Neural Data-to-Text Generation via Jointly Learning the Segmentation and Correspondence

Xiaoyu Shen[1,2] **, Ernie Chang**[3]**, Hui Su**[4]**, Cheng Niu**[4] **and Dietrich Klakow**[1]
[1]Spoken Language Systems (LSV) [2]Max Planck Institute for Informatics
[3]Department of Language Science and Technology, Saarland Informatics Campus
[4]Pattern Recognition Center, Wechat AI, Tencent Inc, China
xshen@mpi-inf.mpg.de,cychang@coli.uni-saarland.de
{aaronsu,niucheng}@tencent.com,dklakow@lsv.uni-saarland.de

Abstract

The neural attention model has achieved great success in data-to-text generation tasks. Though usually excelling at producing fluent text, it suffers from the problem of information missing, repetition and "hallucination". Due to the black-box nature of the neural attention architecture, avoiding these problems in a systematic way is non-trivial. To address this concern, we propose to explicitly segment target text into fragment units and align them with their data correspondences. The segmentation and correspondence are jointly learned as latent variables without any human annotations. We further impose a soft statistical constraint to regularize the segmental granularity. The resulting architecture maintains the same expressive power as neural attention models, while being able to generate fully interpretable outputs with several times less computational cost. On both E2E and WebNLG benchmarks, we show the proposed model consistently outperforms its neural attention counterparts.

1 Introduction

Data-to-text generation aims at automatically producing natural language descriptions of structured database (Reiter and Dale, 1997). Traditional statistical methods usually tackle this problem by breaking the generation process into a set of local decisions that are learned separately (Belz, 2008; Angeli et al., 2010; Kim and Mooney, 2010; Oya et al., 2014). Recently, neural attention models conflate all steps into a single end-to-end system and largely simplify the training process (Mei et al., 2016; Lebret et al., 2016; Shen et al., 2017; Su et al., 2018, 2019; Chang et al., 2020). However, the black-box conflation also renders the generation uninterpretable and hard to control (Wiseman et al., 2018; Shen et al., 2019a). Verifying the generation correctness in a principled way is non-trivial. In practice, it often suffers from the problem of information missing, repetition and "hallucination" (Dušek et al., 2018, 2020).

Source data:
Name[Clowns], PriceRange[more than £30], EatType[pub], FamilyFriendly[no]

Generation:
①Name → ②(Clowns)
③FamilyFriendly → ④(is a child-free)
⑤PriceRange → ⑥(, expensive)
⑦EatType → ⑧(pub.)

Figure 1: Generation from our model on the E2E dataset. Decoding is performed segment-by-segment. Each segment realizes one data record. ①~⑧ mark the decision order in the generation process.

In this work, we propose to explicitly exploit the *segmental structure* of text. Specifically, we assume the target text is formed from a sequence of segments. Every segment is the result of a two-stage decision: (1) Select a proper data record to be described and (2) Generate corresponding text by *paying attention only to the selected data record*. This decision is repeated until all desired records have been realized. Figure 1 illustrates this process.

Compared with neural attention, the proposed model has the following advantages: (1) We can monitor the corresponding data record for every segment to be generated. This allows us to easily control the output structure and verify its correctness[1]. (2) Explicitly building the correspondence between segments and data records can potentially reduce the hallucination, as noted in (Wu et al., 2018; Deng et al., 2018) that hard alignment usually outperforms soft attention. (3) When decoding each segment, the model pays attention only to the

[1]For example, we can perform a similar constrained decoding as in Balakrishnan et al. (2019) to rule out outputs with undesired patterns.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7155–7165
July 5 - 10, 2020.

selected data record instead of averaging over the entire input data. This largely reduces the memory and computational costs [2].

To train the model, we *do not* rely on any human annotations for the segmentation and correspondence, but rather marginalize over all possibilities to maximize the likelihood of target text, which can be efficiently done within polynomial time by dynamic programming. This is essentially similar to traditional methods of inducing segmentation and alignment with semi-markov models (Daumé III and Marcu, 2005; Liang et al., 2009). However, they make strong independence assumptions thus perform poorly as a generative model (Angeli et al., 2010). In contrast, the transition and generation in our model condition on *all previously generated text*. By integrating an autoregressive neural network structure, our model is able to capture unbounded dependencies while still permitting tractable inference. The training process is stable as it does not require any sampling-based approximations. We further add a soft statistical constraint to control the segmentation granularity via posterior regularization (Ganchev et al., 2010). On both the E2E and WebNLG benchmarks, our model is able to produce significantly higher-quality outputs while being several times computationally cheaper. Due to its fully interpretable segmental structure, it can be easily reconciled with heuristic rules or hand-engineered constraints to control the outputs.

