Learning Neural Templates for Text Generation

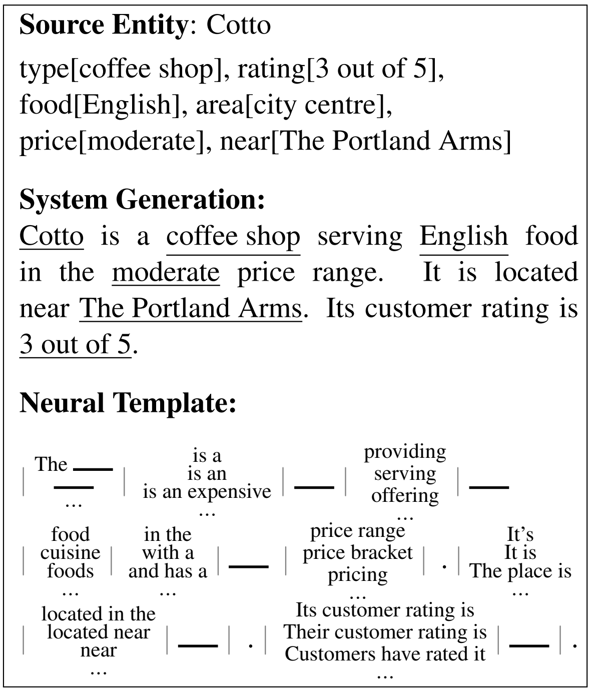
Sam Wiseman **Stuart M. Shieber** **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,shieber,srush}@seas.harvard.edu

Abstract

While neural, encoder-decoder models have had significant empirical success in text generation, there remain several unaddressed problems with this style of generation. Encoder-decoder models are largely (a) uninterpretable, and (b) difficult to control in terms of their phrasing or content. This work proposes a neural generation system using a hidden semi-markov model (HSMM) decoder, which learns latent, discrete templates jointly with learning to generate. We show that this model learns useful templates, and that these templates make generation both more interpretable and controllable. Furthermore, we show that this approach scales to real data sets and achieves strong performance nearing that of encoder-decoder text generation models.

1 Introduction

With the continued success of encoder-decoder models for machine translation and related tasks, there has been great interest in extending these methods to build general-purpose, data-driven natural language generation (NLG) systems (Mei et al., 2016; Dušek and Jurcıcek, 2016; Lebret et al., 2016; Chisholm et al., 2017; Wiseman et al., 2017). These encoder-decoder models (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015) use a neural encoder model to represent a source knowledge base, and a decoder model to emit a textual description word-by-word, conditioned on the source encoding. This style of generation contrasts with the more traditional division of labor in NLG, which famously emphasizes addressing the two questions of "what to say" and "how to say it" separately, and which leads to systems with explicit content selection, macro- and micro-planning, and surface realization components (Reiter and Dale, 1997; Jurafsky and Martin, 2014).



Source Entity: Cotto
type[coffee shop], rating[3 out of 5], food[English], area[city centre], price[moderate], near[The Portland Arms]

System Generation:
Cotto is a coffee shop serving English food in the moderate price range. It is located near The Portland Arms. Its customer rating is 3 out of 5.

Neural Template:

The ___ / ___ / ...
is a / is an / is an expensive / ...
___
providing / serving / offering / ...
___
food / cuisine / foods / ...
in the / with a / and has a / ...
___
price range / price bracket / pricing / ...
.
It's / It is / The place is / ...
located in the / located near / near / ...
___
.
Its customer rating is / Their customer rating is / Customers have rated it / ...
___
.

Figure 1: An example template-like generation from the E2E Generation dataset (Novikova et al., 2017). Knowledge base x (top) contains 6 records, and $\hat{y}$ (middle) is a system generation; records are shown as type[value]. An induced neural template (bottom) is learned by the system and employed in generating $\hat{y}$. Each cell represents a segment in the learned segmentation, and "blanks" show where slots are filled through copy attention during generation.

Encoder-decoder generation systems appear to have increased the fluency of NLG outputs, while reducing the manual effort required. However, due to the black-box nature of generic encoder-decoder models, these systems have also largely sacrificed two important desiderata that are often found in more traditional systems, namely (a) interpretable outputs that (b) can be easily controlled in terms of form and content.

This work considers building interpretable and controllable neural generation systems, and proposes a specific first step: a new data-driven generation model for learning discrete, *template-like*

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3174–3187
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

structures for conditional text generation. The core system uses a novel, neural hidden semi-markov model (HSMM) decoder, which provides a principled approach to template-like text generation. We further describe efficient methods for training this model in an entirely data-driven way by backpropagation through inference. Generating with the template-like structures induced by the neural HSMM allows for the explicit representation of what the system intends to say (in the form of a learned template) and how it is attempting to say it (in the form of an instantiated template).

We show that we can achieve performance competitive with other neural NLG approaches, while making progress satisfying the above two desiderata. Concretely, our experiments indicate that we can induce explicit templates (as shown in Figure 1) while achieving competitive automatic scores, and that we can control and interpret our generations by manipulating these templates. Finally, while our experiments focus on the data-to-text regime, we believe the proposed methodology represents a compelling approach to learning discrete, latent-variable representations of conditional text.

2 Related Work

A core task of NLG is to generate textual descriptions of knowledge base records. A common approach is to use hand-engineered templates (Kukich, 1983; McKeown, 1992; McRoy et al., 2000), but there has also been interest in creating templates in an automated manner. For instance, many authors induce templates by clustering sentences and then abstracting templated fields with hand-engineered rules (Angeli et al., 2010; Kondadadi et al., 2013; Howald et al., 2013), or with a pipeline of other automatic approaches (Wang and Cardie, 2013).

There has also been work in incorporating probabilistic notions of templates into generation models (Liang et al., 2009; Konstas and Lapata, 2013), which is similar to our approach. However, these approaches have always been conjoined with discriminative classifiers or rerankers in order to actually accomplish the generation (Angeli et al., 2010; Konstas and Lapata, 2013). In addition, these models explicitly model knowledge base field selection, whereas the model we present is fundamentally an end-to-end model over generation segments.

