Data-to-text Generation with Variational Sequential Planning

Ratish Puduppully and Yao Fu and Mirella Lapata
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
r.puduppully@sms.ed.ac.uk yao.fu@ed.ac.uk mlap@inf.ed.ac.uk

Abstract

We consider the task of data-to-text generation, which aims to create textual output from non-linguistic input. We focus on generating long-form text, that is, documents with multiple paragraphs, and propose a neural model enhanced with a planning component responsible for organizing high-level information in a coherent and meaningful way. We infer *latent* plans sequentially with a structured variational model, while interleaving the steps of planning and generation. Text is generated by conditioning on previous variational decisions *and* previously generated text. Experiments on two data-to-text benchmarks (RotoWire and MLB) show that our model outperforms strong baselines and is sample-efficient in the face of limited training data (e.g., a few hundred instances).

1 Introduction

Data-to-text generation refers to the task of generating textual output from non-linguistic input such as database tables, spreadsheets, or simulations of physical systems (Reiter and Dale, 1997, 2000; Gatt and Krahmer, 2018). Recent progress in this area (Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017) has been greatly facilitated by the very successful encoder-decoder neural architecture (Sutskever et al., 2014) and the development of large scale datasets. RotoWire (Wiseman et al., 2017) and MLB (Puduppully et al., 2019b) constitute such examples. They both focus on the sports domain, which has historically drawn attention in the generation community (Barzilay and Lapata, 2005; Tanaka-Ishii et al., 1998; Robin, 1994) and consider the problem of generating long target texts from database records.

Figure 1 (reproduced from Puduppully and Lapata, 2021) provides a sample from the MLB dataset, which pairs human written summaries (Table C in Figure 1) with major league baseball game statistics. These are mostly scores (collectively referred to as *box score*) which summarize the performance of teams and players, for example, batters, pitchers, or fielders (Table A in Figure 1) and a *play-by-play* description of the most important events in the game (Table B in Figure 1). Game summaries in MLB are relatively long (540 tokens on average) with multiple paragraphs (15 on average). The complexity of the input and the length of the game summaries pose various challenges to neural models which, despite producing fluent output, are often imprecise, prone to hallucinations, and display poor content selection (Wiseman et al., 2017). Attempts to address these issues have seen the development of special-purpose modules that keep track of salient entities (Iso et al., 2019; Puduppully et al., 2019b), determine which records (see the rows in Tables A and B in Figure 1) should be mentioned in a sentence and in which order (Puduppully et al., 2019a; Narayan et al., 2020), and reconceptualize the input in terms of paragraph plans (Puduppully and Lapata, 2021) to facilitate document-level planning (see Table D in Figure 1).

Specifically, Puduppully and Lapata (2021) advocate the use of *macro plans* for improving the organization of document content and structure. A macro plan is a *sequence* of paragraph plans, and each paragraph plan corresponds to a document paragraph. A macro plan is shown in Table E (Figure 1). Examples of paragraph plans are given in Table D where `<V(entity)>` verbalizes records pertaining to entities and `<V(inning-T/B)>` verbalizes records for the Top/Bottom side of an inning. Verbalizations are sequences of record types followed by their values. Document paragraphs are shown in Table C and have the same color as their corresponding plans in Table E. During training, Puduppully and

Transactions of the Association for Computational Linguistics, vol. 10, pp. 697–715, 2022. https://doi.org/10.1162/tacl_a_00484
Action Editor: Ehud Reiter. Submission batch: 10/2021; Revision batch: 3/2021; Published 6/2022.
© 2022 Association for Computational Linguistics. Distributed under a CC-BY 4.0 license.

(A)

TEAM	Inn1	Inn2	Inn3	Inn4	...	TR	TH	E	...
Orioles	1	0	0	0	...	2	4	0	...
Royals	1	0	0	3	...	9	14	1	...

BATTER	H/V	AB	BR	BH	RBI	TEAM	...
C.Mullins	H	4	2	2	1	Orioles	...
J.Villar	H	4	0	0	0	Orioles	...
W.Merrifield	V	2	3	2	1	Royals	...
R.O'Hearn	V	5	1	3	4	Royals	...
...	...	...	...	...	...	...	

PITCHER	H/V	W	L	IP	PH	PR	ER	BB	K	...
A.Cashner	H	4	13	5.1	9	4	4	3	1	...
B.Keller	V	7	5	8.0	4	2	2	2	4	...
...	...	...	...	...	...	...				

Inn1: runs in innings, TR: team runs, TH: team hits, E: errors, H/V: home/visiting, AB: at-bats, BR: batter runs, BH: batter hits, RBI: runs-batted-in, W: wins, L: losses, IP: innings pitched, PH: hits given, PR: runs given, ER: earned runs, BB: walks, K: strike outs, INN: inning with (T)op/(B)ottom, PL-ID: play id, SCR: score of Royals.

(C)

KANSAS CITY, Mo. – Brad Keller kept up his recent pitching surge with another strong outing. <P> Keller gave up a home run to the first batter of the game – Cedric Mullins – but quickly settled in to pitch eight strong innings in the Kansas City Royals' 9–2 win over the Baltimore Orioles in a matchup of the teams with the worst records in the majors. <P> Keller (7–5) gave up two runs and four hits with two walks and four strikeouts to improve to 3–0 with a 2.16 ERA in his last four starts. <P> Ryan O'Hearn homered among his three hits and drove in four runs, Whit Merrifield scored three runs, and Hunter Dozier and Cam Gallagher also went deep to help the Royals win for the fifth time in six games on their current homestand. <P> With the score tied 1–1 in the fourth, Andrew Cashner (4–13) gave up a sacrifice fly to Merrifield after loading the bases on two walks and a single. Dozier led off the fifth inning with a 423-foot home run to left field to make it 3-1. <P> The Orioles pulled within a run in the sixth when Mullins led off with a double just beyond the reach of Dozier at third, advanced to third on a fly ball and scored on Trey Mancini's sacrifice fly to the wall in right. <P> ...

(B)

BATTER	PITCHER	SCORER	ACTION	TEAM	INN	PL-ID	SCR	...
C.Mullins	B.Keller	—	Home run	Orioles	1-T	1	1	...
H.Dozier	A.Cashner	W.Merrifield	Grounded	Royals	1-B	3	1	...
W.Merrifield	A.Cashner	B.Goodwin	Sac fly	Royals	4-B	5	2	...
H.Dozier	A.Cashner	—	Home run	Royals	5-B	1	3	...
...	...	...	...	...	...	...	...	...

(D)

```
V(Orioles), V(Royals),
V(C.Mullins), V(J.Villar),
V(W.Merrifield), V(R.O'Hearn),
V(A.Cashner), V(B.Keller),
V(H.Dozier), ...,
V(1-T), V(1-B), V(2-T), V(2-B),
V(3-T), V(3-B), ...
```

```
V(Royals) V(Orioles),
V(Orioles) V(C.Mullins), V(Orioles) V(J.Villar),
V(Royals) V(W.Merrifield), V(Royals)
V(R.O'Hearn), V(Orioles) V(A.Cashner), V(Royals)
V(B.Keller), ...,
V(C.Mullins) V(Royals) V(Orioles),
V(J.Villar) V(Royals) V(Orioles), ...
```

(E)

```
V(B.Keller)<P>V(B.Keller) V(C.Mullins) V(Royals) V(Orioles)<P>V(B.Keller)<P>
V(R.O'Hearn) V(W.Merrifield) V(H.Dozier) V(C.Gallagher) <P>V(4-B, 5-B) <P> V(6-T)<P>
```

Figure 1: Example from the MLB dataset reproduced from Puduppully and Lapata (2021) with the authors' permission. Table A is typically referred to as *box score*. It summarizes the data of the game per team and player. Table B reports statistics pertaining to innings or play-by-play scores. Table C contains the game summary. Paragraphs in Table C are separated with <P> delimiters. Table D contains paragraph plans obtained from Tables A and B. Paragraph plans in the first column correspond to a single entity or event. Paragraph plans in the second column describe combinations of entities or events. <V(entity)> verbalizes records pertaining to entities and <V(inning-T/B)> verbalizes records for the Top/Bottom side of an inning. Paragraph plans correspond to paragraphs in Table C. Table E contains the *macro plan* for the document in Table C. A macro plan is a sequence of paragraph plans. Plan-document correspondences are highlighted using the same color.