2 Related Work

Data-to-text generation is traditionally dealt with using a pipeline structure containing content planning, sentence planning and linguistic realization (Reiter and Dale, 1997). Each target text is split into meaningful fragments and aligned with corresponding data records, either by hand-engineered rules (Kukich, 1983; McKeown, 1992) or statistical induction (Liang et al., 2009; Koncel-Kedziorski et al., 2014; Qin et al., 2018). The segmentation and alignment are used as supervision signals to train the content and sentence planner (Barzilay and Lapata, 2005; Angeli et al., 2010). The linguistic realization is usually implemented by template mining from the training corpus (Kondadadi et al., 2013; Oya et al., 2014). Our model adopts a similar pipeline generative process, but integrates all the sub-steps into a single end-to-end trainable neural architecture. It can be considered as a neural extension of the PCFG system in Konstas and Lapata (2013), with a more powerful transition probability considering inter-segment dependence and a state-of-the-art attention-based language model as the linguistic realizer. Wiseman et al. (2018) tried a similar neural generative model to induce templates. However, their model only captures loose data-text correspondence and adopts a weak markov assumption for the segment transition probability. Therefore, it underperforms the neural attention baseline as for generation. Our model is also in spirit related to recent attempts at separating content planning and surface realization in neural data-to-text models (Zhao et al., 2018; Puduppully et al., 2019; Moryossef et al., 2019; Ferreira et al., 2019). Nonetheless, all of them resort to *manual annotations or hand-engineered rules applicable only for a narrow domain*. Our model, instead, automatically learn the optimal content planning via exploring over exponentially many segmentation/correspondence possibilities.

There have been quite a few neural alignment models applied to tasks like machine translation (Wang et al., 2018; Deng et al., 2018), character transduction (Wu et al., 2018; Shankar and Sarawagi, 2019) and summarization (Yu et al., 2016; Shen et al., 2019b). Unlike word-to-word alignment, we focus on learning the alignment between data records and text segments. Some works also integrate neural language models to jointly learn the segmentation and correspondence, e.g., phrase-based machine translation (Huang et al., 2018), speech recognition (Wang et al., 2017) and vision-grounded word segmentation (Kawakami et al., 2019). Data-to-text naturally fits into this scenario since each data record is normally verbalized in one continuous text segment.

3 Background: Data-to-Text

Let X, Y denote a source-target pair. X is structured data containing a set of records and Y corresponds to $y_1, y_2, \ldots, y_m$ which is a text description of X. The goal of data-to-text generation is to learn a distribution $p(Y|X)$ to automatically generate proper text describing the content of the data.

The neural attention architecture handles this task with an encode-attend-decode process (Bahdanau et al., 2015). The input X is processed into a sequence of $x_1, x_2, \ldots, x_n$, normally by flatten-

[2] Coarse-to-fine attention (Ling and Rush, 2017; Deng et al., 2017) was proposed for the same motivation, but they resort to reinforcement learning which is hard to train, and the performance is sacrificed for efficiency.

ing the data records (Wiseman et al., 2017). The encoder encodes each x_i into a vector h_i. At each time step, the decoder attends to encoded vectors and outputs the probability of the next token by $p(y_t|y_{1:t-1}, A_t)$. A_t is a weighted average of source vectors:

$$A_t = \sum_i \alpha_{t,i} h_i$$
$$\alpha_{t,i} = \frac{e^{f(h_i,d_t)}}{\sum_j e^{f(h_j,d_t)}} \quad (1)$$

d_t is the hidden state of the decoder at time step t. f is a score function to compute the similarity between h_i and d_t (Luong et al., 2015).

4 Approach

Suppose the input data X contains a set of records $r_1, r_2, ..., r_K$. Our assumption is that the target text $y_{1:m}$ can be segmented into a sequence of fragments. Each fragment corresponds to one data record. As the ground-truth segmentation and correspondence are not available, we need to enumerate over all possibilities to compute the likelihood of $y_{1:m}$. Denote by $\mathcal{S}_y$ the set containing all valid segmentation of $y_{1:m}$. For any valid segmentation $s_{1:\tau_s} \in \mathcal{S}_y$, $\pi(s_{1:\tau_s}) = y_{1:m}$, where π means concatenation and τ_s is the number of segments. For example, let $m = 5$ and $\tau_s = 3$. One possible segmentation would be $s_{1:\tau_s} = \{\{y_1, y_2, \$\}, \{y_3, \$\}, \{y_4, y_5, \$\}\}$. $\$$ is the end-of-segment symbol and is removed when applying the π operator. We further define $c(*)$ to be the corresponding data record(s) of $*$. The likelihood of each text is then computed by enumerating over all possibilities of $s_{1:\tau_s}$ and $c(s_{1:\tau_s})$:

$$\begin{aligned} p(y_{1:m}|X) &= \sum_{s_{1:\tau_s} \in \mathcal{S}_y} p(s_{1:\tau_s}|X) \\ &= \sum_{s_{1:\tau_s} \in \mathcal{S}_y} \prod_{o=1}^{\tau_s} \sum_{c(s_o)=r_1}^{r_K} p(s_o|\pi(s_{<o}), c(s_o)) \\ &\times p(c(s_o)|\pi(s_{<o}), c(s_{<o})) \end{aligned} \quad (2)$$

Every segment is generated by first selecting the data record based on the *transition probability* $p(c(s_o)|\pi(s_{<o}), c(s_{<o}))$, then generating tokens based on the word *generation probability* $p(s_o|\pi(s_{<o}), c(s_o))$. Figure 2 illustrates the generation process of our model.