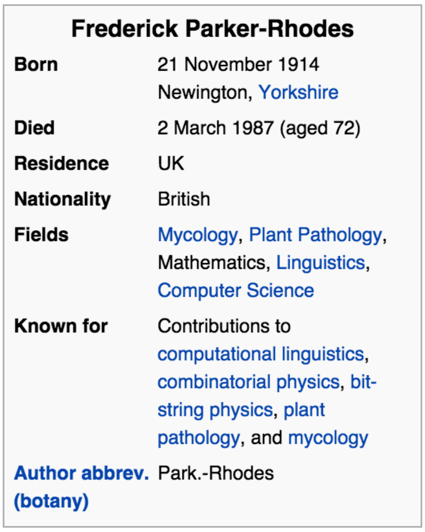
Recently, a new paradigm has emerged around neural text generation systems based on machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015). Most of this work has used unconstrained black-box encoder-decoder approaches. There has been some work on discrete variables in this context, including extracting representations (Shen et al., 2018), incorporating discrete latent variables in text modeling (Yang et al., 2018), and using non-HSMM segmental models for machine translation or summarization (Yu et al., 2016; Wang et al., 2017; Huang et al., 2018). Dai et al. (2017) develop an approximate inference scheme for a neural HSMM using RNNs for continuous emissions; in contrast we maximize the exact log-marginal, and use RNNs to parameterize a discrete emission distribution. Finally, there has also been much recent interest in segmental RNN models for non-generative tasks in NLP (Tang et al., 2016; Kong et al., 2016; Lu et al., 2016).

The neural text generation community has also recently been interested in “controllable” text generation (Hu et al., 2017), where various aspects of the text (often sentiment) are manipulated or transferred (Shen et al., 2017; Zhao et al., 2018; Li et al., 2018). In contrast, here we focus on controlling either the content of a generation or the way it is expressed by manipulating the (latent) template used in realizing the generation.

3 Overview: Data-Driven NLG

Our focus is on generating a textual description of a knowledge base or meaning representation. Following standard notation (Liang et al., 2009; Wiseman et al., 2017), let $x = \{r_1 \ldots r_J\}$ be a collection of records. A record is made up of a *type* ($r.t$), an *entity* ($r.e$), and a *value* ($r.m$). For example, a knowledge base of restaurants might have a record with $r.t$ = `Cuisine`, $r.e$ = `Denny's`, and $r.m$ = `American`. The aim is to generate an adequate and fluent text description $\hat{y}_{1:T} = \hat{y}_1, \ldots, \hat{y}_T$ of x. Concretely, we consider the E2E Dataset (Novikova et al., 2017) and the WikiBio Dataset (Lebret et al., 2016). We show an example E2E knowledge base x in the top of Figure 1. The top of Figure 2 shows an example knowledge base x from the WikiBio dataset, where it is paired with a *reference* text $y = y_{1:T}$ at the bottom.

The dominant approach in neural NLG has been



Frederick Parker-Rhodes

Born	21 November 1914 Newington, Yorkshire
Died	2 March 1987 (aged 72)
Residence	UK
Nationality	British
Fields	Mycology, Plant Pathology, Mathematics, Linguistics, Computer Science
Known for	Contributions to computational linguistics, combinatorial physics, bit-string physics, plant pathology, and mycology
Author abbrev. (botany)	Park.-Rhodes

Frederick Parker-Rhodes (21 March 1914 - 21 November 1987) was an English linguist, plant pathologist, computer scientist, mathematician, mystic, and mycologist.

Figure 2: An example from the WikiBio dataset (Lebret et al., 2016), with a database x (top) for Frederick Parker-Rhodes and corresponding reference generation y (bottom).

to use an encoder network over x and then a conditional decoder network to generate y, training the whole system in an end-to-end manner. To generate a description for a given example, a black-box network (such as an RNN) is used to produce a distribution over the next word, from which a choice is made and fed back into the system. The entire distribution is driven by the internal states of the neural network.

While effective, relying on a neural decoder makes it difficult to understand what aspects of x are correlated with a particular system output. This leads to problems both in controlling fine-grained aspects of the generation process and in interpreting model mistakes.

As an example of why controllability is important, consider the records in Figure 1. Given these inputs an end-user might want to generate an output meeting specific constraints, such as not mentioning any information relating to customer rating. Under a standard encoder-decoder style model, one could filter out this information either from the encoder or decoder, but in practice this would lead to unexpected changes in output that might propagate through the whole system.

As an example of the difficulty of interpreting mistakes, consider the following actual generation from an encoder-decoder style system for the records in Figure 2: ”frederick parker-rhodes (21 november 1914 - 2 march 1987) was an english mycology and plant pathology, mathematics at the university of uk.” In addition to not being fluent, it is unclear what the end of this sentence is even attempting to convey: it may be attempting to convey a fact not actually in the knowledge base (e.g., where Parker-Rhodes studied), or perhaps it is simply failing to fluently realize information that *is* in the knowledge base (e.g., Parker-Rhodes's country of residence).

Traditional NLG systems (Kukich, 1983; McKeown, 1992; Belz, 2008; Gatt and Reiter, 2009), in contrast, largely avoid these problems. Since they typically employ an explicit planning component, which decides which knowledge base records to focus on, and a surface realization component, which realizes the chosen records, the intent of the system is always explicit, and it may be modified to meet constraints.

The goal of this work is to propose an approach to neural NLG that addresses these issues in a principled way. We target this goal by proposing a new model that generates with template-like objects induced by a neural HSMM (see Figure 1). Templates are useful here because they represent a fixed plan for the generation's content, and because they make it clear what part of the generation is associated with which record in the knowledge base.

4 Background: Semi-Markov Models

What does it mean to learn a template? It is natural to think of a template as a sequence of typed text-segments, perhaps with some segments acting as the template's "backbone" (Wang and Cardie, 2013), and the remaining segments filled in from the knowledge base.

A natural probabilistic model conforming with this intuition is the hidden semi-markov model (HSMM) (Gales and Young, 1993; Ostendorf et al., 1996), which models latent segmentations in an output sequence. Informally, an HSMM is much like an HMM, except emissions may last multiple time-steps, and multi-step emissions need not be independent of each other conditioned on the state.

We briefly review HSMMs following Murphy (2002). Assume we have a sequence of observed tokens $y_1 \dots y_T$ and a discrete, latent state $z_t \in \{1, \dots, K\}$ for each timestep. We additionally use two per-timestep variables to model multi-step segments: a length variable $l_t \in \{1, \ldots, L\}$ specifying the length of the current segment, and a deterministic binary variable f_t indicating whether a segment finishes at time t. We will consider in particular *conditional* HSMMs, which condition on a source x, essentially giving us an HSMM decoder.

