Controllable Paraphrase Generation with a Syntactic Exemplar

Mingda Chen **Qingming Tang** **Sam Wiseman** **Kevin Gimpel**
Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
{mchen,qmtang,swiseman,kgimpel}@ttic.edu

Abstract

Prior work on controllable text generation usually assumes that the controlled attribute can take on one of a small set of values known a priori. In this work, we propose a novel task, where the syntax of a generated sentence is controlled rather by a sentential exemplar. To evaluate quantitatively with standard metrics, we create a novel dataset with human annotations. We also develop a variational model with a neural module specifically designed for capturing syntactic knowledge and several multitask training objectives to promote disentangled representation learning. Empirically, the proposed model is observed to achieve improvements over baselines and learn to capture desirable characteristics.[1]

1 Introduction

Controllable text generation has recently become an area of intense focus in the natural language processing (NLP) community. Recent work has focused both on generating text satisfying certain stylistic requirements such as being formal or exhibiting a particular sentiment (Hu et al., 2017; Shen et al., 2017; Ficler and Goldberg, 2017), as well as on generating text meeting structural requirements, such as conforming to a particular template (Iyyer et al., 2018; Wiseman et al., 2018).

These systems can be used in various application areas, such as text summarization (Fan et al., 2018), adversarial example generation (Iyyer et al., 2018), dialogue (Niu and Bansal, 2018), and data-to-document generation (Wiseman et al., 2018). However, prior work on controlled generation has typically assumed a known, finite set of values that the controlled attribute can take on. In this work, we are interested instead in the novel setting where the generation is controlled through an exemplar sentence (where any syntactically valid sentence is a valid exemplar). We will focus in particular on using a sentential exemplar to control the syntactic realization of a generated sentence. This task can benefit natural language interfaces to information systems by suggesting alternative invocation phrases for particular types of queries (Kumar et al., 2017). It can also bear on dialogue systems that seek to generate utterances that fit particular functional categories (Ke et al., 2018; Li et al., 2019).

To address this task, we propose a deep generative model with two latent variables, which are designed to capture semantics and syntax. To achieve better disentanglement between these two variables, we design multi-task learning objectives that make use of paraphrases and word order information. To further facilitate the learning of syntax, we additionally propose to train the syntactic component of our model with word noising and latent word-cluster codes. Word noising randomly replaces word tokens in the syntactic inputs based on a part-of-speech tagger used only at training time. Latent codes create a bottleneck layer in the syntactic encoder, forcing it to learn a more compact notion of syntax. The latter approach also learns interpretable word clusters. Empirically, these learning criteria and neural architectures lead to better generation quality and generally better disentangled representations.

To evaluate this task quantitatively, we manually create an evaluation dataset containing triples of a semantic exemplar sentence, a syntactic exemplar sentence, and a reference sentence incorporating the semantics of the semantic exemplar and the syntax of the syntactic exemplar. This dataset is created by first automatically finding syntactic exemplars and then heavily editing them by ensuring (1) semantic variation between the syntactic inputs and the references, (2) syntactic

[1]Code and data are available at github.com/mingdachen/syntactic-template-generation

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5972–5984
Florence, Italy, July 28 - August 2, 2019.

X: his teammates' eyes got an ugly, hostile expression.
Y: the smell of flowers was thick and sweet.
Z: the eyes of his teammates had turned ugly and hostile.

X: we need to further strengthen the agency's capacities.
Y: the damage in this area seems to be quite minimal.
Z: the capacity of this office needs to be reinforced even further.

Figure 1: Examples from our annotated evaluation dataset of paraphrase generation using semantic input X (red), syntactic exemplar Y (blue), and the reference output Z (black).

similarity between the syntactic inputs and the references, and (3) syntactic variation between the semantic input and references. Examples are shown in Figure 1. This dataset allows us to evaluate different approaches quantitatively using standard metrics, including BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004). As the success of controllability of generated sentences also largely depends on the syntactic similarity between the syntactic exemplar and the reference, we propose a "syntactic similarity" metric based on evaluating tree edit distance between constituency parse trees of these two sentences after removing word tokens.

Empirically, we benchmark the syntactically-controlled paraphrase network (SCPN) of Iyyer et al. (2018) on this novel dataset, which shows strong performance with the help of a supervised parser at test-time but also can be sensitive to the quality of the parse predictor. We show that using our word position loss effectively characterizes syntactic knowledge, bringing consistent and sizeable improvements over syntactic-related evaluation. The latent code module learns interpretable latent representations. Additionally, all of our models can achieve improvements over baselines. Qualitatively, we show that our models do suffer from the lack of an abstract syntactic representation, though we also show that SCPN and our models exhibit similar artifacts.

