Generating Narrative Text in a Switching Dynamical System

Noah Weber [3,1], Leena Shekhar [1*], Heeyoung Kwon[1]
Niranjan Balasubramanian[1], Nathanael Chambers [2]
[1]Stony Brook University
{lshekhar,heekwon,niranjan}@cs.stonybrook.edu
[2]United States Naval Academy
nchamber@usna.edu
[3]Johns Hopkins University
nweber6@jhu.edu

Abstract

Early work on narrative modeling used explicit plans and goals to generate stories, but the language generation itself was restricted and inflexible. Modern methods use language models for more robust generation, but often lack an explicit representation of the scaffolding and dynamics that guide a coherent narrative. This paper introduces a new model that integrates explicit narrative structure with neural language models, formalizing narrative modeling as a *Switching Linear Dynamical System* (SLDS). A SLDS is a dynamical system in which the latent dynamics of the system (i.e. how the state vector transforms over time) is controlled by top-level discrete *switching* variables. The switching variables represent narrative structure (e.g., sentiment or discourse states), while the latent state vector encodes information on the current state of the narrative. This probabilistic formulation allows us to control generation, and can be learned in a semi-supervised fashion using both labeled and unlabeled data. Additionally, we derive a Gibbs sampler for our model that can "fill in" arbitrary parts of the narrative, guided by the switching variables. Our filled-in (English language) narratives outperform several baselines on both automatic and human evaluations.

1 Introduction

A narrative is a textualized sequence of events that serves as a coherent outline for an actual story (Prince, 2003). Effective narratives are typically built on top of higher level *narrative scaffolds*[1] which specify at an abstract level how the story should evolve along different dimensions. Example scaffolds include descriptions of the emotional trajectory of a story (Vonnegut, 1981; Reagan et al., 2016), the goals of characters throughout the story (Meehan, 1977; Turner, 1993), or the abstract types of events that may occur (Martin et al., 2018). The parts of a scaffold are generic, and like Propp's originally proposed narrative functions (Propp, 1928), can be reused across stories. To be fully reusable, one needs to go beyond just identifying *what* the elements of the narrative scaffold are, and also indicate *how* each scaffold element changes the properties of the current story state. We refer to the explication of these transformation/transitions as the *narrative dynamics*.

Prior work on automatic narrative generation has a rich history of modeling both *narrative scaffolds* and *narrative dynamics* (Meehan, 1977; Lebowitz, 1985; Turner, 1993; Riedl and Young, 2006, 2010a). The modeling of both narrative scaffold and dynamics often imbued these systems with a greater degree of control for the user in generating stories, allowing users to flexibly specify desired outcomes or plot points (or more generally, the state of the narrative) that should be achieved at certain sections of the story. Constrained generation (for example, constraining the story to start and end with particular sentences, such as that given in Table 1), is an ability these systems often gained for free through the modeling of dynamics.

Though successful in this regard, this success has only been realized in closed domains, where

Tom didn't know why his internet speed was so slow.
Tom wasn't sure what to do with his computer.
He thought he would fix it himself.
Tom was surprisingly good.
Tom was happy to be surfing the internet again

Table 1: A sample filled in narrative generated by our SLDS model given the first and last sentences as input (grayed out), the middle 3 sentences are imputed by our model (bold).

[*]Author now at Microsoft

[1]We use the term scaffold as an umbrella term to cover many types of plans and structures that underlie stories.

the narrative scaffolds can be specified in a limited ontology and the dynamics operations can be written by hand (such as e.g. the action schemata of Riedl and Young (2010a)). Neural generation has since helped scale to open domains (Roemmele and Gordon, 2015; Khalifa et al., 2017) but not with the same level of control over the narrative. Several recent works have looked at adding the narrative scaffolding component back into neural text generating systems (Fan et al.; Martin et al., 2018; Yao et al., 2019; Xu et al., 2018; Fan et al., 2019). These systems however still do not utilize an explicit model of narrative dynamics, and are thus restricted in the controllability aspect.

In this work, we show how the insight of modeling the structure of a narrative along with general purpose dynamics can be combined with modern neural network based language models. We do this by explicitly modeling the narrative state with a latent vector, and modeling how this state transforms over time as a *Switching Linear Dynamical System* (SLDS). We show how this formulation captures the concepts of narrative dynamics and scaffolds in a way compatible with current neural generation systems. Finally we show that, by explicitly modeling these dynamics, our models obtain the ability to "fill in" narratives; all without being explicitly trained to do so and with no further training required. We evaluate our model with both human evaluation and several automatic measures[2] and show that our model outperforms several strong baselines.

2 A Switching Dynamical System for Narrative Generation

In this section, we give a brief overview of Switching Dynamical systems and how they can be used to capture both a scaffold of the narrative as well as the narrative dynamics. We then describe in detail the components of our model and its relation to existing models.

2.1 Narrative Dynamics in a Dynamical System

The specifics of the narrative (characters, setting, etc.), will differ between stories, but as Propp (1928) notes, the way they transition to the next point in the narrative (what we refer to as "narrative dynamics") is often shared. Let's say that, as done often, we represent the 'narrative specifics' at time step[3] i with a latent vector Z_i. A natural way to explicitly model how this state evolves over time that fits with the above observation is as a *Linear Dynamical System*:

$$Z_{i+1} = AZ_i + \epsilon \, ; \epsilon \sim \mathcal{N}(0, \Sigma)$$

Where A is a matrix, shared across all narratives, and Σ is a noise term that takes into consideration idiosyncrasies different narratives will have[4]. The fact that the shared transition matrix A is linear means that narratives will have linearly analogous trajectories through time, despite having different details (comparable to stories with different settings but matching structures such as *Ran/King Lear*, *Ulysses/Odyssey*, etc). Of course, the fatal flaw of the model is that it assumes there exists only one transition matrix, and thus only one possible way to transition through a narrative!