An HSMM specifies a joint distribution on the observations and latent segmentations. Letting θ denote all the parameters of the model, and using the variables introduced above, we can write the corresponding joint-likelihood as follows

$$p(y, z, l, f \mid x; \theta) = \prod_{t=0}^{T-1} p(z_{t+1}, l_{t+1} \mid z_t, l_t, x)^{f_t} \times \prod_{t=1}^{T} p(y_{t-l_t+1:t} \mid z_t, l_t, x)^{f_t},$$

where we take z_0 to be a distinguished start-state, and the deterministic f_t variables are used for excluding non-segment log probabilities. We further assume $p(z_{t+1}, l_{t+1} \mid z_t, l_t, x)$ factors as $p(z_{t+1} \mid z_t, x) \times p(l_{t+1} \mid z_{t+1})$. Thus, the likelihood is given by the product of the probabilities of each discrete state transition made, the probability of the length of each segment given its discrete state, and the probability of the observations in each segment, given its state and length.

5 A Neural HSMM Decoder

We use a novel, neural parameterization of an HSMM to specify the probabilities in the likelihood above. This full model, sketched out in Figure 3, allows us to incorporate the modeling components, such as LSTMs and attention, that make neural text generation effective, while maintaining the HSMM structure.

5.1 Parameterization

Since our model must condition on x, let $\boldsymbol{r}_j \in \mathbb{R}^d$ represent a real embedding of record $r_j \in x$, and let $\boldsymbol{x}_a \in \mathbb{R}^d$ represent a real embedding of the entire knowledge base x, obtained by max-pooling coordinate-wise over all the $\boldsymbol{r}_j$. It is also useful to have a representation of just the unique *types* of records that appear in x, and so we also define $\boldsymbol{x}_u \in \mathbb{R}^d$ to be the sum of the embeddings of the unique types appearing in x, plus a bias vector and followed by a ReLU nonlinearity.

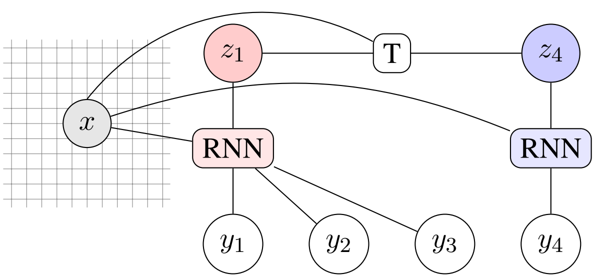


Figure 3: HSMM factor graph (under a known segmentation) to illustrate parameters. Here we assume z_1 is in the "red" state (out of K possibilities), and transitions to the "blue" state after emitting three words. The transition model, shown as T, is a function of the two states and the neural encoded source x. The emission model is a function of a "red" RNN model (with copy attention over x) that generates words 1, 2 and 3. After transitioning, the next word y_4 is generated by the "blue" RNN, but independently of the previous words.

Transition Distribution The transition distribution $p(z_{t+1} \mid z_t, x)$ may be viewed as a $K \times K$ matrix of probabilities, where each row sums to 1. We define this matrix to be

$$p(z_{t+1} \mid z_t, x) \propto \boldsymbol{A}\boldsymbol{B} + \boldsymbol{C}(\boldsymbol{x}_u)\boldsymbol{D}(\boldsymbol{x}_u),$$

where $\boldsymbol{A} \in \mathbb{R}^{K \times m_1}$, $\boldsymbol{B} \in \mathbb{R}^{m_1 \times K}$ are state embeddings, and where $\boldsymbol{C} : \mathbb{R}^d \to \mathbb{R}^{K \times m_2}$ and $\boldsymbol{D} : \mathbb{R}^d \to \mathbb{R}^{m_2 \times K}$ are parameterized non-linear functions of $\boldsymbol{x}_u$. We apply a row-wise $\mathrm{softmax}$ to the resulting matrix to obtain the desired probabilities.

Length Distribution We simply fix all length probabilities $p(l_{t+1} \mid z_{t+1})$ to be uniform up to a maximum length L.[1]

Emission Distribution The emission model models the generation of a text segment conditioned on a latent state and source information, and so requires a richer parameterization. Inspired by the models used for neural NLG, we base this model on an RNN decoder, and write a segment's probability as a product over token-level probabilities,

$$p(y_{t-l_t+1:t} \mid z_t = k, l_t = l, x) = \prod_{i=1}^{l_t} p(y_{t-l_t+i} \mid y_{t-l_t+1:t-l_t+i-1}, z_t = k, x) \times p(\text{</seg>} \mid y_{t-l_t+1:t}, z_t = k, x) \times \mathbf{1}_{\{l_t = l\}},$$

[1] We experimented with parameterizing the length distribution, but found that it led to inferior performance. Forcing the length probabilities to be uniform encourages the model to cluster together functionally similar emissions of different lengths, while parameterizing them can lead to states that specialize to specific emission lengths.

where </seg> is an end of segment token. The RNN decoder uses attention and copy-attention over the embedded records $\boldsymbol{r}_j$, and is conditioned on $z_t = k$ by concatenating an embedding corresponding to the k'th latent state to the RNN's input; the RNN is also conditioned on the entire x by initializing its hidden state with $\boldsymbol{x}_a$.

More concretely, let $\boldsymbol{h}^k_{i-1} \in \mathbb{R}^d$ be the state of an RNN conditioned on x and $z_t = k$ (as above) run over the sequence $y_{t-l_t+1:t-l_t+i-1}$. We let the model attend over records $\boldsymbol{r}_j$ using $\boldsymbol{h}^k_{i-1}$ (in the style of Luong et al. (2015)), producing a context vector $\boldsymbol{c}^k_{i-1}$. We may then obtain scores $\boldsymbol{v}_{i-1}$ for each word in the output vocabulary,

$$\boldsymbol{v}_{i-1} = \boldsymbol{W} \tanh(\boldsymbol{g}^k_1 \circ [\boldsymbol{h}^k_{i-1}, \boldsymbol{c}^k_{i-1}]),$$

with parameters $\boldsymbol{g}^k_1 \in \mathbb{R}^{2d}$ and $\boldsymbol{W} \in \mathbb{R}^{V \times 2d}$. Note that there is a $\boldsymbol{g}^k_1$ vector for each of K discrete states. To additionally implement a kind of slot filling, we allow emissions to be directly copied from the value portion of the records r_j using copy attention (Gülçehre et al., 2016; Gu et al., 2016; Yang et al., 2016). Define copy scores,

$$\rho_j = \boldsymbol{r}_j^{\mathsf{T}} \tanh(\boldsymbol{g}^k_2 \circ \boldsymbol{h}^k_{i-1}),$$

where $\boldsymbol{g}^k_2 \in \mathbb{R}^d$. We then normalize the output-vocabulary and copy scores together, to arrive at

$$\widetilde{\boldsymbol{v}}_{i-1} = \mathrm{softmax}([\boldsymbol{v}_{i-1}, \rho_1, \ldots, \rho_J]),$$

and thus

$$p(y_{t-l_t+i} = w \mid y_{t-l_t+1:t-l_t+i-1}, z_t = k, x) = \widetilde{\boldsymbol{v}}_{i-1,w} + \sum_{j:r_j.m = w} \widetilde{\boldsymbol{v}}_{i-1,V+j}.$$

