Data-to-Text Generation with Content Selection and Planning

Ratish Puduppully, Li Dong, Mirella Lapata
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
r.puduppully@sms.ed.ac.uk, li.dong@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Recent advances in data-to-text generation have led to the use of large-scale datasets and neural network models which are trained end-to-end, without explicitly modeling *what to say* and *in what order*. In this work, we present a neural network architecture which incorporates content selection and planning without sacrificing end-to-end training. We decompose the generation task into two stages. Given a corpus of data records (paired with descriptive documents), we first generate a content plan highlighting which information should be mentioned and in which order and then generate the document while taking the content plan into account. Automatic and human-based evaluation experiments show that our model[1] outperforms strong baselines improving the state-of-the-art on the recently released ROTOWIRE dataset.

1 Introduction

Data-to-text generation broadly refers to the task of automatically producing text from non-linguistic input (Reiter and Dale 2000; Gatt and Krahmer 2018). The input may be in various forms including databases of records, spreadsheets, expert system knowledge bases, simulations of physical systems, and so on. Table 1 shows an example in the form of a database containing statistics on NBA basketball games, and a corresponding game summary.

Traditional methods for data-to-text generation (Kukich 1983; McKeown 1992) implement a pipeline of modules including content planning (selecting specific content from some input and determining the structure of the output text), sentence planning (determining the structure and lexical content of each sentence) and surface realization (converting the sentence plan to a surface string). Recent neural generation systems (Lebret et al. 2016; Mei et al. 2016; Wiseman et al. 2017) do not explicitly model any of these stages, rather they are trained in an end-to-end fashion using the very successful encoder-decoder architecture (Bahdanau et al. 2015) as their backbone.

Despite producing overall fluent text, neural systems have difficulty capturing long-term structure and generating documents more than a few sentences long. Wiseman et al. (2017) show that neural text generation techniques perform poorly at content selection, they struggle to maintain inter-sentential coherence, and more generally a reasonable ordering of the selected facts in the output text. Additional challenges include avoiding redundancy and being faithful to the input. Interestingly, comparisons against template-based methods show that neural techniques do not fare well on metrics of content selection recall and factual output generation (i.e., they often hallucinate statements which are not supported by facts in the database).

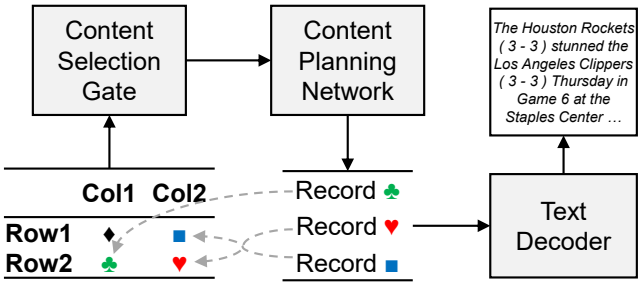

Figure 1: Block diagram of our approach.

In this paper, we address these shortcomings by *explicitly* modeling content selection and planning within a neural data-to-text architecture. Our model learns a content plan from the input and conditions on the content plan in order to generate the output document (see Figure 1 for an illustration). An explicit content planning mechanism has at least three advantages for multi-sentence document generation: it represents a high-level organization of the document structure allowing the decoder to concentrate on the easier tasks of sentence planning and surface realization; it makes the process of data-to-document generation more interpretable by generating an intermediate representation; and reduces redundancy in the output, since it is less likely for the content plan to contain the same information in multiple places.

We train our model end-to-end using neural networks and evaluate its performance on ROTOWIRE (Wiseman et al. 2017), a recently released dataset which contains statistics of NBA basketball games paired with human-written summaries (see Table 1). Automatic and human evaluation shows that modeling content selection and planning improves generation considerably over competitive baselines.

[1]Our code is publicly available at https://github.com/ratishsp/data2text-plan-py.

TEAM	WIN	LOSS	PTS	FG_PCT	RB	AST	...
Pacers	4	6	99	42	40	17	...
Celtics	5	4	105	44	47	22	...

PLAYER	H/V	AST	RB	PTS	FG	CITY	...
Jeff Teague	H	4	3	20	4	Indiana	...
Miles Turner	H	1	8	17	6	Indiana	...
Isaiah Thomas	V	5	0	23	4	Boston	...
Kelly Olynyk	V	4	6	16	6	Boston	...
Amir Johnson	V	3	9	14	4	Boston	...
...	...	...	...	...	...	...	

PTS: points, FT_PCT: free throw percentage, RB: rebounds, AST: assists, H/V: home or visiting, FG: field goals, CITY: player team city.

The **Boston Celtics** defeated the host **Indiana Pacers 105-99** at Bankers Life Fieldhouse on Saturday. In a battle between two injury-riddled teams, the Celtics were able to prevail with a much needed road victory. The key was shooting and defense, as the **Celtics** outshot the **Pacers** from the field, from three-point range and from the free-throw line. Boston also held Indiana to **42 percent** from the field and **22 percent** from long distance. The Celtics also won the rebounding and assisting differentials, while tying the Pacers in turnovers. There were 10 ties and 10 lead changes, as this game went down to the final seconds. Boston (**5–4**) has had to deal with a gluttony of injuries, but they had the fortunate task of playing a team just as injured here. **Isaiah** Thomas led the team in scoring, totaling **23 points and five assists on 4–of–13** shooting. He got most of those points by going 14–of–15 from the free-throw line. **Kelly Olynyk** got a rare start and finished second on the team with his **16 points, six rebounds and four assists**.