2.2 Narrative Scaffolds as Switching Variables

A more fitting model would thus be a *Switching Linear Dynamical System* (Ackerson and Fu, 1970; Chang and Athans, 1978; Murphy, 1998). In an SLDS, we assume there exists a set of K different sets of dynamics, $\{(A_1, \Sigma_1), ...(A_K, \Sigma_K)\}$. At time step $i+1$, *one* of these sets of dynamics is used. The one used depends on the value of a discrete variable at time step $i+1$ called the switching variable, $S_{i+1} \in \{1, ...K\}$:

$$Z_{i+1} = A_{S_{i+1}} Z_i + \epsilon \, ; \epsilon \sim \mathcal{N}(0, \Sigma_{S_{i+1}})$$

There is a switching variable S_i associated with each time step. The switching variable value itself evolves over time by a prior Markov process, $P(S_{i+1}|S_i)$[5]. This top level chain of switching variables thus forms our *narrative scaffold*, indicating *what* transitions we must go through in the narrative, with the dynamics matrices indicating *how* they transition.

2.3 Narrative Scaffold - Emotional Trajectory

What the switching variables actually represent can be chosen by the user. Straightforward narrative

[2]Evaluation in this paper is done on English text data

[3]In our case, we take each sentence in the narrative to be a different timestep. Different levels of granularity for a timestep may be more befitting for other domains.

[4]Note that a bias term may also be added here (we do this in our implementation). We leave the bias off here for clarity

[5]Other ways to formulate this transformation are also possible (Barber, 2006; Linderman et al., 2016). The Markov assumption is a common one and we use it here for simplicity.

scaffolds include event sequences (Martin et al., 2018), keywords (Yao et al., 2019), or latent template ids (Wiseman et al., 2018). More complex but potentially more informative scaffolds may be created using concepts such as story grammar nonterminals (Lakoff, 1972; Thorndyke, 1977), or character action taken throughout a story (Riedl and Young, 2010b).

In our work, we use the sentiment trajectory of the narrative as the scaffold. That is, each S_i for a sentence indicates the overall coarse sentiment of the sentence (Positive, Negative, or Neutral). Though simple, the overall sentiment trajectory of a narrative is important in defining the high level 'shape' of a narrative often shared among different narratives (Vonnegut, 1981; Reagan et al., 2016). Furthermore, sentiment trajectory has been shown to be fairly useful in story understanding tasks (Chaturvedi et al., 2017; Liu et al., 2018). We discuss in the conclusion future directions for using different types of scaffolds.

2.4 The Full Model

The final component of the model is a conditional language model that generates sentence i conditioned on the current Z_i, and all previous sentences, $X_{:i}$. Generation continues until an <eos> is reached. This conditional language model may be parameterized as desired, but in this work, we parameterize it as an RNN neural network language model.

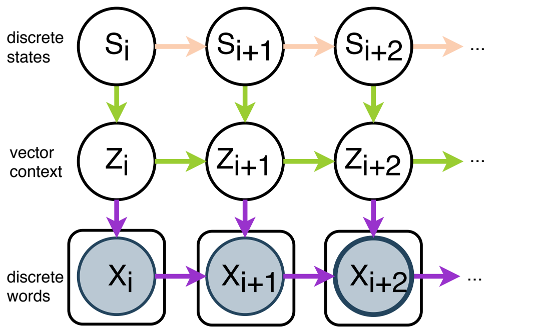

Figure 1: SLDS Generative model-S_i is a discrete state (sentiment of a sentence in a multi-sentence narrative). Z_i is a continuous latent vector that is conditioned on to generate the ith sentence in the narrative , X_i. The dynamics of the narrative are completely captured in the dynamical system controlling the latent vector Z. How to transition from Z_i to Z_{i+1} is determined by the state variable S_{i+1}. Arrows from X_i to X_{i+2} have been left out for clarity.

The graphical model for our SLDS is pictured in Figure 1. The model consists of three sets of variables: (1) Switching variables $S_1, ..., S_N$, (2) Latent state variables $Z_1, ..., Z_N$ capturing the details of the narrative at sentence i, (3) The sentences themselves $X_1, ...X_N$, where each sentence X_i has n_i words, $x^i_1, ...x^i_{n_i}$. The joint over all variables factorizes as below into the following components ($X_{:i}$ stands for all sentences before X_i):

$$P(\mathbf{S}, \mathbf{Z}, \mathbf{X}) = (\prod_i^N \underbrace{P(X_i|Z_i, X_{:i})}_{❸})$$
$$(\prod_i^N \underbrace{P(Z_i|Z_{i-1}, S_i)}_{❷})(\prod_i^N \underbrace{P(S_i|S_{i-1})}_{❶})$$

❶ **Narrative Scaffold Planner**: The factor $P(S_i|S_{i-1})$ is a transition matrix, which we calculate via count based statistics from training. It is fed in as prior knowledge and fixed.
❷ **Narrative Dynamics Network**: The factor $P(Z_i|Z_{i-1}, S_i)$ is determined like a switching linear dynamical system:

$$Z_i = A_{S_i} Z_{i-1} + B_{S_i}\epsilon, \; ; \epsilon \sim \mathcal{N}(0, I)$$

which is equivalent to drawing Z_i from a Normal distribution with mean $A_{S_i} Z_{i-1}$ and variance $B_{S_i} B^T_{S_i}$.
❸ **Conditional Language model**: The factor $P(X_i|Z_i, X_{:i})$ is parameterized by an RNN language model conditioned on the latent Z_i.