2 Related Work

We focus primarily on the task of paraphrase generation, which has received significant recent attention (Quirk et al., 2004; Prakash et al., 2016; Mallinson et al., 2017; Dong et al., 2017; Ma et al., 2018; Li et al., 2018). In order to disentangle the syntactic and semantic aspects of paraphrase generation we learn an explicit latent variable model using a variational autoencoder (VAE) (Kingma and Welling, 2014), which is now commonly applied to text generation (Bowman et al., 2016; Miao et al., 2016; Semeniuta et al., 2017; Serban et al., 2017; Xu and Durrett, 2018; Shen et al., 2019).

In seeking to control generation with exemplars, our approach relates to recent work in controllable text generation. Whereas much work on controllable text generation seeks to control distinct attributes of generated text (e.g., its sentiment or formality) (Hu et al., 2017; Shen et al., 2017; Ficler and Goldberg, 2017; Fu et al., 2018; Zhao et al., 2018; Fan et al., 2018, *inter alia*), there is also recent work which attempts to control structural aspects of the generation, such as its latent (Wiseman et al., 2018) or syntactic (Iyyer et al., 2018) template.

Our work is closely related to this latter category, and to the syntactically-controlled paraphrase generation of Iyyer et al. (2018) in particular, but our proposed model is different in that it simply uses a single *sentence* as a syntactic exemplar rather than requiring a supervised parser. This makes our setting closer to style transfer in computer vision, in which an image is generated that combines the content from one image and the style from another (Gatys et al., 2016). In particular, in our setting, we seek to generate a sentence that combines the semantics from one sentence with the syntax from another, and so we only require a pair of (unparsed) sentences. We also note recent, concurrent work that attempts to use sentences as exemplars in controlling generation (Wang et al., 2019) in the context of data-to-document generation (Wiseman et al., 2017).

Another related line of work builds generation upon sentential exemplars (Guu et al., 2018; Weston et al., 2018; Pandey et al., 2018; Cao et al., 2018; Peng et al., 2019) in order to improve the quality of the generation itself, rather than to allow for control over syntactic structures.

There has been a great deal of work in applying multi-task learning to improve performance on NLP tasks (Plank et al., 2016; Rei, 2017; Augenstein and Søgaard, 2017; Bollmann et al., 2018, *inter alia*). Some recent work used multi-task learning as a way of improving the quality or disentanglement of learned representations (Zhao et al., 2017; Goyal et al., 2017; Du et al., 2018; John et al., 2018).

Part of our evaluation involves assessing the dif-

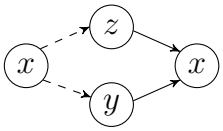

Figure 2: Graphical model. Dashed lines indicate the inference model. Solid lines indicate the generative model.

ferent characteristics captured in the semantic and syntactic encoders, relating them to work on learning disentangled representations in NLP, including morphological reinflection (Zhou and Neubig, 2017), sequence labeling (Chen et al., 2018), and sentence representations (Chen et al., 2019).

3 Methods

Given two sentences X and Y, our goal is to generate a sentence Z that follows the syntax of Y and the semantics of X. We refer to X and Y as the semantic template and syntactic template, respectively.

To solve this problem, we follow Chen et al. (2019) and take an approach based on latent-variable probabilistic modeling, neural variational inference, and multi-task learning. In particular, we assume a generative model that has two latent variables: y for semantics and z for syntax (as depicted in Figure 2). We refer to our model as a vMF-Gaussian Variational Autoencoder (VG-VAE). Formally, following the conditional independence assumptions in the graphical model, the joint probability $p_\theta(x, y, z)$ can be factorized as:

$$\begin{aligned} p_\theta(x, y, z) &= p_\theta(y)p_\theta(z)p_\theta(x \mid y, z) \\ &= p_\theta(y)p_\theta(z)\prod_{t=1}^{T} p_\theta(x_t \mid x_{1:t-1}, y, z), \end{aligned}$$

where x_t is the tth word of x and $p_\theta(x_t \mid x_{1:t-1}, y, z)$ is given by a softmax over a vocabulary of size V. Further details on the parameterization are given below.

When applying neural variational inference, we assume a factorized approximated posterior $q_\phi(y|x)q_\phi(z|x) = q_\phi(y, z|x)$, which has also been used in some prior work (Zhou and Neubig, 2017; Chen et al., 2018). Learning in VGVAE maximizes a lower bound of marginal log-likelihood:

$$\begin{aligned} \log p_\theta(x) \geq & \mathop{\mathbb{E}}_{\substack{y\sim q_\phi(y|x) \\ z\sim q_\phi(z|x)}} [\log p_\theta(x|\ z, y) \\ & - \log \frac{q_\phi(z|x)}{p_\theta(z)} - \log \frac{q_\phi(y|x)}{p_\theta(y)}] \\ = & \mathop{\mathbb{E}}_{\substack{y\sim q_\phi(y|x) \\ z\sim q_\phi(z|x)}} [\log p_\theta(x|z, y)] - KL(q_\phi(z|x)\|p_\theta(z)) \\ & - KL(q_\phi(y|x)\|p_\theta(y)) \end{aligned} \tag{1}$$