Generation Probability We base the generation probability on the same decoder as in neural attention models. The only difference is that *the model can only pay attention to its corresponding data record*. The attention scores of other records are masked out when decoding s_o:

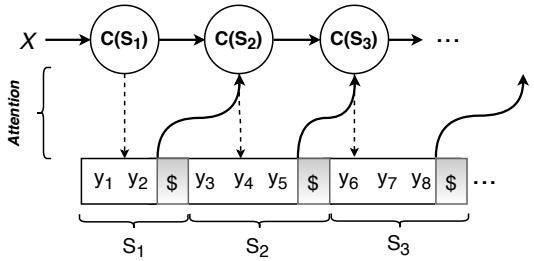

Figure 2: Generation process of our approach. Segment end symbol $ is ignored when updating the state of the decoder. *Solid arrows* indicate the transition model and *dashed arrows* indicate the generation model. Every segment s_o is generated by attending only to the corresponding data record $c(s_o)$.

$$\alpha_{t,i} = \frac{e^{f(h_i,d_t)} \mathbb{1}(x_i \in c(s_o))}{\sum_j e^{f(h_j,d_t)} \mathbb{1}(x_j \in c(s_o))}$$

where $\mathbb{1}$ is the indicator function. This forces the model to learn proper correspondences and enhances the connection between each segment and the data record it describes.

Following the common practice, we define the output probability with the pointer generator (See et al., 2017; Wiseman et al., 2017):

$$\begin{aligned} p_{gen} &= \sigma(\text{MLP}_g([d_t \circ A_t])) \\ p_{vocab} &= \text{softmax}(W_1 d_t + W_2 A_t) \\ p_\theta(y_t|y_{<t}) &= p_{gen} p_{vocab}(y_t) \\ &+ (1 - p_{gen}) \sum_{i:y_t=x_i} a_{t,i} \end{aligned}$$

d_t is the decoder's hidden state at time step t. $\circ$ denotes vector concatenation. A_t is the context vector. MLP indicates multi-layer perceptron and σ normalizes the score between $(0, 1)$. W_1 and W_2 are trainable matrices. p_{gen} is the probability that the word is generated from a fixed vocabulary distribution p_{vocab} instead of being copied. The final decoding probability $p_\theta(y_t)$ is marginalized over p_{vocab} and the copy distribution. The generation probability of s_o factorizes over the words within it and the end-of-segment token:

$$p(s_o|\pi(s_{<o}), c(s_o)) = p_\theta(\$|y_{1:t}) \prod_{y_t \in s_o} p_\theta(y_t|y_{<t})$$

Transition Probability We make a mild assumption that $c(s_o)$ is dependent only on $c(s_{o-1})$ and $\pi(s_{1:o-1})$ but irrelevant of $c(s_{<o-1})$, which is a common practice when modelling alignment (Och

et al., 1999; Yu et al., 2016; Shankar and Sarawagi, 2019). The transition probability is defined as:

$$\begin{aligned} & p(c(s_o) = r_i | c(s_{<o}), \pi(s_{<o})) \\ \approx & p(c(s_o) = r_i | c(s_{o-1}), \pi(s_{<o})) \\ \propto & f(r_i)^T [M^T A_{s_{o-1}} + N^T d_{s_{o-1}}] \end{aligned} \tag{3}$$

A softmax layer is finally applied to the above equation to normalize it as a proper probability distribution. $f(r_i)$ is a representation of r_i, which is defined as a max pooling over all the word embeddings contained in r_i. $A_{s_{o-1}}$ is the attention context vector when decoding the last token in s_{o-1}, defined as in Equation 1. It carries important information from $c(s_{o-1})$ to help predict $c(s_o)$. $d_{s_{o-1}}$ is the hidden state of the neural decoder which goes through all history tokens $\pi(s_{1:o-1})$. M, N are trainable matrices to project $A_{s_{o-1}}$ and $d_{s_{o-1}}$ into the same dimension as $f(r_i)$.

We further add one constraint to prohibit *self-transition*, which can be easily done by zeroing out the transition probability in Equation 3 when $c(s_o) = c(s_{o-1})$. This forces the model to group together text describing the same data record.