3 Learning and Posterior Inference

Due to the conditionals parameterized by neural networks we use amortized variational inference in a manner similar to Variational AutoEncoders (Kingma and Welling, 2013), both to learn an approximate posterior $q(S, Z|X)$ and to learn the generative model parameters by maximizing a lower bound on the data likelihood (ELBO). We assume that the approximate posterior factorizes as follows:

$$q(\mathbf{S}, \mathbf{Z}|\mathbf{X}) =$$
$$(\prod_i^N q(S_i|\mathbf{X}))(\prod_i^N q(Z_i|Z_{i-1}, S_i, X_{:i}, X_i))$$

Like in VAEs, computing these individual factors is done through a parameterized function called the *inference* or *recognition* network whose parameters are trained jointly with the generative model.

In our case there are two forms for the factors in our posterior: (1) The first form, $q(S_i|\mathbf{X}) = q_{S_i}$ is parameterized by a classifier that takes in the set of sentences $\mathbf{X}$ and outputs a categorical distribution over the switching variables. (2) The second form, $q(Z_i|Z_{i-1}, S_i, X_{:i}, X_i) = q_{Z_i}$ is realized by functions $f_\mu(Z_{i-1}, S_i, X_{:i}, X_i)$ and $f_\sigma(Z_{i-1}, S_i, X_{:i}, X_i)$ that output the mean and variance, respectively, of a Gaussian over Z_i.

Borrowing terminology from VAEs, the approximate posterior (the factors given above) act as an 'encoder', while the generative model from the previous section can be seen as the 'decoder'. This type of training has been previously used in (Krishnan et al., 2015, 2017; Fraccaro et al., 2016, 2017; Karl et al., 2017).

3.1 Lower bound formula & exact training algorithm

As mentioned previously, we optimize all parameters (including the variational factor functions) by optimizing a lower bound on the data likelihood. The model may be trained either with supervision labels for the switching states (in our case, sentiment labels) or without supervised labels.

If one is training *without the sentiment labels*, then the lower bound on the marginal likelihood (and thus our optimization objective) may be written as follows:

$$L = \mathbb{E}_{S_1,..S_N \sim q_{S_i}}\left[M - \sum_i^N KL(q_{S_i}||p(S_i|S_{i-1}))\right]$$

$$\text{where, } M = \mathbb{E}_{Z_1,..Z_N \sim q_{Z_i}}\Bigg[\sum_i^N (\log p(X_i|Z_i) - KL(q_{Z_i}||p(Z_i|Z_{i-1}, S_i)))\Bigg]$$

The derivation for this objective is identical to that found in (Krishnan et al., 2017; Fraccaro et al., 2016), and simply relies on using properties of iterated expectations. All expectations are estimated with Monte Carlo samples.

If training *with the sentiment labels* $S_1, ..., S_N$, then the objective is similar (but without the sampling of the switching states), and is augmented with an additional supervision objective as done in Kingma et al. (2014):

$$L_S = M + \sum_i^N q_{S_i}$$

The final training procedure for a single narrative is: (1) For each sentence (starting from the first), sample the switching state S_i from $q(S_i|\mathbf{X})$. (2) For each sentence (starting from the first), sample the latent Z_i from $q(Z_i|S_i, Z_{i-1}, X)$. (3) Evaluate the data likelihood and KL term(s) with these samples. (4) Take the gradients of objective w.r.t. all parameters, using the reparameterization trick for q_{Z_i} (Kingma and Welling, 2013) or the Gumbel-Softmax[6] trick for q_{S_i} (Jang et al., 2017), and optimize.

4 Interpolations via Gibbs Sampling

One of the benefits of probabilistic formulation is the possibility (if an inference procedure can be found) of generating narratives with specific constraints, where the constraints may be specified as clamped variables in the model. In this section, we show how narratives may be generated conditioned on arbitrary bits and pieces of the narrative already filled in, using approximate Gibbs sampling. This allows one to, for example, interpolate a narrative given the first and the last sentence (similar to how earlier story generation systems were able to generate with a given end goal in mind). Some examples of these interpolations generated by our system can be found in Table 3. We give the equations and summarize the algorithm in the next sections.

4.1 Conditionals for Gibbs Sampling

For our Gibbs sampling algorithm we give the narrative scaffold (switching variables), $S_1, ..., S_T \in \mathbf{S}$ and a set of observed sentences, $\mathbf{X}^+$. This may be any set of sentences (the first and last, just the second sentence, etc) as inputs to the system. We wish to find values for the unobserved sentences in set $\mathbf{X}^-$ by sampling from $P(\mathbf{X}^-, Z_1, ..., Z_T|\mathbf{S}, \mathbf{X}^+)$. We perform this sampling via Gibbs sampling. Two different forms of conditionals need to be derived to do this. One over Z_i conditioned on everything else, and one over X_i conditioned on everything else.