Table 1: Example of data-records and document summary. Entities and values corresponding to the plan in Table 2 are boldfaced.

2 Related Work

The generation literature provides multiple examples of content selection components developed for various domains which are either hand-built (Kukich 1983; McKeown 1992; Reiter and Dale 1997; Duboue and McKeown 2003) or learned from data (Barzilay and Lapata 2005; Duboue and McKeown 2001; 2003; Liang et al. 2009; Angeli et al. 2010; Kim and Mooney 2010; Konstas and Lapata 2013). Likewise, creating summaries of sports games has been a topic of interest since the early beginnings of generation systems (Robin 1994; Tanaka-Ishii et al. 1998).

Earlier work on content planning has relied on generic planners (Dale 1988), based on Rhetorical Structure Theory (Hovy 1993) and schemas (McKeown et al. 1997). Content planners are defined by analysing target texts and devising hand-crafted rules. Duboue and McKeown (2001) study ordering constraints for content plans and in follow-on work (Duboue and McKeown 2002) learn a content planner from an aligned corpus of inputs and human outputs. A few researchers (Mellish et al. 1998; Karamanis 2004) select content plans according to a ranking function.

More recent work focuses on end-to-end systems instead of individual components. However, most models make simplifying assumptions such as generation without any content selection or planning (Belz 2008; Wong and Mooney 2007) or content selection without planning (Konstas and Lapata 2012; Angeli et al. 2010; Kim and Mooney 2010). An exception are Konstas and Lapata (2013) who incorporate content plans represented as grammar rules operating on the document level. Their approach works reasonably well with weather forecasts, but does not scale easily to larger databases, with richer vocabularies, and longer text descriptions. The model relies on the EM algorithm (Dempster, Laird, and Rubin 1977) to learn the weights of the grammar rules which can be very many even when tokens are aligned to database records as a preprocessing step.

Our work is closest to recent neural network models which learn generators from data and accompanying text resources. Most previous approaches generate from Wikipedia infoboxes focusing either on single sentences (Lebret et al. 2016; 2017; Sha et al. 2017; Liu et al. 2017) or short texts (Perez-Beltrachini and Lapata 2018). Mei et al. (2016) use a neural encoder-decoder model to generate weather forecasts and soccer commentaries, while Wiseman et al. (2017) generate NBA game summaries (see Table 1). They introduce a new dataset for data-to-document generation which is sufficiently large for neural network training and adequately challenging for testing the capabilities of document-scale text generation (e.g., the average summary length is 330 words and the average number of input records is 628). Moreover, they propose various automatic evaluation measures for assessing the quality of system output. Our model follows on from Wiseman et al. (2017) addressing the challenges for data-to-text generation identified in their work. We are not aware of any previous neural network-based approaches which incorporate content selection and planning mechanisms and generate multi-sentence documents. Perez-Beltrachini and Lapata (2018) introduce a content selection component (based on multi-instance learning) without content planning, while Liu et al. (2017) propose a sentence planning mechanism which orders the contents of a Wikipedia infobox so as to generate a single sentence.

3 Problem Formulation

The input to our model is a table of records (see Table 1 left hand-side). Each record r_j has four features including its type ($r_{j,1}$; e.g., LOSS, CITY), entity ($r_{j,2}$; e.g., Pacers, Miles Turner), value ($r_{j,3}$; e.g., 11, Indiana), and whether a player is on the home- or away-team ($r_{j,4}$; see column H/V in Table 1), represented as $\{r_{j,k}\}_{k=1}^{4}$. The output y is a document containing words $y = y_1 \cdots y_{|y|}$ where $|y|$ is the document length. Figure 2 shows the overall architecture of our model which consists of two stages: (a) *content selection and planning* operates on the input records of a database and produces a content plan specifying which records are to be verbalized in the document and in which order (see Table 2) and (b) *text generation* produces the output text given the content plan as input; at each decoding step, the generation model attends over vector representations of the records in the content plan.

Let $r = \{r_j\}_{j=1}^{|r|}$ denote a table of input records and y the output text. We model $p(y|r)$ as the joint probability of text y and content plan z, given input r. We further decompose $p(y, z|r)$ into $p(z|r)$, a content selection and planning

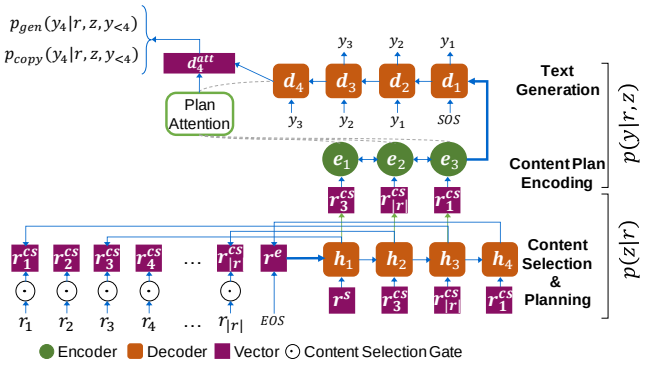

Figure 2: Generation model with content selection and planning; the content selection gate is illustrated in Figure 3.

phase, and $p(y|r,z)$, a text generation phase:

$$p(y|r) = \sum_z p(y,z|r) = \sum_z p(z|r)p(y|r,z)$$

In the following we explain how the components $p(z|r)$ and $p(y|r,z)$ are estimated.