Since Equation 3 conditions on all previously generated text, it is able to capture more complex dependencies as in semi-markov models (Liang et al., 2009; Wiseman et al., 2018).

Null Record In our task, we find some frequent phrases, e.g., "it is", "and", tend to be wrongly aligned with some random records, similar to the garbage collection issue in statistical alignment (Brown et al., 1993). This hurt the model interpretability. Therefore, we introduce an additional null record r_0 to attract these non-content phrases. The context vector when aligned to r_0 is a zero vector so that the decoder will decode words based solely on the language model without relying on the input data.

Training Equation 2 contains exponentially many combinations to enumerate over. Here we show how to efficiently compute the likelihood with the forward algorithm in dynamic programming (Rabiner, 1989). We define the forward variable $\alpha(i, j) = p(y_{1:i}, c(y_i) = j | X)$. With the base $\alpha(1, j) = p(y_1 | c(y_1) = j)$. The recursion goes as follows for $i = 1, 2, \ldots, m-1$:

$$\begin{aligned} \alpha(i+1, j) = & \sum_{p=1}^{i} \sum_{q=r_0}^{r_K} \alpha(p, q) \\ & \times p(c(y_{p+1}) = j | c(y_p) = q, y_{1:p}) \\ & \times p(y_{p+1:i+1} | c(y_{p+1:i+1}) = q, y_{1:p}) \\ & \times p(\$ | c(y_{p+1:i+1}) = q, y_{1:i+1}) \end{aligned} \tag{4}$$

The final likelihood of the target text can be computed as $p(y_{1:m}|X) = \sum_{j=r_0}^{r_K} \alpha(m, j)$. As the forward algorithm is fully differentiable, we maximize the log-likelihood of the target text by backpropagating through the dynamic programming. The process is essentially equivalent to the generalized EM algorithm (Eisner, 2016). By means of the modern automatic differentiation tools, we avoid the necessity to calculate the posterior distribution manually (Kim et al., 2018).

To speed up training, we set a threshold L to the maximum length of a segment as in Liang et al. (2009); Wiseman et al. (2018). This changes the complexity in Equation 4 to a constant $O(LK)$ instead of scaling linearly with the length of the target text. Moreover, as pointed out in Wang et al. (2017), the computation for the longest segment can be reused for shorter segments. We therefore first compute the generation and transition probability for the whole sequence in one pass. The intermediate results are then cached to efficiently proceed the forward algorithm without any re-computation.

One last issue is the numerical precision, it is important to use the log-space binary operations to avoid underflow (Kim et al., 2017).

Near[riverside], Food[French], EatType[pub], Name[Cotto]
1. [Near]$_{\text{Near}}$[the]$_{\text{Null}}$[riverside]$_{\text{Near}}$[is a]$_{\text{Null}}$[French]$_{\text{Food}}$ [pub]$_{\text{EatType}}$[called]$_{\text{Null}}$[Cotto]$_{\text{Name}}$[.]$_{\text{Null}}$
2. [Near the riverside]$_{\text{Near}}$[is]$_{\text{Null}}$[a French]$_{\text{Food}}$[pub]$_{\text{EatType}}$ [called Cotto]$_{\text{Name}}$[.]$_{\text{Null}}$
3. [Near the riverside]$_{\text{Near}}$[is a French]$_{\text{Food}}$[pub]$_{\text{EatType}}$ [called Cotto .]$_{\text{Name}}$
4. [Near the riverside]$_{\text{Near}}$[is a French pub]$_{\text{Food}}$ [called Cotto .]$_{\text{Name}}$

Table 1: Segmentation with various granularities. 1 is too fine-grained while 4 is too coarse. We expect a segmentation like 2 or 3 to better control the generation.

Segmentation Granularity There are several valid segmentations for a given text. As shown in Table 1, when the segmentation (example 1) is too fine-grained, controlling the output information becomes difficult because the content of

one data record is realized in separate pieces [3]. When it is too coarse, the alignment might become less accurate (as in Example 4, "pub" is wrongly merged with previous words and aligned together to the "Food" record). In practice, we expect the segmentation to stay with accurate alignment yet avoid being too brokenly separated. To control the granularity as we want, we utilize posterior regularization (Ganchev et al., 2010) to constrain the expected number of segments for each text [4], which can be calculated by going through a similar forward pass as in Equation 4 (Eisner, 2002). Most computation is shared without significant extra burden. The final loss function is:

$$-\log \mathbb{E}_{\mathcal{S}_y} p(s_{1:\tau_s}|X) + \max\left(\left|\mathbb{E}_{\mathcal{S}_y}\tau_s - \eta\right|, \gamma\right) \quad (5)$$

$\log \mathbb{E}_{\mathcal{S}_y} p(s_{1:\tau_s}|X)$ is the log-likelihood of target text after marginalizing over all valid segmentations. $\mathbb{E}_{\mathcal{S}_y}\tau_s$ is the expected number of segments and η, γ are hyperparameters. We use the max-margin loss to encourage $\mathbb{E}_{\mathcal{S}_y}\tau_s$ to stay close to η under a tolerance range of γ.