Lapata (2021) *learn* to predict a macro plan from a pool of paragraph plans, and produce a game summary based on it. Continuing with our example in Figure 1, plan (E) is obtained from paragraph plans (D), to give rise to game summary (C).

The intermediate macro plan renders generation more interpretable (differences in the output can be explained by differences in macro planning). It also makes modeling easier, the input is no longer a complicated table but a sequence of paragraph plans, which in turn allows us to treat data-to-text generation as a sequence-to-sequence learning problem. Nevertheless, decoding to a long document remains challenging for at least two reasons. Firstly, the macro plan may be encoded as a sequence but a very long one (more than 3,000 tokens), which the decoder has to attend to at each time step in order to generate a summary token-by-token. Secondly, the prediction of the macro plan is conditioned solely on the input (i.e., pool of paragraph plans (D) in Figure 1) and does not make use of information present in the summaries. We hypothesize that planning would be more accurate were it to consider information available in the table (and corresponding paragraph plans) and the generated summary, more so because the plans are coarse-grained and there is a one-to-many relationship between a paragraph plan and its realization. For example, we can see that the plan for <V(B.Keller)> results in two very different realizations in the summary in Figure 1 (see first and third paragraph).

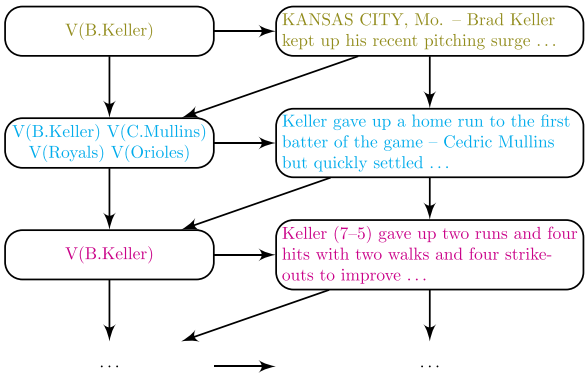


Figure 2: Conceptual sequence of interleaved planning and generation steps. The paragraph plan and its corresponding paragraph have the same color.

In this work, we present a model which interleaves macro planning with text generation (see Figure 2 for a sketch of the approach). We begin by selecting a plan from a pool of paragraph plans (see Table D in Figure 1), and generate the first paragraph by conditioning on it. We select the next plan by conditioning on the previous plan *and* the previously generated paragraph. We generate the next paragraph by conditioning on the currently selected plan, the previously predicted plan, and generated paragraph. We repeat this process until the final paragraph plan is predicted. We model the selection of paragraph plans as a *sequential latent variable process*, which we argue is intuitive since content planing is inherently latent. Contrary to Puduppully and Lapata (2021), we do not a priori decide on a *global* macro plan. Rather, our planning process is *incremental* and as a result less rigid. Planning is informed by generation and vice versa, which we argue should be mutually beneficial (they are conditioned on each other).

During training, the sequential latent model can better leverage the summary to render paragraph plan selection more accurate and take previous decisions into account. We hypothesize that the interdependence between planning and generation allows the model to cope with diversity. In general, there can be many ways in which the input table can be described in the output summary, that is, different plans give rise to equally valid game summaries. The summary in Figure 1 (Table C) focuses on the performance of *Brad Keller*, who is a high scoring pitcher (first three paragraphs). An equally plausible summary might have discussed a high scoring batter first (e.g., *Ryan O'Hearn*). Also notice that the summary describes innings in chronological order. However, another ordering might have been equally plausible, for example, describing innings where the highest runs are scored first or innings that are important in flipping the outcome of the match. In the face of such diversity, there may never be enough data to learn an accurate global plan. It is easier to select a paragraph plan from the pool once some of the summary is known, and different plans can be predicted for the same input. In addition, the proposed model is end-to-end differentiable and gradients for summary prediction also inform plan prediction.

Our contributions can be summarized as follows: (1) We decompose data-to-text generation into sequential plan selection and paragraph generation. The two processes are interleaved and generation proceeds incrementally. We look at what has been already generated, make a plan on what to discuss next, realize the plan, and repeat; (2) in contrast to previous models (Puduppully et al., 2019a; Puduppully and Lapata, 2021), where content plans are monolithic and determined in advance, our approach is more flexible, it simplifies modeling (we do not need to learn alignments between paragraph plans and summary paragraphs), and leads to sample efficiency in low resource scenarios; (3) our approach scales better for tasks involving generation of long multi-paragraph texts, as we do not need to specify the document plan in advance; and (4) experimental results on English and German RotoWire (Wiseman et al., 2017; Hayashi et al., 2019), and MLB (Puduppully et al., 2019b) show that our model is well-suited to long-form generation and generates more factual, coherent, and less repetitive output compared to strong baselines.

We share our code and models in the hope of being useful for other tasks (e.g., story generation, summarization).[1]

2 Related Work

A long tradition in natural language generation views content planning as a central component to identifying important content and structuring it appropriately (Reiter and Dale, 2000). Earlier work has primarily made use of hand-crafted content plans with some exceptions that pioneered

[1] https://github.com/ratishsp/data2text-seq-plan-py.

learning-based approaches. For instance, Duboue and McKeown (2001) learn ordering constraints on the content plan, while Kan and McKeown (2002) learn content planners from semantically annotated corpora, and Konstas and Lapata (2013) predict content plans using grammar rules whose probabilities are learned from training data.

More recently, there have been attempts to equip encoder-decoder models (Bahdanau et al., 2015; Wiseman et al., 2017) with content planning modules. Puduppully et al. (2019a) introduce *micro planning*: They first learn a content plan corresponding to a sequence of records, and then generate a summary conditioned on it. Narayan et al. (2020) treat content selection as a task similar to extractive summarization. Specifically, they post-process Puduppully et al.'s (2019a) micro-plans with special tokens identifying the beginning and end of a sentence. Their model first extracts sentence plans and then verbalizes them one-by-one by conditioning on previously generated sentences. Moryossef et al. (2019b,a) propose a two-stage approach that first predicts a document plan and then generates text based on it. The input to their model is a set of RDF ⟨Subject, Object, Predicate⟩ tuples. Their document plan is a sequence of sentence plans where each sentence plan contains a subset of tuples in a specific order. Text generation is implemented using a sequence-to-sequence model enhanced with attention and copy mechanisms (Bahdanau et al., 2015). They evaluate their model on the WebNLG dataset (Gardent et al., 2017), where the outputs are relatively short (24 tokens on average).

Our approach is closest to Puduppully and Lapata (2021), who advocate *macro planning* as a way of organizing high-level document content. Their model operates over paragraph plans that are verbalizations of the tabular input and predicts a document plan as a sequence of paragraph plans. In a second stage, the summary is generated from the predicted plan making use of attention enriched with a copy mechanism. We follow their formulation of content planning as paragraph plan prediction. Our model thus operates over larger content units compared to related work (Puduppully et al., 2019a; Narayan et al., 2020) and performs the tasks of micro- and macro-planning in one go. In contrast to Puduppully and Lapata (2021), we predict paragraph plans and their corresponding paragraphs *jointly* in an incremental fashion. Our approach is reminiscent of psycholinguistic models of speech production (Levelt, 1993; Taylor and Taylor, 1990; Guhe, 2020), which postulate that different levels of processing (or modules) are responsible for language generation; these modules are incremental, each producing output as soon as the information it needs is available and the output is processed immediately by the next module.

We assume that plans form a sequence of paragraphs, which we treat as a latent variable and learn with a structured variational model. Sequential latent variables (Chung et al., 2015; Fraccaro et al., 2016; Goyal et al., 2017) have previously found application in modeling attention in sequence-to-sequence networks (Shankar and Sarawagi, 2019), document summarization (Li et al., 2017), controllable generation (Li and Rush, 2020; Fu et al., 2020), and knowledge-grounded dialogue (Kim et al., 2020). In the context of data-to-text generation, latent variable models have been primarily used to inject diversity in the output. Shao et al. (2019) generate a sequence of groups (essentially a subset of the input) which specifies the content of the sentence to be generated. Their plans receive no feedback from text generation, they cover a small set of input items, and give rise to relatively short documents (approximately 100 tokens long). Ye et al. (2020) use latent variables to disentangle the content from the structure (operationalized as templates) of the output text. Their approach generates diverse output output by sampling from the template-specific sample space. They apply their model to single-sentence generation tasks (Lebret et al., 2016; Reed et al., 2018).