Record Encoder

The input to our model is a table of unordered records, each represented as features $\{r_{j,k}\}_{k=1}^{4}$. Following previous work (Yang et al. 2017; Wiseman et al. 2017), we embed features into vectors, and then use a multilayer perceptron to obtain a vector representation $\mathbf{r}_j$ for each record:

$$\mathbf{r}_j = \text{ReLU}(\mathbf{W}_r[\mathbf{r}_{j,1};\mathbf{r}_{j,2};\mathbf{r}_{j,3};\mathbf{r}_{j,4}] + \mathbf{b}_r)$$

where $[;]$ indicates vector concatenation, $\mathbf{W}_r \in \mathbb{R}^{n\times 4n}$, $\mathbf{b}_r \in \mathbb{R}^n$ are parameters, and ReLU is the rectifier activation function.

Content Selection Gate

The context of a record can be useful in determining its importance vis-a-vis other records in the table. For example, if a player scores many points, it is likely that other meaningfully related records such as field goals, three-pointers, or rebounds will be mentioned in the output summary. To better capture such dependencies among records, we make use of the content selection gate mechanism as shown in Figure 3.

We first compute attention scores $\alpha_{j,k}$ over the input table and use them to obtain an attentional vector $\mathbf{r}_j^{att}$ for each record r_j:

$$\alpha_{j,k} \propto \exp(\mathbf{r}_j^\intercal \mathbf{W}_a \mathbf{r}_k)$$
$$\mathbf{c}_j = \sum_{k\neq j} \alpha_{j,k}\mathbf{r}_k$$
$$\mathbf{r}_j^{att} = \mathbf{W}_g[\mathbf{r}_j;\mathbf{c}_j]$$

where $\mathbf{W}_a \in \mathbb{R}^{n\times n}$, $\mathbf{W}_g \in \mathbb{R}^{n\times 2n}$ are parameter matrices, and $\sum_{k\neq j}\alpha_{j,k} = 1$.

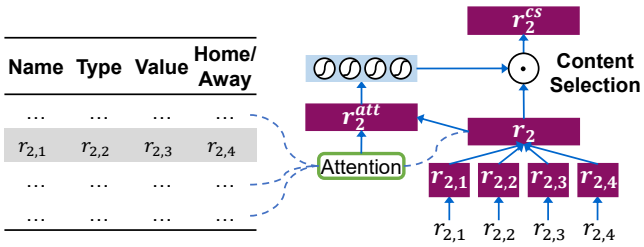

Figure 3: Content selection mechanism.

We next apply the content selection gating mechanism to $\mathbf{r}_j$, and obtain the new record representation $\mathbf{r}_j^{cs}$ via:

$$\mathbf{g}_j = \text{sigmoid}\left(\mathbf{r}_j^{att}\right)$$
$$\mathbf{r}_j^{cs} = \mathbf{g}_j \odot \mathbf{r}_j$$

where $\odot$ denotes element-wise multiplication, and gate $\mathbf{g}_j \in [0,1]^n$ controls the amount of information flowing from $\mathbf{r}_j$. In other words, each element in $\mathbf{r_j}$ is weighed by the corresponding element of the content selection gate $\mathbf{g}_j$.

Content Planning

In our generation task, the output text is long but follows a canonical structure. Game summaries typically begin by discussing which team won/lost, following with various statistics involving individual players and their teams (e.g., who performed exceptionally well or under-performed), and finishing with any upcoming games. We hypothesize that generation would benefit from an explicit plan specifying both *what to say* and *in which order*. Our model learns such content plans from training data. However, notice that ROTOWIRE (see Table 1) and most similar data-to-text datasets do not naturally contain content plans. Fortunately, we can obtain these relatively straightforwardly following an information extraction approach (which we explain in Section 4).

Suffice it to say that plans are extracted by mapping the text in the summaries onto entities in the input table, their values, and types (i.e., relations). A plan is a sequence of pointers with each entry pointing to an input record $\{r_j\}_{j=1}^{|r|}$. An excerpt of a plan is shown in Table 2. The order in the plan corresponds to the sequence in which entities appear in the game summary. Let $z = z_1 \ldots z_{|z|}$ denote the content planning sequence. Each z_k points to an input record, i.e., $z_k \in \{r_j\}_{j=1}^{|r|}$. Given the input records, the probability $p(z|r)$ is decomposed as:

$$p(z|r) = \prod_{k=1}^{|z|} p(z_k|z_{<k}, r)$$

where $z_{<k} = z_1 \ldots z_{k-1}$.

Since the output tokens of the content planning stage correspond to positions in the input sequence, we make use of Pointer Networks (Vinyals et al. 2015). The latter use attention to point to the tokens of the input sequence rather than creating a weighted representation of source encodings. As shown in Figure 2, given $\{r_j\}_{j=1}^{|r|}$, we use an LSTM decoder to generate tokens corresponding to positions in the

Value	Entity	Type	H/V
Boston	Celtics	TEAM-CITY	V
Celtics	Celtics	TEAM-NAME	V
105	Celtics	TEAM-PTS	V
Indiana	Pacers	TEAM-CITY	H
Pacers	Pacers	TEAM-NAME	H
99	Pacers	TEAM-PTS	H
42	Pacers	TEAM-FG_PCT	H
22	Pacers	TEAM-FG3_PCT	H
5	Celtics	TEAM-WIN	V
4	Celtics	TEAM-LOSS	V
Isaiah	Isaiah_Thomas	FIRST_NAME	V
Thomas	Isaiah_Thomas	SECOND_NAME	V
23	Isaiah_Thomas	PTS	V
5	Isaiah_Thomas	AST	V
4	Isaiah_Thomas	FGM	V
13	Isaiah_Thomas	FGA	V
Kelly	Kelly_Olynyk	FIRST_NAME	V
Olynyk	Kelly_Olynyk	SECOND_NAME	V
16	Kelly_Olynyk	PTS	V
6	Kelly_Olynyk	REB	V
4	Kelly_Olynyk	AST	V
...	...	...	...

