THE UNIVERSITY OF CHICAGO

# BEYOND ACCURACY: MODELING TEXT FOR ROBUST NLP

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY

XIAOAN DING

CHICAGO, ILLINOIS
AUGUST 2022

For my supportive family

“This life is the crossing of a sea, where we meet in the same narrow ship.”

# Table of Contents

# List of Figures

# List of Tables

# ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor, Professor Kevin Gimpel, as well as our director, Professor Janos Simon, for giving me the opportunity to work on interesting research problems, for their continuous supports of my Ph.D. study and research, for their patience, enthusiasm, and immense knowledge. Their help and guidance helped me throughout all the time of research and life. They are the best advisors and mentors. Professor Gimpel taught me how to do research step by step, from understanding to exploration, from experiments to writing. I consider myself extremely lucky to be one of Professor Gimpel's students.

Besides my advisors, I would like to thank the rest of my thesis committee, Professor Chenhan Tan and Professor Sam Wiseman, for their support, encouragement and insightful suggestions in projects, thesis, and presentation slides. I am honored to have all of you on my thesis committee.

I would like thank my labmates, Lifu Tu, Qingming Tang, Zewei Chu, Mingda Chen, Lingyu Gao, Freda Shi, and Davis Yoshida, for our countless productive discussions. Also, I would like to thank to my previous collaborators, Tianyu Liu, Sean Gao, Yizhe Zhang, with whom I enjoyed learning and working and had a lot of enlightening.

My sincere thanks goes to computer science faculties at The University of Chicago and Toyota Technological Institute at Chicago. Thank you all those who created welcoming and comfortable environment to study, to do research, and to be productive: professors, fellow students and friends, and the department staff.

Additionally, I want to express my sincere grateful to my mentors and advisors prior to joining the University of Chicago, Yi Liu, Depei Qian, Zhongzhi Luan, at Beihang University, who enlighten me the first glance of research.

Last but not the least, I am grateful to my parents for their love, understanding, and support. You are the best parents in the world! And many thanks to all the friends I met in Chicago, a very long list in my heart, without of you this journey would not be possible.

# ABSTRACT

Research in model robustness has a long history. Improving model robustness generally refers to the goal of ensuring machine learning models are resistant to a variety of imperfect training and testing conditions. With the unprecedented progress in deep learning architectures, large-scale training, and learning algorithms, pretrained models have become pivotal in AI. However, when considering real-world scenarios, these models are still fragile and brittle, which impedes the safe deployment of NLP models in production systems.

In this work, we consider wide applications in NLP and define model robustness to broader aspects: (1) data-efficient: models can adapt to new domains with limited annotated data in both pretrained-finetuned and trained-from-scratch set-ups; (2) resilient: models can perform reliably under uncertainties and challenging circumstances; (3) fair: predictors or generators can make safe decisions and filter undesirable biases especially those imbued with toxicity, hate, and social bias; (4) trusted: models are expected to yield factual and faithful content.

To tackle these robustness issues, on the modeling side, we explored both discrete and continuous latent-variable generative models and various graphical model configurations; on the learning algorithms side, we investigated generative pretraining and various discriminative finetuning objectives in generative classifiers, gradient-based optimization, and the importance-sampled log marginal likelihood on learning deep latent-variable models; on the applications side, we developed document classifiers, textual relation predictors, a controllable story generator, and a hallucinated content detector.

# CHAPTER 1

# INTRODUCTION

Recent advances in large-scale pretrained language models, such as the BERT (Devlin et al., 2019a) and GPT (Radford et al., 2019) families of models, have achieved great progress in many Natural Language Processing (NLP) tasks, including document classification, syntactic or semantic parsing of text, question answering, textual entailment, etc. The performance gain is generally measured and observed on standard benchmarks, under assumptions of ideal training-test scenarios. However, when considering real-world scenarios and deployment to production systems, these models are still fragile and brittle to domain distribution shifting (Hendrycks et al., 2020a; Lin et al., 2022; Lee et al., 2021b), adversarial attacks (McCoy et al., 2019; Wallace et al., 2019), social bias (Zhao et al., 2021a; Sap et al., 2020; Sun et al., 2019), and inconsistent generation issues (Falke et al., 2019; Maynez et al., 2020; Kryscinski et al., 2020). The risk of failure predictions impedes the safe deployment of NLP models.

To better understand robustness issues, I constructed robustness benchmarks and empirically study model performance across multiple NLP tasks. I covered the problem of robustness from four perspectives: (1) *data-efficient*: models can adapt to new domains with limited annotated data in both pretrained-finetuned and trained-from-scratch set-ups; (2) *resilient*: models can perform reliably under uncertainties and challenging circumstances; (3) *fair*: predictors or generators can make safe decisions and filter undesirable biases especially those imbued with toxicity, hate, and social bias; (4) *trusted*: models are expected to yield factual and faithful content.

## 1.1 Contribution

The primary contribution of this thesis is to investigate NLP model robustness under classification and generation tasks, and propose solutions from dataset creation, machine learning modeling, and learning algorithm sides. Under the categorization of applications, my work includes:

**Data-Efficient and Resilient Text Classifiers.** We found the performance of widely-used *discriminative* classifiers drops dramatically under challenging scenarios when labeled data is scarce, noisy, or biased. We proposed **LatGen** (Ding and Gimpel, 2019), in which we introduced discrete latent variables into the generative story and parameterized the distributions using standard neural architectures used in conditional language modeling. In training, we performed learning by directly maximizing the log marginal likelihood via gradient-based optimization, which achieves comparable performance to expectation-maximization with less computing cost. We found LatGen to be more data-efficient compared to the discriminative baselines on six document classification benchmarks. With **GenNLI** (Ding et al., 2020) we applied generative classifiers from document classification with a single text input to sentence relation inference with two text inputs. Our training objective combines generative pretraining and discriminative finetuning, in which we explore various objectives for discriminative finetuning and find strong results with an unbounded modification to log loss. Our experiments show that GenNLI outperforms discriminative baselines including pretrained-finetuned and trained-from-scratch models across several challenging experimental settings.

**Controllable Text Generators.** Large-scale language models (LMs) trained on big data crawled from the web are capable of generating human-like text. Albeit powerful, there are undesirable features learned by LMs that lead to uncomfortable toxic, and biased content. To control generation, we explored frameworks that generate multiple, *diverse* outputs with desirable features via the learned *control variables*. **StoryGen** (Tu et al., 2019) explores a variation of a sequence-to-sequence model with attention, in which we defined an attribute embedding function that maps control attributes to vectors. We experimented with a broad range of control attributes including sentence length, sentiment, predicates, and frame semantics, and demonstrated that our framework can accurately generate story continuations that match the target attribute values. **LatGen** (Ding and Gimpel, 2019) and **GenNLI** (Ding et al., 2020) benefit from the generative frameworks that can be applied for controllable generation. The control attributes are the label and discrete latent

variable index. In **FlowPrior** (Ding and Gimpel, 2021) we explored continuous latent variable modeling of text and focus on variations that learn expressive prior distributions over the latent variable. This leads to models that learn a better sentence-level latent representation that helps generate diverse outputs.

**Hallucination Content Detector.** Motivated from the preliminary findings that top-ranked dialogue generations lack knowledge-consistency with grounded documents, we investigate various approaches for detecting hallucinated content. One key challenge is that there are few in-domain annotated data, while data synthesized from rule-based methods are less challenging because they rarely cover errors in real systems. We proposed an adversarial data augmentation framework that synthesizes data sharing similar types to those incorrectly predicted instances in the dev set.

To summarize, this thesis covers my works in:

1. Latent-variable generative models for data-efficient text classification
2. Discriminatively-tuned generative classifiers for robust Natural Language Inference
3. An empirical study on model-agnostic debiasing strategies for robust Natural Language Inference
4. Generating diverse story continuations with controllable semantics
5. FlowPrior: learning expressive priors for latent variable sentence models
6. Detecting generation inconsistency in grounded dialogue systems

## 1.2 Thesis Organization

Chapter 2 presents fundamental related work. Chapter 3 defines latent generative classifiers and discusses how they help with training efficiency. Chapter 4 presents an approach for discriminatively finetuning the generative classifier to gain better performance. Chapter 5 introduces a

framework for controllable diverse story generation and evaluates the controllability of the framework. Chapter 6 presents an improvement on variational autoencoder and empirically studies the performance of generation via latent variable sampling. Chapter 7 pivots to discuss a data collection pipeline and adversarial data augmentation methods for hallucination detection in the grounded dialogue domain. Finally, I discuss future work and concluding remarks in Chapters 8 and 9, respectively.

# CHAPTER 2

# BACKGROUND

## 2.1 Probablistic Generative Modeling

### *2.1.1 Definition*

Though definitions of generative modeling varies (Goodfellow et al., 2014; Kingma and Welling, 2014a; Devlin et al., 2019b; Liu et al., 2019; Goodfellow et al., 2016), the intuition of generative modeling is from a famous quote from Richard Feynman: "*What I cannot create, I do not understand.*" Connecting to recent advances of deep learning approach in generative models, one can view these as methods that compress the large amount of data into a smaller number of parameters, namely internalize the essence of the data in a fixed size of neural network. Generation is one way to measure the effectiveness of the learned neural network.

In this section we will view the generative models under the lens of probability. A generative model approximates data distribution by modeling a finite set of samples (i.e.: observations $\boldsymbol{x}$) from distribution $\boldsymbol{x} \sim p_{\text{data}}$. In this thesis we only consider parametric generative models.

### *2.1.2 Learning and Inference*

Learning in context of parametric generative models is to pick parameters $\theta$ within a family of model distributions $\mathcal{F}$ that minimizes the distance between probability distributions of $p_\theta$ and $p_{\text{data}}$.

$$\min_{\theta \in \mathcal{F}} \text{distance}(p_{\text{data}}, p_\theta) \tag{2.1}$$

In Section 2.1.3 we will briefly discuss four representative generative models and their learning objectives and optimization procedures. Fundamental inference of generative models include density estimation, sampling, and representation learning. In our thesis we focus on density estimation and sampling.

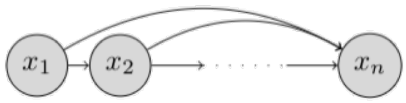



Figure 2.1: Graphical model of autoregressive model.

Besides applying generative models to unlabelled raw data, unsupervised generative models can be combined with discriminative classifiers. Namely one can perform unsupervised learning on unlabeled training data and supervised learning on labeled data, and jointly train the parameters of two. On the other hand, class-conditioned generative models can directly perform classification tasks if we had an approximate (empirical or hypothetical) label distribution. In contrast to discriminative classifiers that directly model $p(y \mid \boldsymbol{x})$, generative classifiers model the joint distribution of data and labels. We'll discuss generative modeling on discriminative tasks in Section 2.1.4.

### *2.1.3 Generative Modeling on Generation Tasks*

**Learning from Un-Annotated Data.** Given an observation $\boldsymbol{x} = \{x_1, x_2, \cdot, x_n\}$ where $x_i$ denotes random variable, generative model models the joint distribution of these $n$ random variables. Representative generative models include *autoregressive models*, *latent variable models*, *flow-based models*, *generative adversarial networks* (GAN), and *energy-based models* (EBM). Since GAN is not referenced throughout the thesis, we will only discuss the other four models in this section.

**Autoregressive Models.** Figure 2.1 shows graphical model of autoregressive model. By applying the chain rule:

$$p(\boldsymbol{x}) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdot p(x_n \mid x_1, \cdot, x_{n_1}) \tag{2.2}$$

Autoregressive models includes fully visible Sigmoid Belief Network (FVSBN) (Gan et al., 2015), Neural Autoregressive Density Estimation (NADE) (Larochelle and Murray, 2011), Masked

Autoencoder for Distribution Estimation (MADE) (Germain et al., 2015). One of the most commonly used autoregressive models is Recurrent Neural Nets (RNN) (Hochreiter, 1998), which keeps the history in order to process long sequences. More recent works include Character RNN (Kim et al., 2016), pixel RNN (van den Oord et al., 2016), WaveNet (Oord et al., 2016).

Training an autoregressive model is to optimize the *maximum likelihood estimation* (MLE) objective via stochastic gradient descent. Inference and sampling follows the sequential procedure.

**Latent Variable Models.** There are lots of variability under the observation $x$. Unless these factors are annotated with input, they can be treated as as latent variables and modeled explicitly.

Latent variables could be discrete (our work in Section 3.3) or continuous (our work in Chapter 6). Training is to encode high-level features into the latent variables $\boldsymbol{z}$. The intuition is that if $\boldsymbol{z}$ is able to capture the key factors, modeling $p(\boldsymbol{x}|\boldsymbol{z})$ can be much easier than modeling $p(\boldsymbol{x})$ directly. The joint distribution $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$, and the generative story follows $\boldsymbol{z} \sim p(\boldsymbol{z})$, $\boldsymbol{x} \sim p(\boldsymbol{x} \mid \boldsymbol{z})$.

*Mixture of Gaussian* (MOG) is a discrete latent variable model:

$$\begin{aligned} \boldsymbol{z} &\sim \text{Categorical}(1, \cdots, K) \\ p(\boldsymbol{x} \mid \boldsymbol{z} = k) &= \mathcal{N}(\mu_k, \text{diag}(\sigma_k^2)) \end{aligned} \tag{2.3}$$

where $K$ is the number of clusters (i.e., mixture components) and the generative story is sampling a data point from the selected Gaussian component.

*Variational Autoencoders* (VAE) is a widely used deep latent variable model with a mixture of an infinite number of Gaussians:

$$\begin{aligned} \boldsymbol{z} &\sim \mathcal{N}(0, \mathbf{1}) \\ p_\theta(\boldsymbol{x} \mid \boldsymbol{z}) &= \mathcal{N}(\mu_\theta(\boldsymbol{z}), \text{diag}(\sigma_\theta^2(\boldsymbol{z}))) \end{aligned} \tag{2.4}$$

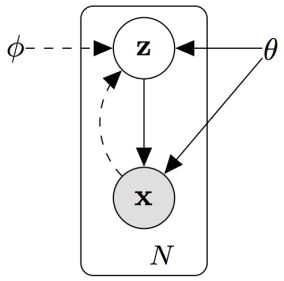



Figure 2.2: Graphical model of VAE.

where $\mu_\theta$ and $\sigma^2_\theta(z)$ are parameterized via neural networks.

Learning a latent variable is to maximize the marginal loglikelihood $\log p(\boldsymbol{x}; \theta)$ which requires to sum or integral over latent variables. This is tractable when we have a smaller set of discrete latent variables. However, for continuous variables, $\log \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}; \theta)\, dx$ is generally intractable.

Figure 2.2 is the graphical model of VAE. VAEs introduce an approximate posterior $q_\phi(z \mid x)$ parameterized using a neural network (i.e., an "inference network"), and maximize the *evidence lower bound* (ELBO) instead of log marginal likelihood:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x) \,||\, \mathcal{N}(0, \mathbf{1})) \tag{2.5}$$

**Flow-based Models.** In standard latent variable model, we generally assume simple prior distribution of latent variables (e.g., $p(\boldsymbol{z}) \sim \mathcal{N}(0, \mathbf{1})$) which are easy to sample from. This motivates the use of normalizing flow, in which one can map a simple distribution to a complex one (multi-modal distribution) with *change of variables*.

In change of variables formula, we start with a base density $p_\epsilon(\epsilon)$, and define a *bijection* function $f : \epsilon \in \mathbb{R}^n \to Z \in \mathbb{R}^n$, that maps from $\epsilon \sim p_\epsilon$ to the target $z \sim p_Z$. Note that $\epsilon$ and $z$ need to be continuous and have the same dimension, and $Z = f(\epsilon)$ and $\epsilon = f^{-1}(Z)$. According to the

*change of variable* formula:

$$\log p_Z(z) = \log p_\epsilon(f^{-1}(z)) + \log \left| \det \left( \frac{\partial f^{-1}(z)}{\partial z} \right) \right| \tag{2.6}$$

where $\frac{\partial f^{-1}(z)}{\partial z}$ is the Jacobian of $f$ at $z$.

A normalizing flow (NF) is a sequence of *invertible*, *deterministic* transformations:

$$\begin{aligned} \boldsymbol{z}_L &= f_L \circ f_{L-1} \circ ... \circ f_1(\boldsymbol{z}_0) \\ \boldsymbol{z}_0 &= f_1^{-1} \circ ... \circ f_{L-1}^{-1} \circ f_L^{-1}(\boldsymbol{z}_L) \end{aligned} \tag{2.7}$$

where $\boldsymbol{z}_i$ is random variable, $f_1, f_2, ..., f_L$ are all bijective functions. By repeatedly applying the rule for change of variables, the base density is transformed into a more complex one. Using the change-of-variables theorem, given a latent variable $z_L$, we can compute the exact density under the prior with the "image" $\boldsymbol{z}_0$ acquired by inverting the transformation:

$$\log p(\boldsymbol{z}_L) = \log p_{\boldsymbol{z}_0}(\boldsymbol{z}_0) - \sum_{l=1}^{L} \log(|\det(\frac{\partial f_l(\boldsymbol{z}_{l-1})}{\partial \boldsymbol{z}_{l-1}})|) \tag{2.8}$$

Networks parameterized using NF can be trained through exact maximum log-likelihood computation. Exact sampling is performed by drawing sample from the base distribution and performing the chain of transformations.

Computing the Jacobian of functions with high dimension and the determinants of large matrices (i.e., the two main computation in NF) are very expensive. Prior work has addressed this challenge by introducing efficient transformations (Dinh et al., 2015, 2016; Germain et al., 2015; Kingma et al., 2016; Kingma and Dhariwal, 2018; Ho et al., 2019). Our work of flow-based prior (Section 6) is based on *real-valued non-volume preserving* (real NVP; Dinh et al., 2016) which is efficient in both training and sampling. The main building block of real NVP transformation is the *affine coupling layer*.

An affine coupling layer is a bijective transformation $f_i : z_{i-1} \to z_i$ that follows the equations:

$$
\begin{aligned}
z_i^{(1:d)} &= z_{i-1}^{(1:d)} \\
z_i^{(d+1:D)} &= z_i^{(d+1:D)} \odot \exp(s(z_{i-1}^{(1:d)})) + t(z_{i-1}^{(1:d)})
\end{aligned}
$$

where $D$ is the dimensionality, $z_i^{(1:d)}$ stands for the first $d$ dimensions of $z_i$ ($d < D$); $s$ and $t$ denote the functions for *scale* and *translation* operations that map from $\mathbb{R}^d \to \mathbb{R}^{D-d}$; and $\odot$ denotes element-wise product.

The Jacobian determinant and inverse of the affine coupling layer are easy to compute. The transformation is flexible because its computation of the Jacobian determinant and inverse do not require any operations with the functions $s$ and $t$, so these two functions could be designed to be arbitrarily complex.

**Energy-based Models.** Probability distribution is the fundamental in generative modeling, in which requires $p(\boldsymbol{x})$ to be both non-negative and normalized. Normalization requires integral over the space of all possible variable configurations, which is intractable in most of cases. This motivates the EBM framework. EBM provides a theoretical framework that generalize across many probabilistic and non-probabilistic models (LeCun et al., 2006). Each configuration of random variables $p(\boldsymbol{x})$ is associated with a scalar called *energy* $E_\theta(\boldsymbol{x})$ that captures the dependencies between variables. $x$ with lower energy (high -$E_\theta(\boldsymbol{x})$) are more likely being observed under the data distribution. The EBM is defined as (LeCun et al., 2006):

$$
p_\theta(x) = \frac{\exp\left(-E_\theta(x)\right)}{Z(\theta)} \tag{2.9}
$$

where $Z$ is the normalization constant. $Z(\theta) = \int \exp(-E(\boldsymbol{x}))\, d\boldsymbol{x}$ is also called partition function.

While EBM benefits from flexibility in the form of energy function, it is difficult to learn and inference due to the intractable normalization constant. For applications that do not require

computing $Z(\theta)$ and only need relative comparisons of the quantity of energies, EBM provides a simple solution. For example, given $x_1$ and $x_2$ we can compare them with

$$p(\boldsymbol{x}_1)/p(\boldsymbol{x}_2) = \exp(-E_\theta(\boldsymbol{x}_1) + E_\theta(\boldsymbol{x}_2)) \tag{2.10}$$

in which the normalization constant is canceled off.

### *2.1.4 Generative Modeling on Classification Tasks*

**Classification Tasks.** In classification tasks, the computer program is asked to specify which of k categories the input belong to. Modeling and learning is to produce a function $f : \mathbb{R} \to \{1, \cdots, k\}$. Given $y = f(\boldsymbol{x})$, the model assigns an input vector $\boldsymbol{x}$ to a category identified by numeric code $y$ (Goodfellow et al., 2016). Another method to formalize the classifier is to define a classification framework $\text{score}(\boldsymbol{x}, y, \theta)$, namely we assign a score to each label $y$ for input $\boldsymbol{x}$, parameterized by parameters $\theta$ (Gimpel, 2018). Learning is to choose parameters $\theta$ that maximize the score of input and gold standard label. At inference time, one search over the space of all labels and predict by $\text{argmax}_y \, \text{score}(\boldsymbol{x}, y, \theta)$.

**Discriminative Classifier.** Discriminative classifier models conditional probability of input and label:

$$\text{score}(\boldsymbol{x}, y, \theta) = p_\theta(y \mid \boldsymbol{x}) \tag{2.11}$$

Training is to maximize the conditional probability of labels given input. Standard discriminative classifiers include Logistic Regression, Multilayer Perceptron (MLP), Conditional Random Field (CRF), Maximum Entropy Markov Model (MEMM), and Sequence-to-sequence Model (Seq2seq).

**Generative Classifier.** Generative classifier models the joint probability of input and label, is often decomposed into:

$$\text{score}(\boldsymbol{x}, y, \theta, \psi) = p_{\theta,\psi}(\boldsymbol{x}, y) = p_\theta(\boldsymbol{x} \mid y) p_\psi(y) \tag{2.12}$$

Training is to maximize the joint probability of input and label. Standard generative classifiers include Naive Bayes, Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), Probabilistic Context-Free Grammars (PCFG), and IBM translation models.

**Discriminative Trained Generative Classifier.** Besides directly maximizing joint probability of input and label, generative classifiers can be trained discriminatively. According to Bayes rule,

$$p_{\theta,\psi}(y \mid \boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x} \mid y) p_\psi(y)}{\sum_{y'} p_\theta(\boldsymbol{x} \mid y') p_\psi(y')} \tag{2.13}$$

Thus, one can optimize parameters of the generative classifier by maximizing the discriminative objective.

**Generative Model in Discriminative Classifier.** Grathwohl et al. (2020) reinterpret the discriminative classifier as a Joint Energy-based Model and use the extra degree of freedom hidden within the logits to define the density function over input $p_\theta(\boldsymbol{x})$, as well as the joint density of input and label $p_\theta(\boldsymbol{x}, y)$.

Here we consider *energy* and *energy function*, a specific type of score and score function defined in EBM (LeCun et al., 2006). In contrast to prior score function that produces a normalized probability for an input and label pair, here the energy function produces an unnormalized scalar, i.e. the energy $E_\theta(\boldsymbol{x}, y)$. With discriminative classifier $g$, energy is interpreted as negative logit. Logits are used to parameterize a categorical distribution using softmax in discirminative classifier:

$$p_\theta(y \mid \boldsymbol{x}) = \frac{\exp(g_\theta(\boldsymbol{x})[y])}{\sum_{y'} \exp(g_\theta(\boldsymbol{x})[y'])} \tag{2.14}$$

$$E_\theta(\boldsymbol{x}, y) = -g_\theta(\boldsymbol{x})[y] \tag{2.15}$$

By definition of EBM,

$$p_\theta(x, y) = \frac{\exp\left(g_\theta(\boldsymbol{x})[y]\right)}{Z} \tag{2.16}$$

Marginalizing over $y$,

$$p_\theta(x) = \frac{\sum_y \exp\left(g_\theta(\boldsymbol{x})[y]\right)}{Z} \tag{2.17}$$

Now we can define the energy of $\boldsymbol{x}$ as

$$E_\theta(x) = -\operatorname{LogSumExp}(g_\theta(\boldsymbol{x})[y]) \tag{2.18}$$

Thus we derived a generative model in discrimintive classifier. Prior works train this joint EBM via noise contrastive estimation (NCE) (He et al., 2021) or stochastic gradient Langevin dynamics (SGLD) (Grathwohl et al., 2020).

### *2.1.5 Generative Classifiers*

While discriminative classifiers directly model the posterior probability of the label given the input, i.e., $p(y \mid x)$, generative classifiers instead model the joint probability $p(x, y)$, typically factoring it into $p(x \mid y)$ and $p(y)$ and making decisions as follows:

$$\hat{y} = \operatorname*{argmax}_y p(x \mid y)p(y)$$

Most neural network classifiers are trained as discriminative classifiers as these work better when conditions are favorable for supervised learning, namely that training data is plentiful and that the training and test data are drawn from the same distribution. While discriminative classifiers are generally preferred in practice, there is certain prior work showing that generative classifiers can have advantages in certain conditions, especially when training data is scarce, noisy, and imbal-

anced (Yogatama et al., 2017; Lewis and Fan, 2019; Ding and Gimpel, 2019).

Ng and Jordan (2002) proved theoretically that generative classifiers can approach their asymptotic error much faster, as naïve Bayes is faster than its discriminative analogue, logistic regression. Yogatama et al. (2017) compared the performance of generative and discriminative classifiers and showed the advantages of neural generative classifiers in terms of sample complexity, data shift, and zero-shot and continual learning settings. Ding and Gimpel (2019) further improved the performance of generative classifiers on document classification by introducing discrete latent variables into the generative story. Lewis and Fan (2019) developed generative classifiers for question answering and achieved comparable performance to discriminative models on the SQuAD (Rajpurkar et al., 2016) dataset, and much better performance in challenging experimental settings.

In this paper, we develop generative models for natural language inference inspired by models for sequence-to-sequence tasks. We additionally contribute an exploration of several discriminative objectives for fine-tuning our generative classifiers, finding multiple choices to outperform log loss used in prior work. We also compare our generative classifiers with fine-tuning of large-scale pretrained models, and characterize performance under other realistic settings such as imbalanced and noisy datasets.

## 2.2 Model Robustness

### *2.2.1 Robustness toward Distribution Shift*

Model robustness towards distribution shift is to ensure model can be generalized to a new domain or new data distribution that is not available during training, i.e., the models' performance should not degrade when test data is different from training data. It is connected to work on transfer learning, domain adaptation, and domain generalization (Pan and Yang, 2009; Muandet et al., 2013; Patel et al., 2015; Wilson and Cook, 2020; Hendrycks et al., 2020b; Gulrajani and Lopez-Paz, 2020). The change of distribution includes the change of data domain, and the change of

users that leads to differences in dialects, languages, word use, and grammar (Blodgett et al., 2016; Miller et al., 2020; Demszky et al., 2021).

### *2.2.2 Robustness towards Adversarial Attacks*

Adversarial examples are generally crafted examples that are meaningful and imperceptible to humans but challenging to models to make the correct predictions. Adversarial examples reveal areas of the input space where the model performs badly, allowing for better understanding and improvement. Adversarial training develops more robust models that may even perform better on non-adversarial cases by utilizing these examples as training data. Recent efforts in adversarial attacks include Ebrahimi et al. (2018b); Alzantot et al. (2018); Elliott (2018); Carton et al. (2018); Ebrahimi et al. (2018a); Wang et al. (2018b).

### *2.2.3 Robustness towards Biases*

There are two main biases studied in the field of NLP robustness, one is robustness towards spurious biases, and the other is the robustness towards undesirable gender biases.

In the first thread, prior works found models utilize spurious correlations between inputs and the labels to make the prediction. However, one can not build causal relation between the spurious features and the tasks' labels. The spurious correlations includes dataset biases from the data collection process, annotation artifacts (Gu et al., 2019; Amiri et al., 2018; Durmus et al., 2022; Wu et al., 2022; Han and Tsvetkov, 2021; Yaghoobzadeh et al., 2021; Tu et al., 2020).

The second thread is often connected to works in filtering unwanted gender stereotypes. These biased systems carry the societal stereotypes present in training data, and have the potential to not only mimic but also amplify stereotypes in society. To mitigate gender biases, recent efforts include Rudinger et al. (2018); Zhao et al. (2018); Park et al. (2018); Zhao et al. (2017a); Bordia and Bowman (2019); Alfaro et al. (2019) .

# CHAPTER 3

# LATENT-VARIABLE GENERATIVE MODELS FOR DATA-EFFICIENT TEXT CLASSIFICATION

## 3.1 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., 1993a; 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 work, 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.

## 3.2 Discriminative and Generative Baselines

Our generative baselines, i.e. generative text classifiers, are essentially the same as those from Yogatama et al. (2017). 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 that uses a separate LSTM for each label. We

instead use a shared LSTM for all labels which has similar performance to the seperate ones.

Our classifiers are trained on datasets $D$ of annotated documents. Each instance $\langle \boldsymbol{x}, y \rangle \in D$ consists of a textual input $\boldsymbol{x} = \{x_1, x_2, ..., x_T\}$, where $T$ is the length of the document, and a label $y \in \mathcal{Y}$.

**Discriminative Baselines.** we encode a document $x$ using an LSTM (Hochreiter and Schmidhuber, 1997a), 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.

**Generative Baselines.** We parameterize $\log p(\boldsymbol{x} \mid y)$ as a conditional LSTM language model using the standard sequential factorization:

$$\log p(\boldsymbol{x} \mid y) = \sum_{t=1}^{T} \log p(x_t \mid \boldsymbol{x}_{<t}, y) \tag{3.1}$$

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. The label prior $p(y)$ is acquired via maximum likelihood estimation and fixed during training of the remaining parameters.

**Inference.** Prediction is made by maximizing $p(y \mid \boldsymbol{x})$ with respect to $y$ for the discriminative classifier and maximizing $p(\boldsymbol{x} \mid y)p(y)$ for the generative classifier.

## 3.3 Latent-Variable Generative Classifiers

We now introduce discrete latent variables into the standard generative classifier as shown in Figure 3.1. 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.

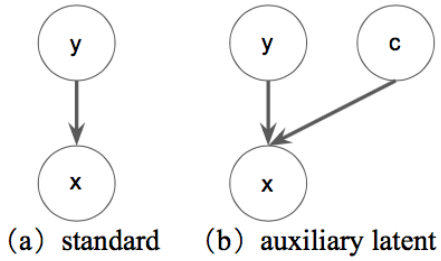



Figure 3.1: Graphical models of (a) standard generative classifier and (b) auxiliary latent generative classifier.

Following the graphical model structure in Figure 3.1(b), we factorize the joint probability $p(\boldsymbol{x}, y, c)$ as follows:

$$p(\boldsymbol{x}, y, c) = p_\Theta(\boldsymbol{x} \mid c, y) p_\Phi(c) p_\Psi(y) \tag{3.2}$$

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

$$\log p_\Theta(\boldsymbol{x} \mid c, y) = \sum_{t=1}^{T} \log p_\Theta(x_t \mid \boldsymbol{x}_{<t}, c, y) \tag{3.3}$$

where $\Theta$ 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:

$$p_\Theta(x_t \mid x_{<t}, c, y) \propto \exp\{u_{x_t}^\top([h_t; v_y; v_c]) + b_{x_t}\} \tag{3.4}$$

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:

$$p_\Phi(c) \propto \exp\{w_c^\top v_c + b_c\} \tag{3.5}$$

where $\Phi$ 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:

$$\max_{\Theta,\Phi,V_\mathcal{C},V_\mathcal{Y}} \sum_{\langle x,y\rangle\in\mathcal{D}} \log \sum_{c\in\mathcal{C}} p_\Theta(x \mid c, y) p_\Phi(c) p_\Psi(y) \tag{3.6}$$

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:

$$\hat{y} = \operatorname*{argmax}_{y\in\mathcal{Y}} \sum_{c\in\mathcal{C}} p_\Theta(x \mid c, y) p_\Phi(c) p_\Psi(y) \tag{3.7}$$

**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,

| Dataset | #Train | #Dev | #Test | #Labels |
|---|---|---|---|---|
| Yelp Polarity | 555k | 5k | 7.6k | 2 |
| Yelp Full | 645k | 5k | 50k | 5 |
| AGNews | 115k | 5k | 7.6k | 4 |
| Sogou | 445k | 5k | 60k | 5 |
| Yahoo | 1395k | 5k | 60k | 10 |
| DBpedia | 555k | 5k | 70k | 14 |

Table 3.1: Text classification datasets used in our experiments.