3 Model

Following Puduppully and Lapata (2021), we assume that at training time our model has access to a pool of paragraph plans $\mathcal{E}$ (see Table D in Figure 1), which represent a clustering of records. We explain how paragraph plans are created from tabular input in Section 4. Given $\mathcal{E}$, we aim to generate a sequence of paragraphs $y = [y^1, \ldots, y^T]$ that describe the data following a sequence of chosen plans $z = [z^1, \ldots, z^T]$. Let y^t denote a paragraph, which can consist of multiple sentences, and T the count of paragraphs in a summary. With a slight abuse of notation, superscripts denote indices rather than exponentiation. So, y_i^t refers to the i-th word in the t-th paragraph.

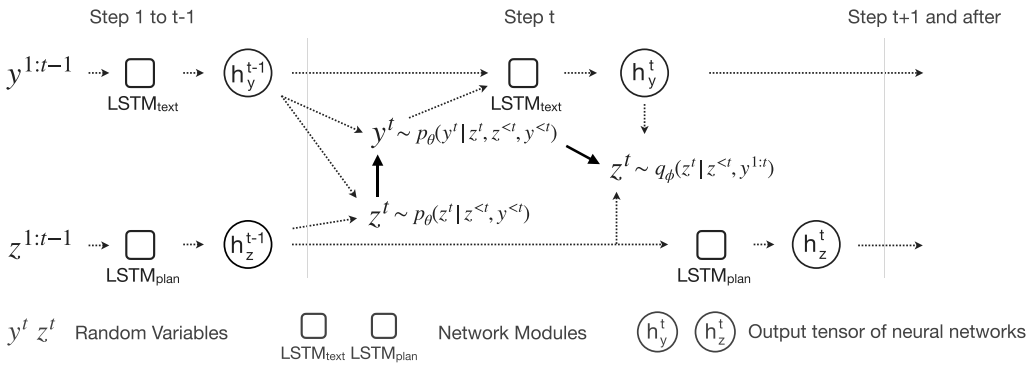


Figure 3: Model workflow. Solid arrows show dependencies between random variables. Dashed arrows show the computation graph whose backbone consists of an $\text{LSTM}_{\text{text}}$ and an $\text{LSTM}_{\text{plan}}$. Note that the variational model and the generative model are tied closely with the shared LSTM. To generate long documents, the model observes what has been already generated, decides on a plan about what to discuss next, uses this plan to guide next stage generation, and repeats until the end.

A plan $z = [z^1, \ldots, z^T]$ is a list of discrete variables where $z^t = j$ means that we choose the j-th item from pool $\mathcal{E}$ of candidate plans to guide the generation of paragraph y^t.

Generation with Latent Plans The core technique of our model is learning the sequence of latent plans that guides long document generation. We consider a conditional generation setting where the input $\mathcal{E}$ is a set of paragraph plans and the output $y_{1:T}$ are textual paragraphs verbalizing the selected sequence $z = z_{1:T}$. Our goal is to induce variables z that indicate which paragraphs are being talked about and in which order. Similar to previous work (Li and Rush, 2020; Fu et al., 2020), we model this process as a conditional generative model that produces both y and z and factorizes as:

$$p_\theta(y, z|\mathcal{E}) = \prod_t p_\theta(z^t|y^{<t}, z^{<t}, \mathcal{E}) p_\theta(y^t|y^{<t}, z^{1:t}, \mathcal{E}) \quad (1)$$

where θ denotes the model parameters and $< t$ all indices smaller than t. We believe this formulation is intuitive, simulating incremental document generation: Inspect $y^{<t}$ (what has been already said), make a plan z^t about what to say next, realize this plan by generating a new paragraph y^t, and so on.

Inference Model We are interested in the posterior distribution $p_\theta(z|y, \mathcal{E})$, that is, the probability over plan sequences z for a known text y and input $\mathcal{E}$. This distribution is intractable to compute in general as the summation of all possible plan sequences z is exponentially complex:

$$p_\theta(z|y, \mathcal{E}) = \frac{p_\theta(y, z|\mathcal{E})}{\sum_z p_\theta(y, z|\mathcal{E})} \quad (2)$$

We use variational inference (Kingma and Welling, 2014; Rezende et al., 2014) to approximate the posterior with a parametrized distribution $q_\phi(z|y, \mathcal{E})$ from which we sample values of z that are likely to produce y (see Doersch 2016 for a tutorial on this topic). Specifically, we employ an autoregressive inference model factorized as:

$$q_\phi(z|y, \mathcal{E}) = \prod_t q_\phi(z^t|y^{1:t}, z^{<t}, \mathcal{E}) \quad (3)$$

Note that a major difference between q above and p in Equation (1) is that p generates y_t under the guidance of z_t (conceptually $z^t \rightarrow y^t$) while q infers z_t given *observed* y_t (conceptually $y^t \rightarrow z^t$).

Neural Parametrization At step t, we start with the encoding of previous paragraphs $y^{<t}$ and plans $z^{<t}$ (see Figure 3 left). Following Yang et al. (2016), we use a Bi-directional LSTM (BiLSTM) with a self-attention layer to encode paragraph y^t as a vector r_y^t at step t:

$$r_y^t = \text{Attn}(q_{\text{text}}, \text{BiLSTM}(y^t)) \quad (4)$$

where q_{text} is a trainable query vector, which is randomly initialized and learned along with the rest of the parameters. $\text{Attn}(\cdot)$ returns the attention probability and output vector over BiLSTM representation y^t with query vector q_{text}.[2] Our

[2]In our notation neural network layers are described by math functions.

model uses the output vector. Next, we encode $r_y^{<t}$ with $\text{LSTM}_{\text{text}}$ as:

$$h_y^{<t} = \text{LSTM}_{\text{text}}(r_y^{<t}) \tag{5}$$

We encode candidate plans in pool $\mathcal{E} = [e_1, .., e_N]$ with a BiLSTM, similar to the paragraph encoding shown in Equation (3), and select one of them at each step. Let r_z^t denote a plan embedding at step t. We encode $r_z^{<t}$ using $\text{LSTM}_{\text{plan}}$ as:

$$h_z^{<t} = \text{LSTM}_{\text{plan}}(r_z^{<t}) \tag{6}$$

The currently selected plan is parametrized as:

$$h^{t-1} = \text{FF}_{\text{plan}}([h_z^{t-1}; h_y^{t-1}]) \tag{7}$$

$$p_\theta(z^t|y^{<t}, z^{<t}, \mathcal{E}) = \text{Attn}(h^{t-1}, \mathcal{E}) \tag{8}$$

where h^{t-1} summarizes information in $y^{<t}$ and $z^{<t}$, $\text{FF}_{\text{plan}}(\cdot)$ denotes a feed-forward layer, and $\text{Attn}(\cdot)$ returns the attention probability (and output vector) of choosing a plan from $\mathcal{E}$ with current state h^{t-1}. Here, we use the attention distribution, which serves essentially as a copy mechanism. Then, a plan z^t is sampled from p (we use greedy decoding in our experiments), and its representation r_z^t is used to update $\text{LSTM}_{\text{plan}}$ (Figure 3 right):

$$h_z^t = \text{LSTM}_{\text{plan}}(r_z^t, h_z^{t-1}) \tag{9}$$

We guide the generation of y^t with current plan z^t and decode each word y_i^t sequentially with an LSTM_{gen} decoder which makes use of beam search. Let s_i denote the i-th decoder state (initialized with the plan encoding). We update it as:

$$s_i = \text{LSTM}_{\text{gen}}(y_{i-1}^t, s_{i-1}, h_y^{t-1}) \tag{10}$$

Note that we feed h_y^{t-1}, representing the context of previous paragraphs, as additional input similar to Serban et al. (2017).