Decoding The segment-by-segment generation process allows us to easily constrain the output structure. Undesirable patterns can be rejected before the whole text is generated. We adopt three simple constraints for the decoder:

1. Segments must not be empty.
2. The same data record cannot be realized more than once (except for the null record).
3. The generation will not finish until all data records have been realized.

Constraint 2 and 3 directly address the information repetition and missing problem. When segments are incrementally generated, the constraints will be checked against for validity. Note that adding the constraints hardly incur any cost, the decoding process is still finished *in one pass*. No post-processing or reranking is needed.

[3] The finer-grained segmentation might be useful if the focus is on modeling the detailed discourse structure instead of the information accuracy (Reed et al., 2018; Balakrishnan et al., 2019), which we leave for future work.

[4] We can also utilize some heuristic rules to help segmentation. For example, we can prevent breaking syntactic elements obtained from an external parser (Yang et al., 2019) or match entity names with handcrafted rules (Chen et al., 2018). The interpretability of the segmental structure allows easy combination with these rules. We focus on a general *domain-agnostic* method in this paper, though heuristic rules might bring further improvement under certain cases.

Computational Complexity Suppose the input data has M records and each record contains N tokens. The computational complexity for neural attention models is $O(MN)$ at each decoding step where the whole input is retrieved. Our model, similar to chunkwise attention (Chiu and Raffel, 2018) or coarse-to-fine attention (Ling and Rush, 2017), reduces the cost to $O(M+N)$, where we select the record in $O(M)$ at the beginning of each segment and attend only to the selected record in $O(N)$ when decoding every word. For larger input data, our model can be significantly cheaper than neural attention models.

5 Experiment Setup

Dataset We conduct experiments on the E2E (Novikova et al., 2017b) and WebNLG (Colin et al., 2016) datasets. E2E is a crowd-sourced dataset containing 50k instances in the restaurant domain. The inputs are dialogue acts consisting of three to eight slot-value pairs. WebNLG contains 25k instances describing entities belonging to fifteen distinct DBpedia categories. The inputs are up to seven RDF triples of the form *(subject, relation, object)*.

Implementation Details We use a bi-directional LSTM encoder and uni-directional LSTM decoder for all experiments. Input data records are concatenated into a sequence and fed into the encoder. We choose the hidden size of encoder/decoder as 512 for E2E and 256 for WebNLG. The word embedding is with size 100 for both datasets and initialized with the pre-trained Glove embedding [5] (Pennington et al., 2014). We use a drop out rate of 0.3 for both the encoder and decoder. Models are trained using the Adam optimizer (Kingma and Ba, 2014) with batch size 64. The learning rate is initialized to 0.01 and decays an order of magnitude once the validation loss increases. All hyperparameters are chosen with grid search according to the validation loss. Models are implemented based on the open-source library PyTorch (Paszke et al., 2019). We set the hyperparameters in Eq. 5 as $\eta = K, \gamma = 1$ (recall that K is the number of records in the input data). The intuition is that every text is expected to realize the content of all K input records. It is natural to assume every text can be roughly segmented into K fragments, each corresponding to one data record. A deviation of

[5] nlp.stanford.edu/data/glove.6B.zip

$K \pm 1$ is allowed for noisy data or text with complex structures.

Metrics We measure the quality of system outputs from three perspectives: (1) *word-level overlap* with human references, which is a commonly used metric for text generation. We report the scores of BLEU-4 (Papineni et al., 2002), ROUGE-L (Lin, 2004), Meteor (Banerjee and Lavie, 2005) and CIDEr (Vedantam et al., 2015) . (2) *human evaluation*. Word-level overlapping scores usually correlate rather poorly with human judgements on fluency and information accuracy (Reiter and Belz, 2009; Novikova et al., 2017a). Therefore, we passed the input data and generated text to human annotators to judge if the text is fluent by grammar (scale 1-5 as in Belz and Reiter (2006)), contains wrong fact inconsistent with input data, repeats or misses information. We report the *averaged score* for fluency and *definite numbers* for others. The human is conducted on a sampled subset from the test data. To ensure the subset covers inputs with all possible number of records ($K \in [3, 8]$ for E2E and $K \in [1, 7]$ for WebNLG), we sample 20 instances for every possible K. Finally,we obtain 120 test cases for E2E and 140 for WebNLG [6]. (3) *Diversity of outputs*. Diversity is an important concern for many real-life applications. We measure it by the number of unique unigrams and trigrams over system outputs, as done in Dušek et al. (2020).