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.

## 3.4 Experiments

**Dataset.** 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. Dataset names and statistics are shown in Table 3.1.

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.

**Training Details.** In all experiments, the word embedding dimension and the LSTM hidden state dimension are set to 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

| models | Yelp P | Yelp F | AGNews | Sogou | Yahoo | DBpedia |
|---|---|---|---|---|---|---|
| generative classifier (ours, shared-LSTM) | 92.61 | 57.36 | 89.88 | 89.57 | 68.87 | 96.46 |
| discriminative classifier (ours) | 96.37 | 65.81 | 90.09 | 96.43 | 73.10 | 98.78 |
| gen LSTM-shared (Yogatama et al., 2017) | 88.20 | 52.70 | 90.60 | 90.30 | 69.30 | 95.40 |
| gen LSTM-independent (Yogatama et al., 2017) | 90.00 | 51.90 | 90.70 | 93.50 | 70.50 | 94.80 |
| disc model (Yogatama et al., 2017) | 92.60 | 59.60 | 92.10 | 94.90 | 73.70 | 98.70 |
| bag of words (Zhang et al., 2015) | 92.20 | 58.00 | 88.80 | 92.90 | 68.90 | 96.60 |
| fastText (Joulin et al., 2017) | 95.70 | 63.90 | 92.50 | 96.80 | 72.30 | 98.60 |
| char-CRNN (Xiao and Cho, 2016) | 94.50 | 61.80 | 91.40 | 95.20 | 71.70 | 98.60 |
| very deep CNN (Conneau et al., 2017b) | 95.70 | 64.70 | 91.30 | 96.80 | 73.40 | 98.70 |

Table 3.2: Summary of the results on the full datasets. Our implementation of the generative model share parameters among classes.

choose 10 or 30 latent variable values with embeddings of dimensionality 10, 50, or 100.

For optimization, we use Adam (Kingma and Ba, 2015a) 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.[1]

**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).

| | 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 3.3: **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.

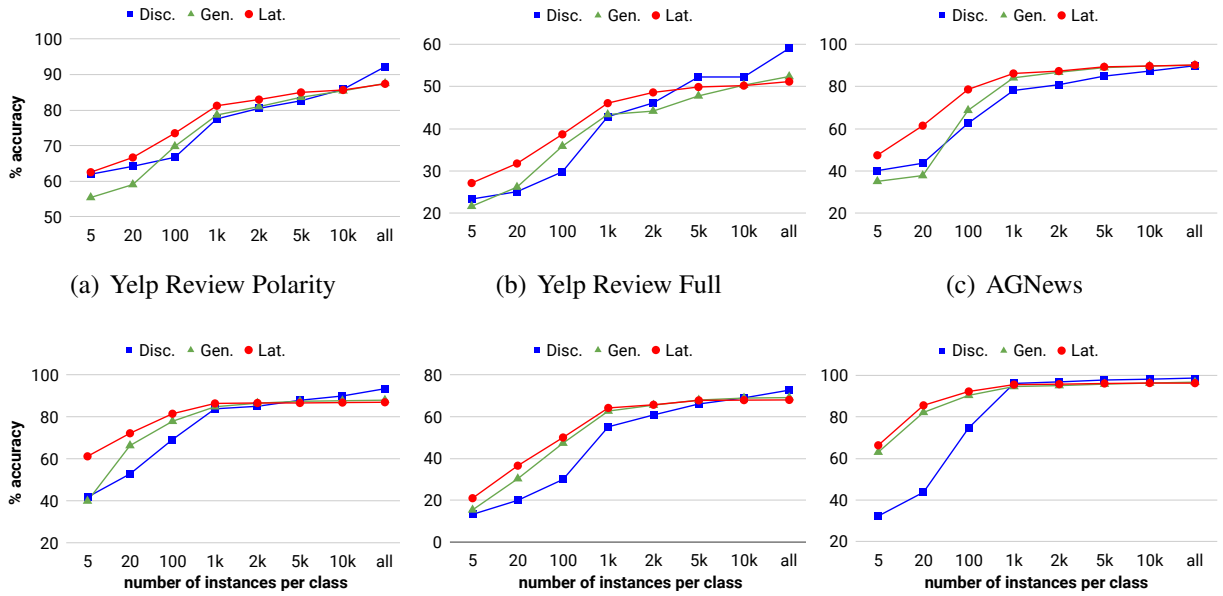
## 3.5 Results

### *3.5.1 Comparison of Baselines*

To indicate we have built a strong baseline, we first compare our implementation of the generative and discriminative classifiers to prior work. Note that here we use the whole training set without any truncation on the sequence length.

Table 3.2 shows that our standard generative and discriminative LSTM models are comparable with Yogatama et al. (2017). All other well-performing models listed in the table are discriminatve models that use more complex modeling methods such as attention to boost performance. Since the focus of our paper is the impact of adding latent variables to generative models, we do not use more complex techniques when building our baselines.

Our results show that our standard generative and discriminative LSTM models are comparable with Yogatama et al. (2017). We also see that the generative models have lower classification accuracies with the full training set, which agrees with the findings in the prior work.




(a) Yelp Review Polarity (b) Yelp Review Full (c) AGNews

(d) Sogou (e) Yahoo (f) DBpedia

Figure 3.2: Comparison of classification accuracy of the discriminative (**Disc.**), standard generative (**Gen.**), and latent generative (**Lat.**) classifiers training across training set sizes.

### *3.5.2 Data Efficiency*

Figure 3.2 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 3.3.

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

1. 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 | 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 3.4: 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.

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

### *3.5.3 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.[2]

For the latent generative classifier, we choose 10 latent variable values with embeddings of dimensionality 10, and a label dimensionality of 100 (**Lat. PC** in Table 3.4). For the standard generative classifier, we choose a label dimensionality of 110 (**Gen. PC** in Table 3.4). So, the numbers of parameters are comparable, since we ensure the same number of parameters in the

2. 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 3.4. The only tuned hyperparameters are the number of latent variable values and the dimensions of their embeddings.

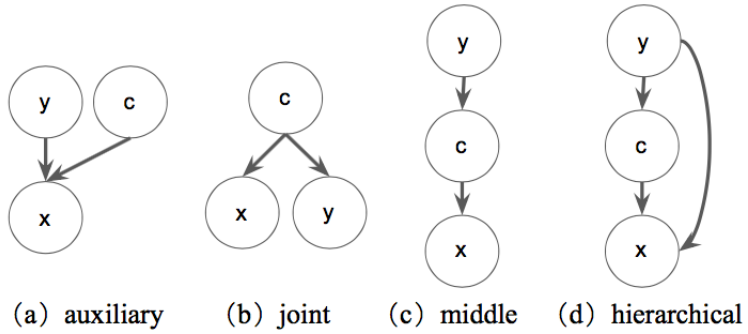



Figure 3.3: Graphical models of (a) auxiliary, (b) joint, (c) middle, and (d) hierarchical latent generative classifiers.

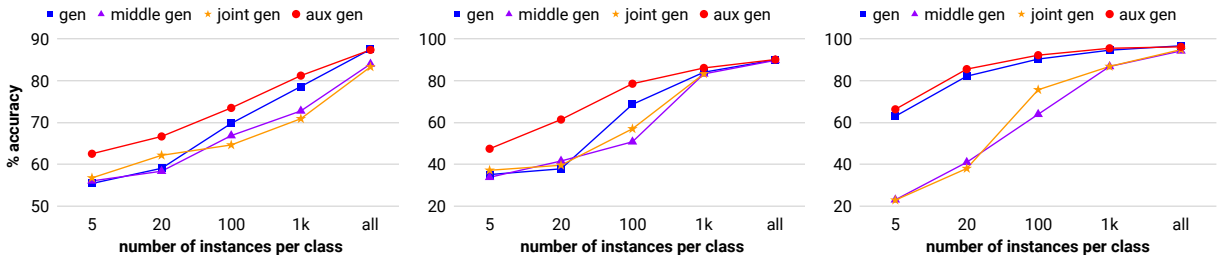
"output" word embeddings in the softmax layer of the language model, which is the decision that most strongly affects the number of parameters.

Table 3.4 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.

### *3.5.4 Effect of Latent Variables*

There are multiple choices to factorize the joint probability of the variables $\boldsymbol{x}$, $y$, and $c$, which correspond to different graphical models. Here we consider other graphical model structures, namely those shown in Figure 3.3. We refer to the model in Figure 3.3(b) as the "joint" latent generative classifier since it uses the latent variable to jointly generate $\boldsymbol{x}$ and $y$. We refer to the model in Figure 3.3(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 $\boldsymbol{x}$ where the embedding of the latent variable is concatenated to the hidden state as in Section 3.3.

Figure 3.4 shows the comparison of the standard and the three latent generative classifiers on




(a) Yelp Review Polarity (b) AGNews (c) DBpedia

Figure 3.4: Comparison of generative classifier (**gen**) and latent generative classifiers (**middle gen, joint gen, aux gen**).

| | **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 3.5: 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.

Yelp Review Polarity, AGNews, and DBpedia.[3] 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.

3. Similar trends are observed for all datasets, so we only show three for brevity.

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 3.3(d). The results in Table 3.5 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 $c$.

### *3.5.5 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).

EM provides a general purpose local search algorithm for learning parameters in probabilistic models with latent variables, and it has been widely used in much prior work and has shown its efficacy in terms of convergence (Ruder et al., 2018; Neal and Hinton, 1998; Dempster et al., 1977).

Salakhutdinov et al. (2003) theoretically study the close relationship between EM and direct optimization approaches with gradient-based methods. Here we empirically characterize the performance of our auxiliary latent generative classifiers with different training methods, namely EM and stochastic gradient descent (SGD) (Bottou, 2010) for direct optimization.

For our latent generative classifiers, the Expectation (E) step computes the posterior distributions over the latent variable:

$$\hat{p}(c \mid x, y) \leftarrow \frac{p(x, y, c)}{\sum_{c' \in \mathcal{C}} p(x, y, c')}$$

| | 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 3.6: 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.

The Maximization (M) step seeks to find new parameter estimates by maximizing the following:

$$\sum_{\langle x,y\rangle \in \mathcal{D}} \sum_{c \in \mathcal{C}} \hat{p}(c \mid x, y) \log p(x, y, c)$$

To speed convergence, we use a mini-batch version of EM, updating the parameters after each mini-batch. Our results in Table 3.6 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.

### *3.5.6 Analysis*

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 3.7.

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.

## 3.6 Analysis

### *3.6.1 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-

| id | description | examples |
|---|---|---|
| 1 | future/progressive tenses | ... Commission **is likely to** follow opinion in the U.S. on the merger suit ...<br>... 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 .<br>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 ...<br>**ABUJA** ... its militias **in Darfur** before they would sign ... |
| 4 | mixture | |
| 5 | abbreviations | Louis advanced to the **N.L.** championship series for the third time in five years ...<br>**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 ....<br>... 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** .<br>... 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 ...<br>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 / finance-QuoteLookup.jhtml** ...<br>**& lt ;** strong **& gt ;** Analysis **& lt ; /** strong **& gt ;** Contracting out the blame ... |

Table 3.7: 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.

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 3.7.

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.

### *3.6.2 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)

| | |
|---|---|
| **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 3.8: 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.

from AGNews, so the generated samples do suffer from the small size of data used in training the language model. Table 3.8 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 set-

ting 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.

## 3.7 Conclusion

In this work we focused 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.

# CHAPTER 4

# DISCRIMINATIVELY-TUNED GENERATIVE CLASSIFIERS FOR ROBUST NATURAL LANGUAGE INFERENCE

## 4.1 Introduction

Natural language inference (NLI) is the task of identifying the relationship between two fragments of text, called the *premise* and the *hypothesis* (Dagan et al., 2005a; Dagan et al., 2013). The task was originally defined as binary classification, in which the labels are *entailment* (the premise implies the hypothesis) or *not entailment*. Subsequent variations added a third *contradiction* label. Most models for NLI are trained and evaluated on standard benchmarks (Bowman et al., 2015a; Williams et al., 2018; Wang et al., 2018a) in a discriminative manner (Conneau et al., 2017a; Chen et al., 2017a). These benchmarks typically have relatively clean, balanced, and abundant annotated data, and there is no distribution shift between the training and test sets.

However, when data quality and conditions are not ideal, there is a substantial performance decrease for existing discriminative models, including both simple model architectures and more complex ones. Prior work on document classification and question answering has shown that **generative classifiers** have advantages over their discriminative counterparts in non-ideal conditions (Yogatama et al., 2017; Lewis and Fan, 2019; Ding and Gimpel, 2019).

In this paper, we develop generative classifiers for NLI. Our model, which we call GenNLI, defines the conditional probability of the hypothesis given the premise and the label, parameterizing the distribution using a sequence-to-sequence model with attention (Luong et al., 2015a) and a copy mechanism (Gu et al., 2016). We explore training objectives for discriminative fine-tuning of our generative classifiers, comparing several classical discriminative criteria. We find that several losses, including hinge loss and softmax-margin, outperform log loss fine-tuning used in prior work (Lewis and Fan, 2019) while similarly retaining the advantages of generative classifiers. We also find strong results with a simple unbounded modification to log loss, which we call the "in-

finilog loss".

Our evaluation focuses on challenging experimental conditions: small training sets, imbalanced label distributions, and label noise. We empirically compare GenNLI with several discriminative baselines and large-scale pretrained language representation models (Devlin et al., 2019a; Yang et al., 2019; Liu et al., 2019) on five standard datasets. GenNLI has better performance than discriminative classifiers under the small data setting. Moreover, when limited to 100 instances per class, GenNLI consistently outperforms all BERT-style pretrained models on four of the five datasets. These results are appealing especially in comparison with BERT-style pretrained baselines. Large-scale pretrained language models have achieved state-of-the-art results on a wide range of NLP tasks, but they still require hundreds or even thousands of annotated examples to outperform GenNLI.

GenNLI also outperforms discriminative classifiers when the training data shows severe label imbalance and when training labels are randomly corrupted. We additionally use GenNLI to generate hypotheses for given premises and labels. While the generations tend to have low diversity due to high lexical overlap with the premise, they are generally fluent and comport with the given labels, even in the small data setting.

## 4.2 A Generative Classifier for NLI

Each example in a natural language inference dataset consists of two natural language texts, known as the premise and the hypothesis, and a label indicating the relation between the two texts. Formally, we denote an instance $\langle x^{(p)}, x^{(h)}, y \rangle$ as a tuple consisting of a premise $x^{(p)} = \{x_1^{(p)}, x_2^{(p)}, ..., x_N^{(p)}\}$, a hypothesis $x^{(h)} = \{x_1^{(h)}, x_2^{(h)}, ..., x_T^{(h)}\}$, and a label $y \in Y$.

Most existing NLI models are trained in a discriminative manner by maximizing the conditional log-likelihood of the label given the input, i.e., $\log p(y \mid x^{(p)}, x^{(h)})$. In this paper, we propose generative classifiers for NLI that are trained instead to estimate the probability of the hypothesis given the premise and the label, i.e., $p(x^{(h)} \mid x^{(p)}, y)$, typically by maximizing log-likelihood. We

decompose this conditional probability using the chain rule, and our final training objective is to minimize the following negative log likelihood:

$$L(x^{(p)}, x^{(h)}, y) = -\sum_{t=1}^{T} \log p(x_t^{(h)} \mid x_{<t}^{(h)}, x^{(p)}, y) \tag{4.1}$$

At inference time, the prediction is made as follows:

$$\underset{y \in Y}{\operatorname{argmax}} \log p(y) + \sum_{t=1}^{T} \log p(x_t^{(h)} \mid x_{<t}^{(h)}, x^{(p)}, y) \tag{4.2}$$

Throughout all of the experiments in this paper, we assume a uniform label prior $p(y)$, so $p(y)$ will not affect the $\operatorname{argmax}$ in Eq. (4.2) and can be omitted.

### *4.2.1 Parameterization*

Our model, which we refer to as GenNLI, is parameterized with a standard RNN-based sequence-to-sequence architecture with attention and a copy mechanism between the encoder and the decoder.[1]

**Encoder.** Our encoder uses a standard bidirectional recurrent neural network (RNN) using long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997a):

$$\mathbf{s}_n = [f_{e_1}(\mathbf{v}_n, \overrightarrow{\mathbf{s}_{n-1}}); f_{e_2}(\mathbf{v}_n, \overleftarrow{\mathbf{s}_{n+1}})]$$

where $f_{e_1}$ and $f_{e_2}$ are forward and backward LSTM recurrences, respectively, $\mathbf{v}_n$ is the word embedding of $x_n^{(p)}$, and $\mathbf{s}_n$ is the concatenation of the forward and backward RNN hidden states at position $n$ in the premise.

1. We also experimented with transformer architectures (Vaswani et al., 2017a) and found similar results.

**Decoder.** Our decoder uses an RNN with dot product attention from Luong et al. (2015a) and a copy mechanism Gu et al. (2016). The decoder hidden state at step $t$ is computed as $\mathbf{h}_t = f_d(\mathbf{w}_t, \mathbf{h}_{t-1})$.

where $f_d$ is the forward LSTM recurrence in the decoder and $\mathbf{w}_t$ is the word embedding of $x_t^{(h)}$. The word distribution at position $t+1$ is computed as follows:

$$\mathbf{p}_{vocab} = \mathrm{softmax}(\mathbf{V}'(\mathbf{V}[\mathbf{h}_t, \mathbf{s}_t^*, \mathbf{v}_y] + b) + b')$$

where $\mathbf{v}_y$ is the label embedding of $y$, $\mathbf{s}_t^*$ is the context vector at step $t$ computed using attention (full details of the attention mechanism are omitted for brevity but can be found in Luong et al., 2015a), and $\mathbf{V}$, $\mathbf{V}'$, $b$, and $b'$ are learnable parameters. Note the presence of the label embedding $\mathbf{v}_y$ concatenated to $\mathbf{h}_t$ and $\mathbf{s}_t^*$ to form the input to the $\mathrm{softmax}$ layer. This enables the label to directly influence the word distribution. We also use label-specific beginning-of-sentence (BOS) tokens as the initial symbol fed to the decoder RNN. Concretely, we create the embeddings for all BOS symbols $BOS_y$ ($y \in Y$) and prepend $BOS_{y'}$ to the hypothesis where $y'$ is the label for the instance.

**Copy mechanism.** In some datasets, hypotheses are written by humans when provided a premise and label (Bowman et al., 2015a). We observed that these hypotheses sometimes appear to be written by slightly modifying the premise according to the label, e.g., adding "not" to negate the premise, or by replacing a phrase with a phrasal hypernym, such as replacing "soccer game" with "sport" (Marelli et al., 2014; Bowman et al., 2015a). The tokens in a premise/hypothesis pair often show a large degree of overlap. So we use a copy mechanism Gu et al. (2016) to (1) reduce the difficulty of word prediction when training sequence-to-sequence models on small datasets and (2) encourage the model to pay more attention to the token differences between the textual input of

$$
\begin{aligned}
\text{perceptron loss:}\quad & -\log p(x^{(h)} \mid x^{(p)}, y) + \max_{y' \in Y} \log p(x^{(h)} \mid x^{(p)}, y') \\
\text{hinge loss:}\quad & -\log p(x^{(h)} \mid x^{(p)}, y) + \max_{y' \in Y} \{\log p(x^{(h)} \mid x^{(p)}, y') + \text{cost}(y, y')\} \\
\text{log loss:}\quad & -\log p(x^{(h)} \mid x^{(p)}, y) + \log \sum_{y' \in Y} p(x^{(h)} \mid x^{(p)}, y') \\
\text{softmax-margin:}\quad & -\log p(x^{(h)} \mid x^{(p)}, y) + \log \sum_{y' \in Y} \exp\{\log p(x^{(h)} \mid x^{(p)}, y') + \text{cost}(y, y')\} \\
\text{Bayes risk:}\quad & \mathbb{E}_{p(y'|x^{(h)}, x^{(p)})}[\text{cost}(y, y')] = \sum_{y' \in Y} \text{cost}(y, y') \frac{p(x^{(h)} \mid x^{(p)}, y')}{\sum_{y'' \in Y} p(x^{(h)} \mid x^{(p)}, y'')} \\
\text{infinilog loss:}\quad & -\log p(x^{(h)} \mid x^{(p)}, y) + \log \sum_{y' \in Y, y' \neq y} p(x^{(h)} \mid x^{(p)}, y')
\end{aligned}
$$

Table 4.1: Discriminative objectives considered for fine-tuning GenNLI in this paper. Each is defined for a single training example $\langle x^{(p)}, x^{(h)}, y \rangle$, where $x^{(p)}$ is the premise, $x^{(h)}$ is the hypothesis, and $y \in Y$ is the label.

the encoder and decoder. We compute:

$$p_{copy} = \sigma(\mathbf{w}_{copy}^{\top}[\mathbf{h}_t, \mathbf{s}_t^*, \mathbf{v}_y] + b_{copy}) \tag{4.3}$$

where $p_{copy} \in [0, 1]$ is the probability of copying a word from the input sequence, the vector $\mathbf{w}_{copy}$ and scalar $b_{copy}$ are learnable parameters, and $\sigma$ represents the logistic sigmoid function. We use an extended vocabulary for a specific sentence pair which includes all the words appearing in the input sentence so that the decoder can copy specific words from the input sentence instead of generating out-of-vocabulary (OOV) words.

## 4.3 Discriminative Fine-Tuning

Lewis and Fan (2019) showed that generative classifiers for question answering can be improved by a discriminative fine-tuning step after estimating the generative classifier distributions. They used log loss as their discriminative objective. We also consider using a discriminative fine-tuning step when training our model, specifically we compare log loss to four other discriminative losses:

- **Perceptron loss**: the loss function underlying the perceptron algorithm Rosenblatt (1958)
- **Hinge loss**: the loss function underlying support vector machines (SVMs) and structured SVMs Wahba et al. (1999); Taskar et al. (2004)
- **Softmax-margin**: which combines log loss with a cost function as in hinge loss Povey et al. (2008); Gimpel and Smith (2010)
- **Bayes risk**: the expectation of the cost function with respect to the model's conditional distribution Kaiser et al. (2000); Smith and Eisner (2006)

Table 4.1 shows these discriminative losses.[2] Some losses use a **cost function**, which can be chosen by the practitioner to penalize different errors differently. In our experiments, we define it as $\text{cost}(y, y') = 1$ for $y \neq y'$ and $\text{cost}(y, y') = 0$ if $y = y'$, where $y$ is the gold label and $y'$ is a candidate label.

In addition, we introduce a very simple loss that is inspired by these other discriminative losses while performing quite well overall in our experiments. We call it the **infinilog loss** and define it as follows:

$$-\log p(x^{(h)} \mid x^{(p)}, y) + \log \sum_{\substack{y' \in Y \\ y' \neq y}} p(x^{(h)} \mid x^{(p)}, y') \tag{4.4}$$

The infinilog loss is different from log loss in that the gold label is excluded from the sum. Therefore, infinilog is not bounded below by zero, unlike all other discriminative losses we consider. It does not approach zero as the model becomes increasingly confident in the correct classification, as is the case with log loss and softmax-margin. Rather, infinilog is unbounded, causing learning to continually seek to increase the score of the correct label and decrease the score of the incorrect labels.

We can view infinilog as softmax-margin with a cost function that returns $-\infty$ when $y = y'$ and 0 otherwise. However, the convention usually assumed when defining cost functions for

2. Again, the label prior $p(y)$ ends up canceling out because it is uniform over labels, so we do not show it.

softmax-margin is for the cost function to be nonnegative (Gimpel and Smith, 2010), and similar conventions are assumed with hinge loss. So we choose to use a distinct name for this loss.

Our results in Section 4.6.1 show that fine-tuning using infinilog or one of the investigated discriminative losses leads to better performance than log loss fine-tuning, which was proposed for generative classifiers by Lewis and Fan (2019).

Though the above objectives appear discriminative due to their direct penalization of incorrect labels, they do so by using the key building blocks of generative classifiers. Thus, this fine-tuning achieves some of the benefits of discriminative classifiers while retaining the advantages of generative classifiers, as shown for question answering by Lewis and Fan (2019) and also shown in our experiments below.

## 4.4 Experiments

### *4.4.1 Datasets*

| Dataset | #Train | #Valid | #Test | #Class |
|---|---|---|---|---|
| SNLI | 549K | 9.8K | 9.8K | 3 |
| MultiNLI | 392K | 9.8K | 9.8K | 3 |
| SICK | 4.5K | 0.5K | 4.9K | 3 |
| RTE | 2.4K | 0.2K | - | 2 |
| MRPC | 4.0K | 1.7K | - | 2 |

Table 4.2: Dataset statistics.

We experiment with five sentence pair datasets, namely the Stanford Natural Language Inference corpus (SNLI; Bowman et al., 2015a), the SICK dataset (Marelli et al., 2014), the Multi-Genre Natural Language Inference corpus (MultiNLI; Williams et al., 2018), the binary Recognizing Textual Entailment (RTE; Dagan et al., 2005a) dataset from the GLUE benchmark (Wang et al., 2018a), and the Microsoft Research Paraphrase Corpus (MRPC; Dolan et al., 2004) also from

GLUE.[3] The statistics of the datasets can be found in Table 4.2.

For MultiNLI, we use the matched dev set and mismatched dev set as our validation and test sets, respectively. Otherwise, we use the standard train, validation, and test splits from the original papers (for SNLI and SICK) or the GLUE benchmark (for RTE and MRPC).[4]

### *4.4.2 Baseline Models*

We compare our GenNLI model to two baseline discriminative models, and three pretrained models as described below.

We consider InferSent (Conneau et al., 2017a) and ESIM (Chen et al., 2017a) as our discriminative baselines. InferSent uses a BiLSTM network with max pooling (Collobert and Weston, 2008) to learn generic sentence embeddings that perform well on several NLI tasks. ESIM has a relatively complicated network structure, including a recursive architecture of local inference modeling (MacCartney, 2009; Parikh et al., 2016) and inference composition. The pretrained models we compare to are BERT (Devlin et al., 2019a), RoBERTa (Liu et al., 2019), and XLNet (Yang et al., 2019).

We select these models as our baselines because (1) they are open-source and are frequently used as baselines for NLI tasks in related work (Peters et al., 2018; Williams et al., 2018), and (2) their performance is strong on standard leaderboards.[5]

3. While MRPC is a binary paraphrase classification task rather than an NLI or entailment task, we treat it as a binary entailment task by choosing one of the sentences arbitrarily as the premise and using the other as the hypothesis.

4. MRPC and RTE have no public test set, so we report their performances on the development sets.

5. GLUE leaderboard: `https://gluebenchmark.com/leaderboard/`; SNLI leaderboard: `https://nlp.stanford.edu/projects/snli/`

### *4.4.3 training details*

Both generative and discriminative models are initialized with GloVe pretrained word embeddings (Pennington et al., 2014a).[6] The word embedding dimension and the LSTM hidden state dimension are set to 300. All parameters, including the word embeddings, are updated during training. The label embedding dimensionality for GenNLI is set to 100. All the experiments are conducted 5 times with different random seeds and we report the median scores.

**GenNLI.** The training includes two steps: the model is first trained with the generative objective only (Equation 4.1) for 20 epochs, followed by the discriminative fine-tuning objective only (one of the objectives in Table 4.1) for 15 epochs. Unless otherwise specified, we use infinilog for discriminative fine-tuning. Section 4.6.1 compares fine-tuning objectives.[7]

**Discriminative baselines.** We run the open source code of InferSent[8] and ESIM.[9] Following their implementation, training stops when the performance on the dev set does not improve across 5 consecutive epochs or the learning rate sufficiently decays (e.g,. less than $e^{-5}$).

For both GenNLI and discriminative baselines, we use the Adam (Kingma and Ba, 2015a) optimizer with learning rates of 0.001 and 0.1, and SGD with learning rates 0.1, 0.5, 1, and 2, and select the model with the best performance on the dev set.

**Pretrained baselines.** We use the Hugging Face PyTorch implementation (Wolf et al., 2019) of pretrained transformer (Vaswani et al., 2017a) models.[10] BERT, XLNet, and RoBERTa are configured with 'bert-base-uncased', 'xlnet-base-cased', and 'roberta-base', respectively. We use

6. All of our experiments use uncased 300-dimensional GloVe vectors trained on 6 billion tokens (`http://nlp.stanford.edu/data/glove.6B.zip`).

7. Our implementation is available at `https://github.com/tyliupku/gen-nli`

8. `github.com/facebookresearch/InferSent`

9. `github.com/coetaur0/ESIM`

10. `github.com/huggingface/transformers`

the vector at the position of the [CLS] token in the last layer as the output of pretrained models, and map the output to NLI classification with a linear transformation. We fine-tune the pretrained models on our training sets for 10 epochs. We observe that the models usually converge within the first 3-5 epochs.

## 4.5 Results

### *4.5.1 Data Efficiency*

We first empirically characterize GenNLI, discriminative baselines, and pretrained baselines in terms of data efficiency. We construct smaller training sets by randomly selecting 5, 20, 100, 500, and 1000 instances per class, and then train separate models across these different-sized training sets. Table 4.3 shows the results.[11]

When using training sets with 100 or fewer instances per class, GenNLI outperforms the pretrained baselines on all datasets except for MRPC. We would hope that pretrained models like BERT would produce generalized text representations that would perform well after fine-tuning with a relatively small number of examples, but here we observe that a thousand or more examples is required to outperform GenNLI on most datasets.

With small training sets, GenNLI also has better performance than the other discriminative baselines, though the performance gap does shrink as the training set gets larger. The accuracies become comparable when we have 1000 instances per label. We also see that on the full training set, the discriminative baselines outperform GenNLI, which accords with our expectations and the findings of prior work in generative classifier on document classification.

11. SICK does not have results in the 1000 column because the 'contradiction' label has only 665 instances.

### 4.5.2 Training Label Noise

To measure robustness to label noise, we construct noisy datasets by randomly flipping the labels of 10%, 30%, or 50% of the training instances in the binary classification tasks. The labels of other instances are unchanged. Evaluation is done on the original validation and test sets.

Table 4.4 shows a comparison of GenNLI, InferSent, and RoBERTa on noisy datasets. In addition, we report the value of the Matthews Correlation Coefficient (MCC) (Matthews, 1975). The value of MCC ranges from -1 to 1, with higher value indicating a better classification model. MCC considers all values in the confusion matrix and describes it with a single number. It is viewed as a balanced measurement when the classes are of very different sizes (Boughorbel et al., 2017).

We find all of the models are robust to slight noise, as the accuracy does not drop dramatically with 10% noisy training data. However, as we increase the proportion of the label noise, the performance of InferSent decreases more rapidly than GenNLI. The results are consistent between the two metrics. It is worth noting that GenNLI works better than RoBERTa under the 50%-noisy-data setting, even though RoBERTa has much stronger performance with the unchanged training set. In other words, GenNLI is more robust as the performance drops only slightly with extremely noisy training data.