Table 2: Content plan for the example in Table 1.

input. The first hidden state of the decoder is initialized by $\text{avg}(\{\mathbf{r}_j^{cs}\}_{j=1}^{|r|})$, i.e., the average of record vectors. At decoding step k, let $\mathbf{h}_k$ be the hidden state of the LSTM. We model $p(z_k = r_j | z_{<k}, r)$ as the attention over input records:

$$p(z_k = r_j | z_{<k}, r) \propto \exp(\mathbf{h}_k^\intercal \mathbf{W}_c \mathbf{r}_j^{cs})$$

where the probability is normalized to 1, and $\mathbf{W}_c$ are parameters. Once z_k points to record r_j, we use the corresponding vector $\mathbf{r}_j^{cs}$ as the input of the next LSTM unit in the decoder.

Text Generation

The probability of output text y conditioned on content plan z and input table r is modeled as:

$$p(y|r,z) = \prod_{t=1}^{|y|} p(y_t | y_{<t}, z, r)$$

where $y_{<t} = y_1 \dots y_{t-1}$. We use the encoder-decoder architecture with an attention mechanism to compute $p(y|r,z)$.

We first encode content plan z into $\{\mathbf{e}_k\}_{k=1}^{|z|}$ using a bidirectional LSTM. Because the content plan is a sequence of input records, we directly feed the corresponding record vectors $\{\mathbf{r}_j^{cs}\}_{j=1}^{|r|}$ as input to the LSTM units, which share the record encoder with the first stage.

The text decoder is also based on a recurrent neural network with LSTM units. The decoder is initialized with the hidden states of the final step in the encoder. At decoding step t, the input of the LSTM unit is the embedding of the previously predicted word y_{t-1}. Let $\mathbf{d}_t$ be the hidden state of the t-th LSTM unit. The probability of predicting y_t from the output vocabulary is computed via:

$$\beta_{t,k} \propto \exp(\mathbf{d}_t^\intercal \mathbf{W}_b \mathbf{e}_k) \quad (1)$$

$$\mathbf{q}_t = \sum_k \beta_{t,k} \mathbf{e}_k$$

$$\mathbf{d}_t^{att} = \tanh(\mathbf{W}_d [\mathbf{d}_t; \mathbf{q}_t])$$

$$p_{gen}(y_t | y_{<t}, z, r) = \text{softmax}_{y_t}(\mathbf{W}_y \mathbf{d}_t^{att} + \mathbf{b}_y) \quad (2)$$

where $\sum_k \beta_{t,k} = 1$, $\mathbf{W}_b \in \mathbb{R}^{n \times n}$, $\mathbf{W}_d \in \mathbb{R}^{n \times 2n}$, $\mathbf{W}_y \in \mathbb{R}^{n \times |\mathcal{V}_y|}$, $\mathbf{b}_y \in \mathbb{R}^{|\mathcal{V}_y|}$ are parameters, and $|\mathcal{V}_y|$ is the output vocabulary size.

We further augment the decoder with a copy mechanism, i.e., the ability to copy words directly from the *value* portions of records in the content plan (i.e., $\{z_k\}_{k=1}^{|z|}$). We experimented with joint (Gu et al. 2016) and conditional copy methods (Gulcehre et al. 2016). Specifically, we introduce a variable $u_t \in \{0,1\}$ for each time step to indicate whether the predicted token y_t is copied ($u_t = 1$) or not ($u_t = 0$). The probability of generating y_t is computed by:

$$p(y_t | y_{<t}, z, r) = \sum_{u_t \in \{0,1\}} p(y_t, u_t | y_{<t}, z, r)$$

where u_t is marginalized out.

Joint Copy The probability of copying from record values and generating from the vocabulary is globally normalized:

$$p(y_t, u_t | y_{<t}, z, r) \propto \begin{cases} \sum_{y_t \leftarrow z_k} \exp(\mathbf{d}_t^\intercal \mathbf{W}_b \mathbf{e}_k) & u_t = 1 \\ \exp(\mathbf{W}_y \mathbf{d}_t^{att} + \mathbf{b}_y) & u_t = 0 \end{cases}$$

where $y_t \leftarrow z_k$ indicates that y_t can be copied from z_k, $\mathbf{W}_b$ is shared as in Equation (1), and $\mathbf{W}_y, \mathbf{b}_y$ are shared as in Equation (2).