6 Results

In this section, we first show the effects of the granularity regularization we proposed, then compare model performance on two datasets and analyze the performance difference. Our model is compared against the neural attention-based pointer generator (**PG**) which does not explicit learn the segmentation and correspondence. To show the effects of the constrained decoding mentioned in §4, Decoding. we run our model with only the first constraint to prevent empty segments (denoted by **ours** in experiments), with the first two constraints to prevent repetition (denoted by **ours (+R)**), and with all constraints to further reduce information missing (denoted by **ours (+RM)**).

Segmentation Granularity We show the effects of the granularity regularization (§4, Segmentation Granularity) in Fig 3. When varying the model size, the segmentation granularity changes much if no regularization is imposed. Intuitively if the generation module is strong enough (larger hidden size), it can accurately estimate the sentence likelihood itself without paying extra cost of switching between segments, then it tends to reduce the number of transitions. Vice versa, the number of transitions will grow if the transition module is stronger (larger embedding size). With the regularization we proposed, the granularity remains what we want regardless of the hyperparameters. We can thereby freely decide the model capacity without worrying about the difference of segmentation behavior.

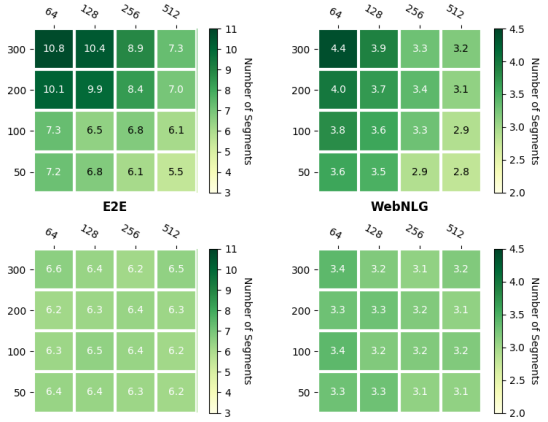

Figure 3: Average expected number of segments with varying hyperparameters. x-axis is the encoder/decoder hidden size and y-axis is the word embedding size. Upper two figures are without the granularity regularization and the bottom two are with regularization.

Results on E2E On the E2E dataset, apart from our implementations, we also compare agianst outputs from the **SLUG** (Juraska et al., 2018), the overall winner of the E2E challenge (seq2seq-based), **DANGNT** (Nguyen and Tran, 2018), the best grammar rule based model, **TUDA** (Puzikov and Gurevych, 2018), the best template based model, and the autoregressive neural template model (**N_TEMP**) from Wiseman et al. (2018). SLUG uses a heuristic slot aligner based on a set of handcrafted rules and combine a complex pipeline of data augmentation, selection, model ensemble and reranker, while our model has a simple end-to-end learning paradigm with no special delexicalizing, training or decoding tricks. Table 2 reports the evaluated results. Seq2seq-based models are more diverse than rule-based models at the cost of higher chances of making errors. As rule-based systems are by design always faithful to the in-

[6] The original human evaluation subset of WebNLG is randomly sampled, most of the inputs contain less than 3 records, so we opt for a new sample for a thorough evaluation.

Metrics	Word Overlap				Human Evaluation				Diversity	
Models	BLEU	R-L	Meteor	CIDEr	Fluent	Wrong	Repeat	Miss	Dist-1	Dist-3
SLUG	**0.662**	**0.677**	0.445	**2.262**	4.94	5	**0**	17	74	507
DANGNT	0.599	0.663	0.435	2.078	4.97	**0**	**0**	21	61	301
TUDA	0.566	0.661	**0.453**	1.821	**4.98**	**0**	**0**	**10**	57	143
N_TEMP	0.598	0.650	0.388	1.950	4.84	19	3	35	**119**	**795**
PG	0.638	0.677	0.449	2.123	4.91	15	1	29	133	822
OURS	0.647	**0.683**	0.453	2.222	**4.96**	**0**	1	15	127	870
OURS (+R)	0.645	0.681	0.452	2.218	4.95	**0**	**0**	13	133	881
OURS (+RM)	**0.651**	0.682	**0.455**	**2.241**	4.95	**0**	**0**	**3**	**135**	**911**

Table 2: Automatic and human evaluation results on E2E dataset. **SLUG**, **DANGNT**, **TUDA** and **N_TEMP** are from previous works and the other models are our own implementations.