In general, training deep neural networks requires abundant clean data. When dealing with potentially noisy data, it may be worthwhile to build both generative and discriminative classifiers.

### 4.5.3 Imbalanced Label Distributions

We also perform experiments in a setting with label imbalance in the training set. Each imbalanced training set is constructed by random sampling and keeping only 10%, 20%, or 50% of the instances from one selected class, and keeping all the instances from the other classes. We use the original validation and test sets. We still use a uniform prior for GenNLI.

Table 4.5 shows the comparison of generative, discriminative, and BERT-based classifiers un-

der various imbalanced training sets.[12] Aside from the 10%-non-entailment RTE dataset, RoBERTa always performs the best. This is unsurprising because, even after subsampling, the training set sizes are on a similar order of magnitude as the full sets, with which RoBERTa excels (Table 4.3). However, RoBERTa does show degradation as the subsampling rate becomes more extreme (more than 10% in MRPC, 8-18% in RTE, and 4-5% on MNLI). GenNLI shows a smaller or comparable decrease in performance, though its overall accuracies are lower. In comparing the generative and discriminative classifiers, GenNLI always outperforms InferSent when keeping only 10% of the instances for the selected class. However, as the percentage of instances in the selected class increases, InferSent begins to perform better than GenNLI.

Another finding is that the different labels have different effects under the imbalanced setting. For example, the performance of RTE/non-entailment decreases more slowly than RTE/entailment for both GenNLI and InferSent, which might suggest that the non-entailment label requires fewer training examples than entailment.

Data efficiency might also affect performance under the label imbalanced setting. We believe it is not the only factor for a performance difference between the generative and discriminative models, as the MNLI dataset has 130k instances per class and the training set still has more than 270k instances in total even under the 10% setting, indicating GenNLI has certain advantages over InferSent when the label distribution is imbalanced.

## 4.6 Analysis

### *4.6.1 Modeling and Training Decisions*

We now empirically assess the importance of major components of modeling and training. As shown in Table 4.6, the copy mechanism is essential, which meets our expectation because we

12. We report the results on these three datasets since they represent different characteristics in terms of training set size, number of candidate labels, and performance difference between GenNLI and InferSent on the full training set.

observe a lot of lexical overlap between the premise and hypothesis in many pairs.[13] We find both generative training and fine-tuning objectives to be helpful, as better results are achieved by training with both objectives.

GenNLI defines the conditional distribution of hypotheses given a premise and label. We could instead model $p(x^{(p)} \mid x^{(h)}, y)$. The final two rows of Table 4.6 compare the two, showing better performance with $p(x^{(h)} \mid x^{(p)}, y)$. The difference is larger in SNLI, which may be due in part to how the dataset was created. If annotators are provided with a premise and label and asked to write hypotheses, as in SNLI, we would expect that a generative model that matches this process would excel. The difference may also be due to the fact that in the entailment pairs, the premise often has more information than the hypothesis, and it is expected to be easier to remove information (when generating the hypothesis from the premise) than to add it.

### *4.6.2 Discriminative Fine-Tuning Comparison*

Table 4.7 compares discriminative fine-tuning objectives. Several choices, including hinge, softmax-margin, and infinilog, consistently outperform the log loss used as discriminative fine-tuning objective by Lewis and Fan (2019). The perceptron loss and Bayes risk also often outperform log loss. It is worth noting that infinilog performs the best when using the full training set on four out of five datasets, while softmax-margin is best with smaller training sets. These results suggest that improving discriminative fine-tuning does not harm the data efficiency benefits of generative classifiers, but rather is able to accentuate them.

### *4.6.3 Data Generation*

One advantage of generative models is that they can be used to generate samples in order to interpret how the model works. Since we include label information in the decoder of GenNLI, we are

13. All the experiments in our paper are in-domain testing. We also test GenNLI in out-of-domain (OOD) datasets to see whether the copy mechanism is helpful in this case. For example, we train on MNLI and test on SICK. The trend is not consistent across different OOD settings.

able to generate various hypotheses for a premise by specifying the label. Table 4.8 shows example generations from two models, one using the full dataset for training and the other using a small training set with only 500 examples per class. We use greedy decoding for these generations.

We observe that the generated examples comport with the labels and premises we have specified, and the generation is of high quality in terms of fluency. However, the diversity is relatively low, with the generated samples looking similar to the premise. This is not surprising since we assume the decoder relies heavily on the copy mechanism when trained on NLI pairs, as some hypotheses differ only slightly from their corresponding premises. The generations are relatively short compared to the gold hypotheses, which is likely due in part to greedy decoding. The model might require more training data and/or a different decoding algorithm to be able to produce more diverse generations. We also note that generations for the entailment label generally look better than those for contradiction.[14]

## 4.7 Conclusion

We proposed GenNLI, a discriminatively-finetuned generative classifier for NLI tasks, and empirically characterized its performance by comparing it to discriminative models and pretrained models. We found several discriminative fine-tuning objectives to outperform log loss, including infinilog, a simple but effective choice. We conducted extensive experiments with GenNLI, showing its robustness across challenging empirical conditions. We also showed its ability to generate hypotheses given premises and particular labels.

14. Future work may consider using these generations for data augmentation. While our preliminary experiments in this direction were not positive, future work will consider fine-tuning pretrained language models as generative classifiers and using them with diverse decoding strategies to automatically expand small training sets.

| | 5 | 20 | 100 | 500 | 1000 | all |
|---|---|---|---|---|---|---|
| | | | **SNLI** | | | |
| GenNLI | **43.5** | **45.6** | **50.6** | 60.6 | 64.2 | 82.2 |
| InferSent | 37.5 | 39.6 | 44.1 | 56.0 | 63.9 | 84.5 |
| ESIM | 38.4 | 38.6 | 46.7 | 58.2 | 65.4 | 87.6 |
| BERT | 33.4 | 37.3 | 47.4 | 70.1 | 78.7 | 90.6 |
| XLNet | 34.1 | 35.6 | 45.1 | 72.3 | 77.3 | 90.9 |
| RoBERTa | 35.1 | 36.0 | 49.3 | **75.9** | **82.8** | **91.7** |
| | | | **MNLI** | | | |
| GenNLI | **44.1** | **47.1** | **49.0** | 60.6 | 63.4 | 67.5 |
| InferSent | 34.1 | 33.7 | 35.2 | 44.9 | 47.9 | 70.4 |
| ESIM | 36.9 | 35.4 | 40.5 | 49.8 | 54.2 | 76.7 |
| BERT | 33.0 | 34.9 | 41.6 | 63.6 | 68.5 | 83.3 |
| XLNet | 35.6 | 35.6 | 39.7 | 68.2 | 74.4 | 86.3 |
| RoBERTa | 33.2 | 34.9 | 42.7 | **68.8** | **74.6** | **87.3** |
| | | | **SICK** | | | |
| GenNLI | **50.6** | **64.7** | **68.7** | 75.2 | - | 80.4 |
| InferSent | 35.5 | 46.3 | 60.2 | 73.2 | - | 83.6 |
| ESIM | 34.5 | 48.4 | 62.9 | 75.4 | - | 84.6 |
| BERT | 36.7 | 56.7 | 63.6 | 78.6 | - | 86.0 |
| XLNet | 34.1 | 55.3 | 62.3 | 79.0 | - | 86.8 |
| RoBERTa | 33.5 | 56.7 | 66.3 | **83.4** | - | **88.5** |
| | | | **RTE** | | | |
| GenNLI | **57.0** | **57.7** | **59.2** | **60.4** | 61.4 | 62.6 |
| InferSent | 49.5 | 47.3 | 52.4 | 54.2 | 55.2 | 56.3 |
| ESIM | 50.1 | 50.3 | 53.5 | 55.8 | 57.3 | 58.9 |
| BERT | 47.3 | 48.0 | 49.1 | 59.9 | 64.3 | 66.4 |
| XLNet | 50.9 | 53.4 | 55.9 | 60.3 | 64.6 | 68.6 |
| RoBERTa | 52.7 | 53.1 | 53.8 | 59.6 | **67.8** | **74.7** |
| | | | **MRPC** | | | |
| GenNLI | **62.8** | 64.1 | 66.2 | 67.8 | 69.9 | 72.9 |
| InferSent | 52.5 | 54.6 | 58.1 | 65.1 | 70.9 | 73.1 |
| ESIM | 54.1 | 54.3 | 59.7 | 64.8 | 71.2 | 75.1 |
| BERT | 53.1 | 55.0 | 57.0 | 69.6 | 74.1 | 82.3 |
| XLNet | 55.3 | 64.7 | **68.5** | 78.7 | 82.5 | 85.2 |
| RoBERTa | 59.8 | **65.3** | 67.5 | **80.3** | **84.4** | **87.1** |

Table 4.3: Comparison of classification accuracy of GenNLI, discriminative baselines, and pre-trained baselines with various amounts of training data. Here 5/20/100/500/1000 indicates the number of training instances per class. The best result for each task and data amount is shown in bold, and the best result between GenNLI and the discriminative baselines is underlined.

| Accuracy | | 50% | 30% | 10% | 0% |
|---|---|---|---|---|---|
| MRPC | InferSent | 40.6 | 61.7 | 72.2 | 73.1 |
| | RoBERTa | 66.5 | **76.8** | **85.3** | **87.1** |
| | GenNLI | **68.5** | 70.0 | 71.7 | 72.9 |
| RTE | InferSent | 50.4 | 50.9 | 54.5 | 56.3 |
| | RoBERTa | 52.0 | **63.5** | **76.2** | **74.7** |
| | GenNLI | **58.8** | 59.9 | 59.6 | 62.6 |
| MCC | | 50% | 30% | 10% | 0% |
| MRPC | InferSent | -0.018 | 0.189 | 0.357 | 0.379 |
| | RoBERTa | 0.000 | **0.447** | **0.664** | **0.707** |
| | GenNLI | **0.214** | 0.245 | 0.303 | 0.352 |
| RTE | InferSent | 0.024 | 0.111 | 0.017 | 0.129 |
| | RoBERTa | 0.030 | **0.266** | **0.521** | **0.501** |
| | GenNLI | **0.173** | 0.190 | 0.191 | 0.230 |

Table 4.4: Classification accuracy and Matthews Correlation Coefficient (MCC) when using noisy training sets. The percentages are the fractions of training instances with flipped labels. 0% is the unchanged training set. The best result for each task and each noisy setting is shown in bold, and the second-best one is underlined.

| Dataset | Subsampled Label | Model | Accuracy | | | | Matthews Correlation Coefficient | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 10% | 20% | 50% | 100% | 10% | 20% | 50% | 100% |
| MRPC | paraphrase | InferSent | 49.2 | 63.1 | 70.6 | 73.1 | 0.244 | 0.362 | 0.372 | 0.379 |
| | | RoBERTa | **74.1** | **83.2** | **86.0** | **87.1** | **0.526** | **0.645** | **0.688** | **0.707** |
| | | GenNLI | 70.2 | 70.7 | 72.0 | 72.9 | 0.301 | 0.367 | 0.318 | 0.352 |
| | non-paraphrase | InferSent | 68.3 | 70.9 | 73.8 | 73.1 | 0.191 | 0.287 | 0.373 | 0.379 |
| | | RoBERTa | **77.2** | **81.2** | **86.3** | **87.1** | **0.469** | **0.568** | **0.697** | **0.707** |
| | | GenNLI | 70.8 | 70.3 | 72.2 | 72.9 | 0.333 | 0.292 | 0.319 | 0.352 |
| RTE | entailment | InferSent | 47.3 | 47.3 | 52.3 | 56.3 | 0.000 | 0.036 | 0.135 | 0.129 |
| | | RoBERTa | **66.7** | **66.7** | **71.5** | **74.7** | **0.226** | **0.230** | **0.426** | **0.501** |
| | | GenNLI | 55.8 | 56.5 | 59.9 | 62.6 | 0.128 | 0.135 | 0.194 | 0.230 |
| | non-entailment | InferSent | 52.7 | 52.7 | 54.0 | 56.3 | 0.001 | 0.035 | 0.065 | 0.129 |
| | | RoBERTa | 56.0 | **62.1** | **72.9** | **74.7** | 0.177 | **0.371** | **0.471** | **0.501** |
| | | GenNLI | **60.5** | 60.3 | 62.2 | 62.6 | **0.209** | 0.204 | 0.181 | 0.230 |
| MNLI | entailment | InferSent | 57.4 | 60.1 | 67.8 | 70.4 | 0.396 | 0.431 | 0.522 | 0.557 |
| | | RoBERTa | **82.4** | **84.8** | **87.0** | **87.3** | **0.747** | **0.776** | **0.806** | **0.809** |
| | | GenNLI | 60.8 | 61.7 | 67.1 | 67.5 | 0.410 | 0.452 | 0.497 | 0.512 |
| | neutral | InferSent | 60.5 | 62.5 | 68.8 | 70.4 | 0.445 | 0.469 | 0.539 | 0.557 |
| | | RoBERTa | **83.0** | **84.5** | **85.9** | **87.3** | **0.754** | **0.769** | **0.790** | **0.809** |
| | | GenNLI | 61.7 | 63.8 | 67.6 | 67.5 | 0.463 | 0.487 | 0.491 | 0.512 |
| | contradiction | InferSent | 60.8 | 64.0 | 67.9 | 70.4 | 0.444 | 0.479 | 0.526 | 0.557 |
| | | RoBERTa | **82.7** | **84.5** | **86.6** | **87.3** | **0.748** | **0.773** | **0.800** | **0.809** |
| | | GenNLI | 61.0 | 62.0 | 65.6 | 67.5 | 0.444 | 0.466 | 0.492 | 0.512 |

Table 4.5: Classification accuracies and Matthews Correlation Coefficients of test sets when training on label-imbalanced training sets. Column headers indicate the percentage of the subsampled label's training instances that are retained in the training set. All training instances are used for the other labels. The best result for each task and each subsample setting is shown in bold, and the second-best one is underlined.

| | SNLI | RTE |
|---|---|---|
| GenNLI | 82.2 | 62.6 |
| no copy mechanism | 74.4 | 54.7 |
| no generative training | 80.1 | 60.3 |
| no discriminative fine-tuning | 79.1 | 61.7 |
| GenNLI, $p(x^{(h)} \mid x^{(p)}, y)$ | 82.2 | 62.6 |
| GenNLI, $p(x^{(p)} \mid x^{(h)}, y)$ | 77.1 | 59.7 |

Table 4.6: Results showing contribution of individual modeling/training decisions on SNLI and RTE.

| | 5 | 20 | 100 | 500 | 1000 | all |
|---|---|---|---|---|---|---|
| **SNLI** | | | | | | |
| perceptron | 41.8 | 44.1 | 49.6 | 58.4 | 62.5 | 80.4 |
| hinge | 42.3 | 45.3 | 49.9 | 58.6 | 63.1 | 81.1 |
| log | 42.1 | 43.2 | 49.1 | 58.6 | 62.3 | 80.7 |
| softmax-margin | **43.5** | 45.3 | **50.6** | **60.6** | **64.2** | 81.9 |
| infinilog | 42.7 | **45.6** | 50.0 | 59.8 | 63.7 | **82.2** |
| Bayes risk | 42.8 | 44.7 | 49.0 | 58.3 | 62.6 | 80.1 |
| **MNLI** | | | | | | |
| perceptron | 42.7 | 45.5 | 46.7 | 58.1 | 61.6 | 66.3 |
| hinge | 43.2 | 46.3 | 48.2 | 60.2 | 62.8 | 67.1 |
| log | 42.1 | 45.4 | 46.7 | 58.3 | 61.4 | 66.2 |
| softmax-margin | **44.1** | **47.1** | **49.0** | **60.6** | **63.4** | **67.5** |
| infinilog | 42.3 | 45.9 | 47.7 | 60.0 | 62.8 | 67.3 |
| Bayes risk | 43.1 | 45.7 | 47.7 | 59.1 | 61.6 | 66.2 |
| **SICK** | | | | | | |
| perceptron | 49.1 | 61.7 | 66.9 | 73.4 | - | 79.7 |
| hinge | **50.6** | 63.8 | 67.8 | 73.6 | - | 80.0 |
| log | 48.6 | 62.1 | 67.5 | 73.1 | - | 79.8 |
| softmax-margin | 50.2 | **64.7** | **68.7** | 74.3 | - | 80.2 |
| infinilog | 48.4 | 62.4 | 68.3 | **75.2** | - | **80.4** |
| Bayes risk | 48.2 | 62.4 | 67.2 | 72.8 | - | 79.7 |
| **RTE** | | | | | | |
| perceptron | 56.1 | 57.4 | 57.9 | 59.4 | 60.1 | 61.1 |
| hinge | 56.4 | 57.1 | 58.8 | 59.2 | 61.3 | 62.2 |
| log | 56.5 | 57.1 | 57.4 | 59.1 | 59.7 | 60.5 |
| softmax-margin | **57.0** | **57.7** | **59.2** | **60.4** | 61.1 | 62.2 |
| infinilog | 56.7 | 57.4 | 58.1 | 59.6 | **61.4** | **62.6** |
| Bayes risk | 56.1 | 57.2 | 58.3 | 59.3 | 60.6 | 61.4 |
| **MRPC** | | | | | | |
| perceptron | 62.1 | 62.5 | 64.6 | 66.1 | 68.6 | 69.8 |
| hinge | 62.3 | 63.8 | 65.4 | 67.1 | 69.0 | 71.8 |
| log | 61.7 | 62.1 | 64.1 | 65.9 | 68.1 | 71.3 |
| softmax-margin | 62.6 | **64.1** | **66.2** | **67.8** | **69.9** | 72.8 |
| infinilog | 62.8 | 63.7 | 65.6 | 67.4 | 69.8 | **72.9** |
| Bayes risk | **63.2** | 63.5 | 65.6 | 67.7 | 69.5 | 72.5 |

Table 4.7: Comparison of discriminative fine-tuning objectives. The best result for each task and data amount is shown in bold, and the second-best one is underlined.

| | | |
|---|---|---|
| | | **GenNLI trained on full SICK training set** |
| | $x^{(p)}$ | A man is sitting near a bike and is writing a note. |
| | $x^{(h)}$ | A man with paint covered clothes is sitting outside in a busy area writing something. |
| N | gen. | **A man is sitting in a bike and is writing a note in a busy area.** |
| | $x^{(p)}$ | People wearing costumes are gathering in a forest and are looking in the same direction. |
| | $x^{(h)}$ | Masked people are looking in the same direction in a forest. |
| E | gen. | **People wearing costumes are looking in a forest.** |
| | $x^{(p)}$ | There is no child holding a water gun or getting sprayed with water. |
| | $x^{(h)}$ | A laughing child is holding a water gun and getting sprayed with water. |
| C | gen. | **A child is holding a water gun.** |
| | | **GenNLI trained on small SICK training set** |
| | $x^{(p)}$ | A little girl and a woman wearing a yellow shirt are getting splashed by a city fountain. |
| | $x^{(h)}$ | The young girl is playing on the edge of a fountain and an older woman is watching her. |
| N | gen. | **A little girl is playing in the background.** |
| | $x^{(p)}$ | A man is playing a flute. |
| E | $x^{(h)}$ | A man is playing the flute. |
| | gen. | **A flute is being played by a man.** |
| | $x^{(p)}$ | There is no man on a rock high above some trees standing in a strange position. |
| | $x^{(h)}$ | A man is on a rock high above some trees and is standing in a strange position. |
| C | gen. | **A man is on a rock high above some trees is standing in a strange position.** |

Table 4.8: Generated hypotheses for premises with given labels (N = neutral, E = entailment, C = contradiction).

# CHAPTER 5

# GENERATING DIVERSE STORY CONTINUATIONS WITH CONTROLLABLE SEMANTICS

## 5.1 Introduction

Automatic story generation has a long history. Early work are primarily based on hand-written rules (Klein et al., 1973; Meehan, 1977; Dehn, 1981; Turner, 1993). Later there are some methods based on planing from artificial intelligence (Theune et al., 2003; Oinonen et al., 2006; Riedl and Young, 2010). Recently, data-driven methods have been applied to story generation (McIntyre and Lapata, 2010; Elson, 2012; Daza et al., 2016; Roemmele and Gordon, 2015; Clark et al., 2018a).

We consider the problem of **automatic story continuation generation**, i.e., how to generate story continuations conditioned on the story context. Inspired by recent work in controllable generation (Hu et al., 2017a; Ficler and Goldberg, 2017), we propose a simple but effective modeling framework for controlled generation of multiple, diverse outputs based on interpretable control variables. Each control variable corresponds to a sentence attribute. Compared to previous work that only seeks to control the values of sentiment, length, and automatically-induced clusters, we further explore neural text generation with predicates from semantic role labeling structures and frame semantic representations.

Recently, Clark et al. (2018a) has proposed a neural text generation method that can explicitly represent entities mentioned. In addition, event sequences (Chaturvedi et al., 2017; Liu et al., 2018) are also important elements in narrative texts but under-explored. We believe this information is very useful for neural text generation because people understand and create new objects by performing mental operations on their existing knowledge (Baker et al., 1998). In this paper, we develop models that control the presence or absence of particular frames from FrameNet (Baker et al., 1998), a curated inventory of semantic frames.

We compare the diversity of story continuations controlled by different sentence attributes and

find that using frames yields the most diverse continuations. Unlike certain other attributes, frames have hundreds of possible values. Some frame values can help to get a natural continuation, while others are not applicable when considering the story context. This leads to the second part of our work: we explore methods to rerank continuations to yield a small number of high-quality outputs.

One potential use case of controllable, diverse story generation is collaborative writing applications. Clark et al. (2018b) indicates that participants find creative writing with a machine in the loop is fun and helpful. However, computers are not able to match human performance in predicting the next step in a given sequence of events and writing the next line of the story. But with suggestions from a collaborative writing system, people can be inspired creatively and produce better pieces. Predicting and controlling with frame values suggests a new way of interaction with collaborative writing system.

To avoid overwhelming the users with the continuations controlled by all potential frame values, we propose a control system to obtain the $k$-best generations from the $n$ frame values[1] as the candidate set of the story continuation. Two different frame ranking methods have been provided: one is to rerank the continuations generated with all frame values using a reverse scoring model (Li et al., 2016a) and get the $k$-best generations; the other is to firstly predict the $k$-most reasonable frame values based on the context, and use these values to control the generations. These $k$-best generation could be the set of suggestions during creative writing.

We conduct both quantitative and qualitative study. We evaluate the controllability of our framework and compare the diversity of generation with baseline systems. Our empirical results show that: (1) our framework is accurate in terms of generating outputs that matches the target control attribute values; (2) our framework increases maximum metrics scores compared to $n$-best list generation with beam search; (3) Controlling frame yields more diverse outputs compared to other control attributes. We also conduct a human evaluation to assess the utility of providing multiple suggestions from our models in a creative writing setting, demonstrating promising results

1. Because predicates are shallow semantic representation, we did not use it in our final system.

for the potential of controllable, generation in a collaborative writing system.

| **Context:** | |
|---|---|
| sandra needed a new phone . her old one had broken . she walked into the store and bought a new one . she was very excited . | |
| **Control Attributes and Generated Continuations:** | |
| **Sentiment**: | |
| positive | sandra was happy to have a new phone . |
| **Predicates**: | |
| loved | sandra loved the new one . |
| gave | sandra 's mom gave sandra a refund . |
| **Frames**: | |
| Calendric_unit | it was the perfect day of her life . |
| Cardinal_numbers | it was a perfect one . |
| Activity_ongoing | she kept it on the couch . |

Table 5.1: Story continuations conditioned on various control attributes generated from our framework.

## 5.2 Task Description and Definitions

Given a story context and a control attribute value, our goal is to generate a story continuation that: (1) conforms to the given attribute value, (2) is relevant to the story context, and (3) is complementary to continuations with other control attribute values, thereby providing a diverse set of continuations when used with multiple attribute values.

We use $x = \langle x_1, x_2, ..., x_{|x|} \rangle$ to denote a story context and $y = \langle y_1, y_2, ..., y_{|y|} \rangle$ to denote a story continuation. The last token $y_{|y|}$ is assumed to be $\langle eos \rangle$. We develop a framework to model tuples $(x, l, y)$, where $l$ is a control attribute. The control attribute represents a characteristic of the continuation, such as its length, sentiment, automatically-induced cluster, verbal predicate, or set of semantic frames. Table 5.1 shows several examples of control attributes and generated continuations corresponding to them from our model.

## 5.3 Model

Our controllable generation framework is a variation of a sequence-to-sequence (seq2seq) model with attention Sutskever et al. (2014); Bahdanau et al. (2015); Luong et al. (2015b). To represent the control attribute values, we define an attribute embedding function $R$ that maps a given attribute value $l$ to a vector $z$: $z = R(l)$. Here $l$ can be a single discrete number or a set of discrete numbers, depending on what attributes are being used. The control variable $z$ contains two parts: $z = [z_{\text{enc}}; z_{\text{dec}}]$ where semicolon (;) denotes vertical concatenation and $z_{\text{enc}}$ and $z_{\text{dec}}$ are additional inputs for the encoder and decoder respectively.

**Encoder.** For our encoder, we use a standard bidirectional recurrent neural network (RNN):

$$\overrightarrow{s_i} = f_{e1}([v_i; z_{\text{enc}}], \overrightarrow{s_{i-1}})$$
$$\overleftarrow{s_i} = f_{e2}([v_i; z_{\text{enc}}], \overleftarrow{s_{i+1}})$$
$$s_i = [\overrightarrow{s_i}; \overleftarrow{s_i}]$$

where $v_i$ is the vector representation of word $x_i$, $s_i \in \mathbb{R}^d$ is the hidden state at time $i$, and $f_{e1}$ and $f_{e2}$ are the forward and backward RNN functions.

**Decoder.** Our decoder uses an RNN with the general global attention scheme from Luong et al. (2015b). An additional input $z_{\text{dec}}$ is fed to the decoder at each time step to reflect the characteristics of the control variable:

$$h_j = f_d([y_{j-1}; z_{\text{dec}}], h_{j-1})$$

where $h_j$ is the decoder RNN hidden vector at time step $j$ and $f_d$ is the decoder RNN function. Then, the conditional probability with controllable generation can be decomposed as follows:

$$\log p_\Theta(y \mid x, l) = \sum_{j=1}^{|y|} \log p_\Theta(y_j \mid y_{<j}, s, l)$$

Here $s$ represents the hidden states of the source sequence and $\Theta$ are the parameters of the seq2seq attention model.

**Training.** Our training objective is:

$$\min_{\Theta, R} \sum_{\langle x,l,y \rangle \in \mathcal{D}} -\log p_\Theta(y \mid x, l) \tag{5.1}$$

where $\mathcal{D}$ is the training data, i.e., we assume attribute values $l$ are provided during training. In practice, these will be predicted automatically using a linguistic analyzer. With certain attributes, we do not update the attribute value embeddings $R$, i.e., we fix $z_{\text{enc}}$ and $z_{\text{dec}}$ to one-hot vectors.

**Inference.** We can specify the value of the control variable and generate specific outputs. By changing the variable values, we obtain multiple continuations for the same context. Beam search is used for decoding.

## 5.4 Control Attributes

In this section, we describe the control attributes we explored using our framework. Table 5.2 shows examples of generated continuations for a single story context with several values for the control attributes described below. Given our simple modeling framework, it would be natural to experiment with combining control attributes via summation or averaging of the attribute representations, but we leave an investigation of this to future work, focusing here on using one attribute at a time.

**Sentiment.** Stories may express sentiment regarding their characters or circumstances. We acquire sentiment labels by running the pretrained analyzer from Socher et al. (2013) on the continuations in the training data. The analyzer produces three labels: "negative", "neutral", or "positive".

During training, $z_{\text{enc}}$ and $z_{\text{dec}}$ are fixed one-hot vectors for each value.

**Length.** Some prior work has generated summaries with a desired length (Kikuchi et al., 2016; Fan et al., 2018). We similarly use length of the continuation as a control attribute. Instead of using an embedding for each integer length value, we group the lengths into a small number of bins (details are provided below). $z_{\text{enc}}$ and $z_{\text{dec}}$ are fixed one-hot vectors for each bin.

**Verbal Predicates.** Semantic role labeling (SRL) is a form of shallow semantic parsing that annotates predicates and their arguments in sentences. We consider predicates from a semantic role labeling as control attributes. We use the SRL system from AllenNLP (Gardner et al., 2018) to automatically obtain predicates for the continuations in our training set. Then, a predicate vector is obtained by first summing up 100-dimensional GloVe embeddings (Pennington et al., 2014b) of the predicted predicates (if there is more than one), then reducing the dimension to 64 using principal component analysis.[2] We wish to clarify that we do not use the argument structure from the SRL system. We restrict our focus to simply the set of verbal predicates in the SRL structure; this would presumably be simpler to use in interactive settings where users would specify attribute values for generating continuations.

**Frame Semantics.** A story is composed of a sequence of meaningful events (Chatman, 1980), often following particular patterns described in various terms such as scripts (Schank and Abelson, 1977) and frames. FrameNet (Baker et al., 1998) is an inventory of semantic frames, which are semantic abstractions describing universal categories of events, concepts, and relationships.

2. The reason we use PCA here is to make all attribute embeddings have comparable embedding size, though we did not systematically evaluate the effect of this choice.

We consider frame semantics as another control attribute in our framework. In order to get a frame semantic representation for a continuation, we use SEMAFOR (Das et al., 2014). SEMAFOR automatically produces a frame-semantic parse for a sentence, which consists of spans that evoke particular frames in FrameNet as well as annotations of textual spans that correspond to frame-specific arguments. For our purposes, we drop the arguments and only use the set containing all frames that are evoked in the sentence. A sentence may contain multiple frames. For example, in the sentence "Roa's advice made Emma a lot happier in her life!", "a lot" evokes the *Quantity* frame while "Emma a lot happier" evokes the *Effect* frame.

The frame set variable $z$ is computed by summing embeddings for the frames in the set:

$$z = R(l) = \sum_{j \in l} R_j \tag{5.2}$$

where $l$ is the frame set and $R_j$ is the representation of frame $j$. The frame embeddings are learned during training.[3] For modeling purposes, we restrict our attention to the 100 most frequent frames in the training data. The rest of the frames are pooled together to form a single additional "catch-all" frame.

**Automatically-Induced Clusters.** We also experiment with running $k$-means clustering on the bag-of-words sentence representations of the continuations in the training set. We treat these automatically-induced cluster labels as control attribute values. Below we describe experiments with different cluster labels and analyze the characteristics of the generated outputs.

**Oracle Bag-of-Words Sentence Representations.** We also consider the use of a bag-of-words (BOW) sentence representation as a control attribute. Naturally, the sentence representation of the continuation is not available before generating the continuation in practice. However, we can use this attribute to verify the capability of our model to reconstruct the continuation from its bag-of-

3. The dimension of the frame vector is 64 in our experiments.

words representation.

## 5.5 Experimental Setup

### *5.5.1 Datasets.*

We experiment with the publicly available ROC story corpus developed by Mostafazadeh et al. (2016). It consists of approximately 100K five-sentence stories of everyday events. We sample 2000 stories as a development set and 2000 as our test set. The remaining stories form our training set. Our goal is to generate the fifth sentence (the "continuation") given the previous four sentences. We use the 10k most frequent words in the training set as our vocabulary. A special token $\langle unk \rangle$ is introduced for unknown words.