An Autoregressive Variant The model as specified assumes segments are independent conditioned on the associated latent state and x. While this assumption still allows for reasonable performance, we can tractably allow interdependence between tokens (but not segments) by having each next-token distribution depend on all the previously generated tokens, giving us an autoregressive HSMM. For this model, we will in fact use $p(y_{t-l_t+i} = w \mid y_{1:t-l_t+i-1}, z_t = k, x)$ in defining our emission model, which is easily implemented by using an additional RNN run over all the preceding tokens. We will report scores for both non-autoregressive and autoregressive HSMM decoders below.

5.2 Learning

The model requires fitting a large set of neural network parameters. Since we assume z, l, and f are unobserved, we marginalize over these variables to maximize the log marginal-likelihood of the observed tokens y given x. The HSMM marginal-likelihood calculation can be carried out efficiently with a dynamic program analogous to either the forward- or backward-algorithm familiar from HMMs (Rabiner, 1989).

It is actually more convenient to use the backward-algorithm formulation when using RNNs to parameterize the emission distributions, and we briefly review the backward recurrences here, again following Murphy (2002). We have:

$$\begin{aligned}
\beta_t(j) &= p(y_{t+1:T} \mid z_t = j, f_t = 1, x) \\
&= \sum_{k=1}^{K} \beta^*_t(k)\, p(z_{t+1} = k \mid z_t = j) \\
\beta^*_t(k) &= p(y_{t+1:T} \mid z_{t+1} = k, f_t = 1, x) \\
&= \sum_{l=1}^{L} \Big[\beta_{t+l}(k)\, p(l_{t+1} = l \mid z_{t+1} = k) \\
&\qquad p(y_{t+1:t+l} \mid z_{t+1} = k, l_{t+1} = l)\Big],
\end{aligned}$$

with base case $\beta_T(j) = 1$. We can now obtain the marginal probability of y as $p(y \mid x) = \sum_{k=1}^{K} \beta^*_0(k)\, p(z_1 = k)$, where we have used the fact that f_0 must be 1, and we therefore train to maximize the log-marginal likelihood of the observed y:

$$\ln p(y \mid x; \theta) = \ln \sum_{k=1}^{K} \beta^*_0(k)\, p(z_1 = k). \qquad (1)$$

Since the quantities in (1) are obtained from a dynamic program, which is itself differentiable, we may simply maximize with respect to the parameters θ by back-propagating through the dynamic program; this is easily accomplished with automatic differentiation packages, and we use `pytorch` (Paszke et al., 2017) in all experiments.

5.3 Extracting Templates and Generating

After training, we could simply condition on a new database and generate with beam search, as is standard with encoder-decoder models. However, the structured approach we have developed allows us to generate in a more template-like way, giving us more interpretable and controllable generations.

[The Golden Palace]$_{55}$ [is a]$_{59}$ [coffee shop]$_{12}$ [providing]$_{3}$ [Indian]$_{50}$ [food]$_{1}$ [in the]$_{17}$ [£20-25]$_{26}$ [price range]$_{16}$ [.]$_{2}$ [It is]$_{8}$ [located in the]$_{25}$ [riverside]$_{40}$ [.]$_{53}$ [Its customer rating is]$_{19}$ [high]$_{23}$ [.]$_{2}$

Figure 4: A sample Viterbi segmentation of a training text; subscripted numbers indicate the corresponding latent state. From this we can extract a template with $S = 17$ segments; compare with the template used at the bottom of Figure 1.

First, note that given a database x and reference generation y we can obtain the MAP assignment to the variables z, l, and f with a dynamic program similar to the Viterbi algorithm familiar from HMMs. These assignments will give us a typed segmentation of y, and we show an example Viterbi segmentation of some training text in Figure 4. Computing MAP segmentations allows us to associate text-segments (i.e., phrases) with the discrete labels z_t that frequently generate them. These MAP segmentations can be used in an exploratory way, as a sort of dimensionality reduction of the generations in the corpus. More importantly for us, however, they can also be used to guide generation.

In particular, since each MAP segmentation implies a sequence of hidden states z, we may run a *template extraction* step, where we collect the most common "templates" (i.e., sequences of hidden states) seen in the training data. Each "template" $z^{(i)}$ consists of a sequence of latent states, with $z^{(i)} = z_1^{(i)}, \ldots z_S^{(i)}$ representing the S distinct segments in the i'th extracted template (recall that we will technically have a z_t for each time-step, and so $z^{(i)}$ is obtained by collapsing adjacent z_t's with the same value); see Figure 4 for an example template (with $S = 17$) that can be extracted from the E2E corpus. The bottom of Figure 1 shows a visualization of this extracted template, where discrete states are replaced by the phrases they frequently generate in the training data.

With our templates $z^{(i)}$ in hand, we can then restrict the model to using (one of) them during generation. In particular, given a new input x, we may generate by computing

$$\hat{y}^{(i)} = \arg\max_{y'} p(y', z^{(i)} \mid x), \qquad (2)$$

which gives us a generation $\hat{y}^{(i)}$ for each extracted template $z^{(i)}$. For example, the generation in Figure 1 is obtained by maximizing (2) with x set to the database in Figure 1 and $z^{(i)}$ set to the template extracted in Figure 4. In practice, the $\arg\max$ in (2) will be intractable to calculate exactly due to the use of RNNs in defining the emission distribution, and so we approximate it with a constrained beam search. This beam search looks very similar to that typically used with RNN decoders, except the search occurs only over a segment, for a particular latent state k.

5.4 Discussion

Returning to the discussion of controllability and interpretability, we note that with the proposed model (a) it is possible to explicitly force the generation to use a chosen template $z^{(i)}$, which is itself automatically learned from training data, and (b) that every *segment* in the generated $\hat{y}^{(i)}$ is typed by its corresponding latent variable. We explore these issues empirically in Section 7.1.