[6]We use the soft variant of Gumbel-Softmax. Rather than forcing a hard choice for S_i, we directly use the Gumbel-Softmax output and combine the transition matrices via a convex combination

One can show[7] that the distribution over Z_i conditioned on everything else will be approximately proportional to the following Gaussian:

$$\begin{aligned}
&\mathcal{N}_{Z_{i+1}}(A_{S_{i+1}}Z_i, \Sigma_{S_{i+1}})\mathcal{N}_{Z_i}(f_\mu(\cdot), f_\sigma(\cdot)) \qquad (1)\\
&\qquad \propto \mathcal{N}_{Z_i}(\mu_*, \Sigma_*) \text{ where,}\\
&\Sigma_* = \left(A^T_{S_{i+1}}\Sigma^{-1}_{S_{i+1}}A_{S_{i+1}} + f_\sigma(\cdot)^{-1}\right)^{-1}\\
&\mu_* = \Sigma^T_*\left(Z_{i+1}\Sigma^{-1}_{S_{i+1}}A_{S_{i+1}} + f_\mu(\cdot)^T f_\sigma(\cdot)^{-1}\right)^T
\end{aligned}$$

To find the second conditional, one can use the d-separation properties of the graph to find that it is proportional to:

$$\begin{aligned}
&P(X_i|Z_i, Z_{i+1}, S_i, S_{i+1}, X_{:i}, X_{i+1})\\
&\propto P(X_{i+1}|X_{:i}, X_i, Z_{i+1})P(X_i|X_{:i}, Z_i)
\end{aligned}$$

These two distributions are simply factors of our conditional language model, and both terms can thus be evaluated easily. In theory, one could use this fact to sample the original conditional via Metropolis-Hastings. Unfortunately, we found this approach to be too slow in practice. We observed that the simple heuristic of deterministically assigning X_i to be the greedy decoded output of the conditional language model $P(X_i|X_{:i}, Z_i)$ works well, as evidenced by the empirical results. We leave it for future work to research different conditional language model parameterizations allowing easy sampling from this conditional[8]

4.2 Gibbs Sampling Interpolation Overview

The variables in the Gibbs sampler are first initialized using some heuristics (see Supplemental). After initialization, performing interpolations with Gibbs sampling follows a two step process: First, for each Z_i, sample a value Z' from equation (1) and set Z_i to Z'. Then, for each X_i in $\mathbf{X}^-$, find a new value for X_i by running greedy decoding using the conditional language model.

5 Training Details

5.1 Dataset and Preprocessing

We use the ROCStories corpora introduced in Mostafazadeh et al. (2016). It contains 98,159 short commonsense stories in English as training, and 1,570 stories for validation and test each. Each story in the dataset has five-sentences and captures causal and temporal commonsense relations. We limit our vocabulary size to 16,983 based on a per-word frequency cutoff set to 5. For sentiment tags, we automatically tag the entirety of the corpus with the rule based sentiment tagger, Vader (Hutto and Gilbert, 2014), and bucket the polarity scores of Vader into three tags: neutral, negative, and positive. These tags form the label set of the S variables in our SLDS model. We tokenize the stories with Spacy tokenizer (Honnibal and Montani, 2017). Each sentences in the input narrative has an `<eos>` tag except for the S2S model discussed below.

5.2 Switching Linear Dynamical System (SLDS)

The SLDS has RNN encoder and decoder networks with single layer GRU cells of hidden size 1024 and an input embedding size of 300. We train the model using Adam with the defaults used by PyTorch. We stop training when the validation loss does not decrease for 3 consecutive epochs. Training details for all models and baselines remain same as above unless otherwise mentioned.

5.3 Baselines

Language Model (LM) : We train a two layer recurrent neural language model with GRU cells of hidden size 512.

Sequence-to-Sequence Attention Model (S2S) We train a two layer neural sequence to sequence model equipped with bi-linear attention function with GRU cells of hidden size 512. Sentiments tags for a narrative (1 for each sentence) are given as input to the model and the corresponding sentences are concatenated together as the output with only one `<eos>` tag at the end. This model is trained with a 0.1 dropout. This model is comparable to the static model of (Yao et al., 2019), and other recent works employing a notion of scaffolding into neural generation (albeit adapted for our setting).

Linear Dynamical System (LDS) We also train a linear dynamical system as discussed in Section 2.1 as one of our baselines for fair comparisons. Apart from having just a *single* transition matrix this model has the same architectural details as SLDS.

Semi-Supervised SLDS (SLDS-X%) To gauge the usability of semi-supervision, we also train semi-supervised SLDS models with varying

[7] See Supplemental for derivation

[8] One possibility is take advantage of 'orderless' pretrained models through sampling (Wang and Cho, 2019). Approaches such as the one recently proposed in Donahue et al. (2020) may also be useful for this purpose.

amount of labelled sentiment tags unlike the original model which uses 100% tagged data. We refer to these as SLDS-X%, where X is the % labelled data used for training: 1%, 10%, 25%, and 50%.

6 Evaluations

As described above, our model is able to perform narrative interpolations via an approximate Gibbs sampling procedure. At the core of our evaluations is thus a fill-in-the-sentences task. We provide 1 or 2 sentences, and require the model to generate the rest of the narrative . We evaluate this via automatic evaluations as well as with crowd-sourced human evaluations. We also report perplexity to evaluate the models' ability to fit the data. Lastly, we look at whether the transitions learned by the SLDS models capture what they are intended to capture: does using the transition matrix associated with a sentiment tag (positive/negative/neutral) lead to a generated sentence with that sentiment?