### *5.5.2 Evaluation.*

Previous work evaluates generation tasks with automatic metrics, such as perplexity (**PPL**), **BLEU** (Papineni et al., 2002),[4] and **ROUGE** (Lin, 2004). We adopt these in our evaluation and add three more metrics using the pretrained story scorer from Sagarkar et al. (2018). The scorer rates a generated continuation given its context along three dimensions: **relevance (R)**, **interestingness (I)**, and **overall quality (O)**. The story scorer does not use a gold standard continuation.

In addition, to evaluate the diversity of the generation, we use **Max-BLEU**[5] and **Max-ROUGE**. First, we compute BLEU and ROUGE scores over a set of outputs $(y_1, y_2, ..., y_n)$ with different

4. In this paper, all BLEU scores are BLEU-2 scores (i.e., using unigrams and bigrams).

5. While max metric scores are not solely measuring diversity, they do provide a sense of the potential of the list. If all entries on the list are the same, the max metric scores would equal the average metric scores. The difference between the max and average metric scores therefore can be viewed as providing a bound on the diversity.

attribute values given the same story context, then we compute the max scores:

$$\text{Max-BLEU} = \max_i \text{BLEU}(y_i, r)$$

$$\text{Max-ROUGE} = \max_i \text{ROUGE}(y_i, r)$$

where $r$ is the gold standard continuation.

We also use **Self-BLEU** (Zhu et al., 2018) to evaluate the diversity of a set of outputs. It is calculated by averaging the BLEU scores computed between all pairs of generated continuations for a given context, then averaging this quantity over all contexts. The smaller the Self-BLEU score is, the more diverse are the generated outputs.

### *5.5.3 Training Details*

Our seq2seq model has a 2-layer biLSTM (Hochreiter and Schmidhuber, 1997b) encoder and a 1-layer LSTM decoder. The hidden dimension of all layers is 512. The word embedding dimension is also 512. For optimization, we use Adam Kingma and Ba (2015a) with learning rate 0.001. We use early stopping based on perplexity on the development set.

## 5.6 Results

We now present our experimental results. Section 5.7 includes results related to how well our generated output matches the desired attribute values. Section 5.7.1 presents results when generating continuations with oracle attribute values. In Section 5.7.2 we use our set-level metrics to evaluate sets of outputs with various attribute values. In Section 5.7.3 we report results when attempting to automatically infer attribute values to generate a small set of high-quality outputs.

## 5.7 Controllability Evaluation

In this section, we evaluate the controllability accuracy of our framework by automatically measuring the match between the attribute values of the generated continuations and the desired values. For certain control variables, like sentiment and frames, this automatic evaluation is prone to errors in the associated analyzers. That is, the metrics that rely on automatic analyzers could become artificially high if our generation models learn to produce outputs that match the biases of the analyzers. We could instead consider manual evaluation of control accuracy. However, we were more interested in devoting our manual evaluation to the question of whether users would find the system outputs useful for a particular goal.

**Sentiment.** We generate three continuations for each story in the development set, one for each sentiment label. Using the same sentiment analyzer from Socher et al. (2013) as above, we obtain predicted sentiment labels for the continuations. Table 5.3 shows the sentiment distribution for each label. We see that the vast majority of the time, the continuations match the desired values. Matching positive sentiment is easiest for our model, followed by neutral.

**Length.** We quantize the generation lengths into bins, each representing a size range. Below are the two settings we consider:

- 3 bins: We use three bins with the following length ranges: [1,7], [8,13], and [14, $\infty$).
- 30 bins: We use a bin for each length. No sentence is longer than 30.

During training, we do not update the representations of the length control variable. After training, we treat the length of the continuation in the development set as the target control variable and generate continuations for each length. The results are shown in Table 5.4 and demonstrate that our model can generate continuations with the desired length with only small differences.

**Verbal Predicates.** We select the top 100 most frequent verbal predicates in the training data. Then for all the stories in the development set, we generate a continuation for each of the 100 predicates. We check whether the predicate appears in the generated continuations. As the results in Table 5.5 show, the framework can nearly always generate outputs with the desired predicates.

**Frame Semantics.** In order to check how frequently the generated output matches the desired frames, we generate continuations for the top 100 frames (one frame for each continuation) for all stories in the development set. We check whether the frame appears in the specific continuation using SEMAFOR. The results are shown in Table 5.6. Most frames have very high match accuracies, but there are a few frames with much lower accuracy, such as "Food" and "Observable_body_parts". These are more concrete frames that may be difficult to reasonably incorporate in certain story contexts.

**Automatically-Induced Clusters.** Given the cluster, the model generates a continuation. Then, we represent the continuation as a bag-of-words sentence embedding (using the same method as when performing the initial clustering) and find the cluster that has the nearest cluster embedding. Then we check whether the two clusters match.

In analyzing the clusters, we observed that cluster 0 corresponds to simple but reasonable continuations. Cluster 2 corresponds to continuations with positive sentiment. Cluster 4 contains continuations with more actions. Some of the generated outputs are shown in Table 5.2. From the results in Table 5.7, we still see controllability for most clusters; however, for target cluster 3, which is rather generic based on our observations, the generated output seems flat.

### *5.7.1 Evaluation with Oracle Attributes*

Table 5.8 shows automatic metric scores with oracle attribute values, i.e., using the attribute values of the gold standard continuations. Unsurprisingly, compared with the seq2seq baseline, the perplexity decreases and the ROUGE and BLEU scores increase with oracle attributes.We also find

that the scores from the story scorer, which does not use the gold standard while scoring, also show improvements over the baseline. We note that frame semantics form one of the most useful control attributes, aside from those that use words directly.

The oracle BOW representation of the gold standard continuation yields the lowest perplexity and highest ROUGE and BLEU scores. It is not surprising that using this attribute would be useful according to metrics that favor matching the gold standard. However, these results do show that our simple modeling framework can make use of the information in the control variable with a high degree of effectiveness. In addition, while the scores from the story scorer are generally higher than for other control attributes, they are roughly on par with those when using predicates and frames.

### *5.7.2 Evaluating Sets of Continuations*

We now evaluate *sets* of continuations using our set-level metrics. Standard methods to generate sets of outputs include beam search (BS) and temperature-based sampling (TS), which we use as baselines. TS with temperature $\tau$ corresponds to transforming probabilities $p_i$ as follows: $\hat{p}_i \propto p_i^{\frac{1}{\tau}}$. A high temperature $\tau$ leads to high variety in generated samples, but also more noise, while lower temperatures lead to samples with less noise but also less diversity.

For each attribute, we generate continuations for each of its values, and compare to BS and TS systems with the same number of outputs. For example, for sentiment, we generate continuations for each of the 3 sentiment values and compare to BS and TS with 3 continuations.

Results are shown in Table 5.9. BS shows the least diversity (as evidenced by its high self-BLEU scores). However, it generally yields high average ROUGE and BLEU scores. TS does very well in terms of diversity, and this diversity enables it to produce higher max scores than BS, but it has lower averages when using small numbers of continuations (3 or 5).

Our sentiment- and cluster-controlled systems outperform TS in max metric scores and BS in diversity (self-BLEU). They also have the highest average BLEU scores, though the differences

are small. With 30 continuations, TS with $\tau = 0.5$ performs best across all metrics; this number of continuations appears to be well-suited for temperature $0.5$. As we move to 100 continuations, we find that using our frame control variable leads to better diversity than TS, suggesting that the move to 100 samples has introduced some amount of repetition. By contrast, the 100 distinct frames and frame sets yield better diversity.

### *5.7.3 Automatically Choosing Attribute Values*

Using our framework, we can generate continuations with any attribute values. However, if we are interested in generating a single continuation, we do not know the ideal attribute values to use. So, we propose two methods to automatically select a small set of values for the frame attribute.

**Frames + Reranking:** Following Li et al. (2016a), we rerank the outputs from the 100 most frequent frame sets by linearly combining the forward score $p(y \mid x)$ and the "reverse score" $\lambda p(x \mid y)$, where the latter comes from a separately-trained seq2seq model. The forward score $p(y \mid x)$ is adjusted by dividing by the length of $y$ in order to not favor shorter outputs.

**Predicted Frames:** We also build a model to automatically predict the frames in the continuation. Given the frames in a sentence $x$, we compute a binary frame vector $f_x$ where entry $j$ is 1 if frame $j$ appears in $x$. We train a model that predicts the frame vector of the continuation given the frame vectors of the previous 4 sentences. The model is an LSTM followed by averaging of hidden states. Mean squared error is minimized during training. After training, the $k$ continuations are selected based on the $k$ frames with the highest predicted score under this frame prediction model.

We use these two methods to produce 3 continuations for each story and report results in Table 5.9. They both achieve a similar balance of quality and diversity as TS with $\tau = 0.6$, with reranking leading to greater diversity than frame prediction and the latter showing higher ROUGE/BLEU scores.

## 5.8 Human Evaluation

Our previous results demonstrate that our frame control system has strong controllability and diversity in generation. In this section, we conduct a human evaluation to assess the utility of providing multiple suggestions from our models in a creative writing setting. We consider four different systems: **BS** with beam size 3; **TS** with 3 continuations using $\tau = 0.6$, which we found to produce outputs with more diversity than $0.5$; **reranking** the 100 most frequent frame sets and using the top 3; and using continuations from the top-3 **predicted** frames under our frame prediction model.[6]

To assess which set of generations from these four systems are most helpful in a collaborative writing setting, we collect annotations using Amazon Mechanical Turk. We randomly select 100 stories. For each story, we generate three outputs as a set of suggestions for each system, so there are 600 comparision pairs in total. We show workers two sets of outputs from different systems and ask them to select which suggestion is more helpful for writing the next line in the story. We also provide a choice of "neither one is helpful at all". We ask them explicitly to imagine they are writing the next line of the given story (see the appendix for more details).

Table 5.10 shows the results.[7] We observe that workers prefer the BS baseline over TS, although TS yields higher diversity. This could be because the continuations from BS are shorter, simpler, and more fluent. In addition, we observe that workers prefer the outputs from the reranking system over BS more often than not. Although the predicted frame system yields more diverse outputs, workers still prefer BS, likely due to the difficulty in predicting frames. The reranking and predicted frame systems are both preferred to TS, though the gap is smaller with the predicted system. We also see that generating helpful suggestions is a difficult task: in many cases workers thought neither system was helpful, especially when given the outputs from BS/TS or TS/predicted.

One may ask why workers do not show a stronger preference for the more diverse sets of

6. The BS and TS baselines do not use control variables.

7. We remove results from 10-question sets where more than half of the questions were answered with the "neither" option, as we were concerned that these annotators did not fully understand the task or did not spend enough time studying the continuations. This occurred in roughly one third of question sets.

outputs. From our own preliminary annotations, we believe this is because diverse outputs tend to be longer and harder to understand, and also because greater diversity increases the chance of producing disfluent or nonsensical outputs. The BS outputs, by comparison, are sensical and mostly on-topic. Even if the suggestions are not creative, they may still help a worker to think about a new direction for the story to take. Nonsensical or disfluent suggestions, however, are rarely helpful.

## 5.9 Conclusion

In this work, we propose a controllable framework that generate the next sentence of a story given its context. Our framework could accurately generate outputs that match the target control attribute values over several attributes: length, sentiment, automatically-induced clusters, predicates, and frame semantic representations. The human study also shows that the multiple suggestions from our model demonstrate promising results in a collaborative writing system.

**Context:** i bought a pair of earbuds at target . i spent ten dollars . someone told me they were cheaper at the dollar store . they were only a dollar .
**Gold Continuation**: i wish i had known that before .

| | |
|---|---|
| **Sentiment** | |
| **negative** | **i was disappointed .** |
| neutral | i decided to keep them . |
| positive | i was able to get a new pair . |
| **Length** | |
| 4 | i was shocked . |
| 5 | i was rather disappointed . |
| 6 | i bought a new pair . |
| 7 | i was able to buy them . |
| **8** | **i was glad i bought them .** |
| 9 | i was able to buy a new pair . |
| 10 | i was able to buy a new pair . |
| 11 | i was able to buy a new pair of shoes . |
| 12 | i was able to buy a new pair of new ones . |
| 13 | i was able to buy a new pair of rubber flavored items . |
| 14 | i was able to buy a new pair and i was very happy . |
| **Verbal Predicates** | |
| **wish, known** | **i wish i will always recognize them .** |
| got | i got a new one . |
| decided | i never decided on the new restaurants . |
| went | i went home with a new friend . |
| is | it is a great price . |
| get | now i have to get a new one . |
| felt | i felt very disappointed . |
| go | i will go to the grocery store . |
| took | i took them to the store . |
| left | i left the store with a new tip . |
| realized | after many years , i realized that . |
| loved | i loved them . |
| ran | i ran back to the store . |
| **Frame Semantics** | |
| **gold frame set*** | **i wanted to be a professional photographer .** |
| Arriving | i got the earbuds . |
| Quantity | it was a lot of fun . |
| Becoming | it ended up being a target . |
| Cardinal_numbers | i paid for $ 50 dollars . |
| Being_obligated | i guess i had a similar card . |
| Kinship | my parents were proud . |
| Statement | i told them i would not be a target . |
| Causation | they sent me to the seller 's desk . |
| Opinion | i think i was a millionaire . |
| Perception_experience | it was a hit . |
| **Clusters** | |
| 0 | i ended up buying a new one . |
| 1 | i bought a new pair of shoes . |
| **2** | **i was a good price .** |
| 3 | i bought a new pair of shoes for free . |
| 4 | i decided to buy a new pair of headphones . |
| **Oracle BOW** | |
| - | i then wish that i had no time . |

Table 5.2: Generated continuations from our framework with different control attribute values. Boldface indicates attribute values of the human-written continuation.

| Target Sentiment | Generated Continuations | | |
|---|---|---|---|
| | negative | neutral | positive |
| negative | **68.5** | 19.6 | 12.0 |
| neutral | 7.0 | **73.9** | 19.2 |
| positive | 0.8 | 3.1 | **96.2** |

Table 5.3: Sentiment match percentages of generated continuations and target sentiment values.

| | $\text{dif} = 0$ | $\text{dif} \leq 1$ | $\text{dif} \leq 2$ | $\text{dif} \leq 3$ |
|---|---|---|---|---|
| 3 bins | 95.8 | 100 | - | - |
| 30 bins | 70.35 | 94.8 | 99.25 | 99.9 |

Table 5.4: Frequency (%) of the generated continuations in the range of $\text{dif} = |l - l_p|$ where $l$ is the continuation length and $l_p$ is the desired length.

| **Predicate** | **M%** | **Predicate** | **M%** |
|---|---|---|---|
| was | 100 | got | 100 |
| had | 100 | decided | 94.4 |
| went | 99.9 | is | 100 |
| made | 99.25 | were | 100 |
| found | 100 | get | 99.95 |
| felt | 99.55 | go | 100 |
| took | 99.2 | ended | 98.25 |
| be | 99.95 | told | 99.9 |
| gave | 99.95 | left | 99.85 |
| said | 100 | bought | 100 |

Table 5.5: Match percentages (M%) showing fraction of stories for which generated continuations contain the desired predicate. The 20 most frequent predicates are shown; additional results are in the Appendix.

| Frame | M% | Frame | M% |
|---|---|---|---|
| Calendric_unit | 99.5 | Locative_relation | 87.4 |
| Arriving | 98.9 | Quantity | 92.5 |
| Temporal_collocation | 99.9 | Becoming | 99.6 |
| Cardinal_numbers | 89.1 | Being_obligated | 97.7 |
| Kinship | 94.4 | Intentionally_act | 99.7 |
| Statement | 98.0 | Causation | 98.6 |
| Emotion_directed | 98.7 | Buildings | 93.0 |
| Personal_relationship | 92.7 | Food | 79.2 |
| Self_motion | 86.4 | Capability | 99.9 |
| Desirability | 98.1 | Observable_body_parts | 74.2 |

Table 5.6: Match percentages (M%) showing fraction of stories for which generated continuations contain the desired frame. Additional results are in the Appendix.

| | Generated Continuations | | | | |
|---|---|---|---|---|---|
| **Target Value** | 0 | 1 | 2 | 3 | 4 |
| 0 | **79.9** | 2.8 | 3.1 | 0.9 | 13.4 |
| 1 | 5.1 | **63.1** | 26.4 | 1.4 | 4.2 |
| 2 | 2.6 | 2.0 | **90.6** | 0.3 | 4.7 |
| 3 | 20.9 | 20.1 | 24.6 | **31.0** | 3.5 |
| 4 | 0.9 | 0.3 | 0.5 | 0.1 | **98.3** |

Table 5.7: Cluster match percentages (%) for each value of the cluster control variable.

| | PPL(↓) | ROUGE | | BLEU (↑) | Story Scorer | | |
|---|---|---|---|---|---|---|---|
| | | ROUGE-1 (↑) | ROUGE-L (↑) | | O (↑) | R (↑) | I (↑) |
| seq2seq | 25.8 | 27.0 | 23.5 | 17.7 | 5.5 | 5.5 | 4.8 |
| sentiment | 25.0 | 26.7 | 23.5 | 17.7 | 5.5 | 5.6 | 4.8 |
| length | 23.1 | 27.3 | 24.6 | 20.3 | 5.7 | 5.8 | 5.0 |
| predicates | 17.1 | 42.9 | 35.1 | 26.4 | 6.0 | 6.2 | 5.2 |
| frames | 15.0 | 41.1 | 35.0 | 27.2 | 5.9 | 6.1 | 5.2 |
| clusters | 24.3 | 28.6 | 25.0 | 18.4 | 5.5 | 6.1 | 5.1 |
| BOW | 5.7 | 64.5 | 54.5 | 45.4 | 6.2 | 6.2 | 5.2 |
| gold standard | - | - | - | - | 6.5 | 6.7 | 5.7 |

Table 5.8: Automatic metrics for baseline system and when using oracle values for control attributes. For the gold standard continuation, we report only the story scorer results because they do not require a gold standard (unlike the other metrics).

| | | ROUGE | | | | BLEU (↑) | | Self-BLEU (↓) |
|---|---|---|---|---|---|---|---|---|
| | | ROUGE-1 (↑) | | ROUGE-L (↑) | | | | |
| | | Max | (Avg) | Max | (Avg) | Max | (Avg) | |
| 3 contin.: | BS, beam = 3 | 31.8 | **(26.7)** | 27.3 | **(22.6)** | 19.7 | (17.0) | 50.5 |
| | TS, $\tau = 0.5$ | 32.5 | (25.5) | 27.8 | (21.6) | 20.3 | (16.3) | 27.0 |
| | TS, $\tau = 0.6$ | 30.5 | (22.2) | 25.9 | (18.8) | 19.0 | (14.8) | **23.8** |
| | sentiment | **32.8** | (25.6) | **28.9** | (22.5) | **21.1** | **(17.1)** | 30.7 |
| | predicted frames | 30.8 | (22.5) | 27.0 | (19.9) | 19.7 | (15.7) | 30.3 |
| | frames + reranking | 30.7 | (21.9) | 26.3 | (18.9) | 18.8 | (14.7) | 25.8 |
| 5 contin.: | BS, beam = 5 | 33.9 | **(26.3)** | 29.4 | **(22.3)** | 21.3 | (16.2) | 68.1 |
| | TS, $\tau = 0.5$ | 35.0 | (25.3) | 30.0 | (21.5) | 21.7 | (16.2) | **40.8** |
| | clusters | **36.1** | (24.5) | **31.7** | (21.4) | **22.9** | **(16.4)** | 43.8 |
| 30 contin.: | BS, beam = 30 | 40.0 | **(25.4)** | 34.3 | (20.8) | 25.6 | (16.0) | 92.5 |
| | TS, $\tau = 0.5$ | **43.0** | **(25.4)** | **37.5** | **(21.6)** | **28.1** | **(16.3)** | **74.0** |
| | length | 42.1 | (24.7) | 35.9 | (20.0) | 26.2 | (14.8) | 82.2 |
| 100 contin.: | BS, beam = 100 | 44.4 | (25.0) | 38.6 | (20.6) | 29.2 | (15.9) | 96.2 |
| | TS, $\tau$= 0.5 | 47.8 | **(25.4)** | 42.3 | **(21.6)** | **32.1** | **(16.3)** | 85.6 |
| | frames (individual) | 47.0 | (24.0) | 41.2 | (20.8) | 29.8 | (15.5) | **72.1** |
| | frames (sets) | **48.3** | (23.1) | **42.7** | (20.1) | 31.2 | (15.1) | 75.5 |

Table 5.9: Metric scores to evaluate the potential of a list of continuations (continu.). We report the maximum and average metric scores over the continuations in each list to evaluate the quality of the lists, and self-BLEU to evaluate diversity. Best results for each metric and each number of outputs are in bold.

| | | **Human Preference** | | |
|---|---|---|---|---|
| system 1 | system 2 | 1 | 2 | neither |
| BS | TS | 43 | 16 | 29 |
| BS | reranking | 18 | 30 | 15 |
| BS | predicted | 38 | 29 | 19 |
| TS | reranking | 18 | 38 | 16 |
| TS | predicted | 18 | 27 | 34 |
| reranking | predicted | 27 | 24 | 21 |

Table 5.10: Human preferences when given three continuations from each pair of systems.

# CHAPTER 6

# LEARNING EXPRESSIVE PRIORS FOR LATENT VARIABLE SENTENCE MODELS

## 6.1 Introduction

Variational autoencoders (VAEs; Kingma and Welling, 2014b) have been widely applied to many natural language processing tasks (Bowman et al., 2016; Zhang et al., 2016; Shen et al., 2017; Kim et al., 2018; Fang et al., 2019; Chen et al., 2019). VAEs provide statistical transparency in describing observations in a latent space and flexibility when used in applications that require directly manipulating the learned representation (Hu et al., 2017b). Recent work (Li et al., 2020) has combined VAEs with BERT/GPT in representation learning and guided generation. However, the representation capacity of VAEs is still limited for modeling sentences due to two main reasons.

One is known as the *posterior collapse* problem, in which the posterior *collapses* to the prior and the generator learns to ignore the latent variable (Bowman et al., 2016). Many methods have been developed to address it: annealing (Fu et al., 2019), weakening the capacity of the generator (Semeniuta et al., 2017; Yang et al., 2017), manipulating training objectives (Burda et al., 2016; Higgins et al., 2017; Zhao et al., 2017b), including the use of "*free bits*" (FB) (Kingma et al., 2016; Li et al., 2019), and changing training (He et al., 2019).

The other reason is the *restrictive assumption* of the parametric forms for the prior and approximate posterior. While these forms are computationally efficient, they limit the expressivity of the model. The main existing solutions (Kingma et al., 2016; Tomczak and Welling, 2018; Razavi et al., 2019) focus on enriching the variational posterior, while other work focuses on learning an expressive prior (Tomczak and Welling, 2018; Serban et al., 2017; Chen et al., 2017b).

In this paper, we follow the latter line of research and draw upon methods in building and learning expressive priors. We first show empirically that the original VAE objective, the evidence lower bound (ELBO), is not effective when learning priors. The issue is not solely due to pos-

terior collapse since it is not resolved by using modifications based on free bits. To address this issue, we propose using a combined objective, adding to the ELBO a second objective (denoted $M_{IS}$) which is a different lower bound on the log marginal likelihood obtained using importance sampling (Burda et al., 2016). Using the combination of the ELBO and $M_{IS}$, we compare multiple choices for the prior, including a mixture of Gaussians, a prior based on a variational mixture of posteriors (VampPrior; Tomczak and Welling, 2018), and a prior based on normalizing flows (NF), specifically real NVP transformations (Dinh et al., 2016). Using a real NVP prior entails creating an invertible mapping from a simple base distribution to the prior distribution of the latent variable in a VAE. This choice allows a flexible prior distribution that is not constrained to a specific parametric family. The hope is that it would be better at modeling the data distribution.

We perform an empirical evaluation of priors and objective functions for training VAE sentence models on four standard datasets. We find the best performance overall when using the flow-based prior and the combined objective in the training objective. We refer to this setting as *FlowPrior*. The generation of prior samples with FlowPrior comports to the training distribution while maintaining a higher diversity than competing models in our quantitative and qualitative evaluation.

To summarize, this paper contributes: (1) a strategy for improved training of sentence VAEs based on combining multiple lower bounds on the log marginal likelihood; (2) the first results applying real NVP to model the prior in sentence VAEs; and (3) comprehensive evaluation and analysis with three expressive priors and training objective variations.

## 6.2 Choice of Prior Family

We now describe the three kinds of priors we will compare in our experiments. The first two are based on Gaussian mixtures (Sec. 6.2.1) and the third is based on normalizing flows (Sec. 2.1.3). We take these three prior families into consideration because they represent the three main categories of work in learning priors: simple Gaussian mixtures (usually as baselines), defining the prior as a function of the approximate posterior (Tomczak and Welling, 2018; Chen et al., 2018),

and flow-based priors (Chen et al., 2017b; Ziegler and Rush, 2019; Ma et al., 2019). Note that we do not make any changes to the approximate posterior distribution. That is, the approximate posterior follows a Gaussian distribution with a diagonal covariance matrix as in standard VAEs.

### *6.2.1 Gaussian Mixture Priors*

Our first choice is a uniform mixture of Gaussians (MoG):

$$p_\psi(z) = \frac{1}{K}\sum_{k=1}^{K} f(z; \mu_k, \mathrm{diag}(\sigma_k^2)) \tag{6.1}$$

where $f(z; \mu, \Sigma)$ is the density function of a $d$-dimensional Gaussian with mean $\mu$ and covariance matrix $\Sigma$. The $\mu_k$ and $\sigma_k$ are learnable parameter vectors with dimensionality $d$ (which is 32 in our experiments). This prior was used as a baseline by Tomczak and Welling (2018). We refer to a VAE that uses this prior as *MoG-VAE*.

Tomczak and Welling (2018) extend MoG-VAE to a "Variational Mixture of Posteriors" prior (VampPrior). This approach parameterizes the prior using a mixture of Gaussians with components given by a variational posterior conditioned on learnable "pseudo-inputs":

$$p_\psi(z) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(z \mid u_k) \tag{6.2}$$

where $K$ is the number of pseudo-inputs, each of which is denoted $u_k$. Pelsmaeker and Aziz (2020) applied this idea to text modeling and we follow their strategy for defining pseudo-inputs. That is, each $u_k$ consists of a sequence of embeddings that have the same dimensionality as word embeddings. For each component $k$, the lengths of pseudo-inputs can vary; they are sampled based on the statistics of the lengths in the training set. We refer to a VAE with this prior as *Vamp-VAE*.

### 6.2.2 *Flow-based Priors*

Our third choice for a prior distribution is to leverage normalizing flows (NF). A normalizing flow is a sequence of invertible, deterministic transformations. By repeatedly applying the rule for change of variables (see Section 2.1.3), the base density is transformed into a more complex one. Networks parameterized using NF can be trained through exact maximum log-likelihood computation. Exact sampling is performed by drawing a sample from the base distribution and performing the chain of transformations. This allows a flexible prior and is expected to have more expressive latent components compared to those based on Gaussian mixtures.

Computing the Jacobian of functions with high dimension and the determinants of large matrices (i.e., the two main computations in NF) are very expensive. Our flow-based prior uses *real-valued non-volume preserving* (real NVP) transformations (Dinh et al., 2016) which are efficient in both training and sampling. These transformations are based on *scale* and *translation* operations. It is worth noting that these two operations are not used in computing the Jacobian determinant and inverse. So one can design arbitrarily complex operations that allow a flexible transformation without incurring large computational cost. Find more details about normalizing flows and real NVP are in Section 2.1.3.

More specifically, we apply real NVP as a prior by creating an invertible mapping between a base distribution $p_0(z_0)$ (in our case, $z_0 \sim \mathcal{N}(0, \mathrm{I})$) and the prior distribution of VAE $z_L \sim p_L(z_L)$:

$$z_L = f_L \circ f_{L-1} \circ ... \circ f_1(z_0) \tag{6.3}$$

where $z_L$ is the sentence latent variable and $f_1, f_2, ..., f_L$ are all bijective functions.

Using the change-of-variables theorem, given a latent variable $z_L$, we can compute the exact

density under the prior with the "image" $z_0$ acquired by inverting the transformation:

$$z_0 = f_1^{-1} \circ ... \circ f_{L-1}^{-1} \circ f_L^{-1}(z_L) \tag{6.4}$$

$$\log p_\psi(z_L) = \log p_0(z_0) - \sum_{l=1}^{L} \log |\det(\frac{\partial f_l(z_{l-1})}{\partial z_{l-1}})| \tag{6.5}$$

We refer to a VAE with a real NVP prior as real NVP-VAE. We find our best setting to consist of a real NVP prior and the combined objective in Section 6.3.1 and we refer to this setting as *FlowPrior*.

## 6.3 Objectives for Learning Priors in VAEs

**ELBO.** Our preliminary experiments found that, when training with the standard ELBO, using more sophisticated priors does not improve perplexity compared to standard Gaussian priors (Table 6.5). Though these priors could potentially be highly multimodal, the learned prior parameters yield approximately unimodal forms (Figure 6.1, left).

Several approaches have been proposed to mitigate or avoid posterior collapse. One method that we include in our experiments is a variation of KL divergence known as "Free Bits" (FB) KL (Li et al., 2019; Kingma et al., 2016). Posterior collapse is mitigated, but the VAE models still do not benefit much from expressive priors (Tables 6.3-6.4). Pelsmaeker and Aziz (2020) made similar observations with an improved FB objective. We speculate these undesirable results are due to the lack of learning signals for the prior parameters.

**Marginal Likelihood via Importance Sampling.** In the ELBO, the prior distribution only appears in the KL term. As a consequence, the prior parameters receive a limited amount of learning signal. The posterior network, by contrast, receives gradient updates from both the reconstruction and KL terms. When minimizing the KL term the potentially expressive prior density can "collapse" to a unimodal form.

We consider optimizing another objective, a different lower bound on the log marginal likelihood obtained using importance sampling (Burda et al., 2016):

$$\log \frac{1}{N} \sum_{i=1}^{N} \frac{p_\theta(x|z^{(i)})p_\psi(z^{(i)})}{q_\phi(z^{(i)}|x)}, \text{s.t. } z^{(i)} \sim q_\phi(z|x)$$

where $x$ is an input in the training data and $N$ is the number of samples in use. This objective was proposed as the training objective in the importance-weighted autoencoder (IWAE; Burda et al., 2016). In this paper, we denote this objective by $\mathrm{M}_{\mathrm{IS}}$.

The use of this term is not only beneficial in training with a tighter objective, but also increasing the flexibility of approximate posterior as shown in (Cremer et al., 2017). With increase of $N$, the approximation posterior has a implicit complex distribution that approaches the true posterior, which is beneficial in learning a expressive prior.

However, use of $\mathrm{M}_{\mathrm{IS}}$ is not always ideal. Rainforth et al. (2018) theoretically proves that a large $N$ is detrimental in learning posterior, which in also shown in our empirical evaluation in Table 6.5.

This term helps particularly when we are learning the prior parameters $\psi$, as we will show in our experiments, but also helps improve performance when we keep the prior fixed as in standard VAEs. It is not surprising that $\mathrm{M}_{\mathrm{IS}}$ would lead to a better test PPL, but we also find adding this term is beneficial to learn a rich representation and avoid posterior collapse (Table 6.5).

**Combination of the Two.** We propose to combine ELBO and $\mathrm{M}_{\mathrm{IS}}$. If we only have $\mathrm{M}_{\mathrm{IS}}$, the posterior only appears in the denominator so learning seeks to make samples from the posterior $q$ less likely under $q$, which could make $q$ a poor proposal distribution. The ELBO, with its reconstruction loss, appears to helpful in learning a better posterior. Similar idea is proposed in prior work (Rainforth et al., 2018) that uses a combined objective to improve IWAE.