Let $r_{z,1}^t, \ldots, r_{z,l}^t$ denote the encoding of tokens of the current plan where $r_{z,k}^t$ is the output of the BiLSTM plan encoder and l the length of the chosen plan. We generate the next word as:

$$c_i = \text{Attn}(s_i, [r_{z,1}^t, \ldots, r_{z,l}^t]) \tag{11}$$

$$p_\theta(y_i^t|z^t, y_{1:i-1}^t, y^{<t}, z^{<t}, \mathcal{E}) = \text{softmax}(\text{FF}_{\text{gen}}([s_i; c_i])) \tag{12}$$

where c denotes the context vector. In Equation 11, we use the output vector from $\text{Attn}(\cdot)$. $\text{FF}_{\text{gen}}(\cdot)$ represents a feed-forward layer. In addition, we equip the decoder with copy attention (See et al., 2017) to enable copying tokens from z^t. As part of this, we learn a probability for copy based on s_i (Gehrmann et al., 2018). Once paragraph y^t has been generated, we obtain its encoding r_y^t with Equation (3), and update $\text{LSTM}_{\text{text}}$ (Figure 3 middle):

$$h_y^t = \text{LSTM}_{\text{text}}(r_y^t, h_y^{t-1}) \tag{13}$$

We parametrize the variational model so that it shares the LSTMs for encoding y and $\mathcal{E}$ with the generative model:

$$\tilde{h}^t = \text{FF}_{\text{v}}([h_z^{t-1}; h_y^t]) \tag{14}$$

$$q_\phi(z^t|y^{1:t}, z^{<t}, \mathcal{E}) = \text{Attn}(\tilde{h}^t, \mathcal{E}) \tag{15}$$

where $\text{FF}_{\text{v}}(\cdot)$ represents a feed-forward layer. Note that Equation (14) differs from Equation (7) in that it uses the updated h_y^t instead of the previous h_y^{t-1} because now y^t is observed. The variational distribution is again parametrized by the attention probability. Essentially, p and q are strongly tied to each other with the shared LSTM encoders.

Although we primarily focus on the inference, and how the latent plan can improve the generation of long documents, we note that the model sketched above could be parametrized differently, for example, by replacing the encoder and decoder with pretrained language models like BART (Lewis et al., 2020). However, we leave this to future work.

Training We optimize the standard evidence lower bound (ELBO) loss:

$$\begin{aligned}\mathcal{L}_0 &= \log p_\theta(y|\mathcal{E}) - D(q_\phi(z|y,\mathcal{E})\|p_\theta(z|y,\mathcal{E})) \\ &= \mathsf{E}_{q_\varphi(z|y,\mathcal{E})}[\log p_\theta(y,z|\mathcal{E}) - \log q_\phi(z|y,\mathcal{E})] \\ &= \mathsf{E}_{q_\varphi(z|y,\mathcal{E})}\Big[\sum_t \Big\{ \log p_\theta(y^t|z^t, y^{<t}, z^{<t}, \mathcal{E}) + \\ &\quad \log\Big(\frac{p_\theta(z^t|y^{<t}, z^{<t}, \mathcal{E})}{q_\phi(z^t|y^{1:t}, z^{<t}, \mathcal{E})}\Big)\Big\}\Big]\end{aligned} \tag{16}$$

where $\log p_\theta(y|\mathcal{E})$ is the log-evidence from the data, and $D(q_\phi(z|y,\mathcal{E})\|p_\theta(z|y,\mathcal{E}))$ is the Kullback-Leibler divergence between q_ϕ and the true posterior p_θ. The objective eventually decomposes to a summation of the reconstruction probability $p_\theta(y^t|\cdot)$ and the ratio between $p_\theta(z^t|\cdot)$ and $q_\phi(z^t|\cdot)$ at each step.

Advantageously, we can exploit oracle plans (see Table E in Figure 1 and the description in Section 4 for how these were created) to obtain

weak labels z^*, which we use as distant supervision to the inference model:

$$\mathcal{L}_1 = \mathsf{E}_{z^*}[\log q_\phi(z^*|y, \mathcal{E})] \quad (17)$$

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_1 \quad (18)$$

Such distant supervision is essential for stabilizing training (it would be extremely challenging to optimize the model in a fully unsupervised way) and for mitigating posterior collapse. We use Gumbel-Softmax (Maddison et al., 2017; Jang et al., 2017) for differentiable sampling (reparameterization) from q. The model is trained with scheduled sampling (Bengio et al., 2015), and follows the curriculum learning strategy using linear decay scheduling. During earlier stages of training predicted plans are less accurate, and we thus sample from oracle plans at a rate which decays linearly with training:

$$\epsilon_k = \max(0, 1 - c * k) \quad (19)$$

where c is the slope of the decay at training step k.

4 Experimental Setup

Data We performed experiments on the RotoWire (Wiseman et al., 2017) and MLB (Puduppully et al., 2019b) datasets and the German RotoWire provided as part of the WNGT 2020 DGT shared task on ''Document-Level Generation and Translation'' (Hayashi et al., 2019). Statistics on these datasets are shown in Table 1. We used the official train/dev/test splits: 3,398/727/728 for RotoWire, 22,821/1,739/1,744 for MLB, and 242/240/241 for German RotoWire. The latter is considerably smaller than its English counterpart and MLB, and serves to illustrate our model's sample efficiency when training data is scarce.

All three datasets were preprocessed following the method of Puduppully and Lapata (2021). A paragraph plan for an entity is constructed by verbalizing its records in a fixed sequence of record type followed by its value. For example, pitcher *B.Keller* from Figure 1 would be verbalized as `<PLAYER> B.Keller <H/V> V <W> 7 <L> 5 <IP> 8 <PH> 4 ...`. We denote this using the shorthand `<V(B.Keller)>`. The paragraph plan for an event is the verbalization of the players in the event followed by the verbalization of play-by-plays. Candidate paragraph plans $\mathcal{E}$ are obtained by enumerating entities and events and their combinations (see Table D in Figure 1). Oracle macro plans are obtained by matching the mentions of entities and events in the gold summary with the input table. We make use of these oracle macro plans during training. The versions of MLB and RotoWire released by Puduppully and Lapata (2021) contain paragraph delimiters for gold summaries; we preprocessed the German RotoWire in a similar fashion.

	RW	MLB	DE-RW
Vocab Size	11.3K	38.9K	9.5K
# Tokens	1.5M	14.3M	234K
# Instances	4.9K	26.3K	723
# Paragraphs	399K	47.7K	7K
# Record Types	39	53	39
Avg Records	628	565	628
Avg Length	337.1	542.1	323.6
Avg Plan length	10.6	15.1	9.5

Table 1: Dataset statistics for RotoWire (RW), MLB, and German RotoWire (DE-RW). Vocabulary size, number of tokens, number of instances (i.e., table-summary pairs), number of paragraphs, number of record types, average number of records, average summary length, average macro plan length measured in terms of number of paragraphs.

Table 1 also shows the average length of the macro plan in terms of the number of paragraph plans it contains. This is 10.6 for RotoWire, 15.1 for MLB, and 9.5 for German RotoWire.

Training Configuration We train our model with the AdaGrad optimizer (Duchi et al., 2011) and tune parameters on the development set. We use a learning rate of 0.15. We learn a joint subword vocabulary (Sennrich et al., 2016) for paragraph plans and summaries with 6K merge operations for RotoWire, 16K merge operations for MLB, and 2K merge operations for German RotoWire. The model is implemented on a fork of OpenNMT-py (Klein et al., 2017). For efficiency, we batch using summaries instead of individual paragraphs. Batch sizes for MLB, RotoWire, and German-RotoWire are 8, 5, and 1, respectively. We set λ to 2 in Equation (18). In Equation (19), c is 1/100000 for MLB, 1/50000 for RotoWire, and 1/30000

for German-RotoWire. We set the temperature of Gumbel-Softmax to 0.1.

During inference in MLB, similar to Puduppully and Lapata (2021), we block the repetition of paragraph plan bigrams (i.e., we disallow the repetition of (z^t, z^{t+1})) and select the paragraph plan with the next higher probability in Equation (8). In addition, we block consecutive repetitions, and more than two repetitions of a unigram. During training we observed high variance in the length of paragraphs y^t, since the same plan can result in a shorter or longer paragraph. For example, <V(B.Keller)> corresponds to two paragraphs (first and third paragraph) with different lengths in Figure 1. We found that this encourages the model to be conservative and generate relatively short output. We control the paragraph length (Fan et al., 2018) by creating discrete bins, each containing approximately an equal number of paragraphs. During training, we prepend the embedding of the bin to the current plan r_z^t (see Equation (11)). For inference, bins are tuned on the validation set.

We run inference for 15 paragraphs on RotoWire and German RotoWire, and for 20 paragraphs on MLB; we stop when the model predicts the end of paragraph plan token *EOP*. Unlike previous work (Wiseman et al., 2017; Puduppully et al., 2019a,b, *inter alia*), we do not make use of truncated Back Propagation Through Time (Williams and Peng, 1990), as we incrementally generate paragraphs instead of long documents.