6.1 Generating the Interpolations

For the SLDS models, the interpolations are generated via the Gibbs sampling algorithm described earlier. In all experiments for the SLDS models we draw **50** samples (including burn in samples) and output the interpolation that maximizes the probability of the given sentence(s). Since the baselines do not have the means for doing interpolations, we simulate 'interpolations' for the baselines; we draw **1000** samples using top k (with k=15) truncated sampling (conditioned on the given initial sentences, if available). We then output the sample that maximizes the probability of the clamped sentences around which we are interpolating the others. We allow the S2S access to the gold sentiment tags. To give a lower bound on the performance of the SLDS model, we do not provide it with gold tags. We instead provide the SLDS model with the semi-noisy[9] tags that are output from $q(S_i|X)$.[10]

6.2 Automatic Evaluation of Interpolations

We automatically evaluate on four different types of interpolations (where different combinations of sentences are removed and the model is forced to regenerate them), We evaluate the generations with the ROUGE (Lin, 2004) and METEOR (Banerjee and Lavie, 2005) metrics using the true sentences as targets. Table 2 shows the automatic evaluation results from interpolations using our proposed models and baselines. The #Sent(s) column indicates which sentence(s) were removed, and then regenerated by the model. We gave the baselines a slight edge over SLDS because they pick the best out of 1000 samples while SLDS is only out of 50. The SLDS models see their largest gain over the baseline models when at least the first sentence is given as an input. The baseline models do better when the first and second sentence need to be imputed. This is likely due to the fact that having access to the earlier sentences allows a better initialization for the Gibbs sampler. Surprisingly, the semi-supervised variants of the SLDS models achieve higher scores. The reasons for this is discussed below in the Perplexity section.

6.3 Human Evaluation of Interpolations

6.3.1 Annotation Scheme

As automatic evaluation metrics are not sufficient to assess the quality of any creative task such as narrative generation, we measure the quality of the generations through human evaluation of 200 stories on the Amazon Mechanical Turk platform. We provided Turkers with two generated narratives from two different models, each with five sentences. The first and last sentences were fed to each model as input, and the middle three sentences were generated. Each pair of narratives is graded by 3 users each with two tasks: (1) to rank on a scale of 0-3 each of the sentences except the first one on the basis of its coherency with the previous sentence(s) and (2) compare and rank the two narratives based on their overall coherency, ie how well the story connects the starting/ending sentences.

6.3.2 Human Evaluation Results

Table 4 reports the result of human evaluations of SLDS and baseline generations. We can observe that people preferred narratives generated by SLDS over the ones generated by baseline models (LM and S2S) as they found the former model more coherent, which is an important criteria for narrative generation. **51.3%** of the time SLDS generates better narratives than the LM model while LM in turn does it only **35.0%** of the times. 13.7% of the generations end up in tie. The mean sentence level coherence score for SLDS is around 12.5% larger than that of the LM, with a slightly lower standard

[9]To confirm that these tags are noisier, we repeat the experiments in Table 2, but feed gold tags to our models. We find that the gold tags lead to a sizable increase in performance for our model. See Appendix for more detail.

[10]Note that missing sentences, X, are used *only* for computing these noisy tags.

# Sent(s)	Metric	SLDS	SLDS-1	SLDS-10	SLDS-25	SLDS-50	S2S	LM	LDS
2^{nd}	R1	17.60	19.36	20.46	**20.92**	19.55	18.79	18.30	17.33
	R2	2.43	3.46	3.86	**4.10**	3.43	3.14	2.83	2.31
	RL	16.43	17.76	19.03	**19.45**	17.87	17.39	16.68	15.97
	M	6.35	6.84	6.98	**7.15**	6.94	7.11	6.76	6.13
4^{th}	R1	16.98	17.90	18.64	18.14	**19.39**	15.38	14.03	17.06
	R2	2.20	2.61	2.74	2.29	**3.23**	1.97	1.40	2.31
	RL	15.20	16.21	16.69	16.08	**17.43**	13.90	12.64	15.24
	M	6.33	7.11	6.92	6.53	**7.18**	5.84	5.61	6.89
$1^{st} + 2^{nd}$	R1	15.40	16.04	16.11	16.33	16.27	**18.91**	17.38	14.32
	R2	1.79	1.65	1.97	2.17	1.83	**2.62**	2.03	1.47
	RL	14.63	15.15	15.23	15.47	15.27	**17.89**	16.48	13.41
	M	5.34	5.27	5.40	5.44	5.42	**6.81**	6.07	4.80
$3^{rd} + 4^{th}$	R1	23.35	23.59	23.57	**23.65**	23.60	20.68	20.01	21.66
	R2	3.77	3.35	3.76	3.58	**3.93**	2.51	1.91	2.94
	RL	21.56	21.49	21.87	**21.88**	21.67	18.87	18.28	20.04
	M	8.28	8.26	8.22	8.12	**8.29**	7.51	7.26	7.87

Table 2: F1 scores for ROUGE-1, 2, and L and METEOR (M) (default mode score) for randomly sampled 500 stories from the test set. #Sents(s) column represents the "fill in" sentence(s) that the models generated using Gibbs sampling. Our SLDS models pick the best of **50** samples, the baselines models pick the best of **1000** samples

Ed was playing baseball in his yard.	**Last week I had an idea.**
He was running down the hill.	**I was so nervous that I decided to make a presentation.**
His ball was coming towards him.	I soon found it hard to come up with new ideas.
It was very scary!	I didn't think it would be so hard.
Ed was scared.	But then, an idea came to me and I was back on track.
Ben has always wanted to learn how to play the piano	Tim was always on his bike during the summer.
His parent bought him one.	**He had a lot of fun.**
Ben enrolls in a piano class with a local tutor.	**One day he decided to cut his bike down.**
Ben practiced every day.	**He hit a rock and fell off the bike and hit a tree.**
He gets better with every lesson.	He broke his arm.