### 6.3.1 Combined Training Objective

Our combined training objective then contains three terms: $\mathrm{M}_{\mathrm{IS}}$, reconstruction, and sample-based KL. We draw $N$ samples from $q_\phi(z|x)$, and compute the three terms using the same samples:

$$\mathcal{L}(\theta, \phi, \psi; x) = \log \frac{1}{N} \sum_{i=1}^{N} \frac{p_\theta(x|z^{(i)}) p_\psi(z^{(i)})}{q_\phi(z^{(i)}|x)} + \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(x|z^{(i)}) - \mathrm{KL}_{\phi,\psi}(x, \{z^{(i)}\}_{i=1}^{N})$$
$$\text{s.t. } z^{(i)} \sim q_\phi(z|x) \quad (6.6)$$

When training with the ELBO alone, one typically uses a single sample from $q_\phi(z|x)$. However, since we draw multiple samples anyway in order to compute $\mathrm{M}_{\mathrm{IS}}$, we use those same samples for the reconstruction term, which can lead to more robust gradients of that term than the standard approach of using one sample.

The reason we use sample-based estimates for the KL divergence is because our choices for the prior preclude the possibility of a closed form for the KL. We consider two different approaches when computing sample-based KLs: *standard KL* and a slightly modified one inspired by "Free Bits" (FB) Li et al. (2019); Pelsmaeker and Aziz (2020); Kingma et al. (2016), which we refer to as *FB KL*.

For standard KL, we use Monte Carlo estimation in computing the KL divergence with $N$ samples:

$$\mathrm{KL}_{\phi,\psi}(x, \{z^{(i)}\}_{i=1}^{N}) = \frac{1}{N} \sum_{i=1}^{N} (\log q_\phi(z^{(i)}|x) - \log p_\psi(z^{(i)})) \quad (6.7)$$

For the FB KL, we follow prior work Kingma et al. (2016) that replaces the KL with a hinge loss term in each latent dimension:

$$\text{FB } \mathrm{KL}_{\phi,\psi}(x, \{z^{(i)}\}_{i=1}^{N}) = \sum_{j=1}^{d} \max(\lambda, \mathrm{KL}^{j}_{\phi,\psi}(x, \{z^{(i)}\}_{i=1}^{N})) \quad (6.8)$$

where $\mathrm{KL}^{j}_{\phi,\psi}$ denotes the KL computed only for dimension $j$ of the latent variable, and $\lambda$ is the

value of the "free bits" reserved.

### *6.3.2 Training Procedure*

We describe our training procedure below for FlowPrior, which combines a real NVP prior with the objective in Eq. 6.6. For simplicity, our description only uses one input $x$. In practice, we use mini-batches with a stochastic gradient based optimizer. All the parameters $(\theta, \phi, \psi)$ are updated simultaneously during training.

1. Sample $N$ instances $z_L^{(1)}, z_L^{(2)}, ..., z_L^{(N)}$ from inference network using reparameterization trick.
2. Perform inverse transformation get the 'image' of the point under the base distribution $z_0^{(1)}, z_0^{(2)}, ..., z_0^{(N)}$.
3. Compute the exact log likelihood of the sample prior with change of variable theorem (Eq. 6.5).
4. Compute and backpropagate the loss (Eq. 6.6).

When using the other priors (standard Gaussian, MoG, and VampPrior), we do not need steps 2 and 3 above because those priors can be computed directly without the inverse transformation or change of variable theorem.

## 6.4 Experiments

### *6.4.1 Datasets*

We consider four widely-used, publicly available English datasets: the Penn Treebank (PTB) Marcus et al. (1993b); Bowman et al. (2016), Yahoo Yang et al. (2017); He et al. (2019), Yelp sentiment Shen et al. (2017), and SNLI Bowman et al. (2015b).

The statistic of our dataset is in Table 6.1. For Yelp and SNLI, we follow Li et al. (2019) and create the dataset with downsampling. For Yahoo, we truncate sentences to length 100 due to computational constraints.

| | # Train | # Dev | # Test | Avg L | Max L | # Vocab |
|---|---|---|---|---|---|---|
| PTB | 42,068 | 3,370 | 3,761 | 21 | 82 | 10,002 |
| Yahoo | 100,000 | 10,000 | 10,000 | 68 | 100 | 19,982 |
| Yelp | 100,000 | 10,000 | 10,000 | 9 | 15 | 9,389 |
| SNLI | 100,000 | 10,000 | 10,000 | 12 | 82 | 19,978 |

Table 6.1: Statistics of the datasets. # Train/Dev/Test is the number of train/dev/test instances. Avg L and Max L are the average and maximum length of the sequences in the training sets. # Vocab is the size of the vocabulary including $\langle$unk$\rangle$, $\langle$sos$\rangle$, $\langle$eos$\rangle$, and $\langle$pad$\rangle$.

### *6.4.2 Baselines*

Our baselines include *standard VAE* with linear KL annealing Bowman et al. (2016); *Cyc-VAE* Fu et al. (2019) in which the KL term is reweighted with a cyclical annealing schedule; *Lag-VAE* He et al. (2019) which updates the encoder multiple times before each decoder update; *VAE+FB* Kingma et al. (2016); Chen et al. (2017b) which replaces the standard KL with FB (i.e., Eq. 6.8 with $N = 1$); and *Pre-VAE+FB* Li et al. (2019) which initializes the VAE with a pretrained autoencoder and replaces standard KL with FB. We evaluated these baselines using their open source implementations.[1]

In addition, we include two prior-learning baselines: MoG-VAE (Eq. 6.1) and Vamp-VAE (Eq. 6.2). We follow Pelsmaeker and Aziz (2020) and set 100 components/pseudo-inputs.

Unlike the earlier baselines, for which we used open source codebases, we implemented the MoG-VAE and Vamp-VAE models on top of our standard VAE implementation, which was also used for FlowPrior.

### *6.4.3 Implementation and Training Details*

We use the open source implementations for other baselines. Across all the experiments for our implemented baselines (i.e., standard VAE, MoG-VAE, Vamp-VAE) and our proposed model FlowPrior, we follow prior work (Kim et al., 2018; He et al., 2019; Li et al., 2019) and use a single-layer

1. The links to their implementations are in the Appendix.

| | PTB | Yahoo | SNLI | Yelp |
|---|---|---|---|---|
| Word Embedding | 256 | 512 | 128 | 128 |
| Encoder Hidden States | 256 | 1024 | 512 | 512 |
| Decoder Hidden States | 256 | 1024 | 512 | 512 |

Table 6.2: The size of word embeddings and hidden states in VAE models used in this paper, which are adopted from prior work.

LSTM encoder and decoder with a 32-dimensional latent variable. We follow the prior work (He et al., 2019; Li et al., 2019) and set the embedding dimension as in Table 6.2. We set a dropout rate of 0.5 to both the input embeddings and the output embeddings before the softmax layer in the decoder. All the parameters are initialized with a uniform distribution $U$(-0.01,0.01). For both MoG and Vamp-VAE we use 100 components/pseudo-inputs in the prior. For real NVP, we use 10 affine coupling layers with 3-layer MLP networks for the parameterized scale and translation operations with the dimensionality of 32. We follow Dinh et al. (2016) to compose the affine coupling layers in an alternative pattern and add batch normalization (Ballé et al., 2016) between adjacent affine coupling layers. For models trained with FB KL, we set the target rate as 2, 4, or 8.

We use a batch size of 32 and train using SGD. All models are trained with the simple linear annealing schedule, with same hyperparameter search space. The optimizer is initialized with learning rate 1 or 0.5, and the learning rate is decayed by 1/2 if the dev loss is not improved in two consecutive epochs. The training stops early after 5 learning rate decay operations. We use a linear annealing schedule that increases the weight from 0 to 1 in the first 10 or 20 epoch for the weight of both KL and infinilog term if they are in the training objective. When training with the combined objective, we start adding infinilog after training ELBO objective 10 epochs. For each model variation, we experiment with 5 different random seeds and report the median numbers in the paper.

### *6.4.4 Evaluation Metrics*

Our evaluation measures language modeling performance, the use of the latent variable, and the quality and diversity of generations from the prior and posterior. The metrics are listed below:

**PPL:** We estimate log marginal likelihood using importance sampling (Burda et al., 2016) and calculate perplexity on the test set.[2]

**KL:** We report the KL term in the ELBO on the test set. When training with FB KL, we still report standard KL. For standard VAE, we compute KL with its closed-form expression. Otherwise, we report the KL estimated with samples.

**MI:** We follow Hoffman and Johnson (2016) and report estimated mutual information between the observation and its latent variable.

**AU:** A dimension $z$ in the latent variable is considered "active" if $\mathrm{Cov}_x(\mathbb{E}_{z \backsim q(z|x)}[z]) > 10^{-2}$. AU is then the number of active latent dimensions (Burda et al., 2016) .

**F-PPL and R-PPL:** These metrics measure the correspondence between generated sentences from the model and the training corpus. We evaluate both F-PPL and R-PPL by estimating 5-gram language models using the KenLM toolkit (Heafield, 2011) with its default smoothing method. For F-PPL, we estimate language models from the actual text and compute the perplexity of the generated samples. For R-PPL, we estimate language models from the generated samples and compute the perplexity of the actual text.[3]

**Self-BLEU:** The self-BLEU metric is one measure of the diversity of a set of samples (Zhu et al., 2018). It is calculated by averaging the BLEU scores computed between all pairs of samples.

2. We use 1000 samples which appears to be more than sufficient for estimation; Ziegler and Rush (2019) found that using more than 50 samples did not even show much difference.

3. Our R-PPL is slightly different from that in Fang et al. (2019). For R-PPL, we always concatenate the training set vocabulary (one word per line) to the set of samples from the models to ensure LMs have seen the entire vocabulary.

| Model | PPL(↓) | Recon(↓) | KL | AU(↑) | MI(↑) |
|---|---|---|---|---|---|
| VAE | 101.40 | 101.28 | 0.00 | 0 | 0.00 |
| Cyc-VAE | 107.73 | 101.17 | 2.01 | 5 | 1.24 |
| Lag-VAE | 100.25 | 100.41 | 1.04 | 3 | 0.79 |
| VAE + FB | 101.56 | 99.84 | 4.46 | 32 | 0.90 |
| Pre-VAE + FB | 96.35 | 94.52 | 8.15 | 32 | 6.30 |
| MoG-VAE | 98.22 | 100.54 | 0.00 | 0 | 0.00 |
| MoG-VAE + FB | 97.50 | 99.44 | 2.35 | 32 | 0.68 |
| Vamp-VAE | 98.27 | 100.56 | 0.00 | 0 | 0.00 |
| Vamp-VAE + FB | 97.83 | 99.53 | 2.31 | 32 | 0.72 |
| FlowPrior | 94.72 | 98.46 | 3.28 | 2 | 2.25 |
| FlowPrior + FB | 93.58 | 99.20 | 7.21 | 31 | 2.83 |

Table 6.3: Language modeling results on PTB dataset.

## 6.5 Results

### *6.5.1 Language Modeling*

We first perform language modeling tasks to characterize models' efficacy at modeling texts in terms of modeling the distribution of language data and making use of the latent variable.

We refer to our model as **FlowPrior**, which uses the training objective in Eq. 6.6 which includes $M_{IS}$ and the standard KL (Eq. 6.7). We use **FlowPrior + FB** to refer to our model with the FB KL (Eq. 6.8).

**Comparison to baselines.** The implementation of FlowPrior is based on the building blocks of VAE. Table 6.3 shows our experiments of our implemented standard VAE baselines on the PTB dataset.

We report the performance of FlowPrior on Yahoo, Yelp, and SNLI in Table 6.4. Since our technical contribution lies in learning the prior instead of changing the training procedure or manipulating the KL term, we set the baselines as standard VAE, MoG, and VampPrior for the rest of the paper.

From Tables 6.3 and 6.4, we observe that FlowPrior consistently outperforms all the baselines

| Model | PPL(↓) | Recon(↓) | KL | AU(↑) | MI(↑) |
|---|---|---|---|---|---|
| **Yahoo** | | | | | |
| VAE | 65.77 | 333.17 | 0.00 | 0 | 0.00 |
| MoG-VAE | 64.60 | 332.90 | 0.00 | 0 | 0.00 |
| Vamp-VAE | 74.81 | 344.61 | 0.01 | 0 | 0.00 |
| FlowPrior | 62.49 | 331.57 | 1.43 | 4 | 1.62 |
| FlowPrior + FB | 68.29 | 345.68 | 10.99 | 25 | 0.61 |
| **Yelp** | | | | | |
| VAE | 35.10 | 35.18 | 0.00 | 0 | 0.00 |
| MoG-VAE | 35.18 | 35.20 | 0.01 | 0 | 0.00 |
| Vamp-VAE | 34.99 | 35.15 | 0.00 | 0 | 0.00 |
| FlowPrior | 31.82 | 30.25 | 4.15 | 2 | 2.46 |
| FlowPrior + FB | 39.03 | 36.87 | 10.13 | 32 | 2.57 |
| **SNLI** | | | | | |
| VAE | 25.97 | 41.34 | 0.00 | 0 | 0.00 |
| MoG-VAE | 28.05 | 40.96 | 0.44 | 1 | 0.41 |
| Vamp-VAE | 25.98 | 41.35 | 0.00 | 0 | 0.00 |
| FlowPrior | 22.41 | 37.89 | 3.83 | 3 | 0.97 |
| FlowPrior + FB | 26.19 | 43.56 | 7.59 | 32 | 3.16 |

Table 6.4: Language modeling results on other datasets.

by a large margin in test set perplexity. It is not surprising since the $\mathrm{M}_{\mathrm{IS}}$ term in our training objective directly targets the perplexity metric because the expressions are identical, differing only in the number of samples used. We noticed though use of FB can help achieve a consistently better AU and higher KL, but it does not always lead to a better test PPL and reconstruction. We report additional results on measuring the impact of FB in Table 6.6.

Another finding is that only enhancing prior is not enough. Though these priors have the potential to be multimodal, they could still be unimodal after training. For example, the MoG-VAE might learn a mixture in which all Gaussians have the same location and scale. Also, the complexity of the prior learned by the Vamp-VAE is dependent upon the inference network, so if the inference network does not learn anything useful, the learned prior may not be useful either.

| Prior | PPL(↓) | KL | AU(↑) |
|---|---|---|---|
| | **PTB** | | |
| Standard | 101.8 / 101.4 / 98.4 | 0.0 / 0.0 / 3.2 | 0 / 0 / 2 |
| MoG | 101.9 / 98.2 / 96.7 | 0.0 / 0.0 / 0.0 | 0 / 0 / 0 |
| Vamp | 101.7 / 98.3 / 96.1 | 0.0 / 0.0 / 3.1 | 0 / 0 / 4 |
| Real NVP | 102.5 / 98.4 / 94.7 | 0.0 / 0.0 / 3.3 | 0 / 0 / 2 |
| | **Yahoo** | | |
| Standard | 65.6 / 65.8 / 63.9 | 0.0 / 0.0 / 2.7 | 0 / 0 / 1 |
| MoG | 65.6 / 64.6 / 62.7 | 0.0 / 0.0 / 0.5 | 0 / 0 / 1 |
| Vamp | 78.5 / 74.8 / 62.9 | 0.0 / 0.0 / 1.5 | 0 / 0 / 2 |
| Real NVP | 65.6 / 65.8 / 62.5 | 0.0 / 0.0 / 1.4 | 0 / 0 / 4 |
| | **Yelp** | | |
| Standard | 35.4 / 35.1 / 33.2 | 0.0 / 0.0 / 2.9 | 0 / 0 / 2 |
| MoG | 36.0 / 35.2 / 34.9 | 0.0 / 0.0 / 0.0 | 0 / 0 / 0 |
| Vamp | 38.0 / 35.0 / 33.7 | 0.0 / 0.0 / 4.1 | 0 / 0 / 1 |
| Real NVP | 35.6 / 35.1 / 31.8 | 0.0 / 0.0 / 4.2 | 0 / 0 / 2 |
| | **SNLI** | | |
| Standard | 27.4 / 26.0 / 25.3 | 0.0/ 0.0 / 1.2 | 0 / 0 / 3 |
| MoG | 27.2 / 28.1 / 24.3 | 0.0 / 0.4 / 4.2 | 0 / 1 / 5 |
| Vamp | 27.6 / 26.0 / 23.7 | 0.0 / 0.0 / 2.8 | 0 / 0 / 2 |
| Real NVP | 27.7 / 26.1 / 22.4 | 0.0 / 0.0 / 3.8 | 0 / 0 / 3 |

Table 6.5: Results of $M_{IS}$, using ELBO, and combination for VAEs with several choices for priors . From left to right in each cell corresponds to training with $M_{IS}$ only, ELBO only, combination of ELBO + $M_{IS}$.

**Impact of selection of objectives.** The learned prior baselines (MoG-VAE and Vamp-VAE) fail to learn to use the latent variable, as shown by the small numbers (nearly zero) for the AU and MI metrics in Tables 6.3-6.4. Same observations found in Pelsmaeker and Aziz (2020). We argue that only improving the prior might not be sufficient, as the ELBO objective is difficult to optimize and little information may be learnable for the prior from the ELBO alone. To show the power of our $M_{IS}$ term, we introduce this term to standard-VAE, MoG-VAE, and Vamp-VAE and evaluate the improved models under the same language model metrics.

Table 6.5 compares models trained with the ELBO and our proposed training objective (Eq.

6.6). We observe that the combined objective is beneficial to all metrics for models with different priors across datasets. Our results meet the suggestion in (Rainforth et al., 2018) that tighter bounds are preferable for training the generative network, while looser bounds are preferable for training the inference network.

Still, FlowPrior (real NVP + $M_{IS}$) performs the best in PPL and MI compared to other models, showing the flexibility and the power of the real NVP architecture.

For the "Standard" setting in Table 6.5 the prior is fixed and not learned while in the other three settings the prior is learned. We observe the combination of ELBO + $M_{IS}$ objective is helpful across all. [4]

**Combining with FB.** Using the Free Bits method can help achieve a consistently better AU and higher KL as shown in the overall results in the main paper. We report additional empirical comparisons to focus on measuring the impact of FB for three models in Table 6.6.

Though adding FB yields higher AU and MI, it is not always true that it leads to a better test PPL and reconstruction. This phenomenon has been pointed out by Razavi et al. (2019) that adding FB makes the objective non-smooth which can lead to optimization difficulties. Possible solutions could be changing to a better training procedure. Li et al. (2019) remedy this issue by combining pretraining with FB, namely using a pretrained autoencoder to initialize the inference network before starting training the VAE networks. This suggests that it may be necessary to pretrain the inference network and decoder to unilaterally benefit from FB.

### *6.5.2 Probing of Latent Space*

One appealing aspect of VAEs for sentence modeling is the potential for learning a smooth, interpretable space for sentences. A qualitative way to explore the latent space is to interpolate between samples from the prior distribution. We randomly sample two latent vectors from the prior and

4. For MoG setting, we perform experiments with setting Gaussian component K=1 and observe comparable or slightly worse test PPL under all 3 choices of training loss.

| Model | PPL(↓) | KL | AU(↑) | MI(↑) |
|---|---|---|---|---|
| | | **PTB** | | |
| VAE | 101.4 / 101.6 | 0.00 / 4.46 | 0 / 32 | 0.0 / 0.1 |
| VAE+$M_{IS}$ | 96.9 / 95.8 | 1.57 / 6.34 | 24 / 32 | 0.6 / 1.5 |
| MoG-VAE | 98.2 / 96.8 | 0.00 / 2.35 | 0 / 32 | 0.00 / 0.68 |
| Vamp-VAE | 98.3 / 97.3 | 0.00 / 2.31 | 0 / 32 | 0.00 / 0.72 |
| FlowPrior | 94.7 / 93.6 | 3.28 / 7.21 | 2 / 31 | 2.3 / 2.8 |
| | | **Yahoo** | | |
| VAE | 65.8 / 64.6 | 0.00 / 4.88 | 0 / 32 | 0.0 / 0.9 |
| VAE+$M_{IS}$ | 63.9 / 61.7 | 2.72 / 13.31 | 1 / 32 | 2.0 / 1.7 |
| MoG-VAE | 64.6 / 67.3 | 0.00 / 1.83 | 0 / 32 | 0.0 / 0.6 |
| Vamp-VAE | 74.8 / 75.9 | 0.01 / 1.24 | 0 / 32 | 0.0 / 0.6 |
| FlowPrior | 62.5 / 68.3 | 1.43 / 10.99 | 4 / 25 | 1.6 / 0.6 |
| | | **Yelp** | | |
| VAE | 35.1 / 37.5 | 0.00 / 3.59 | 0 / 32 | 0.0 / 1.0 |
| VAE+$M_{IS}$ | 33.2 / 39.6 | 2.91 / 4.16 | 28 / 32 | 0.9 / 2.2 |
| MoG-VAE | 35.2 / 39.8 | 0.01 / 1.81 | 0 / 32 | 0.0 / 0.6 |
| Vamp-VAE | 35.0 / 39.4 | 0.00 / 1.78 | 0 / 32 | 0.0 / 0.6 |
| FlowPrior | 31.8 / 39.0 | 4.15 / 10.13 | 2 / 32 | 2.5 / 2.6 |
| | | **SNLI** | | |
| VAE | 26.0 / 30.5 | 0.00 / 1.84 | 0 / 32 | 0.0 / 0.9 |
| VAE+$M_{IS}$ | 25.3 / 17.8 | 1.23 / 15.48 | 23 / 32 | 0.5 / 2.0 |
| MoG-VAE | 28.1 / 27.5 | 0.44 / 2.28 | 1 / 32 | 0.4 / 0.7 |
| Vamp-VAE | 26.0 / 29.3 | 0.00 / 5.11 | 0 / 32 | 0.0 / 0.8 |
| FlowPrior | 22.4 / 26.2 | 3.83 / 7.59 | 3 / 32 | 1.0 / 3.2 |

Table 6.6: Results when comparing standard KL and FB KL for several models: VAE, VAE + $M_{IS}$. The left part in each cell shows training with standard KL and the right part shows using FB KL instead.

linearly interpolate between them with evenly divided intervals Bowman et al. (2016).[5] We use greedy decoding in generation.[6] Table 6.7 shows linear interpolation between prior samples in FlowPrior and Vamp-VAE +$M_{IS}$ (i.e., Vamp-VAE training with combined training objective). We observe substantial improvement with FlowPrior, as it can generate sentences with smooth seman-

5. FlowPrior is slightly different. Instead of directly sampling from the latent variable of VAE (in MoG-VAE and Vamp-VAE), FlowPrior samples from the base distribution of real NVP, interpolates in the base distribution, and maps to the latent with Eq. 6.3. We also experiment with interpolating the two samples after mapping, namely interpolating in the VAE latent space, and find similar results.

6. We additionally tried various sampling methods for decoding. This leads to more noise and becomes harder to interpret. Generations can be found in the Appendices.

| **Vamp-VAE** |
|---|
| Three people are sitting on a bench . |
| People are walking down the street . |
| Man in a blue shirt and jeans is sitting on a bench . |
| Man in a blue shirt and jeans is sitting on a bench . |
| Women in a white dress and a man in a black shirt are standing in front of a microphone . |
| Women in a white dress and a man in a black shirt are standing in front of a microphone . |
| two men are playing soccer |
| two men are playing basketball |
| Two men are playing a game of chess . |
| Two men are playing a game of chess . |
| **FlowPrior** |
| The dog is running through the snow . |
| Two young boys are playing in the snow . |
| There is a man in a blue shirt and a woman in a black shirt and black pants . |
| Three people are sitting on a bench . |
| two men are standing on a bench |
| A girl is sitting on a bench . |
| A young girl is sitting on a bench . |
| A young man is sitting on a bench . |
| A woman in a black shirt is sitting on a bench . |
| A woman is sitting on a bench . |

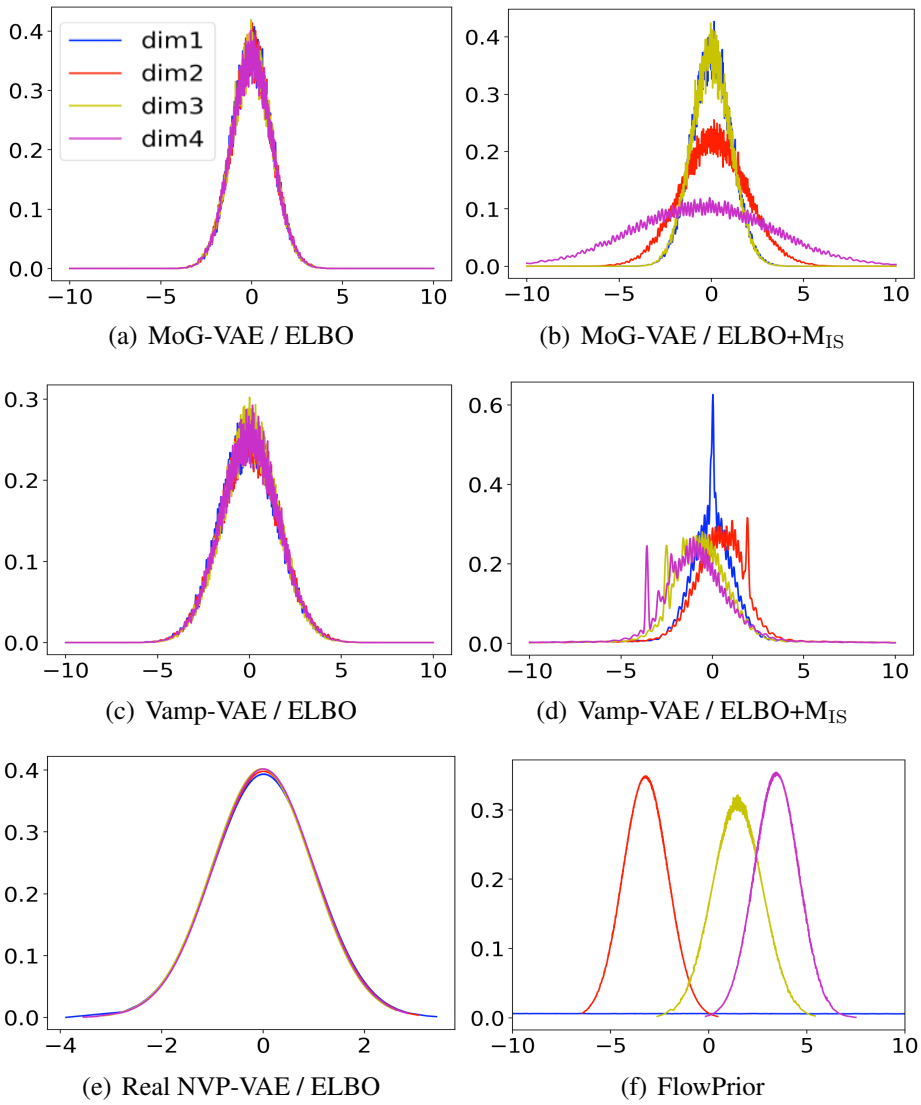
Table 6.7: Interpolation from the prior on SNLI dataset. In each cell, the first and last sentences correspond to two sampled latent codes and between are linearly interpolated samples.

tic evolution while maintaining plausible generations in terms of fluency. This semantic evolution may reflect the complex structure in the learned prior distribution. Interpolations with MoG-VAE + $M_{IS}$ and Vamp-VAE + $M_{IS}$ have more repetitions and do not transit smoothly from one to the other.

### *6.5.3 Visualization of Learned Priors*

We randomly select 4 dimensions from the learned priors per model and plot their densities in Fig. 6.1.

In MoG-VAE, each dimension is a Gaussian mixture with 100 components. When only using




(a) MoG-VAE / ELBO (b) MoG-VAE / ELBO+$M_{IS}$

(c) Vamp-VAE / ELBO (d) Vamp-VAE / ELBO+$M_{IS}$

(e) Real NVP-VAE / ELBO (f) FlowPrior

Figure 6.1: Densities of 4 dimensions of learned priors (SNLI dataset).

the ELBO for training (Fig. 6.1(a)), the four visualized components all have similar shapes. After adding $M_{IS}$ (Fig. 6.1(b)), different dimensions have similar locations but different scales.

Vamp-VAE permits relatively complex components because the means and variances are acquired from the inference network applied to learned pseudo-inputs. Fig. 6.1(c) shows that Vamp-VAE training without $M_{IS}$ does not show much difference compared to MoG-VAE. However, when training with $M_{IS}$ (Fig. 6.1(d)), the distributions in several dimensions appear to be multimodal.

The real NVP prior learns little information when training without $M_{IS}$, as all dimensions are

akin to standard normal distributions. When training with $\mathrm{M}_{\mathrm{IS}}$, different dimensions show distinct placement and shape. The prior in FlowPrior is highly multimodal overall and smooth in each dimension.

### *6.5.4 Generations from Prior Samples*

**Sampling from Prior.** To measure the expressiveness of the prior and the richness of the learned latent variable, we randomly sample 5000 times from the prior distribution and evaluate their greedy-decoded generations qualitatively and quantitatively.

| **MoG-VAE** | **MoG-VAE + $\mathbf{M}_{\mathrm{IS}}$** |
|---|---|
| The man is wearing a black shirt . | An older gentleman in a white shirt is walking in a parking lot . |
| A man is standing in front of a building . | A woman is walking in a field . |
| A man is standing in front of a building . | A young girl in a red shirt is playing with a toy . |
| **Vamp-VAE** | **Vamp-VAE + $\mathbf{M}_{\mathrm{IS}}$** |
| A man is playing a guitar . | Man in a blue shirt and jeans is sitting on a bench . |
| A man is playing a guitar . | The man is wearing a black shirt . |
| A man is playing a guitar . | People are walking down the street . |
| **VAE** | **FlowPrior** |
| A man is sitting on a bench . | Man in a blue shirt and blue jeans is sitting on a rock with a hammer . |
| A man is sitting on a bench . | Two young boys are playing in the snow . |
| A man is sitting on a bench . | A dog is running through the snow . |

Table 6.8: Generations from prior samples with greedy decoding (SNLI dataset)

Table 6.8 shows greedy generations from prior samples. We observe substantial improvements in term of generation diversity when adding $\mathrm{M}_{\mathrm{IS}}$ in the training objective. Note that these diverse samples are achieved with a purely deterministic decoding. A diverse set of samples implies that (1) richer latent codes and a highly multi-modal distribution is learned by the model; (2) and the generator is trained to attend to the latent codes.

| | Yelp | | | SNLI | | |
|---|---|---|---|---|---|---|
| | F-PPL | R-PPL | SB | F-PPL | R-PPL | SB |
| VAE | 4 | 30248 | 96 | 4 | 51127 | 100 |
| VAE+$M_{IS}$ | 5 | 10818 | 30 | 4 | 19047 | 73 |
| Vamp-VAE | 4 | 32504 | 100 | 4 | 56050 | 100 |
| Vamp-VAE+$M_{IS}$ | 7 | 5280 | 10 | 5 | 8420 | 29 |
| FlowPrior | 209 | 1677 | 3 | 42 | 5725 | 13 |

Table 6.9: Forward PPL (F-PPL), Reverse PPL (R-PPL), and Self-BLEU (SB) of greedy-decoded prior samples.

**Sample Mundanity and Coverage.** A strongly-performing generative model should be able to generate samples that comport to the training data distribution. We use the forward and reverse PPL to estimate the similarity between the training data and samples. We can consider F-PPL as a *generation "precision"* as it reflects the amount of information in the samples that is relevant to the actual text. Analogously, we can consider R-PPL as a *generation "recall"* as it reflects how much the samples as a whole provide coverage of the actual text. Moreover, both F-PPL and R-PPL reflect whether the decoder is able to attend to the latent variable in generation.