3.1 Parameterization

vMF Distribution. We choose a von Mises-Fisher (vMF) distribution for the y (semantic) latent variable. vMF can be regarded as a Gaussian distribution on a hypersphere with two parameters: μ and κ. $\mu \in \mathbb{R}^m$ is a normalized vector (i.e., $\|\mu\|_2 = 1$) defining the mean direction. $\kappa \in \mathbb{R}_{\geq 0}$ is often referred to as a concentration parameter analogous to the variance in a Gaussian distribution. We will assume $q_\phi(y|x)$ follows a vMF distribution and $p_\theta(y)$ follows the uniform distribution vMF$(\cdot, 0)$. We follow Davidson et al. (2018) and use an acceptance-rejection scheme to sample from the vMF distribution.

Gaussian Distribution. We assume $q_\phi(z|x)$ follows a Gaussian distribution $\mathcal{N}(\mu_\beta(x), \text{diag}(\sigma_\beta(x)))$ and that the prior $p_\theta(z)$ is $\mathcal{N}(0, I_d)$, where I_d is a $d \times d$ identity matrix.

Encoders. At test time, we want to have different combinations of semantic and syntactic inputs, which naturally suggests separate parameterizations for $q_\phi(y|x)$ and $q_\phi(z|x)$. Specifically, $q_\phi(y|x)$ is parameterized by a word averaging encoder followed by a three-layer feedforward neural network since it has been observed that word averaging encoders perform surprisingly well for semantic tasks (Wieting et al., 2016). $q_\phi(z|x)$ is parameterized by a bidirectional long short-term memory network (LSTM; Hochreiter and Schmidhuber, 1997) also followed by a three-layer feedforward neural network, where we concatenate the forward and backward vectors produced by the biLSTM and then take the average of these vectors.

Decoders. As shown in Figure 3, at each time step, we concatenate the syntactic variable z with the previous word's embedding as the input to the

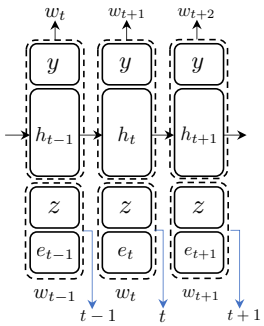

Figure 3: Diagram showing training of the decoder. Blue lines indicate the word position loss (WPL).

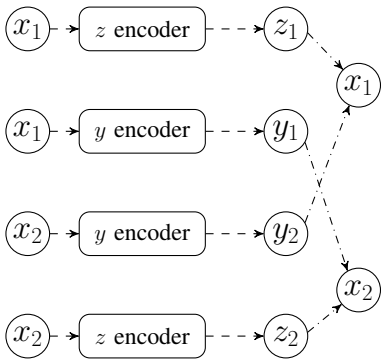

Figure 4: Diagram showing the training process when using the paraphrase reconstruction loss (dash-dotted lines). The pair (x_1, x_2) is a sentential paraphrase pair, the y's are the semantic variables corresponding to each x, and the z's are syntactic variables.

decoder and concatenate the semantic variable y with the hidden vector output by the decoder for predicting the word at the next time step. Note that the initial hidden state of the decoder is always set to zero.

3.2 Latent Codes for Syntactic Encoder

Since what we want from the syntactic encoder is only the syntactic structure of a sentence, using standard word embeddings tends to mislead the syntactic encoder to believe the syntax is manifested by the exact word tokens. An example is that the generated sentence often preserves the exact pronouns or function words in the syntactic input instead of making necessary changes based on the semantics. To alleviate this, we follow Chen and Gimpel (2018) to represent each word with a latent code (LC) for word clusters within the word embedding layer. Our goal is for this to create a bottleneck layer in the word embeddings, thereby forcing the syntactic encoder to learn a more abstract representation of the syntax. However, since our purpose is not to reduce model size (unlike Chen and Gimpel, 2018), we marginalize out the latent code to get the embeddings during both training and testing. That is,

$$e_w = \sum_{c_w} p(c_w) v_{c_w}$$

where c_w is the latent code for word w, v_{c_w} is the vector for latent code c_w, and e_w is the resulting word embedding for word w. In our models, we use 10 binary codes produced by 10 feedforward neural networks based on a shared word embedding, and then we concatenate these 10 individual cluster vectors to get the final word embeddings.

4 Multi-Task Learning

We now describe several additional training losses designed to encourage a clearer separation of information in the semantic and syntactic variables. These losses were also considered in (Chen et al., 2019), but in the context of learning sentence representations.

