Order-Planning Neural Text Generation From Structured Data

Lei Sha[†], Lili Mou[‡], Tianyu Liu[†], Pascal Poupart[‡], Sujian Li[†], Baobao Chang[†], Zhifang Sui[†]
[†]Key Laboratory of Computational Linguistics, Ministry of Education; School of EECS, Peking Univeristy
[‡]David R. Cheriton School of Computer Science, University of Waterloo
[†]{shalei, tianyu0421, lisujian, chbb, szf}@pku.edu.cn
[‡]doublepower.mou@gmail.com, ppoupart@uwaterloo.ca

Abstract

Generating texts from structured data (e.g., a table) is important for various natural language processing tasks such as question answering and dialog systems. In recent studies, researchers use neural language models and encoder-decoder frameworks for table-to-text generation. However, these neural network-based approaches typically do not model the order of content during text generation. When a human writes a summary based on a given table, he or she would probably consider the content order before wording. In this paper, we propose an order-planning text generation model, where order information is explicitly captured by link-based attention. Then a self-adaptive gate combines the link-based attention with traditional content-based attention. We conducted experiments on the WikiBio dataset and achieve higher performance than previous methods in terms of BLEU, ROUGE, and NIST scores; we also performed ablation tests to analyze each component of our model.[1]

Table:

ID	Field	Content
1	Name	*Arthur Ignatius Conan Doyle*
2	Born	*22 May 1859 Edinburgh, Scotland*
3	Died	*7 July 1930 (aged 71) Crowborough, England*
4	Occupation	*Author, writer, physician*
5	Nationality	*British*
6	Alma mater	*University of Edinburgh Medical School*
7	Genre	*Detective fiction fantasy*
8	Notable work	*Stories of Sherlock Homes*

Text: Sir Arthur Ignatius Conan Doyle (22 May 1859 – 7 July 1930) was a British writer best known for his detective fiction featuring the character Sherlock Holmes.

Table 1: An example of a Wikipedia infobox and a reference text.

Introduction

Generating texts from structured data (e.g., a table) is important for various natural language processing tasks such as question answering and dialog systems. Table 1 shows an example of a Wikipedia infobox (containing fields and values) and a text summary.

In early years, text generation was usually accomplished by human-designed rules and templates (Green 2006; Turner, Sripada, and Reiter 2010), and hence the generated texts were not flexible. Recently, researchers applied neural networks to generate texts from structured data (Lebret, Grangier, and Auli 2016), where a neural encoder captures table information and a recurrent neural network (RNN) decodes this information into a natural language sentence.

Although such neural network-based approaches are capable of capturing complicated language and can be trained in an end-to-end fashion, they lack explicit modeling of content order during text generation. That is to say, an RNN generates a word at a time step conditioned on previously generated words as well as table information, which is more or less "shortsighted" and differs from how humans write. As suggested in the book *The Elements of Style*,

> A basic structural design underlies every kind of writing … in most cases, planning must be a deliberate prelude to writing. (William and White 1999)

This motivates order planning for neural text generation. In other words, a neural network should model not only word order (as has been well captured by RNNs), but also the order of content, i.e., fields in a table.

From real summaries, we also observe that table fields by themselves provide illuminating clues and constraints for text generation. In the biography domain, for example, the nationality of a person is typically mentioned before the occupation. This could benefit from explicit planning of content order during neural text generation.

In this paper, we propose an order-planning method for table-to-text generation. Our model is built upon the encoder-decoder framework and uses RNNs for text synthesis with attention to table entries. Different from existing neural models, we design a table field linking mechanism, inspired by temporal memory linkage in the Differentiable Neural Computer (Graves et al. 2016, DNC). Our field linking mechanism explicitly models the relationship between different fields, enabling our neural network to better plan what to say first and what next. Further, we incorporate a copy mechanism (Gu et al. 2016) into our model to cope with rare words.

[1] Code is available at `https://sites.google.com/site/orderplanningnlg/`

We evaluated our order-planning method on the WikiBio dataset (Lebret, Grangier, and Auli 2016). Experimental results show that the proposed approach outperforms previous state-of-the-art results in terms of BLEU, ROUGE, and NIST metrics. Extensive ablation tests verify the effectiveness of each component in our model; we also perform visualization analysis to better understand the proposed order-planning mechanism.

Approach

Our model takes as input a table (e.g., a Wikipedia infobox) and generates a natural language summary describing the information based on an RNN. The neural network contains three main components:

- An encoder captures table information;
- A dispatcher—a hybrid content- and linkage-based attention mechanism over the content of a table—plans what to generate next; and
- A decoder generates a natural language summary using RNN, where we also incorporate a copy mechanism (Gu et al. 2016) to cope with rare words.