System Comparisons We compared our model with: (1) a **Templ**ate-based generator which creates a document consisting of template sentences. We used Wiseman et al.'s (2017) system on RotoWire and Puduppully et al.'s (2019b) system on MLB. They are both similar in that they describe team scores followed by player specific statistics and a concluding statement. In MLB, the template additionally describes play-by-play details. We also created a template system for German RotoWire following a similar approach. (2) **ED+CC**, the best performing model of Wiseman et al. (2017). It consists of an encoder-decoder model equipped with attention and copy mechanisms. (3) **NCP+CC**, the micro planning model of Puduppully et al. (2019a). It first creates a content plan by pointing to input records through the use of Pointer Networks (Vinyals et al., 2015). The content plan is then encoded with a BiLSTM and decoded using another LSTM with an attention and copy mechanism. (4) **ENT**, the entity model of Puduppully et al. (2019b). It creates entity-specific representations which are updated dynamically. At each time step during decoding, their model makes use of hierarchical attention by attending over entity representations and the records corresponding to these. (5) **MACRO**, the two-stage planning model of Puduppully and Lapata (2021), which first makes use of Pointer Networks (Vinyals et al., 2015) to predict a macro plan from a set of candidate paragraph plans. The second stage takes the predicted plan as input and generates the game summary with a sequence-to-sequence model enhanced with attention and copy mechanisms. In addition, we compare with a variant of Macro enhanced with length control (+Bin).

5 Results

Our experiments were designed to explore how the proposed model compares to related approaches which are either not enhanced with planning modules or non-incremental. We also investigated the sample efficiency of these models and the quality of the predicted plans when these are available. The majority of our results focus on automatic evaluation metrics. We also follow previous work (Wiseman et al., 2017; Puduppully et al., 2019a,b; Puduppully and Lapata, 2021) in eliciting judgments to evaluate system output.

5.1 Automatic Evaluation

We evaluate model output using BLEU (Papineni et al., 2002) with the gold summary as a reference. We also report model performance against the Information Extraction (IE) metrics of Wiseman et al. (2017) which are defined based on the output of an IE model which extracts entity (team and player names) and value (numbers) pairs from the summary and predicts the type of relation between them.

Let $\hat{y}$ be the gold summary and y be the model output. *Relation Generation* (RG) measures the precision and count of relations obtained from y that are found in the input table. *Content Selection* (CS) measures the precision, recall, and F-measure of relations extracted from y also found in $\hat{y}$. And *Content Ordering* (CO) measures the

complement of the Damerau-Levenshtein distance between relations extracted from y and $\hat{y}$. Higher values are better for RG Precision, CS F-measure, CO, and BLEU. We reuse the IE model from Puduppully et al. (2019a) for RotoWire, Puduppully and Lapata (2021) for MLB, and Hayashi et al. (2019) for German RotoWire. Our computation of IE metrics for all systems includes duplicate records (Puduppully and Lapata, 2021).

In addition to IE-based metrics, we report the number of errors made by systems according to Number (incorrect number in digits, number spelled in words, etc.), Name (incorrect names of teams, players, days of week, etc.), and Word (errors in usage of words) following the classification of Thomson and Reiter (2020). We detect such errors automatically using the system of Kasner et al. (2021), which scored best against gold standard human annotations of the same type (Thomson and Reiter, 2021). We only report these metrics for English RotoWire, since error annotations (for automatic metric learning) are not available for other datasets. Moreover, with regard to Word errors, we only report errors for incorrect usage of the word *double-double*.[3] We found such errors to be detected reliably, in contrast to Word errors as a whole for which the precision of the system of Kasner et al. (2021) is ∼50%. Lower values are better for the Number, Name, and double-double errors. We note metrics such as RG precision, Number, Name, and double-double errors *directly* compute the accuracy of the generation model. Metrics such as CS, CO, and BLEU measure how similar model output is against a reference summary. Thus, CS, CO, and BLEU measure generation accuracy *indirectly* under the assumption that gold summaries are accurate.

MLB Dataset Table 2 summarizes our results on MLB. Our sequential planning model (SeqPlan) has the highest RG P among neural models and performs best in terms of CS F, CO, and BLEU. The variant of Macro with length control (+Bin) performs comparably or worse than Macro.

To examine the importance of latent sequential planning, we also present a variant of our model that uniformly samples a plan from the pool $\mathcal{E}$ instead of Equation (8) (see row w(ith) Uniform in Table 2). This version obtains lower values compared to SeqPlan across all metrics underscoring the importance of sequential planning. We also present two variants of SeqPlan: (a) one that makes use of oracle (instead of predicted) plans during training to generate y^t; essentially, it replaces z^t with z^* in Equation (12) (row w(ith) Oracle in Table 2); and (b) a two-stage model that trains the planner (Equation (15)) and generator (Equation (12)) separately (row 2-stage in Table 2)—in this case, we use greedy decoding to sample z^t from Equation (15) instead of Gumbel-Softmax and replace z^t with z^* in Equation (12). Both variants are comparable to SeqPlan in terms of RG P but worse in terms of CS F, CO, and BLEU.

Furthermore, we evaluate the accuracy of the inferred plans by comparing them against oracle plans, using the CS and CO metrics (computed over the entities and events in the plan).[4] Table 4 shows that SeqPlan achieves higher CS F and CO scores than Macro. Again, this indicates that planning is beneficial, particularly when taking the table and the generated summary into account.

MLB	RG		CS			CO	BLEU
	#	P%	P%	R%	F%	DLD%	
Templ	62.3	99.9	21.6	55.2	31.0	11.0	4.12
ED+CC	**32.5**	91.3	27.8	40.6	33.0	17.1	9.68
NCP+CC	19.6	81.3	**44.5**	44.1	44.3	21.9	9.68
ENT	23.8	81.1	40.9	49.5	44.8	20.7	11.50
Macro	30.8	94.4	40.8	**54.9**	46.8	21.8	12.62
+Bin	31.2	93.7	38.3	52.4	44.2	21.6	12.32
SeqPlan	28.9	**95.9**	43.3	53.5	**47.8**	**22.7**	**14.29**
w Uniform	18.5	90.9	36.5	30.6	33.3	14.5	10.30
w Oracle	27.6	95.9	42.5	50.4	46.1	22.0	13.13
2-Stage	28.6	95.9	41.4	50.8	45.6	21.3	13.96

Table 2: MLB results (test set); relation generation (RG) count (#) and precision (P%), content selection (CS) precision (P%), recall (R%), and F-measure (F%), content ordering (CO) as complement of normalized Damerau-Levenshtein distance (DLD%), and BLEU. **Highest** and second highest generation models are highlighted.

[3] A double-double occurs when a player scores 10 points or more in two record types: points, rebounds, assists, steals, and blocked shots.

[4] To compute the accuracy of macro plans, entities and events from the model's plan need to be compared against entities and events in the oracle macro plan. Puduppully and Lapata (2021) obtained the entities and events for the oracle macro plan by extracting these from reference summaries. We noted that this includes coreferent or repeat mentions of entities and events within a paragraph. We instead extract entities and events directly from the oracle macro plan.

RW	RG		CS			CO	BLEU
	#	P%	P%	R%	F%	DLD%	
Templ	54.3	99.9	27.1	57.7	36.9	13.1	8.46
WS-2017	34.1	75.1	20.3	36.3	26.1	12.4	14.19
ED+CC	35.9	82.6	19.8	33.8	24.9	12.0	14.99
NCP+CC	40.8	87.6	28.0	51.1	36.2	15.8	**16.50**
ENT	32.7	91.7	**34.7**	48.5	40.5	16.6	16.12
RBF-2020	44.9	89.5	23.9	47.0	31.7	14.3	17.16
Macro	42.1	**97.6**	34.1	57.8	**42.9**	**17.7**	15.46
+Bin	**61.0**	97.2	26.8	**66.1**	38.2	15.8	**16.48**
SeqPlan	46.7	**97.6**	30.6	57.4	39.9	16.7	16.26
w Uniform	22.0	80.2	18.2	19.6	18.9	6.0	8.61
w Oracle	50.4	97.2	29.0	59.1	38.9	16.8	16.32
2-stage	53.4	97.5	28.5	61.3	38.9	16.1	16.61

DE-RW	RG		CS			CO	BLEU
	#	P%	P%	R%	F%	DLD%	
Templ	54.4	99.9	17.2	63.0	27.1	11.6	7.32
ED+CC	24.8	59.3	6.7	18.8	9.9	6.8	5.09
NCP+CC	17.7	52.5	11.3	25.7	15.7	9.6	7.29
ENT	17.4	64.7	13.3	24.0	17.1	9.8	6.52
RBF-2020	0.2	4.0	1.1	0.4	0.6	0.3	2.29
Macro	**30.2**	49.7	5.1	21.0	8.3	6.1	5.15
+Bin	20.4	55.0	7.9	20.0	11.3	8.1	6.18
SeqPlan	13.8	**91.8**	**38.0**	**38.4**	**38.2**	**21.2**	**8.65**

Table 3: Evaluation on RotoWire (RW) and German RotoWire (DE-RW) test sets; relation generation (RG) count (#) and precision (P%), content selection (CS) precision (P%), recall (R%), and F-measure (F%), content ordering (CO) as complement of normalized Damerau-Levenshtein distance (DLD%), and BLEU. **Highest** and second highest generation models are highlighted.