We also note that these properties may be useful for other text applications, and that they offer an additional perspective on how to approach latent variable modeling for text. Whereas there has been much recent interest in learning continuous latent variable representations for text (see Section 2), it has been somewhat unclear what the latent variables to be learned are intended to capture. On the other hand, the latent, template-like structures we induce here represent a plausible, probabilistic latent variable story, and allow for a more controllable method of generation.

Finally, we highlight one significant possible issue with this model – the assumption that segments are independent of each other given the corresponding latent variable and x. Here we note that the fact that we are allowed to condition on x is quite powerful. Indeed, a clever encoder could capture much of the necessary interdependence between the segments to be generated (e.g., the correct determiner for an upcoming noun phrase) in its encoding, allowing the segments themselves to be decoded more or less independently, given x.

6 Data and Methods

Our experiments apply the approach outlined above to two recent, data-driven NLG tasks.

6.1 Datasets

Experiments use the E2E (Novikova et al., 2017) and WikiBio (Lebret et al., 2016) datasets, examples of which are shown in Figures 1 and 2, respectively. The former dataset, used for the

2018 E2E-Gen Shared Task, contains approximately 50K total examples, and uses 945 distinct word types, and the latter dataset contains approximately 500K examples and uses approximately 400K word types. Because our emission model uses a word-level copy mechanism, any record with a phrase consisting of n words as its value is replaced with n positional records having a single word value, following the preprocessing of Lebret et al. (2016). For example, "type[coffee shop]" in Figure 1 becomes "type-1[coffee]" and "type-2[shop]."

For both datasets we compare with published encoder-decoder models, as well as with direct template-style baselines. The E2E task is evaluated in terms of BLEU (Papineni et al., 2002), NIST (Belz and Reiter, 2006), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015), and METEOR (Banerjee and Lavie, 2005).[2] The benchmark system for the task is an encoder-decoder style system followed by a reranker, proposed by Dušek and Jurcıcek (2016). We compare to this baseline, as well as to a simple but competitive non-parametric template-like baseline ("SUB" in tables), which selects a training sentence with records that maximally overlap (without including extraneous records) the unseen set of records we wish to generate from; ties are broken at random. Then, word-spans in the chosen training sentence are aligned with records by string-match, and replaced with the corresponding fields of the new set of records.[3]

The WikiBio dataset is evaluated in terms of BLEU, NIST, and ROUGE, and we compare with the systems and baselines implemented by Lebret et al. (2016), which include two neural, encoder-decoder style models, as well as a Kneser-Ney, templated baseline.

6.2 Model and Training Details

We first emphasize two additional methodological details important for obtaining good performance.

Constraining Learning We were able to learn more plausible segmentations of y by constraining the model to respect word spans $y_{t+1:t+l}$ that appear in some record $r_j \in x$. We accomplish this by giving zero probability (within the backward recurrences in Section 5) to any segmentation that splits up a sequence $y_{t+1:t+l}$ that appears in some r_j, or that includes $y_{t+1:t+l}$ as a subsequence of another sequence. Thus, we maximize (1) subject to these hard constraints.

Increasing the Number of Hidden States While a larger K allows for a more expressive latent model, computing K emission distributions over the vocabulary can be prohibitively expensive. We therefore tie the emission distribution between multiple states, while allowing them to have a different transition distributions.

We give additional architectural details of our model in the Supplemental Material; here we note that we use an MLP to embed $\boldsymbol{r}_j \in \mathbb{R}^d$, and a 1-layer LSTM (Hochreiter and Schmidhuber, 1997) in defining our emission distributions. In order to reduce the amount of memory used, we restrict our output vocabulary (and thus the height of the matrix $\boldsymbol{W}$ in Section 5) to only contain words in y that are *not* present in x; any word in y present in x is assumed to be copied. In the case where a word y_t appears in a record r_j (and could therefore have been copied), the input to the LSTM at time $t+1$ is computed using information from r_j; if there are multiple r_j from which y_t could have been copied, the computed representations are simply averaged.

For all experiments, we set $d = 300$ and $L = 4$. At generation time, we select the 100 most common templates $z^{(i)}$, perform beam search with a beam of size 5, and select the generation with the highest overall joint probability.

For our E2E experiments, our best non-autoregressive model has 55 "base" states, duplicated 5 times, for a total of $K = 275$ states, and our best autoregressive model uses $K = 60$ states, without any duplication. For our WikiBio experiments, both our best non-autoregressive and autoregressive models uses 45 base states duplicated 3 times, for a total of $K = 135$ states. In all cases, K was chosen based on BLEU performance on held-out validation data. Code implementing our models is available at `https://github.com/harvardnlp/neural-template-gen`.

7 Results

Our results on automatic metrics are shown in Tables 1 and 2. In general, we find that the templated baselines underperform neural models, whereas our proposed model is fairly competitive with neural models, and sometimes even out-

[2] We use the official E2E NLG Challenge scoring scripts at `https://github.com/tuetschek/e2e-metrics`.

[3] For categorical records, like "familyFriendly", which cannot easily be aligned with a phrase, we simply select only candidate training sentences with the same categorical value.

	BLEU	NIST	ROUGE	CIDEr	METEOR
			Validation		
D&J	69.25	8.48	72.57	2.40	47.03
SUB	43.71	6.72	55.35	1.41	37.87
NTemp	64.53	7.66	68.60	1.82	42.46
NTemp+AR	67.07	7.98	69.50	2.29	43.07
			Test		
D&J	65.93	8.59	68.50	2.23	44.83
SUB	43.78	6.88	54.64	1.39	37.35
NTemp	55.17	7.14	65.70	1.70	41.91
NTemp+AR	59.80	7.56	65.01	1.95	38.75

Table 1: Comparison of the system of Dušek and Jurcıcek (2016), which forms the baseline for the E2E challenge, a non-parametric, substitution-based baseline (see text), and our HSMM models (denoted "NTemp" and "NTemp+AR" for the non-autoregressive and autoregressive versions, resp.) on the validation and test portions of the E2E dataset. "ROUGE" is ROUGE-L. Models are evaluated using the official E2E NLG Challenge scoring scripts.