We elaborate these components in the rest of this section.

Encoder: Table Representation

We design a neural encoder to represent table information. As shown in Figure 1, the content of each field is split into separate words and the entire table is transformed into a large sequence. Then we use a recurrent neural network (RNN) with long short term memory (LSTM) units (Hochreiter and Schmidhuber 1997) to read the content as well as the corresponding field names.

Concretely, let C be the number of content words in a table; let $\boldsymbol{c}_i$ and $\boldsymbol{f}_i$ be the embeddings of a content and its corresponding field, respectively ($i = 1 \cdots C$). The input of LSTM-RNN is the concatenation of $\boldsymbol{f}_i$ and $\boldsymbol{c}_i$, denoted as $\boldsymbol{x}_i = [\boldsymbol{f}_i; \boldsymbol{c}_i]$, and the output, denoted as $\boldsymbol{h}_i$, is the encoded information corresponding to a content word, i.e.,

$$\begin{aligned} \left[\boldsymbol{g}_{\text{in}}; \boldsymbol{g}_{\text{forget}}; \boldsymbol{g}_{\text{out}}\right] &= \sigma(W_g \boldsymbol{x}_i + U_g \boldsymbol{h}_{i-1}) && (1) \\ \widetilde{\boldsymbol{x}}_i &= \tanh(W_x \boldsymbol{x}_i + U_x \boldsymbol{h}_{i-1}) && (2) \\ \widetilde{\boldsymbol{h}}_i &= \boldsymbol{g}_{\text{in}} \circ \widetilde{\boldsymbol{x}}_i + \boldsymbol{g}_{\text{forget}} \circ \widetilde{\boldsymbol{h}}_{i-1} && (3) \\ \boldsymbol{h}_i &= \boldsymbol{g}_{\text{out}} \circ \tanh(\widetilde{\boldsymbol{h}}_i) && (4) \end{aligned}$$

where $\circ$ denotes element-wise product, and σ denotes the sigmoid function. W's and U's are weights. Bias terms are omitted in the equations for clarity. $\boldsymbol{g}_{\text{in}}$, $\boldsymbol{g}_{\text{forget}}$, and $\boldsymbol{g}_{\text{out}}$ are known as input, forget, and output gates.

Notice that, we have two separate embedding matrices for fields and content words. We observe that the field names of different data samples mostly come from a fixed set of candidates, which is reasonable in a particular domain. Therefore, we assign an embedding to a field, regardless of the number of words in the field name. For example, the field *Notable work* in Table 1 is represented by a single field embedding instead of the embeddings of *notable* and *work*.

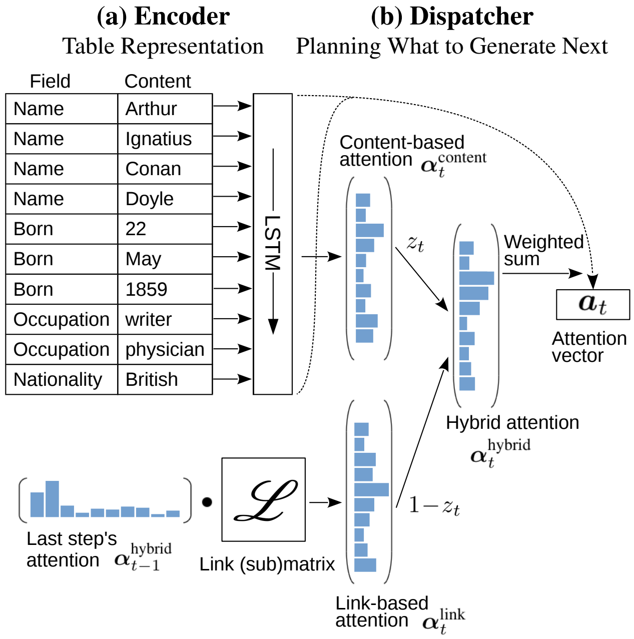

(a) Encoder
Table Representation
(b) Dispatcher
Planning What to Generate Next
Field
Content
Name
Arthur
Name
Ignatius
Name
Conan
Name
Doyle
Born
22
Born
May
Born
1859
Occupation
writer
Occupation
physician
Nationality
British
LSTM
Content-based attention $\boldsymbol{\alpha}_t^{\text{content}}$
z_t
Weighted sum
$\boldsymbol{a}_t$
Attention vector
Hybrid attention $\boldsymbol{\alpha}_t^{\text{hybrid}}$
$1 - z_t$
Last step's attention $\boldsymbol{\alpha}_{t-1}^{\text{hybrid}}$
$\mathscr{L}$
Link (sub)matrix
Link-based attention $\boldsymbol{\alpha}_t^{\text{link}}$

Figure 1: The (a) Encoder and (b) Dispatcher in our model.