Metrics	Word Overlap				Human Evaluation				Diversity	
Models	BLEU	R-L	Meteor	CIDEr	Fluent	Wrong	Repeat	Miss	Dist-1	Dist-3
MELBOURNE	0.450	**0.635**	0.376	**2.814**	4.16	42	22	37	3167	**13,744**
UPF-FORGE	0.385	0.609	0.390	2.500	4.08	**29**	**6**	**28**	**3191**	12,509
PG	0.452	0.652	0.384	2.623	4.13	43	26	42	3,218	13,403
OURS	0.453	0.656	0.388	2.610	4.23	26	19	31	3,377	14,516
OURS (+R)	0.456	**0.657**	0.390	**2.678**	**4.28**	**18**	**2**	24	3,405	14,351
OURS (+RM)	**0.461**	0.654	**0.398**	2.639	4.26	23	4	**5**	**3,457**	**14,981**

Table 3: Automatic and human evaluation results on WebNLG dataset. **MELBOURNE** and **UPFUPF-FORGE** are from previous works and the other models are our own implementations.

Input:	[name the mill][eattype restaurant][food english][pricerange moderate][customerrating 1 out of 5][area riverside] ...
PG:	the mill is a **low - priced** restaurant **in the city centre that delivers take - away** . it is located near café rouge.
Input:	[name the mill][eattype restaurant][food english][pricerange moderate][customerrating 1 out of 5][areariverside] ...
Ours:	[the mill][restaurant][near café rouge][in riverside][serves english food][at moderate prices][. it is kid friendly and]...

Table 4: (E2E) Attention map when decoding the word "low" in the PG model and "moderate" in our model. Hallucinated contents are **bolded**. The PG model wrongly attended to other slots thereby "hallucinated" the content of "low-priced". Our model always attends to one single slot instead of averaging over the whole inputs, the chance of hallucination is largely reduced.

put information, they made zero wrong facts in their outputs. Most models do not have the fact repetition issue because of the relatively simple patterns in the E2E dataset. therefore, adding the (+R) constraint only improves the performance minorly. The (+RM) constraint reduces the number of information missing to 3 without hurting the fluency. All the 3 missing cases are because of the wrong alignment between the period and one data record, which can be easily fixed by defining a simple rule. We put the error analysis in appendix A. N_Temp performs worst among all seq2seq-based systems because of the restrictions we mentioned in §2. As also noted by the author, it trades-off the generation quality for interpretability and controllability. In contrast, our model, despite relying on no heuristics or complex pipelines, *made zero wrong facts with the lowest information missing rate, even surpassing rule-based models*. It also maintains interpretable and controllable without sacrificing the generation quality.

Results on WebNLG Table 3 reports the results evaluated on the WebNLG dataset. We also include results from **MELBOURNE**, a seq2seq-based system achieving highest scores on automatic metrics in the WebNLG challenge and **UPF-FORGE**, a classic grammar-based system that wins in the human evaluation WebNLG contains significantly more distinct types of attributes than E2E, so the chance of making errors or repetitions increases greatly. Nevertheless, our model still *performs on-par on automatic metrics with superior information adequacy and output diversity*. The (+R) decoding constraint becomes important since the outputs in WebNLG are much longer than those in E2E, neural network models have problems tracking the history generation beyond certain range. Models might repeat facts that have been already generated long back before. The (+R) constraint effectively reduces the repetition cases from 19 to 2. These 2 cases are intra-segment repetitions and failed to be detected since our model can only track inter-segment constraints (examples are in appendix A). The (+RM) constraint brings down the information missing cases to 5 with slightly more wrong and repeated facts compared with (+R). Forcing models

Figure 4: Example generations from WebNLG. Relation types are underlined and repeated generations are **bolded**. Segments and corresponding records in our model are marked in the same color. By adding explicit constraints to the decoding process, repetition and missing issues can be largely reduced. (better viewed in color)

to keep generating until coveraging all records will inevitably increase the risk of making errors.

Discussions In summary, our models generates *most diverse outputs, achieves similar or better performances in word-overlap automatic metrics while significantly reduces the information hallucination, repetition and missing problems*. An example of hallucination is shown in Table 4. The standard PG model "hallucinated" the contents of "low-priced", "in the city center" and "delivers takeaway". The visualized attention maps reveal that it failed to attend properly when decoding the word "low". The decoding is driven mostly by language models instead of the contents of input data. In contrast, as we explicitly align each segment to one slot, the attention distribution of our model is *concentrated on one single slot rather than averaged over the whole input*, the chance of hallucinating is therefore largely reduced.

Figure 4 shows some example generations from WebNLG. Without adding the decoding constraints, PG and our model both suffer from the problem of information repetition and missing. However, the interpretability of our model enables us to easily avoid these issues by constraining the segment transition behavior. For the attention-based PG model, there exists no simple way of applying these constraints. We can also explicitly control the output structure similar to Wiseman et al. (2018), examples are shown in appendix B.

7 Conclusion

In this work, we exploit the segmental structure in data-to-text generation. The proposed model significantly alleviates the information hallucination, repetition and missing problems without sacrificing the fluency and diversity. It is end-to-end trainable, domain-independent and allows explicit control over the structure of generated text. As our model is interpretable in the correspondence between segments and input records, it can be easily combined with hand-engineered heuristics or user-specific requirements to further improve the performance.