Table 3: Sample interpolations from Gibbs sampling. Grayed out lines are provided as input and bold sentences are generated by SLDS.

deviation. We see similar results when compared against the S2S model.

System	Sent Coh. (0-3)	Best Story
LM	1.68 ± 1.01	35.0%
SLDS	**1.89 ± 0.96**	**51.3%**
S2S	1.67 ± 1.00	35.1%
SLDS	**1.87 ± 0.97**	**51.9%**

Table 4: Human evaluation scores for filled-in narrative generation. Humans judged sentence coherence and chose which model filled in the *most* coherent narrative overall (13.7% and 13% tie for LM and S2S).

6.4 Language Modeling Perplexity Score

As our models are essentially language models, we evaluated their per-sentence negative log-likelihood and per-word perplexity scores[11], which can be viewed as an indirect measure of how well a system works as a generative model of narrative text. For the SLDS and LDS models these scores are approximations, an upper bound (the negative of the ELBO) to the actual values. For the other two models the scores are exact. A good model should assign low perplexity scores to its test set. In Table 5 SLDS achieves the lowest scores, implying that it is able to model the data distribution well. In Table 6

[11]Note that since S2S appends the eos token only at the end, its per-sentence NLL is slightly lower than that of LM.

we also calculate the perplexity scores for the semi-supervised SLDS models to assess the effectiveness of semi-supervised training. Surprisingly, the models with less supervision scored better in terms of perplexity. One possibility for this might be the use of the soft Gumbel-Softmax in the semi-supervised models. The soft Gumbel-Softmax variant does not commit to using a single transition matrix at each time step (instead linearly combining them, weighted by the Softmax weights). This fact may permit the model greater flexibility in fitting the training data. While this leads to better scores in metrics such as perplexity or BLEU, it does leads to transitions that are worse in capturing the properties they should be capturing, as we shall see in the next section.

System	NLL	PPL
LM	196.30	35.41
S2S	192.25	43.36
LDS	$\leq$186.24	29.49
SLDS	$\leq$**182.17**	**27.39**

Table 5: NLL and PPL scores on the test set. Lower is better for both the metrics. Variance in NLL calculation is in the order of 10^{-3}.

System	NLL	PPL
SLDS-1%	$\leq$**177.60**	**25.19**
SLDS-10%	$\leq$178.81	25.77
SLDS-25%	$\leq$181.11	26.87
SLDS-50%	$\leq$185.07	28.88
SLDS	$\leq$182.17	27.39

Table 6: Approximate NLL and PPL scores for SLDS and semi-supervised SLDS on the test set.

6.5 Evaluation of Transition Dynamics

One matter of interest is whether or not the transitions are capturing what they are supposed to capture, appropriate sentiment. Since we used the sentiment tagger Vader for training tags, we again utilize it to evaluate whether using transitions of a certain sentiment actually leads the model to produce outputs with the given sentiment. To perform this evaluation, we give as input to our models (and the S2S baseline) the sentiment tags for a sentence and allow it to generate a sentence conditioned on these sentiment tags. We then tag the generated sentences with Vader and see if the sentiment tags match the originals. We calculate the F1 score across all sentiment tags and report the macro average. In Table 7 we see that having labels is incredibly important for meaningful transitions. There is a large drop in F1 as the amount of labels given to the model is decreased. The SLDS model that is trained with 100% of the labels performs a little better than even S2S, despite not having direct access to the sentiment labels (SLDS only uses the sentiment labels to decide which transition to use while the S2S model uses attention directly on the sentiment labels).

System	Macro F1
S2S	95.8
SLDS-1%	50.2 ± 1.1
SLDS-10%	51.4 ± 1.1
SLDS-25%	58.7 ± 0.4
SLDS-50%	74.6 ± 0.1
SLDS	$\mathbf{96.1} \pm \mathbf{0.0}$

Table 7: Macro F1 scores on sentiment classification task. Results for SLDS and SLDS-X% are averaged over 5 runs.

7 Related Work

Story/narrative generation has a rich history in the field of AI. Many early systems were based on structured formalisms for describing common narrative structures (Lakoff, 1972; Thorndyke, 1977; Meehan, 1977), many being inspired by the initial work of (Propp, 1928). There has been a swath of recent work that has looked to add some semblance of a 'narrative scaffold' back into generation methods (Fan et al.; Martin et al., 2018; Yao et al., 2019; Xu et al., 2018). Many of these methods work as conditional LMs (conditioned directly on the scaffold). This line of work may be combined with our formalization as well, by conditioning the generation on the switching state as well, as done in the model of Barber (2006). Recent work by Tambwekar et al. (2019) has similar goals to ours in permitting more controlability in generation systems, developing a RL-based system that allows users to specify an end goal for a story (by specifying the event class that is desired to appear at the end). Their work differs from ours in that it does not deal with text directly, modeling only the sequences of events in the narrative. It may be possible to utilize this model as the scaffolding component in our model (utilizing their RL policy for the scaffold planner, rather than the simple

Markovian distribution used here).