For content words, we represent them with conventional word embeddings (which are randomly initialized), and use an LSTM-RNN to summarize the information. In a table, some fields contain a sequence of words (e.g., *Name*="*Arthur Ignatius Conan Doyle*"), whereas other fields contain a set of words (e.g., *Occupation* = "*writer, physician*"). We do not use much human engineering here and let an RNN capture such subtlety by itself.

Dispatcher: Planning What to Generate Next

After encoding table information, we use another RNN to decode a natural language summary (deferred to the next part). During the decoding process, the RNN is augmented with a dispatcher that plans what to generate next.

Generally, a dispatcher is an attention mechanism over table content. At each decoding time step t, the dispatcher computes a probabilistic distribution $\alpha_{t,i}$ ($i = 1 \cdots C$), which is further used for weighting content representations $\boldsymbol{h}_i$. In our model, the dispatcher is a hybrid of content- and link-based attention, discussed in detail as follows.

Content-Based Attention. Traditionally, the computation of attention $\alpha_{t,i}$ is based on the content representation $\boldsymbol{h}_i$ as well as some state during decoding (Bahdanau, Cho, and Bengio 2015; Mei, Bansal, and Walter 2016). We call this *content-based attention*, which is also one component in our dispatcher.

Since both the field name and the content contain important clues for text generation, we compute the attention weights based on not only the encoded vector of table content $\boldsymbol{h}_i$ but also the field embedding $\boldsymbol{f}_i$, thus obtaining the final attention $\alpha_{t,i}^{\text{content}}$ by re-weighting one with the other.

Formally, we have

$$\widetilde{\alpha}_{t,i}^{(f)} = \boldsymbol{f}_i^\top \left(W^{(f)}\boldsymbol{y}_{t-1} + \boldsymbol{b}^{(f)}\right) \tag{5}$$

$$\widetilde{\alpha}_{t,i}^{(c)} = \boldsymbol{h}_i^\top \left(W^{(c)}\boldsymbol{y}_{t-1} + \boldsymbol{b}^{(c)}\right) \tag{6}$$

$$\alpha_{t,i}^{\text{content}} = \frac{\exp\left\{\widetilde{\alpha}_{t,i}^{(f)}\widetilde{\alpha}_{t,i}^{(c)}\right\}}{\sum_{j=1}^{C}\exp\left\{\widetilde{\alpha}_{t,j}^{(f)}\widetilde{\alpha}_{t,j}^{(c)}\right\}} \tag{7}$$

where $W^{(f)}, \boldsymbol{b}^{(f)}, W^{(c)}, \boldsymbol{b}^{(c)}$ are learnable parameters; $\boldsymbol{f}_i$ and $\boldsymbol{h}_i$ are vector representations of the field name and encoded content, respectively, for the ith row. $\alpha_{t,i}^{\text{content}}$ is the content-based attention weights. Ideally, a larger content-based attention indicates a more relevant content to the last generated word.

Link-Based Attention. We further propose a *link-based attention* mechanism that directly models the relationship between different fields.

Our intuition stems from the observation that, a well-organized text typically has a reasonable order of its content. As illustrated previously, the nationality of a person is often mentioned before his occupation (e.g., *a British writer*). Therefore, we propose a link-based attention to explicitly model such information.

We construct a link matrix $\mathscr{L} \in \mathbb{R}^{n_f \times n_f}$, where n_f is the number of possible field names in the dataset. An element $\mathscr{L}[f_j, f_i]$ is a real-valued score indicating how likely the field f_j is mentioned after the field f_i. (Here, $[\cdot,\cdot]$ indexes a matrix.) The link matrix $\mathscr{L}$ is part of the model parameters and learned by backpropagation. Although the link matrix appears to be large in size (1475×1475), a large number of its elements are not used because most fields do not co-occur in at least one data sample; in total, we have 53,422 effective parameters here. In other applications where the link matrix is dense, low-rank approximations may be used to reduce the number of parameters.