4.1 Paraphrase Reconstruction Loss

Our first loss, the paraphrase reconstruction loss (PRL), requires a dataset of sentence paraphrase pairs. The key assumption is that for a pair of paraphrastic sentences x_1, x_2, the semantics is shared but the syntax may differ. As shown in Figure 4, we swap the paraphrases to the semantic encoder during training but keep the input to the syntactic encoder to be the same. It is defined as

$$\begin{aligned} &\mathop{\mathbb{E}}_{\substack{y_2 \sim q_\phi(y|x_2) \\ z_1 \sim q_\phi(z|x_1)}} [\log p_\theta(x_1|y_2, z_1)] + \\ &\mathop{\mathbb{E}}_{\substack{y_1 \sim q_\phi(y|x_1) \\ z_2 \sim q_\phi(z|x_2)}} [\log p_\theta(x_2|y_1, z_2)] \end{aligned} \tag{2}$$

In the following experiments, unless explicitly noted, we will always include PRL as part of the model training and will discuss its effect in Section 7.1.

4.2 Word Position Loss

Since word ordering is relatively unimportant for semantic similarity (Wieting et al., 2016), we assume it is more relevant to the syntax of a sentence than to its semantics. Based on this, we introduce a word position loss (WPL). As shown in

Word tokens	something	funny	happened	this	…
POS tags	NN	JJ	VBD	DT	
	eyebrow	snotty	locked	an	
	loss	muddy	cackled	another	
	concern	green	rebuked	those	
	smoke	spiteful	nodded	all	
	…	…	…	…	

Figure 5: An example of word noising. For each word token in the training sentences, we randomly replace it with other words that share the same POS tags.

Figure 3, WPL is computed by predicting the position at each time step based on the concatenation of word embeddings with the syntactic variable z. That is,

$$\text{WPL} \overset{\text{def}}{=\!=} \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)} \left[\sum_t \log \text{softmax}(f([e_t; z]))_t \right]$$

where $\text{softmax}(\cdot)_t$ indicates the probability at position t. Empirically, we observe that adding WPL to both the syntactic encoder and decoder improves performance, so we always use it in our experiments unless otherwise indicated.

5 Training

5.1 KL Weight

As observed in previous work (Alemi et al., 2017; Bowman et al., 2016; Higgins et al., 2016), the weight of the KL divergence in Equation 1 can be important when learning with latent variables. We attach weights to the KL divergence in Equation 1 and tune them based on development set performance.

5.2 Word Noising via Part-of-Speech Tags

In practice, we often observe that the syntactic encoder tends to remember word types instead of learning syntactic structures. To provide a more flexible notion of syntax, we add word noising (WN) based on part-of-speech (POS) tags. More specifically, we tag the training set using the Stanford POS tagger (Toutanova et al., 2003). Then we group the word types based on the top two most frequent tags for each word type. During training, as shown in Figure 5, we noise the syntactic inputs by randomly replacing word tokens based on the groups and tags we obtained. This provides our framework many examples of word interchangeability based on POS tags, and discourages the syntactic encoder from memorizing the word types in the syntactic input. When using WN, the probability of noising a word is tuned based on development set performance.

6 Experiments

6.1 Training Setup

For training with the PRL, we require a training set of sentential paraphrase pairs. We use ParaNMT (Wieting and Gimpel, 2018), a dataset of approximately 50 million paraphrase pairs. To ensure there is enough variation between paraphrases, we filter out paraphrases with high BLEU score (Papineni et al., 2002) between the two sentences in each pair, which leaves us with around half a million paraphrases as our training set. All hyperparameter tuning is based on the BLEU score on the development set (see appendix for more details).

6.2 Evaluation Dataset and Metrics

To evaluate models quantitatively, we manually annotate 1300 instances based on paraphrase pairs from ParaNMT independent from our training set. Each instance in the annotated data has three sentences: semantic input, syntactic input, and reference, where the semantic input and the reference can be seen as human generated paraphrases and the syntactic input shares its syntax with the reference but is very different from the semantic input in terms of semantics. The differences among these three sentences ensure the difficulty of this task. Figure 1 shows examples.

The annotation process involves two steps. We begin with a paraphrase pair $\langle u, v \rangle$. First, we use an automatic procedure to find, for each sentence u, a syntactically-similar but semantically-different other sentence t. We do this by seeking sentences t with high edit distance of predicted POS tag sequences and low BLEU score with u. Then we manually edit all three sentences to ensure (1) strong semantic match and large syntactic variation between the semantic input u and reference v, (2) strong semantic match between the syntactic input t and its post-edited version, and (3) strong syntactic match between the syntactic input t and the reference v. We randomly pick 500 instances as our development set and use the remaining 800 instances as our test set. We perform additional manual filtering and editing of the test set to ensure quality.