	BLEU	NIST	ROUGE-4
Template KN †	19.8	5.19	10.7
NNLM (field) †	33.4	7.52	23.9
NNLM (field & word) †	34.7	7.98	25.8
NTemp	34.2	7.94	35.9
NTemp+AR	34.8	7.59	38.6
Seq2seq (Liu et al., 2018)	43.65	-	40.32

Table 2: Top: comparison of the two best neural systems of Lebret et al. (2016), their templated baseline, and our HSMM models (denoted "NTemp" and "NTemp+AR" for the non-autoregressive and autoregressive versions, resp.) on the test portion of the WikiBio dataset. Models marked with a † are from Lebret et al. (2016), and following their methodology we use ROUGE-4. Bottom: state-of-the-art seq2seq-style results from Liu et al. (2018).

performs them. On the E2E data, for example, we see in Table 1 that the SUB baseline, despite having fairly impressive performance for a non-parametric model, fares the worst. The neural HSMM models are largely competitive with the encoder-decoder system on the validation data, despite offering the benefits of interpretability and controllability; however, the gap increases on test.

Table 2 evaluates our system's performance on the test portion of the WikiBio dataset, comparing with the systems and baselines implemented by Lebret et al. (2016). Again for this dataset we see that their templated Kneser-Ney model underperforms on the automatic metrics, and that neural models improve on these results. Here the HSMMs are competitive with the best model of Lebret et al. (2016), and even outperform it on ROUGE. We emphasize, however, that recent, sophisticated approaches to encoder-decoder style database-to-text generation have since surpassed the results of Lebret et al. (2016) and our own, and we show the recent seq2seq style results of Liu et al. (2018), who use a somewhat larger model, at the bottom of Table 2.

Travellers Rest Beefeater
name[Travellers Rest Beefeater], customerRating[3 out of 5], area[riverside], near[Raja Indian Cuisine]
1. [Travellers Rest Beefeater]$_{55}$ [is a]$_{59}$ [3 star]$_{43}$ [restaurant]$_{11}$ [located near]$_{25}$ [Raja Indian Cuisine]$_{40}$ [.]$_{53}$ 2. [Near]$_{31}$ [riverside]$_{29}$ [,]$_{44}$ [Travellers Rest Beefeater]$_{55}$ [serves]$_{3}$ [3 star]$_{50}$ [food]$_{1}$ [.]$_{2}$ 3. [Travellers Rest Beefeater]$_{55}$ [is a]$_{59}$ [restaurant]$_{12}$ [providing]$_{3}$ [riverside]$_{50}$ [food]$_{1}$ [and has a]$_{17}$ [3 out of 5]$_{26}$ [customer rating]$_{16}$ [.]$_{2}$ [It is]$_{8}$ [near]$_{25}$ [Raja Indian Cuisine]$_{40}$ [.]$_{53}$ 4. [Travellers Rest Beefeater]$_{55}$ [is a]$_{59}$ [place to eat]$_{12}$ [located near]$_{25}$ [Raja Indian Cuisine]$_{40}$ [.]$_{53}$ 5. [Travellers Rest Beefeater]$_{55}$ [is a]$_{59}$ [3 out of 5]$_{5}$ [rated]$_{32}$ [riverside]$_{43}$ [restaurant]$_{11}$ [near]$_{25}$ [Raja Indian Cuisine]$_{40}$ [.]$_{53}$

Table 3: Impact of varying the template $z^{(i)}$ for a single x from the E2E validation data; generations are annotated with the segmentations of the chosen $z^{(i)}$. Results were obtained using the NTemp+AR model from Table 1.

7.1 Qualitative Evaluation

We now qualitatively demonstrate that our generations are controllable and interpretable.

Controllable Diversity One of the powerful aspects of the proposed approach to generation is that we can manipulate the template $z^{(i)}$ while leaving the database x constant, which allows for easily controlling aspects of the generation. In Table 3 we show the generations produced by our model for five different neural template sequences $z^{(i)}$, while fixing x. There, the segments in each generation are annotated with the latent states determined by the corresponding $z^{(i)}$. We see that these templates can be used to affect the word-ordering, as well as which fields are mentioned in the generated text. Moreover, because the discrete states align with particular fields (see below), it is generally simple to automatically infer to which fields particular latent states correspond, allowing users to choose which template best meets their requirements. We emphasize that this level of controllability is much harder to obtain for encoder-decoder models, since, at best, a large amount of sampling would be required to avoid generating around a particular mode in the conditional distribution, and even then it would be difficult to control the sort of generations obtained.

kenny warren

name: kenny warren, **birth date:** 1 april 1946, **birth name:** kenneth warren deutscher, **birth place:** brooklyn, new york, **occupation:** ventriloquist, comedian, author, **notable work:** book - the revival of ventriloquism in america

1. [kenneth warren deutscher]$_{132}$ [( ]$_{75}$ [born]$_{89}$ [april 1, 1946]$_{101}$ [ )]$_{67}$ [is an american]$_{82}$ [author]$_{20}$ [and]$_{1}$ [ventriloquist and comedian]$_{69}$ [.]$_{88}$
2. [kenneth warren deutscher]$_{132}$ [( ]$_{75}$ [born]$_{89}$ [april 1, 1946]$_{101}$ [ )]$_{67}$ [is an american]$_{82}$ [author]$_{20}$ [best known for his]$_{95}$ [the revival of ventriloquism]$_{96}$ [.]$_{88}$
3. [kenneth warren]$_{16}$ [“kenny” warren]$_{117}$ [( ]$_{75}$ [born]$_{89}$ [april 1, 1946]$_{101}$ [ )]$_{67}$ [is an american]$_{127}$ [ventriloquist, comedian]$_{28}$ [.]$_{133}$
4. [kenneth warren]$_{16}$ [“kenny” warren]$_{117}$ [( ]$_{75}$ [born]$_{89}$ [april 1, 1946]$_{101}$ [ )]$_{67}$ [is a]$_{104}$ [new york]$_{98}$ [author]$_{20}$ [.]$_{133}$
5. [kenneth warren deutscher]$_{42}$ [is an american]$_{82}$ [ventriloquist, comedian]$_{118}$ [based in]$_{15}$ [brooklyn, new york]$_{84}$ [.]$_{88}$

Table 4: Impact of varying the template $z^{(i)}$ for a single x from the WikiBio validation data; generations are annotated with the segmentations of the chosen $z^{(i)}$. Results were obtained using the NTemp model from Table 2.