Formally, let $\alpha_{t-1,i}$ ($i = 1 \dots C$) be an attention probability[2] over table content in the last time step during generation. For a particular data sample whose content words are of fields $f_1, f_2, \cdots, f_C$, we first weight the linking scores by the previous attention probability, and then normalize the weighted score to obtain link-based attention probability, given by

$$\alpha_{t,i}^{\text{link}} = \text{softmax}\left\{\sum_{j=1}^{C}\alpha_{t-1,j}\cdot\mathscr{L}[f_j, f_i]\right\} \tag{8}$$

$$= \frac{\exp\left\{\sum_{j=1}^{C}\alpha_{t-1,j}\cdot\mathscr{L}[f_j, f_{i'}]\right\}}{\sum_{i'=1}^{C}\exp\left\{\sum_{j}\alpha_{t-1,j}\cdot\mathscr{L}[f_j, f_{i'}]\right\}} \tag{9}$$

Intuitively, the link matrix is analogous to the transition matrix in a Markov chain (Karlin 2014), whereas the term $\sum_{j=1}^{C}\alpha_{t-1,j}\cdot\mathscr{L}[f_j, f_i]$ is similar to one step of transition in the Markov chain. However, in our scenario, a table in a particular data sample contains only a few fields, but a field may occur several times because it contains more than one content word. Therefore, we do not require our link matrix $\mathscr{L}$ to be a probabilistic distribution in each row, but normalize the probabilities afterwards by Equation 9, which turns out to work well empirically.

Besides, we would like to point out that the link-based attention is inspired by the Differentiable Neural Computer (Graves et al. 2016, DNC). DNC contains a "linkage-based addressing" mechanism to track consecutively used memory slots and thus to integrate order information during memory addressing. Likewise, we design the link-based attention to capture the temporal order of different fields. But different from the linking strength heuristically defined in DNC, the link matrix in our model is directly parameterized and trained in an end-to-end manner.

Hybrid Attention. To combine the above two attention mechanisms, we use a self-adaptive gate $z_t \in (0, 1)$ represented by a sigmoid unit

$$z_t = \sigma\left(\boldsymbol{w}^\top[\boldsymbol{h}'_{t-1}; \boldsymbol{e}_t^{(f)}; \boldsymbol{y}_{t-1}]\right) \tag{10}$$

where $\boldsymbol{w}$ is a parameter vector. $\boldsymbol{h}'_{t-1}$ is the last step's hidden state of the decoder RNN. $\boldsymbol{y}_{t-1}$ is the embedding of the word generated in the last step; $\boldsymbol{e}_t^{(f)}$ is the sum of field embeddings $\boldsymbol{f}_i$ weighted by the current step's field attention $\alpha_{t,i}^{\text{link}}$. As $\boldsymbol{y}_{t-1}$ and $\boldsymbol{e}_t^{(f)}$ emphasize the content and link aspects, respectively, the self-adaptive gate z is aware of both. In practice, we find z tends to address link-based attention too much[3] and thus adjust it by $\widetilde{z}_t = 0.2z_t + 0.5$ empirically.

Finally, the hybrid attention, a probabilistic distribution over all content words, is given by

$$\boldsymbol{\alpha}_t^{\text{hybrid}} = \widetilde{z}_t \cdot \boldsymbol{\alpha}_t^{\text{content}} + (1 - \widetilde{z}_t)\cdot\boldsymbol{\alpha}_t^{\text{link}} \tag{11}$$

Decoder: Sentence Generation

The decoder is an LSTM-RNN that predicts target words in sequence. We also have an attention mechanism (Bahdanau, Cho, and Bengio 2015) that summarizes source information, i.e., the table in our scenario, by weighted sum, yielding an attention vector $\boldsymbol{a}_t$ by

$$\boldsymbol{a}_t = \sum_{i=1}^{C}\alpha_{t,i}^{\text{hybrid}}\boldsymbol{h}_i \tag{12}$$

where $\boldsymbol{h}_i$ is the hidden representation obtained by the table encoder. As $\alpha_{t,i}^{\text{hybrid}}$ is a probabilistic distribution—determined by both content and link information—over content words, it enables the decoder RNN to focus on relevant information at a time, serving as an order-planning mechanism for table-to-text generation.

[2] Here, $\alpha_{t-1,i}$ uses the hybrid content- and link-based attention, which will be introduced shortly.

[3] One author of this paper has similar experience in another application (Meng, Mou, and Jin 2017), where combining content and temporal information (analogous to order information in this paper) in a naïve fashion does not yield high performance. Our conjecture is that, if we directly use Equation 10, the link-based attention is much simpler, and hence easier to train, than content-based attention, thus bypassing content information. More in-depth analysis is needed as future work.