For evaluation, we consider two categories of

	BLEU (↑)	ROUGE-1 (↑)	ROUGE-2 (↑)	ROUGE-L (↑)	METEOR (↑)	ST (↓)
Return-input baselines						
Semantic input	18.5	50.6	23.2	47.7	28.8	12.0
Syntactic input	3.3	24.4	7.5	29.1	12.1	5.9
Our work						
VGVAE	3.5	24.8	7.3	29.7	12.6	10.6
VGVAE + WPL	4.5	26.5	8.2	31.5	13.3	10.0
VGVAE + LC	3.3	24.0	7.2	29.4	12.5	9.1
VGVAE + LC + WPL	5.9	29.1	10.2	33.0	14.5	9.0
VGVAE + WN	13.0	43.2	20.2	47.0	23.8	6.8
VGVAE + WN + WPL	13.2	43.4	20.3	47.0	23.9	6.7
VGVAE + LC + WN + WPL	13.6	44.7	21.0	48.3	24.8	6.7
Prior work using supervised parsers						
SCPN + template	17.8	47.9	22.8	48.5	27.3	9.9
SCPN + full parse	19.2	50.4	26.1	53.5	28.4	5.9

Table 1: Test results. The final metric (ST) measures the syntactic match between the output and the reference.

automatic evaluation metrics, designed to capture different components of the task. To measure roughly the amount of semantic content that matches between the predicted output and the reference, we report BLEU score (BL), METEOR score (MET; Banerjee and Lavie, 2005) and three ROUGE scores, including ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-L (R-L). Even though these metrics are not purely based on semantic matching, we refer to them in this paper as "semantic metrics" to differentiate them from our second metric category, which we refer to as a "syntactic metric". For the latter, to measure the syntactic similarity between generated sentences and the reference, we report the syntactic tree edit distance (ST). To compute ST, we first parse the sentences using Stanford CoreNLP (Manning et al., 2014), and then compute the tree edit distance (Zhang and Shasha, 1989) between constituency parse trees after removing word tokens.

6.3 Baselines

We report results for three baselines. The first two baselines directly output the corresponding syntactic or semantic input for each instance. For the last baseline, we consider SCPN (Iyyer et al., 2018). As SCPN requires parse trees for both the syntactic and semantic inputs, we follow the process in their paper and use the Stanford shift-reduce constituency parser (Manning et al., 2014) to parse both, then use the parsed sentences as inputs to SCPN. We report results for SCPN when using only the top two levels of the parse as input (template) and using the full parse as input (full parse).

6.4 Results

As shown in Table 1, simply outputting the semantic input shows strong performance across the BLEU, ROUGE, and METEOR scores, which are more relevant to semantic similarity, but shows much worse performance in terms of ST. On the other hand, simply returning the syntactic input leads to lower BLEU, ROUGE, and METEOR scores but also a very strong ST score. These trends provide validation of the evaluation dataset, as they show that the reference and the semantic input match more strongly in terms of their semantics than in terms of their syntax, and also that the reference and the syntactic input match more strongly in terms of their syntax than in terms of their semantics. The goal in developing systems for this task is then to produce outputs with higher semantic metric scores than the syntactic input baseline and simultaneously higher syntactic scores than the semantic input baseline.

Among our models, adding WPL leads to gains across both the semantic and syntactic metric scores. The gains are much larger without WN, but even with WN, adding WPL improves nearly all scores. Adding LC typically helps the semantic metrics (at least when combined with WPL) without harming the syntactic metric (ST). We see the largest improvements, however, by adding WN, which uses an automatic part-of-speech tagger at training time only. Both the semantic and syntactic metrics increase consistently with WN, as the syntactic variable is shown many examples of

	BL	R-1	R-2	R-L	MET	ST
VGVAE w/o PRL	2.0	23.4	4.3	26.4	11.3	11.8
VGVAE w/ PRL	3.5	24.8	7.3	29.7	12.6	10.6

Table 2: Test results when including PRL.

	BL	R-1	R-2	R-L	MET	ST
VGVAE w/o WPL	3.5	24.8	7.3	29.7	12.6	10.6
Dec. hidden state	3.6	24.9	7.3	29.7	12.6	10.5
Enc. emb.	3.9	26.1	7.8	31.0	12.9	10.2
Dec. emb.	4.1	26.3	8.1	31.3	13.1	10.1
Enc. & Dec. emb.	4.5	26.5	8.2	31.5	13.3	10.0

Table 3: Test results with WPL at different positions.

word interchangeability based on POS tags.