Interpretable States Discrete states also provide a method for interpreting the generations produced by the system, since each segment is explicitly typed by the current hidden state of the model. Table 4 shows the impact of varying the template $z^{(i)}$ for a single x from the WikiBio dataset. While there is in general surprisingly little stylistic variation in the WikiBio data itself, there is variation in the information discussed, and the templates capture this. Moreover, we see that particular discrete states correspond in a consistent way to particular pieces of information, allowing us to align states with particular field types. For instance, birth names have the same hidden state (132), as do names (117), nationalities (82), birth dates (101), and occupations (20).

To demonstrate empirically that the learned states indeed align with field types, we calculate the average purity of the discrete states learned for both datasets in Table 5. In particular, for each discrete state for which the majority of its generated words appear in some r_j, the *purity* of a state's record type alignment is calculated as the percentage of the state's words that come from the most frequent record type the state represents. This calculation was carried out over training examples that belonged to one of the top 100 most frequent templates. Table 5 indicates that discrete states learned on the E2E data are quite pure. Discrete states learned on the WikiBio data are less pure, though still rather impressive given that there are approximately 1700 record types represented in the WikiBio data, and we limit the number of states to 135. Unsurprisingly, adding autoregressiveness to the model decreases purity on both datasets, since the model may rely on the autoregressive RNN for typing, in addition to the state's identity.

	NTemp	NTemp+AR
E2E	89.2 (17.4)	85.4 (18.6)
WikiBio	43.2 (19.7)	39.9 (17.9)

Table 5: Empirical analysis of the average purity of discrete states learned on the E2E and WikiBio datasets, for the NTemp and NTemp+AR models. Average purities are given as percents, and standard deviations follow in parentheses. See the text for full description of this calculation.

8 Conclusion and Future Work

We have developed a neural, template-like generation model based on an HSMM decoder, which can be learned tractably by backpropagating through a dynamic program. The method allows us to extract template-like latent objects in a principled way in the form of state sequences, and then generate with them. This approach scales to large-scale text datasets and is nearly competitive with encoder-decoder models. More importantly, this approach allows for controlling the diversity of generation and for producing interpretable states during generation. We view this work both as the first step towards learning discrete latent variable template models for more difficult generation tasks, as well as a different perspective on learning latent variable text models in general. Future work will examine encouraging the model to learn maximally different (or minimal) templates, which our objective does not explicitly encourage, templates of larger textual phenomena, such as paragraphs and documents, and hierarchical templates.

Acknowledgments

SW gratefully acknowledges the support of a Siebel Scholars award. AMR gratefully acknowledges the support of NSF CCF-1704834, Intel Research, and Amazon AWS Research grants.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

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.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(04):431–455.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of nlg systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR*, abs/1702.06235.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Hanjun Dai, Bo Dai, Yan-Ming Zhang, Shuang Li, and Le Song. 2017. Recurrent hidden semi-markov model. In *International Conference on Learning Representations*.

Ondrej Dušek and Filip Jurcıcek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 45.

Mark JF Gales and Steve J Young. 1993. *The theory of segmental hidden Markov models*. University of Cambridge, Department of Engineering.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *ACL*.

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

Blake Howald, Ravikumar Kondadadi, and Frank Schilder. 2013. Domain adaptable semantic clustering in statistical nlg. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 143–154.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *International Conference on Learning Representations*.

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*. Pearson London.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1406–1415.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *International Conference on Learning Representations*.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)*, 48:305–346.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *ACL*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213.

J. Li, R. Jia, H. He, and P. Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *North American Association for Computational Linguistics (NAACL)*.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*, pages 91–99. Association for Computational Linguistics.

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

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. Segmental recurrent neural networks for end-to-end speech recognition. *Interspeech 2016*, pages 385–389.

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

Kathleen McKeown. 1992. *Text generation - using discourse strategies and focus constraints to generate natural language text*. Studies in natural language processing. Cambridge University Press.

Susan W McRoy, Songsak Channarukul, and Syed S Ali. 2000. Yag: A template-based generator for real-time systems. In *Proceedings of the first international conference on Natural language generation-Volume 14*, pages 264–267. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT*, pages 720–730.

Kevin P Murphy. 2002. Hidden semi-markov models (hsmms). *unpublished notes*, 2.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany.

Mari Ostendorf, Vassilios V Digalakis, and Owen A Kimball. 1996. From hmm's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on speech and audio processing*, 4(5):360–378.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS 2017 Autodiff Workshop*.

Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

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 6833–6844.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Hao Tang, Weiran Wang, Kevin Gimpel, and Karen Livescu. 2016. End-to-end training approaches for discriminative segmental models. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 496–502. IEEE.

Ke M Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Chong Wang, Yining Wang, Po-Sen Huang, Abdelrahman Mohamed, Dengyong Zhou, and Li Deng. 2017. Sequence modeling via segmentations. In *International Conference on Machine Learning*, pages 3674–3683.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1395–1405.

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.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *CoRR*, abs/1611.01628.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1307–1316.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pages 5897–5906.

A Supplemental Material

A.1 Additional Model and Training Details

Computing $\boldsymbol{r}_j$ A record r_j is represented by embedding a feature for its type, its position, and its word value in $\mathbb{R}^d$, and applying an MLP with ReLU nonlinearity (Nair and Hinton, 2010) to form $\boldsymbol{r}_j \in \mathbb{R}^d$, similar to Yang et al. (2016) and Wiseman et al. (2017).

LSTM Details The initial cell and hidden-state values for the decoder LSTM are given by $\boldsymbol{Q}_1 \boldsymbol{x}_a$ and $\tanh(\boldsymbol{Q}_2 \boldsymbol{x}_a)$, respectively, where $\boldsymbol{Q}_1, \boldsymbol{Q}_2 \in \mathbb{R}^{d \times d}$.

When a word y_t appears in a record r_j, the input to the LSTM at time $t + 1$ is computed using an MLP with ReLU nonlinearity over the concatenation of the embeddings for r_j's record type, word value, position, and a feature for whether it is the final position for the type. If there are multiple r_j from which y_t could have been copied, the computed representations are averaged. At test time, we use the MAP r_j to compute the input, even if there are multiple matches. For y_t which could not have been copied, the input to the LSTM at time $t+1$ is computed using the same MLP over y_t and three dummy features.

For the autoregressive HSMM, an additional 1-layer LSTM with d hidden units is used. We experimented with having the autoregressive HSMM consume either tokens $y_{1:t}$ in predicting y_{t+1}, or the average embedding of the field *types* corresponding to copied tokens in $y_{1:t}$. The former worked slightly better for the WikiBio dataset (where field types are more ambiguous), while the latter worked slightly better for the E2E dataset.