Conditional Copy The variable u_t is first computed as a switch gate, and then is used to obtain the output probability:

$$p(u_t = 1 | y_{<t}, z, r) = \text{sigmoid}(\mathbf{w}_u \cdot \mathbf{d}_t + b_u)$$

$$p(y_t, u_t | y_{<t}, z, r) = \begin{cases} p(u_t | y_{<t}, z, r) \sum_{y_t \leftarrow z_k} \beta_{t,k} & u_t = 1 \\ p(u_t | y_{<t}, z, r) p_{gen}(y_t | y_{<t}, z, r) & u_t = 0 \end{cases}$$

where $\beta_{t,k}$ and $p_{gen}(y_t | y_{<t}, z, r)$ are computed as in Equations (1)–(2), and $\mathbf{w}_u \in \mathbb{R}^n$, $b_u \in \mathbb{R}$ are parameters. Following Gulcehre et al. (2016) and Wiseman et al. (2017), if y_t appears in the content plan during training, we assume that y_t is copied (i.e., $u_t = 1$).[2]

[2]We learn whether y_t can be copied from candidate z_k by applying supervision during training. Specifically, we retain z_k when the record entity and its value occur in the same sentence in y.

Training and Inference

Our model is trained to maximize the log-likelihood of the gold[3] content plan given table records r *and* the gold output text given the content plan and table records:

$$\max \sum_{(r,z,y)\in\mathcal{D}} \log p\left(z|r\right) + \log p\left(y|r,z\right)$$

where $\mathcal{D}$ represents training examples (input records, plans, and game summaries). During inference, the output for input r is predicted by:

$$\hat{z} = \arg\max_{z'} p(z'|r)$$
$$\hat{y} = \arg\max_{y'} p(y'|r,\hat{z})$$

where z' and y' represent content plan and output text candidates, respectively. For each stage, we utilize beam search to approximately obtain the best results.

4 Experimental Setup

Data We trained and evaluated our model on ROTOWIRE (Wiseman et al. 2017), a dataset of basketball game summaries, paired with corresponding box- and line-score tables. The summaries are professionally written, relatively well structured and long (337 words on average). The number of record types is 39, the average number of records is 628, the vocabulary size is 11.3K words and token count is 1.6M. The dataset is ideally suited for document-scale generation. We followed the data partitions introduced in Wiseman et al. (2017): we trained on 3,398 summaries, tested on 728, and used 727 for validation.

Content Plan Extraction We extracted content plans from the ROTOWIRE game summaries following an information extraction (IE) approach. Specifically, we used the IE system introduced in Wiseman et al. (2017) which identifies candidate entity (i.e., player, team, and city) and value (i.e., number or string) pairs that appear in the text, and then predicts the type (aka relation) of each candidate pair. For instance, in the document in Table 1, the IE system might identify the pair "Jeff Teague, 20" and then predict that that their relation is "PTS", extracting the record (Jeff Teague, 20, PTS). Wiseman et al. (2017) train an IE system on ROTOWIRE by determining word spans which could represent entities (i.e., by matching them against players, teams or cities in the database) and numbers. They then consider each entity-number pair in the same sentence, and if there is a record in the database with matching entities and values, the pair is assigned the corresponding record type or otherwise given the label "none" to indicate unrelated pairs.

We adopted their IE system architecture which predicts relations by ensembling 3 convolutional models and 3 bidirectional LSTM models. We trained this system on the training portion of the ROTOWIRE corpus.[4] On held-out data it achieved 94% accuracy, and recalled approximately 80% of the relations licensed by the records. Given the output of the IE system, a content plan simply consists of (entity, value, record type, H/V) tuples in their order of appearance in a game summary (the content plan for the summary in Table 1 is shown in Table 2). Player names are pre-processed to indicate the individual's first name and surname (see Isaiah and Thomas in Table 2); team records are also pre-processed to indicate the name of team's city and the team itself (see Boston and Celtics in Table 2).

Training Configuration We validated model hyperparameters on the development set. We did not tune the dimensions of word embeddings and LSTM hidden layers; we used the same value of 600 reported in Wiseman et al. (2017). We used one-layer pointer networks during content planning, and two-layer LSTMs during text generation. Input feeding (Luong et al. 2015) was employed for the text decoder. We applied dropout (Zaremba et al. 2014) at a rate of 0.3. Models were trained for 25 epochs with the Adagrad optimizer (Duchi et al. 2011); the initial learning rate was 0.15, learning rate decay was selected from $\{0.5, 0.97\}$, and batch size was 5. For text decoding, we made use of BPTT (Mikolov et al. 2010) and set the truncation size to 100. We set the beam size to 5 during inference. All models are implemented in OpenNMT-py (Klein et al. 2017).

5 Results

Automatic Evaluation We evaluated model output using the metrics defined in Wiseman et al. (2017). The idea is to employ a fairly accurate IE system (see the description in Section 4) on the gold and automatic summaries and compare whether the identified relations align or diverge.

Let $\hat{y}$ be the gold output, and y the system output. *Content selection* (CS) measures how well (in terms of precision and recall) the records extracted from y match those found in $\hat{y}$. *Relation generation* (RG) measures the factuality of the generation system as the proportion of records extracted from y which are also found in r (in terms of precision and number of unique relations). *Content ordering* (CO) measures how well the system orders the records it has chosen and is computed as the normalized Damerau-Levenshtein Distance between the sequence of records extracted from y and $\hat{y}$. In addition to these metrics, we report BLEU (Papineni et al. 2002), with human-written game summaries as reference.

Our results on the development set are summarized in Table 3. We compare our Neural Content Planning model (NCP for short) against the two encoder-decoder (ED) models presented in Wiseman et al. (2017) with joint copy (JC) and conditional copy (CC), respectively. In addition to our own re-implementation of these models, we include the best scores reported in Wiseman et al. (2017) which were obtained with an encoder-decoder model enhanced with con-

[3] Strictly speaking, the content plan is not gold since it was not created by an expert but is the output of a fairly accurate IE system.