English and German RotoWire Results on RotoWire are presented in Table 3 (top). In addition to Templ, ED+CC, NCP+CC, and ENT, we compare with the models of Wiseman et al. (2017) (WS-2017) and Rebuffel et al. (2020) (RBF-2020). WS-2017 is the best performing model of Wiseman et al. (2017). Note that ED+CC is an improved re-implementation of WS-2017. RBF-2020 represents the current state-of-the-art on RotoWire, and is composed of a Transformer encoder-decoder architecture (Vaswani et al., 2017) with hierarchical attention on entities and their records. The models of Saleh et al. (2019), Iso et al. (2019), and Gong et al. (2019) are not comparable as they make use of information additional to the table such as previous/next games or the author of the game summary. The model of Narayan et al. (2020) is also not comparable as it relies on a pretrained language model (Rothe et al., 2020) to generate the summary sentences.

Table 3 (bottom) shows our results on German RotoWire. We compare against NCP+CC's entry in the WNGT 2019 shared task[5] (Hayashi et al., 2019), and our implementation of Templ, ED+CC, ENT, Macro, and RBF-2020. Saleh et al. (2019) are not comparable as they pretrain on 32M parallel and 420M monolingual data. Likewise, Puduppully et al. (2019c) make use of a jointly trained multilingual model by combining RotoWire with German RotoWire.

We find that SeqPlan achieves highest RG P among neural models, and performs on par with Macro (it obtains higher BLEU but lower CS F and CO scores). The +Bin variant of Macro performs better on BLEU but worse on other metrics. As in Table 2, w Uniform struggles across metrics corroborating our hypothesis that latent sequential planning improves generation performance. The other two variants (w Oracle and 2-Stage) are worse than SeqPlan in RG P and CS F, comparable in CO, and slightly higher in terms of BLEU.

On German, our model is best across metrics, achieving an RG P of 91.8% which is higher by 42% (absolute) compared to Macro. In fact, the RG P of SeqPlan is superior to Saleh et al. (2019), whose model is pretrained with additional data and is considered state of the art (Hayashi et al., 2019). RG# is lower mainly because of a bug in the German IE that is excludes number records. RG# for NCP+CC and Macro is too high because the summaries contain considerable repetition. The same record will repeat at least once with NCP+CC and three times with Macro, whereas only 7% of the records are repeated with SeqPlan.

Table 4 evaluates the quality of the plans inferred by our model on the RotoWire dataset. As can be seen, SeqPlan is slightly worse than Macro in terms of CS F and CO. We believe this is because summaries in RotoWire are somewhat formulaic, with a plan similar to Templ: an opening statement is followed by a description of the top scoring players, and a conclusion describing the next match. Such plans can be learned well by Macro without access to the summary. MLB texts show much more diversity in terms of length, and

[5]We thank Hiroaki Hayashi for providing us with the output of the NCP+CC system.

	Datasets	CS P%	CS R%	CS F%	CO DLD%
MLB	Macro	73.6	45.9	56.5	27.0
	SeqPlan	74.4	51.1	60.6	27.1
RW	Macro	81.5	62.7	70.9	36.3
	SeqPlan	79.1	61.6	69.3	35.5
DE-RW	Macro	86.8	34.2	49.0	30.1
	SeqPlan	73.1	60.8	66.4	31.0

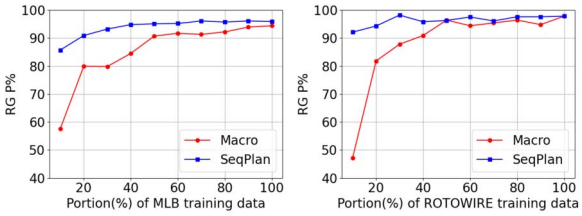
Table 4: Evaluation of macro planning stage (test set); content selection (CS) precision (P%), recall (R%), and F-measure (F%), content ordering (CO) as complement of normalized Damerau-Levenshtein distance (DLD%).

	Number	Name	double-double
Templ	0.08*	3.05*	0.00*
WS-2017	13.01*	9.66*	0.36*
ED+CC	8.11*	8.29*	0.31*
NCP+CC	7.89*	7.76*	0.14
ENT	5.89*	7.24*	0.15
RBF-2020	6.20*	8.39*	0.41*
Macro	2.57	4.60*	0.18
SeqPlan	2.70	6.56	0.20

Table 5: Number, Name, and double-double (Word) errors per example. Systems significantly different from SeqPlan are marked with an asterisk * (using a one-way ANOVA with posthoc Tukey HSD tests; $p \leq 0.05$).

the sequencing of entities and events. The learning problem is also more challenging, supported by the fact that the template system does not do very well in this domain (i.e., it is worse in BLEU, CS F, and CO compared to ROTOWIRE). In German ROTOWIRE, SeqPlan plans achieve higher CS F and CO than Macro.

Table 5 reports complementary automatic metrics on English ROTOWIRE aiming to assess the factuality of generated output. We find that Templ has the least Number, Name, and double-double errors. This is expected as it simply reproduces facts from the table. SeqPlan and Macro have similar Number errors, and both are significantly better than other neural models. SeqPlan has significantly more Name errors than Macro, and significantly fewer than other neural models. Inspection of Name errors revealed that these are mostly due to incorrect information about next games. Such information is not part of the input and models are prone to hallucinate. SeqPlan fares worse as it attempts to discuss next games for both teams while Macro focuses on one team only. In terms of double-double errors, SeqPlan is comparable to Macro, ENT, and NCP+CC, and significantly better than WS-2017, ED+CC, and RBF-2020.



(a) (b)

Figure 4: Sample efficiency for (a) MLB and (b) ROTOWIRE datasets. SeqPlan and Macro are trained on different portions (%) of the training dataset and performance is measured with RG P%.

5.2 Sample Efficiency

We also evaluated whether SeqPlan is more sample-efficient in comparison to Macro, by examining how RG P varies with (training) data size. As shown in Figure 4, the difference between SeqPlan and Macro is more pronounced when relatively little data is available. For example, with 10% of training data, RG P for SeqPlan on MLB is 85.7% and 92.1% on ROTOWIRE. In contrast, Macro obtains 57.5% on MLB and 47.1% on ROTOWIRE. As more training data becomes available, the difference in RG P decreases. The slope of increase in RG P for Macro is higher for ROTOWIRE than MLB. We hypothesize that this is because MLB has longer summaries with more paragraphs, and it is thus more difficult for Macro to learn alignments between paragraph plans and text paragraphs in the game summary.

5.3 Human Evaluation

We used the Amazon Mechanical Turk crowdsourcing platform for our judgment elicitation study. To ensure consistent ratings (van der Lee et al., 2019), we required that raters have completed at least 1,000 tasks, and have at least 98% approval rate. Participants were restricted to English-speaking countries (USA, UK, Canada,

Australia, Ireland, or New Zealand) and were allowed to provide feedback or ask questions. Raters were paid an average of $0.35 for each task, ensuring that the remuneration is higher than the minimum wage per hour in the United States. We compared SeqPlan with Gold, Templ, ED+CC, and Macro; we did not compare against ENT, as previous work (Puduppully and Lapata, 2021) has shown that it performs poorly against Macro. For RotoWire, we additionally compared against RBF-2020.