While the SCPN yields very strong metric scores, there are several differences that make the SCPN results difficult to compare to those of our models. In particular, the SCPN uses a supervised parser both during training and at test time, while our strongest results merely require a POS tagger and only use it at training time. Furthermore, since ST is computed based on parse trees from a parser, systems that explicitly use constituency parsers at test time, such as SCPN, are likely to be favored by such a metric. This is likely the reason why SCPN can match the syntactic input baseline in ST. Also, SCPN trains on a much larger portion of ParaNMT.

We find large differences in metric scores when SCPN only uses a parse template (i.e., the top two levels of the parse tree of the syntactic input). In this case, the results degrade, especially in ST, showing that the performance of SCPN depends on the quality of the input parses. Nonetheless, the SCPN results show the potential benefit of explicitly using a supervised constituency parser at both training and test time. Future work can explore ways to combine syntactic parsers with our models for more informative training and more robust performance.

7 Analysis

7.1 Effect of Multi-Task Training

Effect of Paraphrase Reconstruction Loss. We investigate the effect of PRL by removing PRL from training, which effectively makes VGVAE a variational autoencoder. As shown in Table 2, making use of pairing information can improve performance both in the semantic-related metrics and syntactic tree edit distance.

Effect of Position of Word Position Loss. We also study the effect of the position of WPL by (1) using the decoder hidden state, (2) using the concatenation of word embeddings in the syntactic encoder and the syntactic variable, (3) using the concatenation of word embeddings in the decoder and the syntactic variable, or (4) adding it on both the encoder embeddings and decoder word embeddings. Table 3 shows that adding WPL on hidden states can help improve performance slightly but not as good as adding it on word embeddings. In practice, we also observe that the value of WPL tends to vanish when using WPL on hidden states, which is presumably caused by the fact that LSTMs have sequence information, making the optimization of WPL trivial. We also observe that adding WPL to both the encoder and decoder brings the largest improvement.

	Semantic var.	Syntactic var.
VGVAE	64.8	14.5
VGVAE + WPL	65.2	10.5
VGVAE + LC	67.2	29.0
VGVAE + LC + WPL	67.9	8.5
VGVAE + WN	71.1	10.2
VGVAE + WN + WPL	72.9	9.8
VGVAE + LC + WN + WPL	74.3	7.4

Table 4: Pearson correlation (%) for STS Benchmark test set.

7.2 Encoder Analysis

To investigate what has been learned in the encoder, we evaluate $q_\phi(y|x)$ and $q_\phi(z|x)$ on both semantic similarity tasks and syntactic similarity tasks and also inspect the latent codes.

Semantic Similarity. We use cosine similarity between two variables encoded by the inference networks as the predictions and then compute Pearson correlations on the STS Benchmark test set (Cer et al., 2017). As shown in Table 4, the semantic variable y always outperforms the syntactic variable z by a large margin, suggesting that different variables have captured different information. Every time when we add WPL the differences in performance between the two variables increases. Moreover, the differences between these two variables are correlated with the performance of models in Table 1, showing that a better generation system has a more disentangled latent representation.

Syntactic Similarity. We use the syntactic evaluation tasks from Chen et al. (2019) to evaluate the syntactic knowledge encoded in the encoder. The tasks are based on a 1-nearest-neighbor con-

	Semantic var.		Syntactic var.	
	F_1	Acc.	F_1	Acc.
Random	19.2	12.9	-	-
Best	71.1	62.3	-	-
VGVAE	20.7	24.9	25.9	28.8
VGVAE + WPL	21.2	25.3	31.1	33.3
VGVAE + LC	21.6	25.5	29.0	32.4
VGVAE + LC + WPL	18.9	23.5	31.2	33.5
VGVAE + WN	20.6	18.1	28.4	30.4
VGVAE + WN + WPL	20.0	24.6	43.7	40.8
VGVAE + LC +WN + WPL	20.3	24.8	43.7	40.9

Table 5: Labeled F_1 score (%) and accuracy (%) on syntactic similarity tasks from Chen et al. (2019).

12	does must could shall do wo 's did ai 'd 'll should
451	watching wearing carrying thrown refuse drew
11	? : * ≫ ! ;) . " , '
18	maybe they because if where but we when how
41279	elvish festive freeway anteroom jennifer terrors
10	well ⟨unk⟩ anyone okay now everybody someone
165	supposedly basically essentially rarely officially
59	using on by into as the with within under quite

Table 6: Examples of learned word clusters. Each row is a different clusters. Numbers in the first column indicate the number of words in that cluster.

stituency parser or POS tagger. To understand the difficulty of these two tasks, Table 5 shows results for two baselines. "Random" means randomly pick candidates as predictions. The second baseline ("Best") is to compute the pairwise scores between the test instances and the sentences in the candidate pool and then take the maximum values. It can be seen as the upper bound performance for these tasks.

As shown in Table 5, similar trends are observed as in Tables 1 and 4. When adding WPL or WN, there is a boost in the syntactic similarity for the syntactic variable. Adding LC also helps the performance of the syntactic variable slightly.