8 Conclusion and Future Work

In this paper, we formulated the problem of narrative generation as a switching dynamical system. We showed how this formulation captures notions important in narrative generation, such as narrative dynamics and scaffolds. We developed an approximate Gibbs sampling algorithm for the model that permits the system to generate interpolations conditioned on arbitrary parts of the narrative, and evaluated these interpolations using both human and automatic evaluations. Though in this work we used sentiment tags for our scaffolds/switching variables, future work may look at utilizing different kinds of information to guide the generation of narratives. Utilizing the main predicate of a sentence as a scaffold would be a logical next step, and may prove more informative than the sentiment trajectory. A scaffold such as this can take on many more possible values than a sentiment tag, and as such, it may prove difficult to assign a set of dynamics to each value. Another avenue for future work would deal with this possible problem. One potential solution could be to associate each switching variable value with a (learned) vector in a probability simplex, and use this vector to combine a small set of "primitive" dynamics matrices in order to get that value's associated set of dynamics.

References

G Ackerson and K Fu. 1970. On state estimation in switching environments. *IEEE Transactions on Automatic Control*, 15(1):10–17.

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, Ann Arbor, Michigan. Association for Computational Linguistics.

David Barber. 2006. Expectation correction for smoothed inference in switching linear dynamical systems. *Journal of Machine Learning Research*, 7(Nov):2515–2540.

C. B. Chang and M. Athans. 1978. State estimation for discrete systems with switching parameters. *IEEE Transactions on Aerospace and Electronic Systems*, AES-14(3):418–425.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2492–2501, Online. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. 2017. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *NeurIPS 2017*.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *NeurIPS 2016*.

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

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *AAAI conference on weblogs and social media*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

Maximilian Karl, Maximilian Sölch, Justin Bayer, and Patrick van der Smagt. 2017. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *ICLR*.

Ahmed Khalifa, Gabriella AB Barros, and Julian Togelius. 2017. Deeptingle. *ICCC*.

Diederik P Kingma and Max Welling. 2013. Autoencoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *NeurIPS 2014*.

Rahul G. Krishnan, Uri Shalit, and David Sontag. 2015. Deep kalman filters. *CoRR*.

Rahul G. Krishnan, Uri Shalit, and David Sontag. 2017. Structured inference networks for nonlinear state space models. In *AAAI 2017*.

George Lakoff. 1972. Structural complexity in fairy tales.

Michael Lebowitz. 1985. Story-telling as planning and learning. *Poetics*, 14(6):483–502.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL 2004*.

Scott W Linderman, Andrew C Miller, Ryan P Adams, David M Blei, Liam Paninski, and Matthew J Johnson. 2016. Recurrent switching linear dynamical systems. *AISTATS*.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018. Narrative modeling with memory chains and semantic supervision. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 278–284, Melbourne, Australia. Association for Computational Linguistics.

Lara J. Martin, Prithviraj Ammanabrolu, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. 2018. Event representations for automated story generation with deep neural nets. *AAAI*.

James R. Meehan. 1977. Tale-spin, an interactive program that writes stories. In *IJCAI*, pages 91–98.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. 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. [link].

Kevin P Murphy. 1998. Switching kalman filters.

Gerald Prince. 2003. *A dictionary of narratology*.

Vladimir Propp. 1928. *Morphology of the Folktale*.

Andrew J Reagan, Lewis Mitchell, Dilan Kiley, Christopher M Danforth, and Peter Sheridan Dodds. 2016. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*, 5(1):31.

Mark O. Riedl and R. Michael Young. 2006. Story planning as exploratory creativity: Techniques for expanding the narrative search space. *New Generation Computing*, 24(3):303–323.

Mark O. Riedl and Robert Michael Young. 2010a. Narrative planning: Balancing plot and character. *J. Artif. Intell. Res.*, 39:217–268.

Mark O. Riedl and Robert Michael Young. 2010b. Narrative planning: Balancing plot and character. *J. Artif. Intell. Res.*, 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.

Pradyumna Tambwekar, Murtaza Dhuliawala, Animesh Mehta, Lara J Martin, Brent Harrison, and Mark O Riedl. 2019. Controllable neural story generation via reinforcement learning. *IJCAI*.

Perry W Thorndyke. 1977. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive psychology*, 9(1):77–110.

Scott R. Turner. 1993. *Minstrel: A Computer Model of Creativity and Storytelling*. Ph.D. thesis.

Kurt Vonnegut. 1981. *Palm Sunday*. Rosetta Books.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov random field language model. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Jingjing Xu, Yi Zhang, Qi Zeng, Xuancheng Ren, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4306–4315, Brussels, Belgium. Association for Computational Linguistics.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. *AAAI*, abs/1811.05701.