Transition Distribution The function $\boldsymbol{C}(\boldsymbol{x}_u)$, which produces hidden state embeddings conditional on the source, is defined as $\boldsymbol{C}(\boldsymbol{x}_u) = \boldsymbol{U}_2(\mathrm{ReLU}(\boldsymbol{U}_1 \boldsymbol{x}_u))$, where $\boldsymbol{U}_1 \in \mathbb{R}^{m_3 \times d}$ and $\boldsymbol{U}_2 \in \mathbb{R}^{K \times m_2 \times m_3}$; $\boldsymbol{D}(x)$ is defined analogously. For all experiments, $m_1 = 64$, $m_2 = 32$, and $m_3 = 64$.

Optimization We train with SGD, using a learning rate of 0.5 and decaying by 0.5 each epoch after the first epoch in which validation log-likelihood fails to increase. When using an autoregressive HSMM, the additional LSTM is optimized only after the learning rate has been decayed. We regularize with Dropout (Srivastava et al., 2014).

A.2 Additional Learned Templates

In Tables 6 and 7 we show visualizations of additional templates learned on the E2E and WikiBio data, respectively, by both the non-autoregressive and autoregressive HSMM models presented in the paper. For each model, we select a set of five dissimilar templates in an iterative way by greedily selecting the next template (out of the 200 most frequent) that has the highest percentage of states that do not appear in the previously selected templates; ties are broken randomly. Individual states within a template are visualized using the three most common segments they generate.

1.	The Waterman The Golden Palace Browns Cambridge …	is a is an is a family friendly …	Italian French fast food …	restaurant pub place …	with a with with an …	average high low …	customer rating price range rating …	.						
2.	There is a There is a cheap There is an …	restaurant coffee shop French restaurant …	The Mill Bibimbap House The Twenty Two …	located in the located on the located north of the …	centre of the city river city centre …	that serves serving that provides …	fast food sushi take-away deliveries …	.						
3.	The Olive Grove The Punter The Cambridge Blue …	restaurant pub …	serves offers has …	fast food sushi take-away deliveries …	.									
4.	The Child friendly The average priced …	restaurant coffee shop French restaurant …	The Mill Bibimbap House The Twenty Two …	serves offers has …	English Indian Italian …	food cuisine dishes …	.							
5.	The Strada The Dumpling Tree Alimentum …	provides serves offers …	Indian Chinese English …	food in the food at a food and has a …	customer rating of price range of rating of …	1 out of 5 average 5 out of 5 …	.							
1.	The Eagle The Golden Curry Zizzi …	provides providing serves …	Indian Chinese English …	food cuisine Food …	in the with a and has a …	high moderate average …	price range customer rating rating …	.	It is They are It's …	near located in the located near …	riverside city centre Cafe Sicilia …	.	Its customer rating is It has a The price range is …	1 out of 5 average high … .
2.	Located near Located in the Near …	The Portland Arms riverside city centre …	is an is a family friendly there is a …	Italian fast food French …	restaurant called place called restaurant named …	The Waterman Cocum Loch Fyne …	.							
3.	A An A family friendly …	Italian fast food French …	restaurant pub coffee shop …	is called named …	The Waterman Cocum Loch Fyne …	.								
4.	Located near Located in the Near …	The Portland Arms riverside city centre …	,	The Eagle The Golden Curry Zizzi …	is a is a family friendly is an …	cheap family-friendly family friendly …	Italian fast food French …	restaurant pub coffee shop …	.					
5.	A An A family friendly …	Italian fast food French …	restaurant pub coffee shop …	near located in the located near …	riverside city centre Cafe Sicilia …	is called named …	The Waterman Cocum Loch Fyne …	.						

Table 6: Five templates extracted from the E2E data with the NTemp model (top) and the Ntemp+AR model (bottom).

1.	william henry george augustus frederick marie anne de bourbon ...	(was (; ...	born born on born 1 ...	1968 1960 1970 ...	)])] ...	is an american is a russian was an american ...	politician actor football player ...	.	
2.	sir captain lieutenant ...	john herbert hartley donald charles cameron ...	was a was a british was an english ...	world war i world war first world war ...	national team organization super league ...	.			

3.	john herbert
	hartley
	donald charles cameron
	...
		is a
		was a
		is an
		...
			indie rock
			death metal
			ska
			...
				band
				midfielder
				defenceman
				...
					from
					for
					based in
					...
						australia
						los angeles, california
						chicago
						...
							.		

4.	john herbert hartley donald charles cameron ...	was a is a is a former ...	american major league baseball australian ...	football professional baseball professional ice hockey ...	midfielder defender goalkeeper ...	.			
5.	james william john william ...	“ billy ” wilson smith “ jack ” henry ...	(	1900 c. 1894 1913 ...	–	france budapest buenos aires ...	)	is an american is an english was an american ...	footballer professional footballer rules footballer ...
	who plays for who currently plays for who played with ...	paganese south melbourne fc dynamo kyiv ...	in the of the and the ...	vicotiral football league national football league australian football league ...	(	vfl nfl afl ...	)	.	
1.	aftab ahmed anderson da silva david jones ...	(; ...	born born on born 1 ...	1951 1970 1974 ...	)] ...	is an american was an american is an english ...	actor actress cricketer ...	.	
2.	aftab ahmed anderson da silva david jones ...	was a is a former is a ...	world war i liberal baseball ...	member of the party member of the recipient of the ...	austrian pennsylvania montana ...	house of representatives legislature senate ...	.		
3.	adjutant lieutenant captain ...	aftab ahmed anderson da silva david jones ...	was a is a former is a ...	world war i liberal baseball ...	member of the party member of the recipient of the ...	knesset scottish parliament fc lokomotiv liski ...	.		
4.	william john william james “ ...	“ billy ” watson smith jim ” edward ...	(	1913 c. 1900 1913 ...	– in - ...	1917 surrey, england british columbia ...	)	was an american was an australian is an american ...	football player rules footballer defenceman ...
	who plays for who currently plays for who played with ...	collingwood st kilda carlton ...	in the of the and the ...	victorial football league national football league australian football league ...	(	vfl afl nfl ...	)	.	
5.	aftab ahmed anderson da silva david jones ...	is a is a former is a female ...	member of the party member of the recipient of the ...	knesset scottish parliament fc lokomotiv liski ...	.				

Table 7: Five templates extracted from the WikiBio data with the NTemp model (top) and the Ntemp+AR model (bottom).