[4] A bug in the code of Wiseman et al. (2017) excluded number words from the output summary. We corrected the bug and this resulted in greater recall for the relations extracted from the summaries. See the supplementary material for more details.

Model	RG		CS		CO	BLEU
	#	P%	P%	R%	DLD%	
TEMPL	54.29	99.92	26.61	59.16	14.42	8.51
WS-2017	23.95	75.10	28.11	35.86	15.33	14.57
ED+JC	22.98	76.07	27.70	33.29	14.36	13.22
ED+CC	21.94	75.08	27.96	32.71	15.03	13.31
NCP+JC	33.37	87.40	32.20	48.56	17.98	14.92
NCP+CC	33.88	87.51	33.52	51.21	18.57	16.19
NCP+OR	21.59	89.21	88.52	85.84	78.51	24.11

Table 3: Automatic evaluation on ROTOWIRE development set using relation generation (RG) count (#) and precision (P%), content selection (CS) precision (P%) and recall (R%), content ordering (CO) in normalized Damerau-Levenshtein distance (DLD%), and BLEU.

Model	RG		CS		CO	BLEU
	#	P%	P%	R%	DLD%	
ED+CC	21.94	75.08	27.96	32.71	15.03	13.31
CS+CC	24.93	80.55	28.63	35.23	15.12	13.52
CP+CC	33.73	84.85	29.57	44.72	15.84	14.45
NCP+CC	33.88	87.51	33.52	51.21	18.57	16.19
NCP	34.46	—	38.00	53.72	20.27	—

Table 4: Ablation results on ROTOWIRE development set using relation generation (RG) count (#) and precision (P%), content selection (CS) precision (P%) and recall (R%), content ordering (CO) in normalized Damerau-Levenshtein distance (DLD%), and BLEU.

ditional copy (WS-2017). Table 3 also shows results when NCP uses oracle content plans (OR) as input. In addition, we report the performance of a template-based generator (Wiseman et al. 2017) which creates a document consisting of eight template sentences: an introductory sentence (who won/lost), six player-specific sentences (based on the six highest-scoring players in the game), and a conclusion sentence.

As can be seen, NCP improves upon vanilla encoder-decoder models (ED+JC, ED+CC), irrespective of the copy mechanism being employed. In fact, NCP achieves comparable scores with either joint or conditional copy mechanism which indicates that it is the content planner which brings performance improvements. Overall, NCP+CC achieves best content selection and content ordering scores in terms of BLEU. Compared to the best reported system in Wiseman et al. (2017), we achieve an absolute improvement of approximately 12% in terms of relation generation; content selection precision also improves by 5% and recall by 15%, content ordering increases by 3%, and BLEU by 1.5 points. The results of the oracle system (NCP+OR) show that content selection and ordering do indeed correlate with the quality of the content plan and that any improvements in our planning component would result in better output. As far as the template-based system is concerned, we observe that it obtains low BLEU and CS precision but scores high on CS recall and RG metrics. This is not surprising as the template system is provided with domain knowledge which our model does not have, and thus represents an upper-bound on content selection and relation generation. We also measured the degree to which the game summaries generated by our model contain redundant information as the proportion of non-duplicate records extracted from the summary by the IE system. 84.5% of the records in NCP+CC are non-duplicates compared to Wiseman et al. (2017) who obtain 72.9% showing that our model is less repetitive.

Model	RG		CS		CO	BLEU
	#	P%	P%	R%	DLD%	
TEMPL	**54.23**	**99.94**	26.99	**58.16**	14.92	8.46
WS-2017	23.72	74.80	29.49	36.18	15.42	14.19
NCP+JC	34.09	87.19	32.02	47.29	17.15	14.89
NCP+CC	34.28	87.47	**34.18**	51.22	**18.58**	**16.50**

Table 5: Automatic evaluation on ROTOWIRE test set using relation generation (RG) count (#) and precision (P%), content selection (CS) precision (R%) and recall (R%), content ordering (CO) in normalized Damerau-Levenshtein distance (DLD%), and BLEU.

We further conducted an ablation study with the conditional copy variant of our model (NCP+CC) to establish whether improvements are due to better content selection (CS) and/or content planning (CP). We see in Table 4 that content selection and planning individually contribute to performance improvements over the baseline (ED+CC), and accuracy further increases when both components are taken into account. In addition we evaluated these components on their own (independently of text generation) by comparing the output of the planner (see $p(z|r)$ block in Figure 2) against gold content plans obtained using the IE system (see row NCP in Table 4. Compared to the full system (NCP+CC), content selection precision and recall are higher (by 4.5% and 2%, respectively) as well as content ordering (by 1.8%). In another study, we used the CS and CO metrics to measure how well the generated text follows the content plan produced by the planner (instead of arbitrarily adding or removing information). We found out that NCP+CC generates game summaries which follow the content plan closely: CS precision is higher than 85%, CS recall is higher than 93%, and CO higher than 84%. This reinforces our claim that higher accuracies in the content selection and planning phase will result in further improvements in text generation.

The test set results in Table 5 follow a pattern similar to the development set. NCP achieves higher accuracy in all metrics including relation generation, content selection, content ordering, and BLEU compared to Wiseman et al. (2017). We provide examples of system output in Figure 4 and the supplementary material.