Appendix

9 Gibbs Sampling Derivation

With the d-separation properties of the graph (and substituting in our variational posterior approximation Q), we can write the conditional distribution

# Sent(s)	Metric	SLDS	SLDS-1	SLDS-10	SLDS-25	SLDS-50	S2S	LM	LDS
$1^{st} + 2^{nd}$	R1	17.89	17.93	18.06	18.51	18.44	**18.91**	17.38	14.32
	R2	2.30	1.98	2.35	2.57	2.21	**2.62**	2.03	1.47
	RL	15.01	15.96	16.41	16.73	16.31	**17.89**	16.48	13.41
	M	6.06	5.83	6.21	6.49	6.37	**6.81**	6.07	4.80
$3^{rd} + 4^{th}$	R1	23.41	23.78	24.03	**24.08**	23.60	20.68	20.01	21.66
	R2	3.79	3.51	3.88	3.62	**3.93**	2.51	1.91	2.94
	RL	21.97	21.86	22.00	**22.10**	21.67	18.87	18.28	20.04
	M	8.32	8.27	8.32	8.30	**8.33**	7.51	7.26	7.87

Table 8: F1 scores for ROUGE-1, 2, and L and METEOR (M) (default mode score) for randomly sampled 500 stories from the test set. #Sents(s) column represents the “fill in” sentence(s) that the models generated using Gibbs sampling. Our SLDS models pick the best of **50** samples, the baselines models pick the best of **1000** samples

of some Z_i given everything else as follows:

$$\begin{aligned}
&P(Z_i|Z_{i-1}, Z_{i+1}, S_i, S_{i+1}, X_i, X_{i-1}) \\
&\propto P(Z_{i+1}, S_{i+1}|Z_{i-1}, Z_{i+1}, S_i, S_{i+1}, X_i, X_{i-1}, Z_i) \\
&* P(Z_i|Z_{i-1}, S_i, X_i, X_{i-1}) \\
&\approx P(Z_{i+1}, S_{i+1}|Z_{i-1}, S_i, S_{i+1}, X_i, X_{i-1}, Z_i) \\
&* Q(Z_i|Z_{i-1}, S_i, X_i, X_{i-1}) \\
&= P(Z_{i+1}|S_{i+1}, Z_{i-1}, S_i, S_{i+1}, X_i, X_{i-1}, Z_i) \\
&* P(S_{i+1}|S_i) Q(Z_i|Z_{i-1}, S_i, X_i, X_{i-1}) \\
&\propto P(Z_{i+1}|S_{i+1}, Z_{i-1}, S_i, S_{i+1}, X_i, X_{i-1}, Z_i) \\
&* Q(Z_i|Z_{i-1}, S_i, X_i, X_{i-1}) \\
&= P(Z_{i+1}|S_{i+1}, Z_i) Q(Z_i|Z_{i-1}, S_i, X_i, X_{i-1}) \\
&= \mathcal{N}_{Z_{i+1}}(A_{S_{i+1}} Z_i, \Sigma_{S_{i+1}}) \mathcal{N}_{Z_i}(f_\mu(\cdot), f_\sigma(\cdot))
\end{aligned}$$

The rest can be derived by taking the PDFs of the two Gaussian densities above, getting rid of constants that don’t depend on Z_i, multiplying them together, and completing the square to obtain the numerator of a Gaussian over Z_i (such that Z_i appears nowhere else in the equation). This numerator can then be multiplied by the normalizing constant (that does not depend on Z_i) to obtain exactly a Gaussian pdf with the mean and variance as given below:

$$\begin{aligned}
&\mathcal{N}_{Z_{i+1}}(A_{S_{i+1}} Z_i, \Sigma_{S_{i+1}}) \mathcal{N}_{Z_i}(f_\mu(\cdot), f_\sigma(\cdot)) \\
&\qquad \propto \mathcal{N}_{Z_i}(\mu_*, \Sigma_*) \ \text{ where,} \\
\Sigma_* &= \left(A^T_{S_{i+1}} \Sigma^{-1}_{S_{i+1}} A_{S_{i+1}} + f_\sigma(\cdot)^{-1}\right)^{-1} \\
\mu_* &= \Sigma^T_* \left(Z_{i+1} \Sigma^{-1}_{S_{i+1}} A_{S_{i+1}} + f_\mu(\cdot)^T f_\sigma(\cdot)^{-1}\right)^T
\end{aligned}$$

10 Manual Error Analysis of Generations

We evaluate the quality of sentences of 20 generated narratives are not considered coherent by Turkers. We find that, in a broader context, 16 stories out of 20 are not good enough in terms of connecting the ending with the previous sentences. Also, in 14 out of 20 stories, *mild* off-topic sentences are introduced, which are aligned with the main topic of the story but not along with local coherency (i.e. the previous and the next sentences). When considering a narrower context, or sentence level, we confirm that only 9 out of 60 generated sentences are ungrammatical, so they fail to deliver their meaning.

11 Initializing the Gibbs Sampler

Initializing Z : We first initialize the Z variables in the sampler. This is done as follows: The Z variables are initialized in increasing order. If sentence X_i is provided as input, then we sample from the approximate posterior q_Z in order to initialize Z_i. If X_i is missing, then we sample using the dynamics distribution, $P(Z_i|Z_{i-1}, S_i)$. Since we initialize in increasing order, we are guaranteed to have Z_{i-1}.

Initializing X : We next initialize the missing text. Initializing the missing text X_i is simply done by greedy decoding from the language model conditioned on Z_i, and previous sentences.

12 Automatic Evaluation Results with Gold Labels

Below in Table 8 we provide the results for the automatic experiments with gold labels fed to our method. We find a sizable increase in performance by providing the gold labels, and thus use the automatic labels as noisy proxies in order to establish a lower bound on performance.