Table 6.9 shows the F-PPL and R-PPL with greedy generation from prior samples. While Fang et al. (2019) treats a lower F-PPL as an indicator of better samples, we argue that it is not necessarily true. A model could achieve a low F-PPL by simply generating identical (or nearly-identical) high-probability sequences, like those observed from the VAE, MoG-VAE, and Vamp-VAE in Table 6.8. This reflects how an overly-simplified or restrictive assumption in prior can lead to less diversity in samples.

Indeed, we find that models with very low F-PPL values often have very high R-PPL values. A lower R-PPL indicates the distribution of generated samples matches the distribution of the training data. From Table 6.9 we observe that adding $M_{IS}$ is beneficial as it leads to a lower R-PPL. FlowPrior has the best R-PPL score, and shows the capability of capturing characteristics of the target distribution that are not captured by simpler priors.

**Generation Diversity.** To identify which model has richer usage of latent variables, we use self-BLEU to measure the diversity of a set of samples. We observe significant improvements in Flow-Prior in Table 6.9, which implies a diverse latent representation and a better utilization of the latent variable.

## 6.6 Conclusion

We build our model FlowPrior by introduce normalizing flow into the VAE prior, and propose adding the importance-sampled marginal likelihood ($M_{IS}$) as a second term to the standard VAE objective. Our empirical results shows FlowPrior yields a substantial improvement in language modeling tasks and generation tasks.

# CHAPTER 7

# DETECTING GENERATION INCONSISTENCY IN GROUNDED DIALOGUE SYSTEMS

## 7.1 Introduction

Grounded dialogue systems (Shuster et al., 2020; Adiwardana et al., 2020) are often formulated as open-domain controllable generation models that produce diverse responses based on grounded knowledge. With the advent of large-scale language model architectures (Vaswani et al., 2017b; Radford et al., 2019), recent works (Zhang et al., 2020b; Adiwardana et al., 2020; Roller et al., 2021) are capable of generating responses that are highly grammatical, fluent, and human-like. However, studies have shown that the generated responses lack global consistency with the conversation context (Zhang et al., 2019b; Kim et al., 2020; Gao et al., 2020), or contain hallucinated content that contradicts grounded knowledge or world knowledge (Dziri et al., 2021; Honovich et al., 2021). We focus on the latter problem, aiming to automatically identify hallucinated responses generated by grounded dialogue systems.

The prevalence of hallucinated content impedes the real-world deployment of neural dialogue systems, as they might unintentionally create or propagate misinformation resulting in deleterious consequences to human society. With the goal of mitigating imperfect generations, the first step is to detect the inconsistency between generated responses and grounded knowledge. We study document-grounded DialoGPT in Zhang et al. (2020b, 2021) that trained to attend to grounded documents when generating responses. In Table 7.1 we show examples that are fact-inconsistency with the given documents.

Inspired by recent work in hallucination detection in text summarization and neural machine translation (Falke et al., 2019; Kryscinski et al., 2020; Maynez et al., 2020; Wang et al., 2020; Durmus et al., 2020; Pasunuru and Bansal, 2018; Arumae and Liu, 2019; Zhou et al., 2021), we measure the fact-consistency of output sequences using external semantic models for natural lan-

| |
|---|
| **Document**: Following the conclusion of his sophomore year, Iverson declared for the 1996 NBA draft. He was the first player under Coach Thompson to leave Georgetown early for the NBA. |
| **Context**: TIL at the age of 17, Allen Iverson was sentenced to 15 years in prison after participating in a brawl at a bowling alley. He was only released due to a pardon by the Virginia Governor after a Tom Brockaw interview caused a public outcry. |
| **C Resp.**: Iverson was the first player under coach Thompson to leavege early for the NBA. |
| **IC Resp.**: He was the first player under coach Thompson to leave Gonzaga early for the NBA |
| **IC Resp.**: He was also the first player to leave College early to join the NBA. |

Table 7.1: Examples of factual-consistenct (**C**) and factual-inconsistenct (**IC**) responses (**Resp.**) generated by document-grounded DialoGPT conditioned on the same document and context.

guage inference (Dagan et al., 2005b; Bowman et al., 2015b). We note that methods in text summarization can not be directly applied to the dialogue domain due to the open-ended nature in dialogue. For example, new information in dialogue is typically allowed as long as it does not contradict with the conversation history and reference documents. This open-ended nature of dialogue renders the possible generations to be much more diverse than summarization, thus posing new challenges for suppressing confounding issues other than faithfulness. Figure 7.1 shows examples of confounding issues such as blandness of the response and irrelevance when the response is not relevant to the given reference documents. They may be legit in human conversation (Grice, 1975) and agreed by social norms (Grice, 1989), but are not suitable for studying fact-consistency with given reference grounding documents, which is the focus of this paper.

To close this gap, we build an efficient data filtering pipeline and collect a **g**rounded **d**ialogue **b**enchmark (GDB) for evaluating fact-consistency of generated responses. We annotate GDB to indicate whether the generations are fact-consistent with the grounded documents, and their cor-

responding rationale. We create and evaluate baselines with off-the-shelf methods developed for semantic similarity, in-context-learning, and proxy tasks, and further improve the baselines with two different data augmentation approaches based on crafted templates (ADVTEMP) and model generation (GPTAUG).

ADVTEMP relies on manually crafted templates, and synthesizes data via text transformations defined in the templates. Though these text transformations are efficient and effective, the process of template creation requires extensive amount of expertise and effort. Moreover, outputs from transformation lack diversity because templates only inject local noises into the linguistic space with shallow modification to words and phrases. GPTAUG leverages the power of GPT-3 (Brown et al., 2020) to automatically identify patterns with in-context examples of documents, responses, and their relations, and generate new samples conditioned on the specified relations. The generated augmentations follow the data distribution, and are diverse in sentence structure and word use. However, these generation does not always comport with the specified relation. Thus, the labels in the synthesized data pairs need to be relabeled or filtered to be ultimately useful.

Our experiments shows that proxy-task-based method is a strong baseline. Data augmentation improves the performance further. ADVTEMP and GPTAUG tend to capture different aspects of knowledge as we expected. We did error analysis on models with breakdown under types of detection knowledge needed, and found data augmentation performs better than proxy-task-based method in most augmented categories. Our analysis shows the effectiveness and limitations of the two augmentation methods. We found augmented instances from GPTAUG are more diverse in sentence structure than ADVTEMP, however, the original labels from GPTAUG are not as clean as those in ADVTEMP.

To summarize, our contributions in this work are: (1) We propose a data filtering pipeline to facilitate efficient data collection under grounded dialogue domain. (2) We introduce a **g**rounded **d**ialogue **b**enchmark (GDB) with crowdsource annotation of consistency label and and types of knowledge required for prediction. (3) We establish baselines with a range of off-the-shelf methods

and further improve the performance with two different augmentation approaches.

## 7.2 Related Work

### *7.2.1 Hallucination Detection*

Generations from large-scale pretrained language models are fluent but problematic in presenting faithful content. Recent works (Falke et al., 2019; Kryscinski et al., 2020; Maynez et al., 2020; Wang et al., 2020; Durmus et al., 2020; Pasunuru and Bansal, 2018; Arumae and Liu, 2019; Zhou et al., 2021) attempt to quantify this issue in abstractive summarization. Some work (Falke et al., 2019; Maynez et al., 2020; Kryscinski et al., 2020) frames the hallucination detection problem as a textual entailment task (Dagan et al., 2005b) and uses BERT-based models (Devlin et al., 2019a) fine-tuned on MultiNLI (Williams et al., 2018) to verify the factual consistency of the generated summaries. Other works (Wang et al., 2020; Durmus et al., 2020) verify faithfulness by employing a question answering (QA) modules to identify whether similar answers are produced by a QA system from the source document and the summary.

Recently, in the dialogue domain, Gupta et al. (2021b) introduce a fact verification benchmark focusing on evidence retrieval and claim verification. Honovich et al. (2021) measure faithful consistency with automatic question generation and question answering. Dziri et al. (2021) introduce a benchmark to measure and detect consistency between generation and grounding. Similarly, Dziri et al. (2021) they leverage the MNLI dataset as part of its training set, and synthesize training data with templates. Our work differs from Dziri et al. (2021) in 3 ways: (1) in addition to templatic transformation, we leverage GPT-3 in-context learning to generate diverse in-domain samples. Doing so eliminates the efforts in template creation. (2) instead of separating confounding instances with crowdsource annotations, we build a data filtering pipeline that automatically alleviating confounding factors. (3) we analyze data with minimum types of knowledge needed for detection to shed light on future research.

### *7.2.2 Synthetic Datasets*

Synthetic natural language datasets have been used for assessing model robustness towards adversarial attacks (Ebrahimi et al., 2018b; Wallace et al., 2019; Pruthi et al., 2019) and improving generalizability in model training (Wei and Zou, 2019). The creation procedure of these datasets comes in two flavors: directly manipulating text to inject noise and using neural networks to generate new data while maintaining consistency to the desired label.

Common text manipulation approaches include edit-based changes like replacement, insertion, swap-within-sentence, and deletion (Ebrahimi et al., 2018b; Pruthi et al., 2019; Wei and Zou, 2019), entity substitution (Jiang et al., 2020), word replacement Alzantot et al. (2018), and syntactic transformations (Kryscinski et al., 2020; Dziri et al., 2021).

Neural network approaches include generating paraphrased data (Sennrich et al., 2016; Fadaee et al., 2017; Iyyer et al., 2018), inserting words via natural language generation (Zhang et al., 2019a; Gupta et al., 2021a), and finetuning pretrained language models (Wu et al., 2019; Kumar et al., 2020; Anaby-Tavor et al., 2020; Lee et al., 2021a).

More recently, GPT-3 has been used to generate training data. Sahu et al. (2022) uses GPT-3 to generate labeled training data for intent classification. Yoo et al. (2021) considers distilling knowledge from GPT-3 with synthetic data. Wang et al. (2021) employs GPT-3 to label unannotated data.

## 7.3 Grounded Dialogue Fact-Inconsistency Detection

The goal of knowledge-grounded dialogue systems is to generate *responses* that fit into the *context* (Jurafsky and Martin, 2021) and employ *knowledge* from structured knowledge bases (Moon et al., 2019; Tuan et al., 2019) or unstructured documents (Dinan et al., 2018; Feng et al., 2020). We define the "context" as the sequence of turns in the dialogue prior to the utterance to be generated by a system. We define the "knowledge" as one specific document called the *grounded document*. The "response" is the output generated by a system when given the context and grounded document.

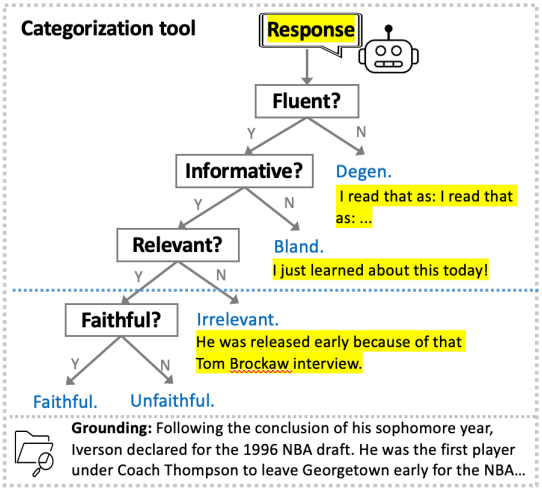



Figure 7.1: Visualization of response categorization tool. Degenerated (*Degen.*), *Bland.*, and *Irrelevant.* do not employ information in the grounding. These are confounding factors need to be removed when determining factual-consistency. In this example the generated responses are conditioned on the same grounding and context as those in Table 7.1.

We assume that the grounded document is always present and is consistent with the context. The task of grounded dialogue hallucination detection is to predict whether the generated response is consistent with the grounded document. It is challenging due to the open-ended nature of dialogue and the scarcity of currently available data for the hallucination detection task.

Figure 7.1 and Table 7.1 show generated responses from the grounded dialogue system. A large proportion of these generations include confounding factors that are neither factual-consistency nor factual-inconsistency. To avoid spending annotation power, we build a data pipeline to automatically filter out those generations, then feed the remaining instances to crowdsource platform.

### *7.3.1 Response Generation*

In particular, we investigate the system of document-grounded DialoGPT in (Zhang et al., 2021). It is initialized from DialoGPT (Zhang et al., 2020b) which has the similar model architecture as GPT-2 (Radford et al., 2019) and trained on dataset extracted from DSTC-7 grounded response generation challenge (Galley et al., 2019). In the training set, each instance is a dialogue session, consisting of a context, a response and an oracle document.

Following Zhang et al. (2021), we generate 16 hypotheses with top-10 sampling for each document and dialogue context and collect the top-ranked one with Maximum Mutual Information (MMI; Li et al., 2016b). This encourages sample diversity while ensuring generations are less likely to go off-topic.

### *7.3.2 Data Filter Pipeline*

To reduce the impact of other confounding factors and focus on the fact-consistency issue, we build a data pipeline (Figure 7.1) based on quantitative metrics. It follows a hierarchical structure and categorizes responses into four classes: *Degen.*, *Bland.*, *Irrelevant.*, and *Relevant.*. The workflow of the data pipeline is as follow: It first checks the fluency of the response. if it is fluent, then it determines whether the response is informative, and, if so, whether it is relevant. We only collect responses that are fluent, informative, and relevant. The corresponding quantitative metrics are:

- **Fluent**: It has been shown that GPT-style models and standard decoding can lead to repetitive outputs (Holtzman et al., 2020). We use *portion of unique n-grams* (Welleck et al., 2019) on generation $s$ to measure fluency and filter out degenerate instances:

$$p_{\text{uniq-n-grams}}(s) = \frac{|\text{uniq-n-grams(s)}|}{|\text{n-grams}(s)|} \tag{7.1}$$

- **Informative**: We consider a response to be informative if it is specific under the given pair of grounded document and context. We use *n-gram entropy* to measure the informativeness of a response $s$ and filter the bland response that contain a large proportion of frequent tokens. The n-gram entropy is calculated as

$$\text{ent-n-grams}(s) = -\frac{\sum_{x \in \text{n-grams(s)}} \log p(x)}{|\text{n-grams(s)}|} \tag{7.2}$$

where $p(x)$ is the empirical n-gram distribution estimated from gold standard responses in

document-grounded DialoGPT training set. A rare n-gram that yields a higher entropy is considered as a more informative phrase.

- **Relevant**: A response is considered relevant to the grounded document if there is overlapping information between. Here we use ROUGE-2 recall (Lin and Hovy, 2003) and sentence embedding cosine similarity between the generated response and grounded document to measure relevance. For the latter, sentence embedding is acquired from SentenceBERT (Reimers and Gurevych, 2019).[1]

### *7.3.3 Data Annotation*

We solicit annotations via Amazon Mechanical Turk on the remaining generations (i.e.,: the relevant response) from our data filtering pipeline. We ask annotators to select one choice from three candidates: 'Irrelevant', 'Relevant but contradicted', and 'Relevant and consistent'. As a tutorial for annotators, we provide several labeled examples with explanations. To ensure annotation quality, the hiring criteria for workers are having at least 5,000 HITS done with an acceptance rate of 90% and above. In addition, we perform annotations in batches, and examine samples in each batch and exclude workers with poor annotation quality from future work. For each annotation instance, we assign three different annotators. An annotation is valid if the label is agreed upon by all annotators.

### *7.3.4 Data Statistics*

We remove instances labeled as 'Irrelevant' and create a testbed which has 1002 instances in total. We name it as **g**rounded **d**ialogue **b**enchmark (GDB). GDB is a binary classification dataset, with labels of fact-inconsistent (i.e.,: the 'Relevant but contradicted' class), and fact-consistent (i.e.,: the 'Relevant and consistent' class), among which 500 instances are DEV and 502 instances are

1. We use the HuggingFace model `bert-base-nli-mean-tokens`.

TEST. In DEV, 301 are fact-consistency and 199 are fact-inconsistency, while in TEST 297 are fact-consistency and 205 are fact-inconsistency.

To better understand generation issues and explore the *minimum types of knowledge needed* to build a hallucination detector, we annotate 200 instances in GDB. Table 7.2,7.3 shows the breakdown under categorization of knowledge, which has 6 classes: entity swap (*Ent*), replacement (*Replace*), negation or self-contradiction (*Neg/Con*), commentary (*Comn*), information extraction and paraphrasing (*Xtract*), and new information introduction (*NewInf*). It differs from hallucination detection under summarization, as in summarization (Kryscinski et al., 2020; Maynez et al., 2020) the dominant types are Ent, Replace, Neg/Con, and Xtract, here in grounded dialogue Comn and NewInf are two additional common types.

## 7.4 Baselines

we establish several baselines on GDB with a range of off-the-shelf methods.

### *7.4.1 Score-based Methods*

We examine four score-based methods using off-the-shelf models developed for semantic similarity. These models outputs a score given a pair of document and response. A higher score indicates higher lexical and/or semantic similarity, and is more likely to be factual-consistent. For scores that are not normalized, we linearly re-scaled them with their empirical lower bound $l$ and upper bound $h$ calculated on DEV:

$$\text{score}^* = \frac{\text{score} - l}{h - l} \tag{7.3}$$

The intention is to standardize the outputs for unified evaluation criterion. The score-based baselines include:

**MMI** (Li et al., 2016b) measures the mutual information between the generated response $r$ and grounded document $d$. The score is the sum of the forward NLL $-\log p(r \mid d)$ and the reverse

NLL $-\log p(d \mid r)$. Forward NLL is acquired from grounded DialoGPT, while the reverse NLL is from a similar model that is trained to generate the document conditioned on the response on the same dataset.

**SBERT** (Reimers and Gurevych, 2019) yields fixed-sized vectors for sentences, which are used in calculating sentence similarities. The score here is the cosine similarity between the embeddings of responses and grounded documents.

**BERTScore** (Zhang et al., 2020a) computes the score of two sentences as a sum of cosine similarities between their tokens' embeddings. Compared to SBERT, BERTScore takes token alignments into account. **BERTScore-R** computes recall by matching each token in the document to its generated response. **BERTScore-P** computes precision by matching each token in the response to tokens in the document. **BERTScore-F1** combines BERTScore-P and BERTScore-R.

**BLEURT** (Sellam et al., 2020) is a BERT-based metric that benefits from training on a synthetic dataset. It is trained under the supervision of a collection of quantitative metrics including BLEU, ROUGE, BERTScore, backtransation likelihood, backtranslation flag, and textual entailment.

### *7.4.2 In-Context Learning*

**GPT-3** (Brown et al., 2020) is a large-scale transformer language model trained on a large text corpus. Recent studies demonstrate GPT-3's ability to perform *in-context learning*, i.e., GPT-3 learns a downstream task based on several examples in its context and predicts outputs on a new input via sequence generation. Here, we feed GPT-3 10 randomly selected instances (5 per class) from DEV as examples, and let the model generate labels for new inputs.

### *7.4.3 Proxy-task-based Methods*

We also consider applying models trained on other natural language understanding benchmark datasets to the hallucination detection task. We use the textual entailment model and claim verification model in measuring factual-consistency between document and response to examine whether

these these models can transfer to the dialogue hallucination detection. In particular, we use models traine on MNLI (Williams et al., 2018) and FEVER datasets (Thorne et al., 2018). MNLI is a Natural Language Inference (NLI) dataset that judges the relationship between a sentence pair (i.e.: the *premise* and the *hypothesis*) by picking a label from 'NEUTRAL', 'ENTAILMENT', and 'CONTRADICTION'. FEVER is a dataset that requires to verify a *claim* given the Wikipedia pages the claim is extracted or altered from. It has three classes: 'SUPPORTED', 'REFUTED' or 'NOTENOUGHINFO'. We treat the premises and Wikipedia pages as grounding, and the hypotheses and claims as responses. We remove instances labeled as 'NEUTRAL' and 'NOT ENOUGH INFO', and treat 'ENTAILMENT' and 'SUPPORT' as fact-consistency, and 'CONTRADICTION' and 'REFUTE' as fact-inconsistency.

## 7.5 Synthetic Training Data

We hypothesize that the baseline methods described above may have a data distribution mismatch issue, and a lack of the minimum knowledge needed to make correct predictions. We consider approaches that can automatically synthesize data from grounded documents at a marginal cost. The goal is to synthesize data that carries information beneficial for dialogue hallucination detection. In particular, we are interested in adversarial instances that are challenging to the strongest baselines.

Figure 7.2 shows the augmentation procedure. We explore two approaches. The first forms templates and then transforms documents to synthesize responses. The other directly generates responses using GPT-3 with in-context examples. The coverage of knowledge types is shown in Table 7.4.

### *7.5.1 Adversarial Template-based Data Augmentation (*ADVTEMP*)*

This approach augment data in an *adversarial* manner because the template is created by identifying patterns that are challenging for current best-performed model. We examine instances on

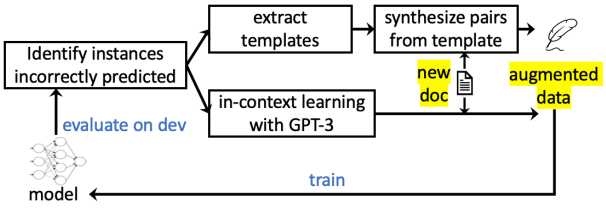



Figure 7.2: Data Augmentation Pipeline

DEV that fail in the strongest baseline (i.e., textual-entailment-based method, FEVER+MNLI in Table 7.5) and form templates generalized to the findings on those instances.

Concretely, we take one sentence $s$ from a document $d$ as a faithful *base pair*. The transformation is applied by altering words in $s$ following template rules. The details of templates are:

- EnSwp: A transformation can be applied when there are at least two distinct entities under the same mention NER tag in $d$, and at least one of these entities is in $s$. We replace the entity in $s$ with another one.[2] The transformation forms unfaithful pairs.

- Quan: The transformation is similar to EnSwp, except template works on entities labeled as number-related tags. The transformation yields unfaithful pairs.

- Neg: This transformation forms unfaithful pairs. The template looks for an auxiliary verb (e.g.:, can, could, need, be ) in $s$. For verbs that appear with the word "not", we remove the "not". Otherwise, we add "not" after the auxiliary verb.

- Xtract: We directly use base pairs. These are faithful instances.

There are two main limitations of ADVTEMP: (1) Templates are designed to be generalized, which is at a cost of flexibility in sentence structures and word use. (2) Template creation requires human efforts and needs to be constantly updated based on new observations. In the next section, we explore a neural-network-based method that learns transformation via in-context examples and generates diverse pairs.

2. Entities are extracted by running SpaCy NER tagger (Honnibal and Montani, 2017) which labels entities with tags of Person, Location, Organization, Product, Event, Quantity, Date, Time, Percent, etc.

### 7.5.2 *GPT-3 Data Augmentation (*GPTAUG*)*

We use GPT-3 to generate instances with desired features via choice of in-context examples. These examples are randomly selected from DEV instances that are misclassified by the strongest baseline. They cover a comprehensive set of knowledge types as shown in Table 7.4.

GPT-3 is known to be highly sensitive to the choice of prompt (Reynolds and McDonell, 2021; Zhao et al., 2021b). Our prompt is composed of meta-information as guidance and in-context examples. We found that using triplets of document, response, and label as examples, and generating responses conditioned on a new document and a specified label is challenging for GPT-3. While GPT-3 can generate in-distribution and fluent faithful instances, it often fails in generating unfaithful ones by ignoring the provided label and simply extracting information from the document. We speculate that forming unfaithful pairs directly might be too challenging for GPT-3.

To alleviate the issue, we divide the process of generating unfaithful responses into two steps. We first generate a faithful response given a document, then generate an unfaithful response conditioned on both the generated faithful response and the document. In addition, to ensure the quality of the augmented pair, we use a simple yet effective method which labels these generations with the strongest baseline. This labeling process, similar to pseudo-labeling (Lee et al., 2013; Shi et al., 2018) or self-labeling (Yarowsky, 1995; McClosky et al., 2006), has been shown effective in training better models with samples filtered on labels using previously trained models.

## 7.6 Experiments

### 7.6.1 *Experimental Setup*

We measure the performance of score-based and in-context learning baselines on GDB. More model and parameter details can be found in Appendix. We had separate experiments evaluating models fine-tuned on MNLI, FEVER, ADVTEMP, and GPTAUG. The statistic of these dataset is in Appendix.

In Proxy-task-based method, we combine the training set of MNLI and FEVER. In Data augmentation method, we combine data from ADVTEMP and GPTAUG. For ADVTEMP, we combine the training set of MNLI, FEVER, and the templatic augmented data. For GPTAUG, we use models trained on MNLI and FEVER to psuedolabel the augmented data from GPT-3.

On these datasets we finetune RoBERTa-large with a training batch size of 16. We run 60,000 iterations for each setting, using the Adam optimizer (Kingma and Ba, 2015b) with a learning rate of $1e-6$. PyTorch and RTX GPUs are used in the experiments.

### *7.6.2 Evaluation Metrics*

We take AUC and F1 as evaluation metrics that provide comprehensive measurements on binary classifiers. The classification thresholds are located by picking values that lead to the largest geometric means, which seeks to balance between the sensitivity and the specificity of the model:

$$\text{g\_means} = \sqrt{(\text{TPR} * (1 - \text{FPR}))} \tag{7.4}$$

where TPR is classifier's true-positive rate and FPR is the false-positive rate.

For classification using in-context learning, we conducted experiments 10 times with prompts that were randomly sampled from DEV. For methods that involve training neural networks, we run experiments with 10 different random seeds and use early stopping on DEV. The classification thresholds are selected based on predictions on DEV.

### *7.6.3 Results*

Table 7.5 compares different methods on DEV and TEST. Among all the score-based methods, BERTScore-P which estimates the alignment from the generated response to the document performs the best. GPT-3 has undesirable performance, as it always predicts the sentence pair as faithful no matter what combination of instances we had in the prompt. The reason could be that

similar data or tasks are not prevalent in the GPT-3 pretraining data, and in-context learning with few examples are not enough. The subtle differences between sentence pairs are challenging to be identified. We observe proxy-task-based method is a strong baseline on the dataset combining MNLI and FEVER, as it shows significant improvement compared to score-based method. Augmentation-based method has the strongest and stablest performance with the highest AUC and F1 score, and lowest standard deviations.

## 7.7 Analysis

### *7.7.1 Model Ablation*

Table 7.6 shows the ablation on proxy-task-based methods and data augmentation methods. We found that though the performance of classifiers trained on MNLI-only and on FEVER-only have comparable performance, they tend to capture different aspects of knowledge, as a better performance is achieved when training on the combination of two. Similarly, the performance of two augmentation methods are comparable, and the performance increases when combining the two. We also found that labeling process is essential to GPTAUG as the noisy label largely degrades the power of the augmented data.

### *7.7.2 Error Analysis*

Table 7.7 shows a performance breakdown according to categories of knowledge needed. The augmentation methods outperform the proxy-task-based methods in all classes except the category of NewInf. NewInf is challenging for all methods, which suggests future work could improve the performance by utilizing external knowledge and/or performing better understanding and reasoning. In Xtract category, the augmentation methods do not show much of an advantage over proxy-task-based ones. This is likely because the MNLI and FEVER datasets contain abundant entailed or paraphrased instances. In the Comn category, instances that introduce filler information

and subjective comments are very likely to be predicted as fact-inconsistent by proxy-task-based methods due to domain mismatch, so using the augmentation methods yields better performance.

### *7.7.3 Qualitative Analysis on Synthesized Data*

We conduct qualitative analysis on data synthesized from ADVTEMP and GPTAUG. Table 7.8 shows several representative examples. Unsurprisingly, samples from ADVTEMP are less diverse in sentence structure than those from GPTAUG. For example, augmented response 3 from GPTAUG is paraphrased from the document instead of directly copying the sentence and augmented response 4 combines material from multiple sentences and alters a single word to make the utterance factual-inconsistent. ADVTEMP has better quality in labels because the transformation is strictly constrained by conditions and rules. However, there are still exceptions which indicates the importance of taking care of edge cases. For example, augmented response 4 from ADVTEMP does not form a contradicted pair, though it is problematic as it repeats 'Bulgaria' when listing countries, after swapping entities.