Acknowledgements

This research was funded in part by the DFG collaborative research center SFB 1102. Ernie Chang is supported by SFB 248 Foundations of Perspicuous Software Systems (E2); Xiaoyu Shen is supported by IMPRS-CS fellowship. We sincerely thank the anonymous reviewers for their insightful comments that helped us to improve this paper.

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 *International Conference on Learning Representations*.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338. Association for Computational Linguistics.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):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*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Ernie Chang, David Ifeoluwa Adelani, Xiaoyu Shen, and Vera Demberg. 2020. Unsupervised pidgin text generation by pivoting english data and self-training. *arXiv preprint arXiv:2003.08272*.

Mingje Chen, Gerasimos Lampouras, and Andreas Vlachos. 2018. Sheffield at e2e: structured prediction approaches to end-to-end language generation. *arxiv*.

Chung-Cheng Chiu and Colin Raffel. 2018. Monotonic chunkwise attention. *ICLR*.

Emilie Colin, Claire Gardent, Yassine M'rabet, Shashi Narayan, and Laura Perez-Beltrachini. 2016. The WebNLG challenge: Generating text from DBPedia data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 163–167, Edinburgh, UK. Association for Computational Linguistics.

Hal Daumé III and Daniel Marcu. 2005. Induction of word and phrase alignments for automatic document summarization. *Computational Linguistics*, 31(4):505–530.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 980–989. JMLR. org.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*, pages 9712–9724.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the e2e nlg challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. *EMNLP*.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. *ICLR*.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 152–162.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. 2019. Unsupervised word discovery with segmental neural language models. *ACL*.

Joohyun Kim and Raymond J Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *ICLR*.

Yoon Kim, Sam Wiseman, and Alexander M Rush. 2018. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ICLR*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi. 2014. Multi-resolution language grounding with weak supervision. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 386–396.

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.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213. Association for Computational Linguistics.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Jeffrey Ling and Alexander Rush. 2017. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 33–42.

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*, pages 1412–1421. Association for Computational Linguistics.

Kathleen McKeown. 1992. *Text generation*. Cambridge University Press.

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

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277.

Dang Tuan Nguyen and Trung Tran. 2018. Structurebased generation system for e2e nlg challenge. *arxiv*.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017b. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 45–53.

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, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2e nlg challenge: Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 463–471.

Guanghui Qin, Jin-Ge Yao, Xuening Wang, Jinpeng Wang, and Chin-Yew Lin. 2018. Learning latent semantic annotations for grounding natural language to structured data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3761–3771.

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

Lena Reed, Shereen Oraby, and Marilyn Walker. 2018. Can neural generators for dialogue learn sentence planning and discourse structuring? In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 284–295.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

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

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Shiv Shankar and Sunita Sarawagi. 2019. Posterior attention models for sequence to sequence learning. *ICLR*.

Xiaoyu Shen, Youssef Oualil, Clayton Greenberg, Mittul Singh, and Dietrich Klakow. 2017. Estimation of gap between current language models and human performance. *Proc. Interspeech 2017*, pages 553–557.

Xiaoyu Shen, Jun Suzuki, Kentaro Inui, Hui Su, Dietrich Klakow, and Satoshi Sekine. 2019a. Select and attend: Towards controllable content selection in text generation. *arXiv preprint arXiv:1909.04453*.

Xiaoyu Shen, Yang Zhao, Hui Su, and Dietrich Klakow. 2019b. Improving latent alignment in text summarization by generalizing the pointer generator. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3753–3764.

Hui Su, Xiaoyu Shen, Wenjie Li, and Dietrich Klakow. 2018. Nexus network: Connecting the preceding and the following in dialogue generation. *arXiv preprint arXiv:1810.00671*.

Hui Su, Xiaoyu Shen, Rongzhi Zhang, Fei Sun, Pengwei Hu, Cheng Niu, and Jie Zhou. 2019. Improving multi-turn dialogue modelling with utterance rewriter. *arXiv preprint arXiv:1906.07004*.

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 *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3674–3683. JMLR. org.

Weiyue Wang, Derui Zhu, Tamer Alkhouli, Zixuan Gan, and Hermann Ney. 2018. Neural hidden Markov model for machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 377–382, Melbourne, Australia. Association for Computational Linguistics.

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

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2018. Learning neural templates for text generation. *EMNLP*.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438.

Ze Yang, wei wu, Jian Yang, Can Xu, and zhoujun li. 2019. Low-resource response generation with template prior. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1886–1897, Hong Kong, China. Association for Computational Linguistics.

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.

Yang Zhao, Xiaoyu Shen, Hajime Senuma, and Akiko Aizawa. 2018. A comprehensive study: Sentence compression with linguistic knowledge-enhanced gated neural network. *Data & Knowledge Engineering*, 117:307–318.