Human-Based Evaluation We conducted two human evaluation experiments using the Amazon Mechanical Turk (AMT) crowdsourcing platform. The first study assessed relation generation by examining whether improvements in relation generation attested by automatic evaluation metrics are indeed corroborated by human judgments. We compared our best performing model (NCP+CC), with gold reference

The Golden State Warriors (10–2) defeated the Boston Celtics (6–6) 104–88. Klay Thompson scored 28 points (12–21 FG, 3–6 3PT, 1–1 FT) to go with 4 rebounds. Kevin Durant scored 23 points (10–13 FG, 1–2 3PT, 2–4 FT) to go with 10 rebounds. Isaiah Thomas scored 18 points (4–12 FG, 1–6 3PT, 9–9 FT) to go with 2 rebounds. Avery Bradley scored 17 points (7–15 FG, 2–4 3PT, 1–2 FT) to go with 10 rebounds. Stephen Curry scored 16 points (7–20 FG, 2–10 3PT, 0–0 FT) to go with 3 rebounds. Terry Rozier scored 11 points (3–5 FG, 2–3 3PT, 3–4 FT) to go with 7 rebounds. The Golden State Warriors' next game will be at home against the Dallas Mavericks, while the Boston Celtics will travel to play the Bulls.

The Golden State Warriors defeated the Boston Celtics 104–88 at TD Garden on Friday. The Warriors (10–2) came into this game winners of five of their last six games, but the Warriors (6–6) were able to pull away in the second half. Klay Thompson led the way for the Warriors with 28 points on 12–of–21 shooting, while Kevin Durant added 23 points, 10 rebounds, seven assists and two steals. Stephen Curry added 16 points and eight assists, while Draymond Green rounded out the box score with 11 points, eight rebounds and eight assists. For the Celtics, it was Isaiah Thomas who shot just 4–of–12 from the field and finished with 18 points. Avery Bradley added 17 points and 10 rebounds, while the rest of the Celtics combined to score just seven points. Boston will look to get back on track as they play host to the 76ers on Friday.

Figure 4: Example output from TEMPL (top) and NPC+CC (bottom). Text that accurately reflects a record in the associated box or line score is in blue, erroneous text is in red.

summaries, a template system and the best model of Wiseman et al. (2017). AMT workers were presented with a specific NBA game's box score and line score, and four (randomly selected) sentences from the summary. They were asked to identify supporting and contradicting facts mentioned in each sentence. We randomly selected 30 games from the test set. Each sentence was rated by three workers.

The left two columns in Table 6 contain the average number of supporting and contradicting facts per sentence as determined by the crowdworkers, for each model. The template-based system has the highest number of supporting facts, even compared to the human gold standard. TEMPL does not perform any content selection, it includes a large number of facts from the database and since it does not perform any generation either, it exhibits a few contradictions. Compared to WS-2017 and the Gold summaries, NCP+CC displays a larger number of supporting facts. All models are significantly[5] different in the number of supporting facts (#Supp) from TEMPL (using a one-way ANOVA with post-hoc Tukey HSD tests). NCP+CC is significantly different from WS-2017 and Gold. With respect to contradicting facts (#Cont), Gold and TEMPL are not significantly different from each other but are significantly different from the neural systems (WS-2017, NCP+CC).

In the second experiment, we assessed the generation quality of our model. We elicited judgments for the same 30 games used in the first study. For each game, participants were asked to compare a human-written summary, NCP with conditional copy (NCP+CC), Wiseman et al.'s (2017) best model, and the template system. Our study used *Best-Worst Scaling* (BWS; Louviere, Flynn, and Marley 2015), a technique shown to be less labor-intensive and providing more

[5]All significance differences reported throughout this paper are with a level less than 0.05.

	#Support	#Contra	Gram	Cohere	Concise
Gold	2.98	0.28	11.78	16.00	13.78
TEMPL	6.98	0.21	-0.89	-4.89	1.33
WS-2017	3.19	1.09	-4.22	-4.89	-6.44
NCP+CC	4.90	0.90	-2.44	-2.44	-3.55

Table 6: Average number of supporting (#Support) and contradicting (#Contra) facts in game summaries and *best-worst scaling* evaluation (higher is better) for grammaticality (Gram), Coherence (Cohere), and Conciseness (Concise).

reliable results as compared to rating scales (Kiritchenko and Mohammad 2017). We arranged every 4-tuple of competing summaries into 6 pairs. Every pair was shown to three crowdworkers, who were asked to choose which summary was *best* and which was *worst* according to three criteria: *Grammaticality* (is the summary fluent and grammatical?), *Coherence* (is the summary easy to read? does it follow a natural ordering of facts?), and *Conciseness* (does the summary avoid redundant information and repetitions?). The score of a system for each criterion is computed as the difference between the percentage of times the system was selected as the best and the percentage of times it was selected as the worst (Orme 2009). The scores range from -100 (absolutely worst) to $+100$ (absolutely best).

The results of the second study are summarized in Table 6. Gold summaries were perceived as significantly better compared to the automatic systems across all criteria (again using a one-way ANOVA with post-hoc Tukey HSD tests). NCP+CC was perceived as significantly more grammatical than WS-2017 but not compared to TEMPL which does not suffer from fluency errors since it does not perform any generation. NCP+CC was perceived as significantly more coherent than TEMPL and WS-2017. The template fairs poorly on coherence, its output is stilted and exhibits no variability (see top block in Table 4). With regard to conciseness, the neural systems are significantly worse than TEMPL, while NCP+CC is significantly better than WS-2017. By design the template cannot repeat information since there is no redundancy in the sentences chosen to verbalize the summary.