Latent Code Analysis. We look into the learned word clusters by taking the argmax of latent codes and treating it as the cluster membership of each word. Although these are not the exact word clusters we would use during test time (because we marginalize over the latent codes), it provides us intuition on what individual cluster vectors have contributed to the final word embeddings. As shown in Table 6, the words in the first and last rows are mostly function words. The second row has verbs. The third row has special symbols. The fourth row also has function words but somewhat different from the first row. The fifth row is a large cluster populated by content words, mostly nouns and adjectives. The sixth row has words that are not very important semantically and the seventh row has mostly adverbs. We also observe that the size of clusters often correlates with how strongly it relates to topics. In Table 6, clusters that have size under 20 are often function words while the largest cluster (5th row) has words with the most concrete meanings.

We also compare the performance of LC by using a single latent code that has 50 classes. The results in Table 7 show that it is better to use smaller number of classes for each cluster instead of using a cluster with a large number of classes.

	BL	R-1	R-2	R-L	MET	ST
LC	13.6	44.7	21.0	48.3	24.8	6.7
Single LC	12.9	44.2	20.3	47.4	24.1	6.9

Table 7: Test results when using a single code.

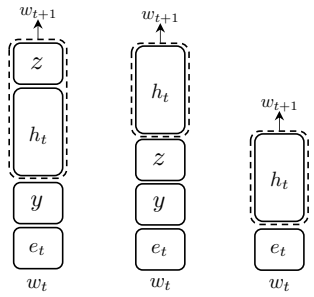

Figure 6: Variants of decoder. Left (SWAP): we swap the position of variable y and z. Middle (CONCAT): we concatenate word embedding with y and z as input to decoder. Right (INIT): we use word embeddings as input to the decoder and use the concatenation of y and z to compute the initial hidden state of the decoder.

7.3 Effect of Decoder Structure

As shown in Figure 6, we evaluate three variants of the decoder, namely INIT, CONCAT, and SWAP. For INIT, we use the concatenation of semantic variable y and syntactic variable z for computing the initial hidden state of decoder and then use the word embedding as input and hidden state to predict the next word. For CONCAT, we move both y and z to the input of the decoder and use the concatenation of these two variables as input to the decoder and use the hidden state for predicting the next word. For SWAP, we swap the position of y and z to use the concatenation of y and word embeddings as input to the decoder and the concatenation of z and hidden states as output for predicting the next word. Results for these three settings are shown in Table 9. INIT performs

Semantic input	Syntactic input	Reference	SCPN + full parse	Our best model
don't you think that's a quite aggressive message?	that's worth something, ain't it?	that's a pretty aggressive message, don't you think?	that's such news, don't you?	that's impossible message, aren't you?
if i was there, i would kick that bastard in the ass.	they would've delivered a verdict in your favor.	i would've kicked that bastard out on his ass.	you'd have kicked the bastard in my ass.	she would've kicked the bastard on my ass.
with luck, it may turn out you're right.	of course, i could've done better.	if lucky, you will be proved correct.	with luck, i might have gotten better.	of course, i'll be getting lucky.
they can't help, compassion is unbearable.	love is straightforward and it is lasting.	their help is impossible and compassion is insufferable.	compassion is unbearable but it is excruciating.	compassion is unacceptable and it is intolerable.
her yelling sounds sad.	she looks beautiful. shining like a star.	she sounds sad. yelling like that.	she's sad. screaming in the air.	she sounds sad. screaming like a scream.
me, scare him?	how dare you do such thing?	how can i scare him?	why do you have such fear?	why do you scare that scare?

Table 8: Examples of generated sentences.

	BL	R-1	R-2	R-L	MET	ST
VGVAE	4.5	26.5	8.2	31.5	13.3	10.0
INIT	3.5	22.7	6.0	24.9	9.8	11.5
CONCAT	4.0	23.9	6.6	27.9	11.2	10.9
SWAP	4.3	25.6	7.5	30.4	12.5	10.5

Table 9: Test results with decoder variants.

the worst across the three settings. Both CONCAT and SWAP have variables in each time step in the decoder, which improves performance. SWAP arranges variables in different positions in the decoder and further improves over CONCAT in all metrics.

7.4 Generated Sentences

We show several generated sentences in Table 8. We observe that both SCPN and our model suffer from the same problems. When comparing syntactic input and results from both our models and SCPN, we find that they are always the same length. This can often lead to problems like the first example in Table 8. The length of the syntactic input is not sufficient for expressing the semantics in the semantic input, which causes the generated sentences from both models to end at "you?" and omit the verb "think". Another problem is in the consistency of pronouns between the generated sentences and the semantic inputs. An example is the second row in Table 8. Both models alter "i" to be either "you" or "she" while the "kick that bastard in the ass" becomes "kicked the bastard in my ass".