Supported and Contradicted Facts Our first eliciation study provided raters with box scores (and play-by-plays in the case of MLB), along with sentences randomly extracted from game summaries. We asked them to count supported and contradicting facts (ignoring hallucinations). Participants were given a cheatsheet to help them understand box score and play-by-play statistics as well as examples of sentences with the correct count of supported and contradicting facts. This evaluation was conducted on 40 summaries (20 for each dataset), with four sentences per summary, each rated by three participants. For MLB, this resulted in 300 tasks (5 systems $\times$ 20 summaries $\times$ 3 raters) and for RotoWire in 360 (6 systems $\times$ 20 summaries $\times$ 3 raters). Altogether, we had 177 participants. The agreement between raters using Krippendorff's α for supported facts and contradicting facts was 0.43.

Table 6 (columns #Supp and #Contra) presents our results. Lower is better for contradicting facts. In case of supporting facts, the count should neither be too high nor too low. A high count of supporting facts indicates indicates poor content selection. A low count of supporting facts with a high count of contradicting facts indicates low accuracy of generation.

Templ achieves the lowest count of contradicting facts and the highest count of supported facts for both the datasets. This is no surprise as it essentially regurgitates facts (i.e., records) from the table. On MLB, all systems display a comparable count of *supported* facts (differences are not statistically significant), with the exception of Templ, which contains significantly more. In terms of *contradicting* facts, SeqPlan performs on par with Macro, Gold, and Templ, and is significantly better than ED+CC. On RotoWire, in terms of supported facts, SeqPlan performs on par with the other neural models, is significantly higher than Gold, and significantly lower than Templ. In terms of contradicting facts, SeqPlan performs on par with Macro, Gold, and Templ, and significantly better than ED+CC and RBF-2020.

MLB	#Supp	#Contra	Gram	Coher	Concis
Gold	3.59	0.14	21.67	29.17	14.17
Templ	4.21*	0.04	−58.33*	−48.33*	9.17
ED+CC	3.42	0.72*	−32.50*	−18.33*	−48.33*
Macro	3.76	0.25	37.50	15.00	22.50
SeqPlan	3.68	0.19	31.67	22.50	2.50

RotoWire	#Supp	#Contra	Gram	Coher	Concis
Gold	3.63*	0.07	42.67*	40.67	28.00
Templ	7.57*	0.08	−57.33*	−55.33*	−34.67*
ED+CC	3.92	0.91*	4.00	−14.67*	−13.33
RBF-2020	5.08	0.67*	6.00	1.33	−0.67
Macro	4.00	0.27	0.67	7.33	10.00
SeqPlan	4.84	0.17	4.00	20.67	10.67

Table 6: Average number of supported (#Supp) and contradicting (#Contra) facts in game summaries and *best-worst scaling* evaluation for Coherence (Coher), Conciseness (Concis), and Grammaticality (Gram). Lower is better for contradicting facts; higher is better for Coherence, Conciseness, and Grammaticality. Systems significantly different from SeqPlan are marked with an asterisk * (using a one-way ANOVA with post hoc Tukey HSD tests; $p \leq 0.05$).

Coherence, Grammaticality, and Conciseness In our second study, raters were asked to choose the better summary from a pair of summaries based on *Coherence* (Is the summary well structured and well organized and does it have a natural ordering of the facts?), *Conciseness* (Does the summary avoid unnecessary repetition including whole sentences, facts or phrases?), and *Grammaticality* (Is the summary written in well-formed English?). For this study, we required that the raters be able to comfortably comprehend summaries of NBA/MLB games. We obtained ratings using Best-Worst scaling (Louviere and Woodworth, 1991; Louviere et al., 2015), an elicitation paradigm shown to be more accurate than Likert scales. The score for a system is obtained by the number of times it is rated best minus the number of times it is rated worst (Orme, 2009). Scores range between −100 (absolutely worst) and +100 (absolutely best); higher is better. We assessed 40 summaries from the test set (20 for each dataset). Each summary pair was rated by three participants.

For MLB, we created 1,800 tasks (10 system pairs × 20 summaries × 3 raters × 3 dimensions) and 2,700 for RotoWire (15 pairs of systems × 20 summaries × 3 raters × 3 dimensions). Altogether, 377 raters participated in this task. The agreement between the raters using Krippendorff's α was 0.49.

On MLB, SeqPlan is significantly more coherent than ED+CC and Templ, and is comparable with Gold and Macro. A similar picture emerges with grammaticality. SeqPlan is as concise as Gold, Macro, and Templ, and significantly better than ED+CC. On RotoWire, SeqPlan is significantly more coherent than Templ and ED+CC, but on par with Macro, RBF-2020, and Gold. In terms of conciseness, SeqPlan is comparable to Gold, Macro, RBF-2020, and ED+CC, and significantly better than Templ. In terms of grammaticality, SeqPlan is comparable to Macro, RBF-2020, and ED+CC, significantly better than Templ, and significantly worse than Gold.

6 Discussion

In this work, we proposed a novel sequential latent variable model for joint macro planning and generation. Key in our approach is the creation of a latent plan in a sequential manner, while interleaving the prediction of plans and the generation of corresponding paragraphs. We proposed to deconstruct monolithic long document generation into smaller units (paragraphs, in our case), which affords flexibility and better communication between planning and generation. Taken together, the results of automatic and human evaluation suggest that SeqPlan performs best in terms of factuality and coherence, it generates diverse, and overall fluent, summaries, and is less data-hungry compared with strong systems like Macro and NCP+CC. As SeqPlan does not have to learn alignments between the macro plan and the output text, it is better suited for long-form generation. Potential applications include summarizing books (Kryściński et al., 2021), where the output can be longer than 1,000 tokens, or generating financial reports (Kogan et al., 2009; Händschke et al., 2018), where the output exceeds 9,000 tokens. Existing approaches for long-form generation summarize individual paragraphs independently (Kryściński et al., 2021) or adopt a hierarchical approach (Wu et al., 2021), where summaries of paragraphs form the basis of chapter summaries which in turn are composed into a book summary.

<V(Cardinals)> → <V(9-T)>→ <V(Cardinals)>→ <V(Cardinals)>→ <V(Brewers)>→ <V(9-T)>→ <V(8-B)>→ <V(8-T)>→ <V(8-B)>→ <V(9-B)>→ <V(Brewers)>→ <V(Adam Wainwright)>→ <V(Brewers)>→ <V(3-T)>→ <V(3-B)>→ <V(Carlos Villanueva)>
ST. LOUIS – The **St. Louis Cardinals** have been waiting for their starting rotation. <P> Skip Schumaker drove in the go-ahead run with a double in the **ninth** inning, and the Cardinals beat the Milwaukee Brewers 4–3 on Wednesday night to avoid a three-game sweep. <P> The **Cardinals** have won four of five, and have won four in a row. <P> The **Cardinals** have won four of five, including a three-game sweep by the **Brewers**. <P> Brian Barton led off the **ninth** with a pinch-hit double off Derrick Turnbow (0–1) and moved to third on Cesar Izturis' sacrifice bunt. Schumaker drove in Barton with a double down the left-field line. <P> Ryan Braun, who had two hits, led off the **eighth** with a double off Ryan Franklin (1–1). Braun went to third on a wild pitch and scored on Corey Hart's triple into the right-field corner. <P> Albert Pujols was intentionally walked to load the bases with one out in the **eighth**, and Guillermo Ankiel flied out. Troy Glaus walked to load the bases for Kennedy, who hit a sacrifice fly off Guillermo Mota. <P> Ryan Franklin (1–1) got the win despite giving up a run in the **eighth**. Ryan Braun led off with a double and scored on Corey Hart's one-out triple. <P> Jason Isringhausen pitched a perfect **ninth** for his seventh save in nine chances. He has converted his last six save opportunities and has n't allowed a run in his last three appearances. <P> The **Brewers** lost for the seventh time in eight games. <P> **Wainwright** allowed two runs and four hits in seven innings. He walked four and struck out six. <P> **Brewers** manager Ron Roenicke was ejected by home plate umpire Bill Miller for arguing a called third strike. <P> The Cardinals took a 2–0 lead in the **third**. Albert Pujols walked with two outs and Rick Ankiel walked. Glaus then lined a two-run double into the left-field corner. <P> The Brewers tied it in the **third**. Jason Kendall led off with a double and scored on Rickie Weeks' double. Ryan Braun's RBI single tied it at 2. <P> **Villanueva** allowed two runs and three hits in seven innings. He walked four and struck out one.

Table 7: Predicted macro plan (top) and generated output from our model. Transitions between paragraph plans are shown using →. Paragraphs are separated with <P> delimiters. Entities and events in the summary corresponding to the macro plan are boldfaced.