Taken together, our results show that content planning improves data-to-text generation across metrics and systems. We find that NCP+CC overall performs best, however there is a significant gap between automatically generated summaries and human-authored ones.

6 Conclusions

We presented a data-to-text generation model which is enhanced with content selection and planning modules. Experimental results (based on automatic metrics and judgment elicitation studies) demonstrate that generation quality improves both in terms of the number of relevant facts contained in the output text, and the order according to which these are presented. Positive side-effects of content planning are additional improvements in the grammaticality, and conciseness of the generated text. In the future, we would like to learn more detail-oriented plans involving inference over multiple facts and entities. We would also like to verify our approach across domains and languages.

References

Angeli, G.; Liang, P.; and Klein, D. 2010. A simple domain-independent probabilistic approach to generation. In *EMNLP*, 502–512.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Barzilay, R., and Lapata, M. 2005. Collective content selection for concept-to-text generation. In *EMNLP*.

Belz, A. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.* 14(4):431–455.

Chisholm, A.; Radford, W.; and Hachey, B. 2017. Learning to generate one-sentence biographies from wikidata. In *EACL*, 633–642.

Dale, R. 1988. *Generating referring expressions in a domain of objects and processes.* Ph.D. Dissertation, University of Edinburgh.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)* 1–38.

Duboue, P. A., and McKeown, K. R. 2001. Empirically estimating order constraints for content planning in generation. In *ACL*, 172–179.

Duboue, P., and McKeown, K. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of the International Natural Language Generation Conference*, 89–96.

Duboue, P. A., and McKeown, K. R. 2003. Statistical acquisition of content selection rules for natural language generation.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Gatt, A., and Krahmer, E. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *JAIR* 61:65–170.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, 1631–1640.

Gulcehre, C.; Ahn, S.; Nallapati, R.; Zhou, B.; and Bengio, Y. 2016. Pointing the unknown words. In *ACL*, 140–149.

Hovy, E. H. 1993. Automated discourse generation using discourse structure relations. *Artificial intelligence* 63(1-2):341–385.

Karamanis, N. 2004. *Entity coherence for descriptive text structuring*. Ph.D. Dissertation, School of Informatics, University of Edinburgh.

Kim, J., and Mooney, R. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Coling 2010: Posters*, 543–551.

Kiritchenko, S., and Mohammad, S. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *ACL*, volume 2, 465–470.

Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; and Rush, A. M. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Konstas, I., and Lapata, M. 2012. Unsupervised concept-to-text generation with hypergraphs. In *NAACL*, 752–761.

Konstas, I., and Lapata, M. 2013. A global model for concept-to-text generation. *J. Artif. Int. Res.* 48(1):305–346.

Kukich, K. 1983. Design of a knowledge-based report generator. In *ACL*.

Lebret, R.; Grangier, D.; and Auli, M. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, 1203–1213.

Liang, P.; Jordan, M.; and Klein, D. 2009. Learning semantic correspondences with less supervision. In *ACL*, 91–99.

Liu, T.; Wang, K.; Sha, L.; Chang, B.; and Sui, Z. 2017. Table-to-text generation by structure-aware seq2seq learning. *arXiv preprint arXiv:1711.09724*.

Louviere, J. J.; Flynn, T. N.; and Marley, A. A. J. 2015. *Best-worst scaling: Theory, methods and applications*. Cambridge University Press.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, 1412–1421.

McKeown, K. R.; Pan, S.; Shaw, J.; Jordan, D. A.; and Allen, B. A. 1997. Language generation for multimedia healthcare briefings. In *ANLP*, 277–282.

McKeown, K. 1992. *Text generation*. Cambridge University Press.

Mei, H.; Bansal, M.; and Walter, M. R. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL*, 720–730.

Mellish, C.; Knott, A.; Oberlander, J.; and O'Donnell, M. 1998. Experiments using stochastic search for text planning. *Natural Language Generation*.

Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

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

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 311–318.

Perez-Beltrachini, L., and Lapata, M. 2018. Bootstrapping generators from noisy data. *arXiv preprint arXiv:1804.06385*.

Reiter, E., and Dale, R. 1997. Building applied natural language generation systems. *Nat. Lang. Eng.* 3(1):57–87.

Reiter, E., and Dale, R. 2000. *Building natural language generation systems*. New York, NY: Cambridge University Press.

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

Sha, L.; Mou, L.; Liu, T.; Poupart, P.; Li, S.; Chang, B.; and Sui, Z. 2017. Order-planning neural text generation from structured data. *arXiv preprint arXiv:1709.00155*.

Tanaka-Ishii, K.; Hasida, K.; and Noda, I. 1998. Reactive content selection in the generation of real-time soccer commentary. In *ACL*.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *NIPS*. Curran Associates, Inc. 2692–2700.

Wiseman, S.; Shieber, S.; and Rush, A. 2017. Challenges in data-to-document generation. In *EMNLP*, 2253–2263.

Wong, Y. W., and Mooney, R. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *NAACL*, 172–179.

Yang, Z.; Blunsom, P.; Dyer, C.; and Ling, W. 2017. Reference-aware language models. In *EMNLP*, 1851–1860.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.