neural network classifiers and discriminative training criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2), 1995.

Ng, Andrew Y. and Jordan, Michael I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Proc. of NIPS*, 2001.

Peng, Fuchun and Schuurmans, Dale. Combining naive Bayes and n-gram language models for text classification. In *Proc. of ECIR*, 2003.

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

Rubinstein, Y. Dan and Hastie, Trevor. Discriminative vs informative learning. In *Proc. KDD*, 1997.

Rusu, Andrei A., Rabinowitz, Neil C., Desjardins, Guillaume, Soyer, Hubert, Kirkpatrick, James, Kavukcuoglu, Koray, Pascanu, Razvan, and Hadsell, Raia. Progressive neural networks. *arXiv preprint*, 2016.

Titsias, Michalis K. One-vs-each approximation to softmax for scalable estimation of probabilities. In *Proc. of NIPS*, 2017.

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

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. In *Proc. of ICLR*, 2017.

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