We found that our models sometimes can generate nonsensical sentences, for example the last row in Table 8. while SCPN, which is trained on a much larger corpus, does not have this problem. Also, our models can sometimes be distracted by the word tokens in the syntactic input as shown in the 3rd row in Table 8, where our model directly copies "of course" from the syntactic input while since SCPN uses a parse tree, it outputs "with luck". In some rare cases where the function words in both syntactic inputs and the references are the exactly the same, our models can perform better than SCPN, e.g., the last two rows in Table 8. Generated sentences from our model make use of the word tokens "and" and "like" while SCPN does not have access to this information and generates inferior sentences.

8 Conclusion

We proposed a novel setting for controlled text generation, which does not require prior knowledge of all the values the control variable might take on. We also proposed a variational model accompanied with a neural component and multiple multi-task training objectives for addressing this task. The proposed approaches do not rely on a test-time parser or tagger and outperform our baselines. Further analysis shows the model has learned both interpretable and disentangled representations.

Acknowledgments

We would like to thank the anonymous reviewers, NVIDIA for donating GPUs used in this research, and Google for a faculty research award to K. Gimpel that partially supported this research.

References

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. 2017. Deep variational information bottleneck. In *Proceedings of ICLR*.

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 341–346. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72. Association for Computational Linguistics.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 19–24. 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. Association for Computational Linguistics.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14. Association for Computational Linguistics.

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

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. A multi-task approach for disentangling syntax and semantics in sentence representations. 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 2453–2464, Minneapolis, Minnesota. Association for Computational Linguistics.

Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M. Tomczak. 2018. Hyperspherical variational auto-encoders. *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)*.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886. Association for Computational Linguistics.

Jiachen Du, Wenjie Li, Yulan He, Ruifeng Xu, Lidong Bing, and Xuan Wang. 2018. Variational autoregressive decoder for neural response generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3154–3163.

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.

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.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423.

Anirudh Goyal Alias Parth Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, pages 6713–6723.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

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

Mohamed, and Alexander Lerchner. 2016. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of ICLR*.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia. PMLR.

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. Association for Computational Linguistics.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for non-parallel text style transfer. *arXiv preprint arXiv:1808.04339*.

Pei Ke, Jian Guan, Minlie Huang, and Xiaoyan Zhu. 2018. Generating informative responses with controlled sentence function. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1499–1508, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of ICLR*.

Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Bjorn Hoffmeister, Markus Dreyer, Stanislav Peshterliev, Ankur Gandhe, Denis Filiminov, Ariya Rastrow, et al. 2017. Just ASK: building an architecture for extensible self-service spoken language understanding. In *1st Workshop on Conversational AI at NIPS 2017 (NIPS-WCAI)*.

Juntao Li, Lisong Qiu, Bo Tang, Dongmin Chen, Dongyan Zhao, and Rui Yan. 2019. Insufficient data can also rock! learning to converse using smaller data with augmentation. In *Thirty-Third AAAI Conference on Artificial Intelligence*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li, and Xuancheng Ren. 2018. Query and output: Generating words by querying distributed word representations for paraphrase generation. 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 196–206. Association for Computational Linguistics.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1727–1736, New York, New York, USA. PMLR.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:373–389.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1329–1338, Melbourne, Australia. Association for Computational Linguistics.

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*.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. 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 2555–2565, Minneapolis, Minnesota. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934. The COLING 2016 Organizing Committee.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130. Association for Computational Linguistics.

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. 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. Association for Computational Linguistics.

Dinghan Shen, Asli Celikyilmaz, Yizhe Zhang, Liqun Chen, Xin Wang, Jianfeng Gao, and Lawrence Carin. 2019. Towards generating long and coherent text with multi-level latent variable models.

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.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Wentao Wang, Zhiting Hu, Zichao Yang, Haoran Shi, Frank Xu, and Eric Xing. 2019. Toward unsupervised text content manipulation. *arXiv preprint arXiv:1901.09501*.

Jason Weston, Emily Dinan, and Alexander Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 87–92, Brussels, Belgium. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceedings of ICLR*.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462. Association for Computational Linguistics.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

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

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513. Association for Computational Linguistics.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262.

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, and Yann LeCun. 2018. Adversarially Regularized Autoencoders. In *Proceedings of ICML*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 310–320. Association for Computational Linguistics.

A Appendices

A.1 Hyperparameters

We use 100 dimensional word embeddings in both encoders and 100 dimensional word embeddings for the decoder. These word embeddings are all initialized by GloVe vectors (Pennington et al., 2014). The syntactic encoder uses 100 dimensions per direction and the decoder is a 100 dimensional unidirectional LSTM. When performing early stopping, we use greedy decoding. During testing, we use beam search with size of 10.