## 7.8 Conclusion

In this paper, we explore methods in detecting fact-inconsistency in the domain of grounded dialogue system. It is challenging due to the dialogue's open-ended natural. We build a data filtering pipeline that helps us alleviate confounding factors and focus on the high-quality responses. We collect a dataset GDB for better understanding the fact-inconsistency issues. We improve the performance on GDB with two data augmentation methods.

| Category | Type | Description | Count IC | Count C | Dev % | Test % | Examples (D = Document; R = Response) |
|---|---|---|---|---|---|---|---|
| **EnSwp** | | Swap one entity with another in document | 32 | 0 | 17 | 15 | **D**: The last confirmed wild bird is thought to have been shot in 1901... Martha, thought to be the last passenger pigeon, died on September 1, 1914, at the Cincinnati Zoo.<br>**R**: The last confirmed wild bird died on September 1, 1914 at the Cincinnati Zoo... |
| **Replace** | **Ent** | Replace entity with another in vocabulary | 7 | 0 | 3 | 4 | **D**: ...The steel was melted down at Amite Foundry and Machine in Amite, Louisiana, to cast the ship's bow section...<br>**R**: The steel was melted down at Amite Wellness and Machine in Amite Louisiana to cast the ship's bow section. |
| | **Desc** | Change description of entity | 4 | 0 | 2 | 2 | **D**: ...In 1982, Lundgren graduated with a master's degree in chemical engineering ...<br>**R**: Dolph Lundgren has an open degree in chemical engineering ... |
| | **Verb** | Replace verb phrase | 5 | 0 | 2 | 3 | **D**: Commander James Bond, CMG, RNVR, is a fictional character created by the British journalist and novelist Ian Fleming in 1953.<br>**R**: James Bond is named after the author Ian Fleming who wrote the novels. |
| | **Quan** | Replace quantity | 14 | 0 | 8 | 6 | **D**: Ronald Wilson Reagan (February 6, 1911 - June 5, 2004) was an American statesman who served as the 40th President of the United States from 1981 to 1989.<br>**R**: Ronald Wilson Reagan, the man responsible for saving a lot of lives, was the 40th President of the United States from 1989 to 1993. |
| **Neg/Con** | **Neg** | Negate or replace with antonym | 7 | 0 | 3 | 4 | **D**: ...Although popular since the 19th century, the day is not a public holiday in any country.<br>**R**: In France, April fools day ends at 9pm, and April fools day is a public holiday. |
| | **Con** | Self-contradiction | 4 | 0 | 2 | 2 | **D**: ... Indian immigrants to Africa, particularly in South Africa, brought vegetarianism with them ...<br>**R**: India has a high number of vegetarians, but a low number of vegetarians. |

Table 7.2: Annotation of 200 instances in categories according to the minimal types of knowledge needed for hallucination detection, along with examples showing snippets of documents (D) and responses (R). Some categories have multiple subtypes, e.g., the Replace category includes replacements of entities, descriptions, verbs, or quantities. We show the number of factual-inconsistency (IC) and factual-consistency (C) instances for each type, and their percentages in DEV and TEST. Some instances are associated with multiple types, so the sum of counts is greater than 200.

| Category | Type | Description | Count IC | Count C | Dev % | Test % | Examples (D = Document; R = Response) |
|---|---|---|---|---|---|---|---|
| **Comn** | | Add subjective comments or filler information | 0 | 40 | 23 | 17 | **D**: Nintendocore (also known as Nintendo rock, nerdcore, and video game rock) is a broadly defined music genre that fuses chiptune and video game music ...<br>**R**: I've been listening to Nintendocore and I can say that the music fits the genre perfectly. |
| **Xtract** | | Extract or paraphrase information from document | 0 | 38 | 20 | 18 | **D**: In 2017 he met the Queen of Denmark in Amalienborg Palace.<br>**R**: He met the Queen of Denmark as well. |
| **NewInf** | **Reason** | Require reasoning | 8 | 3 | 8 | 3 | **D**: ... The movie depicts Kroc's franchise development, nationwide expansion, and ultimate acquisition of McDonald's, while being critical of his treatment of the founding McDonald's brothers.<br>**R**: Kroc's founder, John Kroc, founded McDonald's... |
| | **Xternal** | Need external knowledge | 31 | 15 | 17 | 29 | **D**: In March 2015, Davidson was a roaster on the Comedy Central Roast of Justin Bieber ... Davidson, whose firefighter father died during the September 11 attacks...<br>**R**: TIL that Pete Davidson is the son of the Roaster at the Roast of Justin Bieber... |

Table 7.3: Annotation of 200 instances in categories according to the minimal types of knowledge needed for hallucination detection, along with examples showing snippets of documents (D) and responses (R). Some categories have multiple subtypes, e.g., the Replace category includes replacements of entities, descriptions, verbs, or quantities. We show the number of factual-inconsistency (IC) and factual-consistency (C) instances for each type, and their percentages in DEV and TEST. Some instances are associated with multiple types, so the sum of counts is greater than 200.

| type | ADVTEMP | GPTAUG |
|---|---|---|
| **EnSwp** | ✓ | ✓ |
| **Ent** | | ✓ |
| **Desc** | | ✓ |
| **Verb** | | ✓ |
| **Quan** | ✓ | ✓ |
| **Neg** | ✓ | ✓ |
| **Con** | | |
| **Comn** | | ✓ |
| **Xtract** | ✓ | ✓ |
| **Reason** | | |
| **Xternal** | | ✓ |

Table 7.4: Coverage of knowledge types in two data augmentation methods.

| | | TEST | | DEV | |
|---|---|---|---|---|---|
| | | AUC | F1 | AUC | F1 |
| Score-based | MMI | .547 | .411 | .516 | .411 |
| | SBERT | .531 | .408 | .550 | .472 |
| | BERTScore-F1 | .600 | .507 | .523 | .451 |
| | BERTScore-R | .591 | .514 | .518 | .464 |
| | BERTScore-P | .605 | .569 | .530 | .532 |
| | BLEURT | .590 | .547 | .517 | .518 |
| In-context Learning | GPT-3 | - | 0 | - | 0 |
| Proxy-task-based | FEVER+MNLI | .772 (.010) | .667 (.015) | .757 (.008) | .656 (.011) |
| Data augmentation | ADVTEMP + GPTAUG | **.795 (.003)** | **.671 (.010)** | **.799 (.004)** | **.683 (.004)** |

Table 7.5: Comparison of various methods on DEV and TEST. We report the median and standard deviation of performance metrics.

| | TEST AUC | DEV AUC |
|---|---|---|
| MNLI-only | .763 (.011) | .759 (.007) |
| FEVER-only | .763 (.027) | .724 (.010) |
| ADVTEMP only | .780 (.010) | .797 (.009) |
| - w/o MNLI & FEVER | .630 (.012) | .565 (.007) |
| GPTAUG only | .780 (.009) | .795 (.008) |
| - w/o label process | .763 (.017) | .730 (.010) |

Table 7.6: Ablation of proxy-task-based method and data augmentation method. For proxy-task-based method, we show results only have FEVER or MNLI dataset. For data argumentation methods, we show results of ADVTEMP only, GPTAUG only, and their performance before ensembling with MNLI and FEVER.

| | Label | Augment | Proxy | GPTAUG | ADVTEMP | MNLI | FEVER | BERTScore-P |
|---|---|---|---|---|---|---|---|---|
| EnSwp | IC | **74** | 52 | 61 | 61 | 35 | 39 | 48 |
| Replace | IC | **68** | 56 | **68** | 56 | 48 | 48 | 40 |
| Neg/Con | IC | **82** | 73 | 73 | 82 | 68 | 73 | 64 |
| Comn | C | **71** | 59 | 65 | 53 | 32 | 32 | 24 |
| Xtract | C | 84 | 82 | **86** | 73 | 82 | 73 | 55 |
| NewInf | mixed | 56 | 56 | 62 | 56 | 60 | 56 | **64** |

Table 7.7: Comparison of prediction accuracy under categories of detecting skills.

| # | Document | Augmented Response | Type |
|---|---|---|---|
| 1 | ... Within weeks there will be graffiti and vandalism ... | Within weeks there will not be graffiti and vandalism... | Neg |
| 2 | ...iOS 7 is now 64 - bit compatible . All of the included apps run in 64 - bit , and Apple says developers will have a ‘ seamless ’ transition to port over to 64 - bit apps . | All of the included apps run in 7 - bit , and Apple says developers will have a ‘ seamless ’ transition to port over to 64 - bit apps . | Quan |
| 3 | ‘So Danny Ainge drafts De ’ Aaron Fox or Josh Jackson 3rd this year , signs Gordon Hayward this offseason , and then trades for Anthony Davis at the deadline . Hand him the Executive of the Year award now | ‘So Danny Ainge drafts De ’ Aaron Fox or Josh Jackson 3rd this year , signs Anthony Davis this offseason , and then trades for Anthony Davis at the deadline | EnSwp |
| 4 | Gasiorowski identified Bulgaria , Cuba , Czechoslovakia , East Germany , Hungary , Mongolia Poland , North Vietnam , and Rumania as ‘ strong client states ’ of the Soviet Union , and Afghanistan Egypt , Guinea , North Korea , Somalia , and Syria as moderately important client states ... | Gasiorowski identified Bulgaria , Cuba , Czechoslovakia , Bulgaria , Hungary , Mongolia Poland , North Vietnam , and Rumania as ‘ strong client states ’ of the Soviet Union ... | EnSwp |
| 1 | The film was dubbed into English by Walt Disney Pictures , under the supervision of Pixar’s John Lasseter ... | The film was dubbed into English by Pixar, under the supervision of Walt Disney Pictures. | EnSwp |
| 2 | Windshield wiper replacement blades are measured in inches . | Windshield wiper replacement blades are measured in millimeters. | Desc |
| 3 | In the meantime , you can’t go wrong with Arkham . While Arkham Knight had its fair share of hiccups , Arkham Asylum and Arkham City remain two of the best superhero games ever made . | In the meantime, you can’t go wrong with Arkham Asylum and Arkham City. Both games are two of the best superhero games ever made. | Para |
| 4 | To the guy who said Hines Ward hasnt been any good since 2004 … … He was the Super bowl MVP 2006 . | Hines Ward was the Super bowl MVP 2004. | Quan |
| 5 | Roman concrete , like any concrete , consists of an aggregate and hydraulic mortar – a binder mixed with water that hardens over time ... | The concrete used in construction is a mixture of cement and water. Cement is a binder that sets and hardens over time... | Ent |
| 6 | ... One look at that long , naked appendage and people instantly think “ rat , ” an association not to the opossum ’ s advantage . Yet the tail itself is much to the opossum ’ s advantage , serving as a veritable fifth limb . | Their tail is naked and long. They use it for balance while climbing. | NewInf |

Table 7.8: Synthesized data from AdvTemp (upper) and GptAug (lower).

# CHAPTER 8

# FUTURE WORK

**Faithful Grounded Dialogue Response Generation.** In prior work, we found top-ranked generations with high informativeness have the propensity to "hallucinate" facts and we attempt to boost accuracy of generation with a reranking module to model outputs. Albeit it is flexible due to being model-agnostic, there is a scant guarantee of faithfulness when the generator fails to attend to grounded knowledge or yield reasonable word distributions. Potential future work includes employing controllable frameworks and altering training objectives that correct the aforementioned undesirable behaviors.

**Robustness Test Benchmarks.** One direction of my past research has focused on innovating models and learning methods for robust classifiers. In empirical characterization we noticed a standard test suite that covers different robustness aspects is in demand. With one of our completed works targeting an adversarial aspect (i.e., data contains certain biases that prevailing models failed to generalize) in sentence relation inference (Liu et al., 2020), we are interested in continuing to work on other dimensions.

**Robustness Failure Identification** Existing methods in identifying robustness failure heavily rely on human knowledge and efforts where the general process is to perform error analysis on existing systems and connect the predicted results with human priors. Future work can mimic how humans analyze the data and build (semi-)automatic analysis tools that help to identify the robustness failure comprehensively. This does not only reduce the extensive amount effort in the current identification process but also helps mitigate the subjective biases in human analysis.

# CHAPTER 9

# CONCLUSION

Motivated by the long term goal to improve NLP model robustness, we have explored and improve model robustness from modeling, learning algorithms, and application side. In this thesis we packaged my findings from data-efficient, resilient, fair, trusted perspective in model robustness. We outline findings from five separate-but-related research expeditions in the following.

In Chapter 2, we discussed that 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. 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. 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.

In Chapter 3, we propose GenNLI, a generative classifier for NLWe tasks, and empirically characterize its performance by comparing it to five baselines, including discriminative models and large-scale pretrained language representation models like BERT. We explore training objectives for discriminative fine-tuning of our generative classifiers, showing improvements over log loss fine-tuning from prior work (Lewis and Fan, 2019). In particular, we find strong results with a simple unbounded modification to log loss, which we call the "infinilog loss". My experiments show that GenNLI outperforms both discriminative and pretrained baselines across several challenging NLI experimental settings, including small training sets, imbalanced label distributions, and label noise.

In chapter 4, we propose a simple but effective modeling framework for controlled generation

of multiple, diverse outputs. We focus on the setting of generating the next sentence of a story given its context. As controllable dimensions, we consider several sentence attributes, including length, sentiment, automatically-induced clusters, predicates, and frame semantic representations. Our empirical results demonstrate: (1) our framework is accurate in terms of generating outputs that match the target control attribute values;

In chapter 5, we focus on variations that learn expressive prior distributions over the latent variable. We find that existing training strategies are not effective for learning rich priors, so we add the importance-sampled log marginal likelihood as a second term to the standard VAE objective to help when learning the prior. Doing so improves results for all priors evaluated, including a novel choice for sentence VAEs based on normalizing flows (NF). Priors parameterized with NF are no longer constrained to a specific distribution family, allowing a more flexible way to encode the data distribution. Our model, which we call FlowPrior, shows a substantial improvement in language modeling tasks compared to strong baselines. We demonstrate that FlowPrior learns an expressive prior with analysis and several forms of evaluation involving generation.

In chapter 6, we introduce a small-scale benchmark with a semi-automatic data collection pipeline, and establish baselines to better understand the inconsistent issue under the domain of grounded dialogue. We propose a fine-grained categorization of detection knowledge and manually annotate our benchmark under this category. Doing so helps in understanding the behavior of predictors under different models. We hope this can foster future work on inconsistency detection and faithful generation. We improve baselines with two different augmentation approaches. In particular, we validate the effectiveness and observe the limitations of prompt-based data augmentation with the power of GPT-3.

I envision that the work presented in this thesis, when combined with additional work in related areas, will lead to the construction of robust NLP systems in the future.

# References


Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Felipe Alfaro, Marta R. Costa-jussà, and José A. R. Fonollosa. 2019. BERT masked language modeling for co-reference resolution. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 76–81, Florence, Italy. Association for Computational Linguistics.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Hadi Amiri, Timothy Miller, and Guergana Savova. 2018. Spotting spurious data with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2006–2016, New Orleans, Louisiana. Association for Computational Linguistics.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7383–7390.

Kristjan Arumae and Fei Liu. 2019. Guiding extractive summarization with question-answering rewards. 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 2566–2577, Minneapolis, Minnesota. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *COLING-ACL '98: Proceedings of the Conference*, pages 86–90, Montreal, Canada.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. 2016. Density modeling of images using a generalized normalization transformation. In *International Conference on Learning Representations*.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1119–1130, Austin, Texas. Association for Computational Linguistics.

Shikha Bordia and Samuel R. Bowman. 2019. Identifying and reducing gender bias in word-level language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 7–15, Minneapolis, Minnesota. Association for Computational Linguistics.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. 2017. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PloS one*, 12(6):e0177678.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015b. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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

Samuel Carton, Qiaozhu Mei, and Paul Resnick. 2018. Extractive adversarial networks: High-recall explanations for identifying personal attacks in social media posts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3497–3507, Brussels, Belgium. Association for Computational Linguistics.

Seymour Benjamin Chatman. 1980. *Story and discourse: Narrative structure in fiction and film*. Cornell University Press.

Snigdha Chaturvedi, Haoruo Peng, and Dan Roth. 2017. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. 2018. Variational sequential labelers for semi-supervised learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 215–226, Brussels, Belgium. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017a. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017b. Variational lossy autoencoder. In *International Conference on Learning Representations*.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018a. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260. Association for Computational Linguistics.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018b.

Creative writing with a machine in the loop: Case studies on slogans and stories. In *IUI*, pages 329–340. ACM.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017a. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017b. 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.

Chris Cremer, Quaid Morris, and David Duvenaud. 2017. Reinterpreting importance-weighted autoencoders. In *International Conference on Learning Representations (Workshop Track)*.

I. Dagan, D. Roth, F. Zanzotto, and M. Sammons. 2013. *Recognizing Textual Entailment: Models and Applications*. Morgan & Claypool.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005a. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005b. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40:1:9–56.

Angel Daza, Hiram Calvo, and Jesús Figueroa-Nazuno. 2016. Automatic text generation by learning from literary structures. In *Proceedings of the Fifth Workshop on Computational Linguistics for Literature*, pages 9–19.

Natalie Dehn. 1981. Story generation after TALE-SPIN. In *IJCAI*, pages 16–18.

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.

Dorottya Demszky, Devyani Sharma, Jonathan Clark, Vinodkumar Prabhakaran, and Jacob Eisenstein. 2021. Learning to recognize dialect features. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2315–2338, Online. Association for Computational Linguistics.

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. 2019a. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. 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.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Xiaoan Ding and Kevin Gimpel. 2019. Latent-variable generative models for data-efficient text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 507–517, Hong Kong, China. Association for Computational Linguistics.

Xiaoan Ding and Kevin Gimpel. 2021. FlowPrior: Learning expressive priors for latent variable sentence models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3242–3258, Online. Association for Computational Linguistics.

Xiaoan Ding, Tianyu Liu, Baobao Chang, Zhifang Sui, and Kevin Gimpel. 2020. Discriminatively-Tuned Generative Classifiers for Robust Natural Language Inference. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8189–8202, Online. Association for Computational Linguistics.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2015. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations*.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 350–356, Geneva, Switzerland.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

Esin Durmus, Faisal Ladhak, and Tatsunori Hashimoto. 2022. Spurious correlations in reference-free evaluation of text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1443–1454, Dublin, Ireland. Association for Computational Linguistics.

Nouha Dziri, Hannah Rashkin, Tal Linzen, and David Reitter. 2021. Evaluating groundedness in dialogue systems: The begin benchmark. *arXiv preprint arXiv:2105.00071*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018a. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 653–663, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018b. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Desmond Elliott. 2018. Adversarial evaluation of multimodal machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2974–2978, Brussels, Belgium. Association for Computational Linguistics.

David K. Elson. 2012. *Modeling Narrative Discourse*. Ph.D. thesis, Columbia University.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for*

*Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2214–2220, Florence, Italy. Association for Computational Linguistics.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54. Association for Computational Linguistics.

Le Fang, Chunyuan Li, Jianfeng Gao, Wen Dong, and Changyou Chen. 2019. Implicit deep latent variable models for text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3946–3956, Hong Kong, China. Association for Computational Linguistics.

Song Feng, Kshitij Fadnis, Q. Vera Liao, and Luis A. Lastras. 2020. Doc2dial: A framework for dialogue composition grounded in documents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13604–13605.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. 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 240–250, Minneapolis, Minnesota. Association for Computational Linguistics.

Michel Galley, Chris Brockett, Xiang Gao, Jianfeng Gao, and Bill Dolan. 2019. Grounded response generation task at dstc7. In *AAAI Dialog System Technology Challenges Workshop*.

Zhe Gan, Ricardo Henao, David Carlson, and Lawrence Carin. 2015. Learning deep sigmoid belief networks with data augmentation. In *Artificial Intelligence and Statistics*, pages 268–276. PMLR.

Xiang Gao, Yizhe Zhang, Michel Galley, Chris Brockett, and Bill Dolan. 2020. Dialogue response ranking training with large-scale human feedback data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 386–395, Online. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6. Association for Computational Linguistics.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. 2015. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889.

Kevim Gimpel. 2018. Lecture 4: Text classification. `https://ttic.uchicago.edu/~kgimpel/teaching/31190-s18/lectures/lect4-textcat1.pdf`.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736, Los Angeles, California. Association for Computational Linguistics.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. 2020. Your classifier is secretly an energy based model and you should treat it like one. *ICLR*.

Herbert P Grice. 1975. Logic and conversation. In *Speech acts*, pages 41–58. Brill.

Paul Grice. 1989. *Studies in the Way of Words*. Harvard University Press.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. 2019. Improved zero-shot neural machine translation via ignoring spurious correlations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1258–1268, Florence, Italy. Association for Computational Linguistics.

Ishaan Gulrajani and David Lopez-Paz. 2020. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*.

Prakhar Gupta, Yulia Tsvetkov, and Jeffrey Bigham. 2021a. Synthesizing adversarial negative responses for robust response ranking and evaluation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3867–3883, Online. Association for Computational Linguistics.

Prakhar Gupta, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2021b. Dialfact: A benchmark for fact-checking in dialogue. *arXiv preprint arXiv:2110.08222*.

Xiaochuang Han and Yulia Tsvetkov. 2021. Influence tuning: Demoting spurious correlations via instance attribution and instance-driven updates. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4398–4409, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. In *International Conference on Learning Representations*.

Tianxing He, Bryan McCann, Caiming Xiong, and Ehsan Hosseini-Asl. 2021. Joint energy-based model training for better calibrated natural language understanding models. *arXiv preprint arXiv:2101.06829*.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020a. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online. Association for Computational Linguistics.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020b. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick,

Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. 2019. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997b. Long short-term memory. *Neural Computation*.

Matthew D Hoffman and Matthew J Johnson. 2016. ELBO surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *ICLR*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Or Honovich, Leshem Choshen, Roee Aharoni, Ella Neeman, Idan Szpektor, and Omri Abend. 2021. $q^2$: Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7856–7870, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017a. Toward controlled generation of text. In *Proc. of ICML*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017b. Toward controlled generation of text. In *ICML*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. 2020. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3441–3460, Online. Association for Computational Linguistics.

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.

Dan Jurafsky and James H. Martin. 2021. *Speech and Language Processing (3rd ed. draft)*. Pearson Prentice Hall, Upper Saddle River, N.J.

J. Kaiser, B. Horvat, and Z. Kacic. 2000. A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Proc. of ICSLP*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference*

*on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas. Association for Computational Linguistics.

Hyunwoo Kim, Byeongchang Kim, and Gunhee Kim. 2020. Will I sound like me? improving persona consistency in dialogues through pragmatic self-consciousness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 904–916, Online. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Yoon Kim, Sam Wiseman, Andrew C Miller, David Sontag, and Alexander M Rush. 2018. Semi-amortized variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*.

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

Diederik P. Kingma and Jimmy Ba. 2015b. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Diederik P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*.

Diederik P Kingma and Max Welling. 2014a. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014b. Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In D. D. Lee,

M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4743–4751. Curran Associates, Inc.

Sheldon Klein, John F. Aeschlimann, David F. Balsiger, Steven L. Converse, Claudine Court, Mark Foster, Robin Lao, John D. Oakley, and Joel Smith. 1973. Automatic novel writing: A status report. Technical Report 186, University of Wisconsin-Madison.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online. Association for Computational Linguistics.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

Hugo Larochelle and Iain Murray. 2011. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37. JMLR Workshop and Conference Proceedings.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021a. Neural data augmentation via example extrapolation. *arXiv preprint arXiv:2102.01335*.

Seanie Lee, Minki Kang, Juho Lee, and Sung Ju Hwang. 2021b. Learning to perturb word embeddings for out-of-distribution QA. In *Proceedings of the 59th Annual Meeting of the Association*

*for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5583–5595, Online. Association for Computational Linguistics.

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

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. 2019. A surprisingly effective fix for deep latent variable modeling of text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3603–3614, Hong Kong, China. Association for Computational Linguistics.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4678–4699, Online. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016b. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Bill Yuchen Lin, Sida Wang, Xi Lin, Robin Jia, Lin Xiao, Xiang Ren, and Scott Yih. 2022. On continual model refinement in out-of-distribution data streams. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3128–3139, Dublin, Ireland. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

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.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018. Narrative modeling with memory chains and semantic supervision. *CoRR*, abs/1805.06122.

Tianyu Liu, Zheng Xin, Xiaoan Ding, Baobao Chang, and Zhifang Sui. 2020. An empirical study on model-agnostic debiasing strategies for robust natural language inference. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 596–608, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods*

*in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4282–4292, Hong Kong, China. Association for Computational Linguistics.

Bill MacCartney. 2009. Natural language inference. In *P.hD Thesis*.

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

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

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572.

James R. Meehan. 1977. TALE-SPIN, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 91–98.

John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. 2020. The effect of natural distribution shift on question answering models. In *International Conference on Machine Learning*, pages 6905–6916. PMLR.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854, Florence, Italy. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. 2013. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR.

Radford M Neal and Geoffrey E Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

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.

K.M. Oinonen, Mariet Theune, Antinus Nijholt, and J.R.R. Uijlings. 2006. *Designing a story database for use in automatic story generation*, Lecture Notes in Computer Science, pages 298–301. Springer Verlag.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018. Reducing gender bias in abusive language detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2799–2804, Brussels, Belgium. Association for Computational Linguistics.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. 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 646–653, New Orleans, Louisiana. Association for Computational Linguistics.

Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. 2015. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69.

Tom Pelsmaeker and Wilker Aziz. 2020. Effective estimation of deep generative language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7220–7236, Online. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014a. 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.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014b. 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.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah. 2008. Boosted MMI for model and feature space discriminative training. In *Proc. of ICASSP*.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Tom Rainforth, Adam Kosiorek, Tuan Anh Le, Chris Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. 2018. Tighter variational bounds are not necessarily better. In *International Conference on Machine Learning*, pages 4277–4285. PMLR.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Em-*

*pirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. 2019. Preventing posterior collapse with delta-vaes. In *International Conference on Learning Representations*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Mark O. Riedl and Robert Michael Young. 2010. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268.

Melissa Roemmele and Andrew S. Gordon. 2015. Creative help: a story writing assistant. In *International Conference on Interactive Digital Storytelling*, pages 81–92. Springer.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online. Association for Computational Linguistics.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Sebastian Ruder, Ryan Cotterell, Yova Kementchedjhieva, and Anders Søgaard. 2018. A discriminative latent-variable model for bilingual lexicon induction. *arXiv preprint arXiv:1808.09334*.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. 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 8–14, New Orleans, Louisiana. Association for Computational Linguistics.

Manasvi Sagarkar, John Wieting, Lifu Tu, and Kevin Gimpel. 2018. Quality signals in generated stories. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics (*SEM 2018)*, New Orleans, Louisiana.

Gaurav Sahu, Pau Rodriguez, Issam Laradji, Parmida Atighehchian, David Vazquez, and Dzmitry Bahdanau. 2022. Data augmentation for intent classification with off-the-shelf large language models. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 47–57, Dublin, Ireland. Association for Computational Linguistics.

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

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. 2020. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5477–5490, Online. Association for Computational Linguistics.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 627–637, Copenhagen, Denmark. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Iulian Vlad Serban, Alexander G. Ororbia, Joelle Pineau, and Aaron Courville. 2017. Piecewise latent variables for neural variational text processing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 422–432, Copenhagen, Denmark. Association for Computational Linguistics.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. 2018. Transductive semi-supervised deep learning using min-max features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 299–315.

Kurt Shuster, Da Ju, Stephen Roller, Emily Dinan, Y-Lan Boureau, and Jason Weston. 2020. The dialogue dodecathlon: Open-domain knowledge and image grounded conversational agents. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2453–2470, Online. Association for Computational Linguistics.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models.

In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 787–794, Sydney, Australia. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1630–1640, Florence, Italy. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press.

Mariët Theune, Sander Faas, Anton Nijholt, and Dirk Heylen. 2003. The virtual storyteller: story creation by intelligent agents. In *Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, pages 204–215. Springer.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

*Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Jakub M Tomczak and Max Welling. 2018. VAE with a VampPrior. In *Proceedings of AISTATS*.

Lifu Tu, Xiaoan Ding, Dong Yu, and Kevin Gimpel. 2019. Generating diverse story continuations with controllable semantics. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 44–58, Hong Kong. Association for Computational Linguistics.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An empirical study on robustness to spurious correlations using pre-trained language models. *Transactions of the Association for Computational Linguistics*, 8:621–633.

Yi-Lin Tuan, Yun-Nung Chen, and Hung-yi Lee. 2019. DyKgChat: Benchmarking dialogue generation grounding on dynamic knowledge graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1855–1865, Hong Kong, China. Association for Computational Linguistics.

Scott R. Turner. 1993. *Minstrel: a computer model of creativity and storytelling*. Ph.D. thesis, University of California at Los Angeles.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017a. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017b. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Grace Wahba et al. 1999. Support vector machines, reproducing kernel hilbert spaces and the randomized gacv. *Advances in Kernel Methods-Support Vector Learning*, 6:69–87.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiaozhi Wang, Xu Han, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018b. Adversarial multilingual neural relation extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1156–1166, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in*

*Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Garrett Wilson and Diane J Cook. 2020. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional bert contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer.

Yuxiang Wu, Matt Gardner, Pontus Stenetorp, and Pradeep Dasigi. 2022. Generating data to mitigate spurious correlations in natural language inference datasets. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2660–2676, Dublin, Ireland. 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*.

Yadollah Yaghoobzadeh, Soroush Mehri, Remi Tachet des Combes, T. J. Hazen, and Alessandro Sordoni. 2021. Increasing robustness to spurious correlations using forgettable examples. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3319–3332, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3881–3890. JMLR. org.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

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

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural*

*Language Processing*, pages 521–530, Austin, Texas. Association for Computational Linguistics.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019a. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

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.

Yizhe Zhang, Xiang Gao, Sungjin Lee, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2019b. Consistent dialogue generation with self-supervised feature learning. *arXiv preprint arXiv:1903.05759*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020b. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Xiang Gao, Yuwei Fang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2021. Joint retrieval and generation training for grounded text generation. *arXiv preprint arXiv:2105.06597*.

Jieyu Zhao, Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Kai-Wei Chang. 2021a. Ethical-advice taker: Do language models understand natural language interventions? In *Find-*

*ings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4158–4164, Online. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017a. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989, Copenhagen, Denmark. Association for Computational Linguistics.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. 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 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2017b. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1393–1404, Online. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. *SIGIR*.

Zachary M Ziegler and Alexander M Rush. 2019. Latent normalizing flows for discrete sequences. In *International Conference on Learning Representations*.

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

# APPENDIX A

# APPENDIX: LATENT-VARIABLE GENERATIVE MODELS FOR DATA-EFFICIENT TEXT CLASSIFICATION

## A.1 Alternative Inference Criteria

The classification accuracies of the auxiliary latent generative model in the main text are based on predictions made while marginalizing out the latent variable. In addition, we experiment with two other inference objectives. One uses the posterior $p(c \mid x, y)$ instead of the learned prior $p_\Phi(c)$ during marginalization:

$$\hat{y} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{c \in \mathcal{C}} p_\Theta(x \mid c, y) p(c \mid x, y) p_\Psi(y)$$

The other way is to predict by maximizing the latent variable:

$$\hat{y} = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \max_{c \in \mathcal{C}} p_\Theta(x \mid c, y) p_\Phi(c) p_\Psi(y)$$

We find very similar performance with all three inference criteria, which agrees with our observation that the classifiers learn peaked prior and posterior distributions over the discrete latent variables.

## A.2 Additional Results with Training Sizes

While the main paper contained these results in plots, for completeness we provide the numerical classification accuracies of the discriminative (**Disc.**), generative (**Gen.**), and latent-variable generative (**Lat.**) classifiers trained with various training sizes in Tables A.1, A.2, A.3, A.4, A.5, and A.6.

| # per class | Disc. | Gen. | Lat. |
|---|---|---|---|
| 5 | 61.97 | 55.42 | 62.53 |
| 20 | 64.19 | 59.06 | 66.67 |
| 100 | 66.72 | 69.80 | 73.50 |
| 1k | 77.53 | 78.62 | 81.22 |
| 2k | 80.48 | 80.98 | 82.96 |
| 5k | 82.62 | 83.60 | 84.97 |
| 10k | 85.83 | 85.41 | 85.65 |
| all | 92.20 | 87.51 | 87.38 |

Table A.1: Comparison of classification accuracy on Yelp Review Polarity dataset.

| # per class | Disc. | Gen. | Lat. |
|---|---|---|---|
| 5 | 23.38 | 21.67 | 27.16 |
| 20 | 25.12 | 26.20 | 31.78 |
| 100 | 29.82 | 35.87 | 38.67 |
| 1k | 42.85 | 43.36 | 46.05 |
| 2k | 46.09 | 44.16 | 48.58 |
| 5k | 52.23 | 47.75 | 49.86 |
| 10k | 52.23 | 50.30 | 50.19 |
| all | 59.00 | 52.34 | 51.14 |

Table A.2: Comparison of classification accuracy on Yelp Review Full dataset.

| # per class | Disc. | Gen. | Lat. |
|---|---|---|---|
| 5 | 40.20 | 35.12 | 47.46 |
| 20 | 43.68 | 37.86 | 61.45 |
| 100 | 62.58 | 68.70 | 78.58 |
| 1k | 78.08 | 84.08 | 86.12 |
| 2k | 80.80 | 86.70 | 87.25 |
| 5k | 84.87 | 88.88 | 89.26 |
| 10k | 87.25 | 89.67 | 89.63 |
| all | 89.79 | 90.00 | 90.14 |

Table A.3: Comparison of classification accuracy on AG News dataset.

| # per class | Disc. | Gen. | Lat. |
|---|---|---|---|
| 5 | 41.75 | 39.89 | 61.19 |
| 20 | 52.80 | 66.32 | 72.18 |
| 100 | 69.18 | 77.88 | 81.48 |
| 1k | 83.83 | 84.81 | 86.40 |
| 2k | 85.04 | 86.50 | 86.61 |
| 5k | 87.90 | 87.42 | 86.62 |
| 10k | 89.94 | 87.67 | 86.81 |
| all | 93.40 | 87.95 | 86.95 |

Table A.4: Comparison of classification accuracy on Sogou dataset.

| # per class | Disc. | Gen. | Lat. |
|---|---|---|---|
| 5 | 13.26 | 15.39 | 21.00 |
| 20 | 19.98 | 30.33 | 36.55 |
| 100 | 29.97 | 47.33 | 50.04 |
| 1k | 55.15 | 62.68 | 64.18 |
| 2k | 60.83 | 65.52 | 65.70 |
| 5k | 66.07 | 67.95 | 67.79 |
| 10k | 69.00 | 68.90 | 67.92 |
| all | 72.70 | 69.14 | 68.02 |

Table A.5: Comparison of classification accuracy on Yahoo dataset.

| # per class | Disc. | Gen. | Lat. |
|---|---|---|---|
| 5 | 32.27 | 63.02 | 66.33 |
| 20 | 43.72 | 82.17 | 85.56 |
| 100 | 74.73 | 90.37 | 92.24 |
| 1k | 96.11 | 94.62 | 95.54 |
| 2k | 96.85 | 95.06 | 95.75 |
| 5k | 97.76 | 95.78 | 96.09 |
| 10k | 98.15 | 96.29 | 96.30 |
| all | 98.70 | 96.73 | 96.25 |

Table A.6: Comparison of classification accuracy on DBpedia dataset.