Table 7 gives an example of SeqPlan output. We see that the game summary follows the macro plan closely. In addition, the paragraph plans and the paragraphs exhibit coherent ordering. Manual inspection of SeqPlan summaries reveals that a major source of errors in MLB relate to attention diffusing over long paragraph plans. As an example, consider the following paragraph produced by SeqPlan "*Casey Kotchman had three hits and three RBIs, including a two-run double in the second inning that put the Angels up 2–0. Torii Hunter had three hits and drove in a run.*" In reality, *Torii Hunter* had two hits but the model incorrectly generates hits for *Casey Kotchman*.

The corresponding paragraph plan is 360 tokens long and attention fails to discern important tokens. A more sophisticated encoder, for example, based on Transformers (Vaswani et al., 2017), could make attention more focused. In RotoWire, the majority of errors involve numbers (e.g., team attributes) and numerical comparisons. Incorporating pre-executed operations such as min, max (Nie et al., 2018) could help alleviate these errors.

Finally, it is worth mentioning that although the template models achieve highest RG precision for both MLB and RotoWire (Tables 2 and 3), this is mainly because they repeat facts from the table. Template models score low against CS F, CO, and BLEU metrics. In addition, they obtain lowest scores in Grammaticality and Coherence (Table 6), which indicates that they are poor at selecting records from the table and ordering them correctly in a fluent manner.

Acknowledgments

We thank the Action Editor, Ehud Reiter, and the anonymous reviewers for their constructive feedback. We also thank Parag Jain for helpful discussions. We acknowledge the financial support of the European Research Council (award number 681760, ''Translating Multiple Modalities into Text'').

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 331–338, Vancouver, British Columbia, Canada. Association for Computational Linguistics. https://doi.org/10.3115/1220575.1220617

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1171–1179, Cambridge, MA, USA. MIT Press.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Carl Doersch. 2016. Tutorial on variational autoencoders. *CoRR*, abs/1606.05908.

Pablo A. Duboue and Kathleen R. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 172–179, Toulouse, France. Association for Computational Linguistics.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

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

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Yao Fu, Chuanqi Tan, Bin Bi, Mosha Chen, Yansong Feng, and Alexander Rush. 2020. Latent template induction with Gumbel-CRFS. In *Advances in Neural Information Processing Systems*, volume 33, pages 20259–20271. Curran Associates, Inc.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1017

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170. https://doi.org/10.1613/jair.5477

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1443

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). 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 3143–3152, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1310

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. 2017. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Markus Guhe. 2020. *Incremental Conceptualization for Language Production*. Psychology Press. https://doi.org/10.4324/9781003064398

Sebastian G. M. Händschke, Sven Buechel, Jan Goldenstein, Philipp Poschmann, Tinghui Duan, Peter Walgenbach, and Udo Hahn. 2018. A corpus of corporate annual and social responsibility reports: 280 million tokens of balanced organizational writing. In *Proceedings of the First Workshop on Economics and Natural Language Processing*, pages 20–31, Melbourne, Australia. Association for Computational Linguistics. https://doi.org/10.18653/v1/W18-3103

Hiroaki Hayashi, Yusuke Oda, Alexandra Birch, Ioannis Konstas, Andrew Finch, Minh-Thang Luong, Graham Neubig, and Katsuhito Sudoh. 2019. Findings of the third workshop on neural generation and translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 1–14, Hong Kong. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-5601

Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. 2019. Learning to select, track, and generate for data-to-text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2102–2113, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1202

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with Gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview.net.

Min-Yen Kan and Kathleen R. McKeown. 2002. Corpus-trained text generation for summarization. In *Proceedings of the International Natural Language Generation Conference*, pages 1–8, Harriman, New York, USA. Association for Computational Linguistics.

Zdeněk Kasner, Simon Mille, and Ondřej Dušek. 2021. Text-in-context: Token-level error detection for table-to-text generation. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 259–265, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Byeongchang Kim, Jaewoo Ahn, and Gunhee Kim. 2020. Sequential latent knowledge selection for knowledge-grounded dialogue. In *International Conference on Learning Representations*. https://doi.org/10.1145/3459637.3482314

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine

translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-4012

Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 272–280, Boulder, Colorado. Association for Computational Linguistics. https://doi.org/10.3115/1620754.1620794

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514, Seattle, Washington, USA. Association for Computational Linguistics.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization.

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, Austin, Texas. Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1128

Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368, Tokyo, Japan. Association for Computational Linguistics.

Willem J. M. Levelt. 1993. *Speaking: From Intention to Articulation*, volume 1. MIT Press. https://doi.org/10.7551/mitpress/6393.001.0001

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.703

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100, Copenhagen, Denmark. Association for Computational Linguistics.

Xiang Lisa Li and Alexander Rush. 2020. Posterior control of blackbox generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2731–2743, Online. Association for Computational Linguistics.

Jordan J. Louviere, Terry N. Flynn, and Anthony Alfred John Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge University Press. https://doi.org/10.1017/CBO9781107337855

Jordan J. Louviere and George G. Woodworth. 1991. *Best-Worst Scaling: A Model for the Largest Difference Judgments*. University of Alberta: Working Paper.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview.net.

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 *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, San Diego, California. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019a. Improving quality and efficiency in plan-based neural data-to-text generation. In *Proceedings of the 12th International*

Conference on Natural Language Generation, pages 377–382, Tokyo, Japan. Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-8645

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019b. 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Shashi Narayan, Joshua Maynez, Jakub Adamek, Daniele Pighin, Blaz Bratanic, and Ryan McDonald. 2020. Stepwise extractive summarization and planning with structured transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4143–4159, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.339

Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1422

Bryan Orme. 2009. Maxdiff analysis: Simple counting, individual-level logit, and HB. *Sawtooth Software*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019a. Data-to-text generation with content selection and planning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. Honolulu, Hawaii. https://doi.org/10.1609/aaai.v33i01.33016908

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019b. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1195

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text generation with macro planning. *Transactions of the Association for Computational Linguistics*, abs/2102.02723. https://doi.org/10.1162/tacl_a_00381

Ratish Puduppully, Jonathan Mallinson, and Mirella Lapata. 2019c. University of Edinburgh's submission to the document-level generation and translation shared task. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 268–272, Hong Kong. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-5630

Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. 2020. A hierarchical model for data-to-text generation. In *European Conference on Information Retrieval*, pages 65–80. Springer. https://doi.org/10.1007/978-3-030-45439-5_5

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, Tilburg University, The Netherlands. Association for Computational Linguistics. https://doi.org/10.18653/v1/W18-6535

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87. https://doi.org/10.1017/S1351324997001502

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*, Cambridge University Press, New York, NY. https://doi.org/10.1017/CBO9780511519857

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org.

Jacques Robin. 1994. *Revision-based generation of Natural Language Summaries providing historical Background*. Ph.D. thesis, Columbia University.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280. https://doi.org/10.1162/tacl_a_00313

Fahimeh Saleh, Alexandre Berard, Ioan Calapodescu, and Laurent Besacier. 2019. Naver Labs Europe's systems for the document-level generation and translation task at WNGT 2019. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 273–279, Hong Kong. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-5631

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, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1162

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3295–3301. AAAI Press.

Shiv Shankar and Sunita Sarawagi. 2019. Posterior attention models for sequence to sequence learning. In *International Conference on Learning Representations*. https://doi.org/10.18653/v1/D18-1065

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. 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 3257–3268, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1321

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Kumiko Tanaka-Ishii, Koiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1282–1288, Montreal, Quebec, Canada. Association for Computational Linguistics. https://doi.org/10.3115/980691.980778

M. Martin Taylor and Insup Taylor. 1990. Book reviews: Speaking: From intention to articulation. *Computational Linguistics*, 16(1).

Craig Thomson and Ehud Reiter. 2020. A gold standard methodology for evaluating accuracy in data-to-text systems. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 158–168, Dublin, Ireland. Association for Computational Linguistics.

Craig Thomson and Ehud Reiter. 2021. Generation challenges: Results of the accuracy evaluation shared task. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 240–248, Aberdeen,

Scotland, UK. Association for Computational Linguistics.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015, Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Ronald J. Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501. https://doi.org/10.1162/neco.1990.2.4.490

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, Copenhagen, Denmark. Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1239

Jeff Wu, Long Ouyang, Daniel M. Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul F. Christiano. 2021. Recursively summarizing books with human feedback. *CoRR*, abs/2109.10862.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1174

Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. 2020. Variational template machine for data-to-text generation. In *International Conference on Learning Representations*.