## A.3 Total Number of Parameters

Table A.7 shows the hyperparameter settings for our classifiers. There are various choices of latent variable values and dimensionalities. We select the ones with the best classification accuracy

| | word embed | hidden state | label embed | # latent variable | latent embedding |
|---|---|---|---|---|---|
| **Disc.** | 100 | 100 | 100 | - | - |
| **Gen.** | 100 | 100 | 100 | - | - |
| **Lat.** | 100 | 100 | 100 | 10, 30, 50 | 10, 50, 100 |
| **Gen. PC** | 100 | 100 | 110 | - | - |
| **Lat. PC** | 100 | 100 | 100 | 10 | 10 |

Table A.7: Hyperparameter settings of Discriminative (**Disc.**), Generative (**Gen.**), Latent Generative (**Lat.**), Generative PC (**Gen. PC**), Latent Generative PC (**Lat. PC**) classifiers. **PC** stands for "Parameter-comparison Configuration." More description can be found in the main paper.

according to the development sets.

Table A.8 lists the number of parameters in each classifier. It is related to the discussion about effect of latent structure in the main paper. We created **Gen. PC** and **Lat. PC** 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.

## A.4 Results with Larger Models

We increase the model capacity by increasing dimensionality of the word embedding, LSTM hidden embedding, and label embedding to 200 and refer to the resulting models as the large discriminative (**Disc L.**), generative (**Gen L.**), and latent-variable generative (**Lat L.**) classifiers. Note that we did not change the number of values or dimensionality of the latent variables. We only experimented with two datasets due to GPU memory limits.[1] Table A.9 shows the performance comparison between standard (reported in the main paper) and larger classifiers. We find that the trend **Disc L.** $<$ **Gen L.** $<$ **Lat L.** still holds in most cases in the small-data setting, though the performance gaps shrink as the capacity increases.

1. The GPU memory consumption is affected by the number of labels in the generative and latent generative classifiers. These two datasets have relatively small numbers of labels.

| Dataset | # per class | Disc. | Gen. | Lat. | Gen. PC | Lat. PC |
|---|---|---|---|---|---|---|
| Yelp P | 5 | 4,082,414 | 12,642,828 | 16,122,903 | 12,481,640 | 12,521,733 |
| | 20 | | | 14,123,253 | | |
| | 100 | | | 14,123,253 | | |
| | 1k | | | 14,123,253 | | |
| | all | | | 16,122,903 | | |
| Yelp F | 5 | 4,082,414 | 12,642,828 | 12,522,233 | 12,481,640 | 12,521,733 |
| | 20 | | | 12,522,233 | | |
| | 100 | | | 12,522,233 | | |
| | 1k | | | 12,522,233 | | |
| | all | | | 14,122,553 | | |
| AGNews | 5 | 4,082,414 | 12,642,828 | 11,430,985 | 10,104,780 | 10,137,185 |
| | 20 | | | 10,137,185 | | |
| | 100 | | | 11,430,985 | | |
| | 1k | | | 10,137,585 | | |
| | all | | | 12,522,333 | | |
| Sogou | 5 | 4,082,414 | 12,642,828 | 13,162,568 | 11,634,120 | 11,671,448 |
| | 20 | | | 13,163,568 | | |
| | 100 | | | 15,028,468 | | |
| | 1k | | | 11,676,935 | | |
| | all | | | 12,522,233 | | |
| Yahoo | 5 | 4,082,414 | 12,642,828 | 12,522,533 | 12,482,520 | 12,522,533 |
| | 20 | | | 14,124,053 | | |
| | 100 | | | 12,522,733 | | |
| | 1k | | | 12,522,533 | | |
| | all | | | 16,123,703 | | |
| DBpedia | 5 | 4,082,414 | 12,642,828 | 12,522,933 | 12,482,960 | 12,522,933 |
| | 20 | | | 14,124,453 | | |
| | 100 | | | 14,124,453 | | |
| | 1k | | | 16,126,103 | | |
| | all | | | 12,522,933 | | |

Table A.8: Number of parameters in each classifier.

| | # per class | Disc L. | Gen L. | Lat L. | Disc. | Gen. | Lat. |
|---|---|---|---|---|---|---|---|
| AGNews | 5 | 37.34 | 40.55 | 47.91 | 40.20 | 35.12 | 47.46 |
| | 20 | 44.53 | 47.42 | 62.36 | 43.68 | 37.86 | 61.45 |
| | 100 | 62.74 | 76.95 | 79.63 | 62.58 | 68.70 | 78.58 |
| | 1k | 80.67 | 84.82 | 86.79 | 78.08 | 84.08 | 86.12 |
| | all | 90.54 | 90.16 | 89.68 | 89.79 | 90.00 | 90.14 |
| Yelp Polarity | 5 | 60.82 | 60.65 | 63.07 | 61.97 | 55.42 | 62.53 |
| | 20 | 61.11 | 64.44 | 67.50 | 64.19 | 59.06 | 66.67 |
| | 100 | 68.55 | 71.79 | 74.37 | 66.72 | 69.80 | 73.50 |
| | 1k | 77.93 | 78.91 | 81.82 | 77.53 | 78.62 | 81.22 |
| | all | 92.48 | 87.76 | 87.34 | 92.20 | 87.51 | 87.38 |

Table A.9: Comparision of classification accuracies between standard and larger classifiers.

# APPENDIX B

# APPENDIX: DISCRIMINATIVELY-TUNED GENERATIVE CLASSIFIERS FOR ROBUST NATURAL LANGUAGE INFERENCE

## B.1 Discriminative Fine-Tuning Comparison

Table B.1 lists the full comparison results of different discriminative fine-tuning objectives. Several choices, including hinge, softmax-margin, and infinilog, consistently outperform the log loss used as discriminative fine-tuning objective by Lewis and Fan (2019). It is worth noting that infinilog performs the best when using the full training set on four out of five datasets.

## B.2 Data Generation

Table B.2 shows example generations from two models, one using the full dataset for training and the other using a small training set with only 500 examples per class.

## B.3 Ablation of Copy Mechanism in Generation

Table B.3 shows the generated hypotheses of the proposed generative classifier. Comparing the generative classifiers with and without copy mechanism, we find that the copy mechanism can help the model capture key differences between premise and hypothesis sentences given the specified labels. For example, we see 'There is no child' versus 'A child' given the label 'contradiction', and 'another animal' versus 'a brown dog' given the label 'neutral'. The copy mechanism also helps to avoid excessive semantic drift, e.g., generating the same subject as the premise and maintaining a reasonable amount of text with the premise.

Although classification accuracy increases by adopting discriminative finetuning after generative training, the finetuning method can lead to ungrammatical or repetitive generated sentences, as

demonstrated in Table B.3. This shows that generated text with higher quality does not necessarily lead to better performance in NLI classification.

| | 5 | 20 | 100 | 500 | 1000 | all |
|---|---|---|---|---|---|---|
| **SNLI** | | | | | | |
| perceptron | 41.8 | 44.1 | 49.6 | 58.4 | 62.5 | 80.4 |
| hinge | 42.3 | 45.3 | 49.9 | 58.6 | 63.1 | 81.1 |
| log | 42.1 | 43.2 | 49.1 | 58.6 | 62.3 | 80.7 |
| softmax-margin | **43.5** | 45.3 | **50.6** | **60.6** | **64.2** | 81.9 |
| infinilog | 42.7 | **45.6** | 50.0 | 59.8 | 63.7 | **82.2** |
| Bayes risk | 42.8 | 44.7 | 49.0 | 58.3 | 62.6 | 80.1 |
| **MNLI** | | | | | | |
| perceptron | 42.7 | 45.5 | 46.7 | 58.1 | 61.6 | 66.3 |
| hinge | 43.2 | 46.3 | 48.2 | 60.2 | 62.8 | 67.1 |
| log | 42.1 | 45.4 | 46.7 | 58.3 | 61.4 | 66.2 |
| softmax-margin | **44.1** | **47.1** | **49.0** | **60.6** | **63.4** | **67.5** |
| infinilog | 42.3 | 45.9 | 47.7 | 60.0 | 62.8 | 67.3 |
| Bayes risk | 43.1 | 45.7 | 47.7 | 59.1 | 61.6 | 66.2 |
| **SICK** | | | | | | |
| perceptron | 49.1 | 61.7 | 66.9 | 73.4 | - | 79.7 |
| hinge | **50.6** | 63.8 | 67.8 | 73.6 | - | 80.0 |
| log | 48.6 | 62.1 | 67.5 | 73.1 | - | 79.8 |
| softmax-margin | 50.2 | **64.7** | **68.7** | 74.3 | - | 80.2 |
| infinilog | 48.4 | 62.4 | 68.3 | **75.2** | - | **80.4** |
| Bayes risk | 48.2 | 62.4 | 67.2 | 72.8 | - | 79.7 |
| **RTE** | | | | | | |
| perceptron | 56.1 | 57.4 | 57.9 | 59.4 | 60.1 | 61.1 |
| hinge | 56.4 | 57.1 | 58.8 | 59.2 | 61.3 | 62.2 |
| log | 56.5 | 57.1 | 57.4 | 59.1 | 59.7 | 60.5 |
| softmax-margin | **57.0** | **57.7** | **59.2** | **60.4** | 61.1 | 62.2 |
| infinilog | 56.7 | 57.4 | 58.1 | 59.6 | **61.4** | **62.6** |
| Bayes risk | 56.1 | 57.2 | 58.3 | 59.3 | 60.6 | 61.4 |
| **MRPC** | | | | | | |
| perceptron | 62.1 | 62.5 | 64.6 | 66.1 | 68.6 | 69.8 |
| hinge | 62.3 | 63.8 | 65.4 | 67.1 | 69.0 | 71.8 |
| log | 61.7 | 62.1 | 64.1 | 65.9 | 68.1 | 71.3 |
| softmax-margin | 62.6 | **64.1** | **66.2** | **67.8** | **69.9** | 72.8 |
| infinilog | 62.8 | 63.7 | 65.6 | 67.4 | 69.8 | **72.9** |
| Bayes risk | **63.2** | 63.5 | 65.6 | 67.7 | 69.5 | 72.5 |

Table B.1: Comparison of discriminative fine-tuning objectives. The best result for each task and data amount is shown in bold, and the second-best one is underlined.

| | | |
|---|---|---|
| **GenNLI trained on full RTE training set** | | |
| E | $x^{(p)}$ | Only a week after it had no comment on upping the storage capacity of its hotmail e-mail service , microsoft early thursday announced it was boosting the allowance to 250mb to follow similar moves by rivals such as google , yahoo , and lycos. |
| | $x^{(h)}$ | Microsoft 's hotmail has raised its storage capacity to 250mb. |
| | gen. | **Microsoft was boosting of its hotmail e-mail.** |
| N | $x^{(p)}$ | The name for the newest james bond film has been announced today . the 22nd film , previously known only as " bond 22 " , will be called " quantum of solace " . Eon productions who are producing the film made the announcement today at pinewood studios , where production for the film has been under way since last year . The name of the film was inspired by a short story of the same name from for your eyes only by bond creator , ian fleming. |
| | $x^{(h)}$ | James bond was created by ian fleming. |
| | gen. | **James bond is a member of the film.** |
| **GenNLI trained on small RTE training set** | | |
| E | $x^{(p)}$ | Lin piao , after all , was the creator of mao 's " little red book " of quotations. |
| | $x^{(h)}$ | Lin piao wrote the " little red book " . |
| | gen. | **Lin piao 's " little red book ".** |
| N | $x^{(p)}$ | A dog is pushing a toddler into a rain puddle. |
| | $x^{(h)}$ | A dog is pulling a toddler out of a rain puddle. |
| | gen. | **A dog is pushing a rain puddle.** |

Table B.2: Generated hypotheses for premises with given labels (N = not entailment, E = entailment).

| | | |
|---|---|---|
| Neutral | $x^{(p)}$ | A brown dog is attacking another animal in front of the man in pants. |
| | $x^{(h)}$ | Two dogs are fighting. |
| | gen. | **A brown dog is attacking a brown dog in front of the man.** |
| | gen. w/ finetune | **A man is sitting on a black shirt is standing on a black shirt.** |
| | gen. w/o copy | **A man is wearing a black shirt and is sitting on a dirt ball.** |
| Entailment | $x^{(p)}$ | A group of children in uniforms is standing at a gate and one is kissing the mother. |
| | $x^{(h)}$ | A group of children wearing the same clothes is waiting at a gate and one is kissing the mother |
| | gen. | **A group of children in uniforms is standing at a gate.** |
| | gen. w/ finetune | **A group in uniforms at uniforms is gate and one is kissing mother.** |
| | gen. w/o copy | **A man is sitting on a ball in the water.** |
| Contradiction | $x^{(p)}$ | There is no child holding a water gun or getting sprayed with water. |
| | $x^{(h)}$ | A laughing child is holding a water gun and getting sprayed with water. |
| | gen. | **A child is holding a water gun.** |
| | gen. w/ finetune | **There is child child holding a water gun with water.** |
| | gen. w/o copy | **A dog is jumping in the water.** |

Table B.3: Generated hypotheses for premises with given labels using models trained on the full SICK dataset. When generating using the discriminatively-finetuned model, the outputs show more repetition, while without the copy mechanism, they drift more from the premise.

# APPENDIX C

# APPENDIX: FLOWPRIOR: LEARNING EXPRESSIVE PRIORS FOR LATENT VARIABLE SENTENCE MODELS

## C.1 Additional Results with Free Bits KL

| Model | PPL(↓) | KL | AU(↑) | MI(↑) |
|---|---|---|---|---|
| **PTB** | | | | |
| VAE | 101.4 / 101.6 | 0.00 / 4.46 | 0 / 32 | 0.0 / 0.1 |
| VAE+infinilog | 96.9 / 95.8 | 1.57 / 6.34 | 24 / 32 | 0.6 / 1.5 |
| MoG-VAE | 98.2 / 96.8 | 0.00 / 2.35 | 0 / 32 | 0.00 / 0.68 |
| Vamp-VAE | 98.3 / 97.3 | 0.00 / 2.31 | 0 / 32 | 0.00 / 0.72 |
| FlowPrior | 94.7 / 93.6 | 3.28 / 7.21 | 2 / 31 | 2.3 / 2.8 |
| **Yahoo** | | | | |
| VAE | 65.8 / 64.6 | 0.00 / 4.88 | 0 / 32 | 0.0 / 0.9 |
| VAE+infinilog | 63.9 / 61.7 | 2.72 / 13.31 | 1 / 32 | 2.0 / 1.7 |
| MoG-VAE | 64.6 / 67.3 | 0.00 / 1.83 | 0 / 32 | 0.0 / 0.6 |
| Vamp-VAE | 74.8 / 75.9 | 0.01 / 1.24 | 0 / 32 | 0.0 / 0.6 |
| FlowPrior | 62.5 / 68.3 | 1.43 / 10.99 | 4 / 25 | 1.6 / 0.6 |
| **Yelp** | | | | |
| VAE | 35.1 / 37.5 | 0.00 / 3.59 | 0 / 32 | 0.0 / 1.0 |
| VAE+infinilog | 33.2 / 39.6 | 2.91 / 4.16 | 28 / 32 | 0.9 / 2.2 |
| MoG-VAE | 35.2 / 39.8 | 0.01 / 1.81 | 0 / 32 | 0.0 / 0.6 |
| Vamp-VAE | 35.0 / 39.4 | 0.00 / 1.78 | 0 / 32 | 0.0 / 0.6 |
| FlowPrior | 31.8 / 39.0 | 4.15 / 10.13 | 2 / 32 | 2.5 / 2.6 |
| **SNLI** | | | | |
| VAE | 26.0 / 30.5 | 0.00 / 1.84 | 0 / 32 | 0.0 / 0.9 |
| VAE+infinilog | 25.3 / 17.8 | 1.23 / 15.48 | 23 / 32 | 0.5 / 2.0 |
| MoG-VAE | 28.1 / 27.5 | 0.44 / 2.28 | 1 / 32 | 0.4 / 0.7 |
| Vamp-VAE | 26.0 / 29.3 | 0.00 / 5.11 | 0 / 32 | 0.0 / 0.8 |
| FlowPrior | 22.4 / 26.2 | 3.83 / 7.59 | 3 / 32 | 1.0 / 3.2 |

Table C.1: Results when comparing standard KL and FB KL for several models. The left part in each cell shows training with standard KL and the right part shows using FB KL instead.

Using the Free Bits method can help achieve a consistently better AU and higher KL as shown in the overall results in the main text. We report additional empirical comparisons to focus on

| | Yelp | | SNLI | |
|---|---|---|---|---|
| | KL | FB KL | KL | FB KL |
| VAE | 0.26 | 0.49 | 1.66 | 2.18 |
| VAE+infinilog | 0.65 | 0.71 | 1.91 | 3.57 |
| FlowPrior | 1.50 | 0.74 | 4.87 | 3.64 |

Table C.2: Test set reconstruction BLEU scores.

measuring the impact of FB for three models in Table C.1.

Though adding FB yields higher AU and MI, it is not always true that it leads to a better test PPL and reconstruction. This phenomenon has been pointed out by Razavi et al. (2019) that adding FB makes the objective non-smooth which can lead to optimization difficulties. Possible solutions could be changing to a better training procedure. Li et al. (2019) remedy this issue by combining pretraining with FB, namely using a pretrained autoencoder to initialize the inference network before starting training the VAE networks. This suggests that it may be necessary to pretrain the inference network and decoder to unilaterally benefit from FB.

Table C.2 considers standard VAEs and FlowPrior when comparing the use of standard KL to FB KL. Using FB KL does not lead to a higher BLEU score in FlowPrior, though FB does improve BLEU when combined with standard VAE and VAE+infinilog.

## C.2 Additional Results with FB and infinilog

Table C.3 shows the impact of infinilog and FB on F-PPL, R-PPL, and self-BLEU with greedy generation from prior samples.

## C.3 Reconstruction Results with Sampling

Tables C.4-C.5 show the reconstruction performance with standard sampling and nucleus sampling with $p = 0.9$ (Holtzman et al., 2020). We observe the trends are consistent with the results that use greedy decoding.

| | Yelp | | | SNLI | | |
|---|---|---|---|---|---|---|
| | F-PPL | R-PPL | SB | F-PPL | R-PPL | SB |
| VAE | 4 | 30248 | 96 | 4 | 51127 | 100 |
| VAE+infinilog | 5 | 10818 | 30 | 4 | 19047 | 73 |
| MoG-VAE | 4 | 30413 | 100 | 3 | 45979 | 77 |
| MoG-VAE+infinilog | 4 | 33624 | 100 | 6 | 5257 | 26 |
| Vamp-VAE | 4 | 32504 | 100 | 4 | 56050 | 100 |
| Vamp-VAE+infinilog | 7 | 5280 | 10 | 5 | 8420 | 29 |
| FlowPrior | 209 | 1677 | 3 | 42 | 5725 | 13 |
| VAE+FB | 7 | 7517 | 29 | 4 | 22536 | 42 |
| VAE+infinilog+FB | 8 | 5713 | 13 | 4 | 24204 | 48 |
| FlowPrior+FB | 8 | 5179 | 9 | 15 | 4876 | 11 |

Table C.3: Forward PPL (F-PPL), Reverse PPL (R-PPL), and Self-BLEU (SB) of greedy-decoded prior samples.

## C.4 Interpolation with Sampling

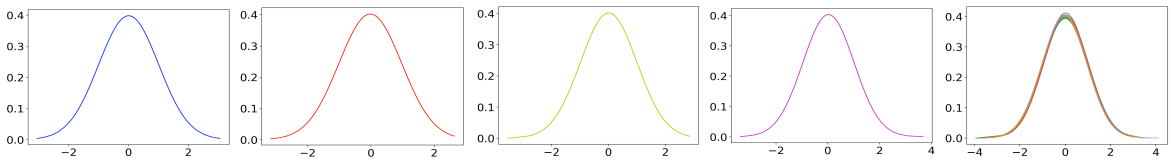
Table C.6 shows more examples of interpolation-based generation with greedy decoding. We show results with sampling methods for decoding in Tables C.7, C.8, C.9, and C.10. The results with greedy decoding provide a lower-variance way to interpret the learned latent space. The additional results with sampling methods provide a richer picture as they also capture the randomness in the relationship between the latent variable and the text. This is especially helpful when we observe repetition in neighboring samples with greedy decoding, as we see with MoG-VAE and Vamp-VAE in Table C.6. Even with sampling, FlowPrior shows a smoother semantic evolution in the latent space than MoG-VAE and Vamp-VAE, at least in terms of aspects of the subjects of the generated sentences.

## C.5 Sampling from Priors

Table C.11 shows more greedy generations from prior samples. We observe substantial improvements in term of generation diversity in FlowPrior and FlowPrior + FB. While standard VAE always yields identical samples because the latent variable is ignored, using FB KL yields bet-

| | Yelp | | SNLI | |
|---|---|---|---|---|
| | ELBO | ELBO+infinilog | ELBO | ELBO+infinilog |
| Standard | 0.07 | 0.15 | 0.43 | 0.57 |
| MoG | 0.08 | 0.00 | 0.66 | 2.36 |
| Vamp | 0.06 | 0.32 | 0.55 | 0.76 |
| Real NVP | 0.28 | 0.60 | 0.97 | 1.91 |
| | KL | FB KL | KL | FB KL |
| VAE | 0.07 | 0.13 | 0.43 | 0.74 |
| VAE+infinilog | 0.15 | 0.28 | 0.57 | 0.95 |
| MoG | 0.08 | 0.08 | 0.66 | 0.54 |
| Vamp | 0.06 | 0.10 | 0.55 | 0.78 |
| FlowPrior | 0.60 | 0.34 | 1.91 | 0.91 |

Table C.4: Test set reconstruction BLEU scores using standard sampling in decoding.



(a) dim 1 (b) dim 2 (c) dim 3 (d) dim 4 (e) all 32 dims

Figure C.1: Visualization of dimensions of learned prior when using real NVP on SNLI dataset. Plots from left to right are first dimension alone, second dimension alone, third dimension alone, fourth dimension alone, and all 32 dimensions together.

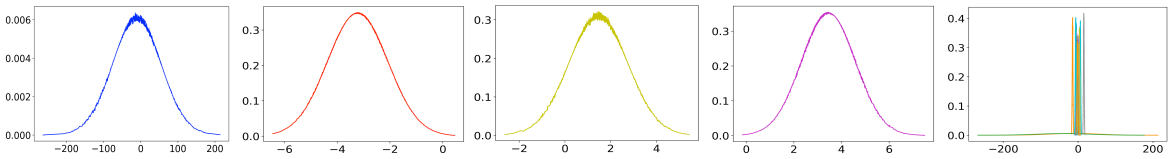
ter sample diversity as it encourages more information encoded into latent variable during training. This can be easily observed by comparing the samples from standard VAE with those from VAE+FB, comparing MoG-VAE with MoG-VAE + infinilog, comparing Vamp-VAE with Vamp-VAE + infinilog.

## C.6 More Visualizations of Real NVP Prior and FlowPrior

Visualizations of each dimension alone and all dimensions together are in Figures C.1 and C.2.

| | Yelp | | SNLI | |
|---|---|---|---|---|
| | ELBO | ELBO+infinilog | ELBO | ELBO+infinilog |
| Standard | 0.08 | 0.20 | 0.56 | 0.72 |
| MoG | 0.09 | 0.06 | 0.80 | 2.66 |
| Vamp | 0.08 | 0.44 | 0.71 | 1.01 |
| Real NVP | 0.30 | 0.72 | 1.31 | 2.40 |
| | KL | FB KL | KL | FB KL |
| VAE | 0.08 | 0.17 | 0.56 | 1.03 |
| VAE+infinilog | 0.20 | 0.36 | 0.72 | 1.22 |
| MoG | 0.09 | 0.16 | 0.80 | 0.82 |
| Vamp | 0.08 | 0.13 | 0.71 | 1.07 |
| FlowPrior | 0.72 | 0.40 | 2.40 | 1.23 |

Table C.5: Test set reconstruction BLEU scores using nucleus sampling in decoding.



(a) dim 1 (b) dim 2 (c) dim 3 (d) dim 4 (e) all 32 dims

Figure C.2: Visualization of dimensions of learned prior when using real NVP with infinilog (i.e., FlowPrior) on SNLI dataset. Plots from left to right are first dimension alone, second dimension alone, third dimension alone, fourth dimension alone, and all 32 dimensions together.

| **Mog-VAE** |
| --- |
| The man is wearing a black shirt . |
| The man is wearing a black shirt . |
| The man is wearing a black shirt . |
| A man is standing in front of a building . |
| A man is standing in front of a building . |
| A man is standing in front of a building . |
| A man is standing in front of a building . |
| A man is standing in front of a building . |
| A man is standing in front of a building . |
| A man is standing in front of a building . |
| **Vamp-VAE** |
| Three people are sitting on a bench . |
| People are walking down the street . |
| People are walking down the street . |
| People are walking down the street . |
| Man in a blue shirt and jeans is sitting on a bench . |
| Man in a blue shirt and jeans is sitting on a bench . |
| Man in a blue shirt and jeans is sitting on a bench . |
| Man in a blue shirt and jeans is sitting on a bench . |
| Man in a blue shirt and jeans is sitting on a bench . |
| Man in a blue shirt and jeans is sitting on a bench . |
| **FlowPrior** |
| The dog is running through the snow . |
| Two young boys are playing in the snow . |
| There is a man in a blue shirt and a woman in a black shirt and black pants . |
| Three people are sitting on a bench . |
| two men are standing on a bench |
| A girl is sitting on a bench . |
| A young girl is sitting on a bench . |
| A young man is sitting on a bench . |
| A woman in a black shirt is sitting on a bench . |
| A woman is sitting on a bench . |

Table C.6: Interpolation between two prior samples with greedy decoding. Dataset used is SNLI. In each cell, the first sentence and the last sentence correspond to the two sampled latent codes, and between are linearly interpolated samples.

| **MoG-VAE** |
| --- |
| A girl is laughing at the beach . |
| The dog is walking in the water . |
| Man in khaki jacket painting an elephant . |
| The man is breakdancing . |
| People are outside on a sunny day . |
| Some people are playing in the snow . |
| Two men are working in a lab . |
| The boy is at the beach . |
| Five soccer players playing soccer . |
| Men stand on a pier . |
| **Vamp-VAE** |
| A person is riding a bicycle at a parade . |
| A brown and white dog with a brown collar is climbing over its hind legs while lying on the floor , talking and fabric in the grass . |
| A little girl wearing a yellow shirt looks at a fountain while a man is kneeling next to her and a child . |
| Two men play dominoes . |
| Young lady in blue dress waits at a bus . |
| A woman carrying a small child looking through a window . Three men are fighting with swords . |
| The little boy is riding his scooter down the paved road . |
| A man wearing a yellow suit eats a hotdog on a wooden table . |
| A group of friends are smiling. |

Table C.7: Interpolation between two prior samples with nucleus sampling for decoding. Dataset used is SNLI. In each cell, the first sentence and the last sentence correspond to the two sampled latent codes, and between are linearly interpolated samples.

| **FlowPrior** |
|---|
| three dogs are in the water |
| 3 people walking down the street with their hands in the air . |
| Man getting a picture |
| Three people are on the beach . |
| There are two dogs standing near each other . |
| Two men in white uniforms are cleaning on a mess of an escalator . |
| Two girls are getting some exercise together . |
| Two women working in a restaurant outdoors . |
| The children are riding . |
| The child is standing on the sidewalk in front of an apartment building . |

Table C.8: Interpolation between two prior samples with nucleus sampling for decoding. Dataset used is SNLI. In each cell, the first sentence and the last sentence correspond to the two sampled latent codes, and between are linearly interpolated samples.

| **MoG-VAE** |
| --- |
| A girl is attending a birthday of popcorn . |
| The dog is walking in the green snow . |
| Man lady on the beach . |
| The man is breakdancing . |
| People are outside on a sunny day . |
| Some people are rooting in an outdoor resaurant . |
| Two men both face out for directions on a street . |
| Big hikers . |
| Five soccer players playing soccer after fifty finish in a field . |
| Men stand on a doorstep . |
| **Vamp-VAE** |
| Two farmers share a drink , while one looks at a woolen . |
| A couple in black and white with a long blue scarf are standing in a store wearing a yellow hat . |
| A three people are riding a white bike through the desert along a road . |
| A tribesman near a playground . |
| People all gathered in the street looking at something on a woman . |
| A couple of angel making corn on a mountainside in a city . |
| A man holding a sign can and the young female . |
| The man playing a guitar concert festival . |
| The shirtless woman and woman performing on sand . |
| dog is outside . |

Table C.9: Interpolation between two prior samples with standard sampling for decoding. Dataset used is SNLI. In each cell, the first sentence and the last sentence correspond to the two sampled latent codes, and between are linearly interpolated samples.

| **FlowPrior** |
|---|
| A dog is jumping through the air to catch a Frisbee in the air . |
| A brown and white dog chewing on a red disc . |
| A crowd of people are blowing in a brown down balloon . |
| A woman is pushing her cart . |
| A person is skiing through a snowy mountain . |
| A woman carrying a small child is playing with her friend on a busy street . |
| A man with a black shirt and brown long-sleeve shirt is standing near a graffiti that has come poles off around two . |
| A man , dressed with purple and black stands in bottoms while bandannas disbelief . |
| A man is looking at shoulder on a rack . |
| A man is about to fall . |

Table C.10: Interpolation between two prior samples with standard sampling for decoding. Dataset used is SNLI. In each cell, the first sentence and the last sentence correspond to the two sampled latent codes, and between are linearly interpolated samples.

| **VAE** | **VAE + FB** |
|---|---|
| A man is sitting on a bench . | Two men are playing basketball . |
| A man is sitting on a bench . | A man is playing a guitar . |
| A man is sitting on a bench . | The man is wearing a blue shirt . |
| A man is sitting on a bench . | The man is wearing a blue shirt . |
| A man is sitting on a bench . | Two men are playing basketball . |
| **MoG-VAE** | **MoG-VAE + infinilog** |
| The man is wearing a black shirt . | An older gentleman in a white shirt is walking in a parking lot . |
| A man is standing in front of a building . | A dog is running . |
| A man is standing in front of a building . | A woman is walking in a field . |
| A man is standing in front of a building . | A young girl in a red shirt is playing with a toy . |
| A man is standing in front of a building . | An older gentleman in a white shirt and white pants is standing on a ladder with a large ladder on his right hand |
| **Vamp-VAE** | **Vamp-VAE + infinilog** |
| A man is playing a guitar . | Women in a white dress and a man in a black shirt are standing in front of a microphone . |
| A man is playing a guitar . | Man in a blue shirt and jeans is sitting on a bench . |
| A man is playing a guitar . | The man is wearing a black shirt . |
| A man is playing a guitar . | People are walking down the street . |
| A man is playing a guitar . | Two men are playing a game of chess . |
| **FlowPrior** | **FlowPrior + FB** |
| Man in a blue shirt and blue jeans is sitting on a rock with a hammer . | Children are standing in the middle of a building with a man in a blue shirt and black pants . |
| Two young boys are playing in the snow . | A man is standing in the middle of a large building . |
| A dog is running through the snow . | The man is wearing a black shirt . |
| Two men are standing on a boat . | Two men are playing basketball . |
| A young man is sitting on a bench . | Girl in a blue shirt and black jacket standing on a bench . |

Table C.11: Greedy generation from prior samples.