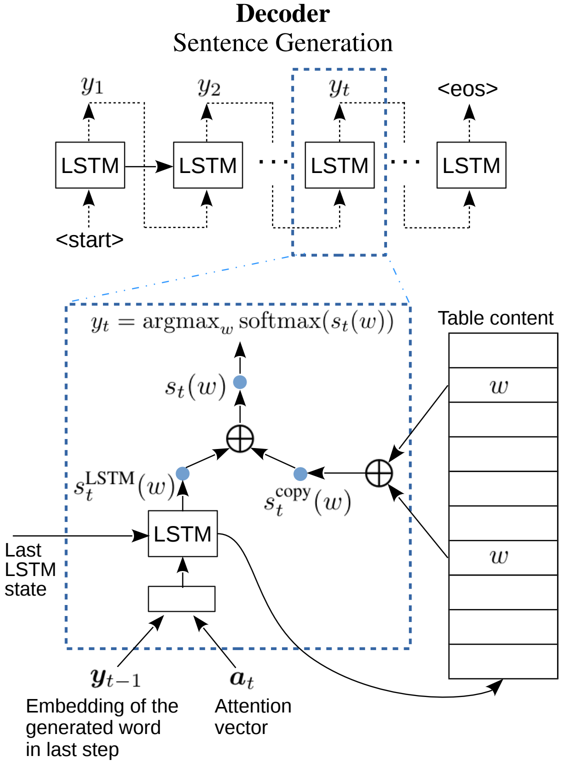


Figure 2: The decoder RNN in our model, which is enhanced with a copy mechanism.

Then we concatenate the attention vector $\boldsymbol{a}_t$ and the embedding of the last step's generated word $\boldsymbol{y}_{t-1}$, and use a single-layer neural network to mix information before feeding to the decoder RNN. In other words, the decoder RNN's input (denoted as $\boldsymbol{x}_t$) is

$$\boldsymbol{x}_t = \tanh(W_d[\boldsymbol{a}_t; \boldsymbol{y}_{t-1}] + \boldsymbol{b}_d) \tag{13}$$

where W_d and b_d are weights. Similar to Equations 1–4, at a time step t during decoding, the decoder RNN yields a hidden representation $\boldsymbol{h}'_t$, based on which a score function $\boldsymbol{s}^{\text{LSTM}}$ is computed suggesting the next word to generate. The score function is given by

$$\boldsymbol{s}_t^{\text{LSTM}} = W_s \boldsymbol{h}'_t + \boldsymbol{b}_s \tag{14}$$

where $\boldsymbol{h}'_t$ is the decoder RNN's state. (W_s and $\boldsymbol{b}_s$ are weights.) The score function can be thought of as the input of a softmax layer for classification before being normalized to a probabilistic distribution. We incorporate a copy mechanism (Gu et al. 2016) into our approach, and the normalization is accomplished after considering a copying score, introduced as follows.

Copy Mechanism. The copy mechanism scores a content word c_i by its hidden representation $\boldsymbol{h}_i$ in the encoder side, indicating how likely the content word c_i is directly copied during target generation. That is,

$$s_{t,i} = \sigma(\boldsymbol{h}_i^\top W_c)\boldsymbol{h}'_t \tag{15}$$

and $s_{t,i}$ is a real number for $i = 1, \cdots, C$ (the number of content words). Here W_c is a parameter matrix, and $\boldsymbol{h}'$ is the decoding state.

In other words, when a word appears in the table content, it has a copying score computed as above. If a word w occurs multiple times in the table content, the scores are added as follows

$$s_t^{\text{copy}}(w) = \sum_{i=1}^{C} s_{t,i} \cdot \mathbb{1}_{\{c_i = w\}} \tag{16}$$

where $\mathbb{1}_{\{c_i = w\}}$ is a Boolean variable indicating whether the content word c_i is the same as the word w we are considering.

Finally, the LSTM score and the copy score are added for a particular word and further normalized to obtain a probabilistic distribution over candidate words, given by

$$s_t(w) = s_t^{\text{LSTM}}(w) + s_t^{\text{copy}}(w) \tag{17}$$

$$p_t(w) = \mathrm{softmax}\left(s_t(w)\right) = \frac{\exp\{s_t(w)\}}{\sum_{w' \in \mathcal{V} \bigcup \mathcal{C}} \exp\{s_t(w')\}} \tag{18}$$

where $\mathcal{V}$ refers to the vocabulary list and $\mathcal{C}$ refers to the set of content words in a particular data sample. In this way, the copy mechanism can either generate a word from the vocabulary or directly copy a word from the source side. This is helpful in our scenario because some fields in a table (e.g., *Name*) may contain rare or unseen words and the copy mechanism can cope with them naturally.

Training and Inference

Our training objective is the negative log-likelihood of a sentence $y_1 \cdots y_T$ in the training set.

$$J = -\sum_{t=1}^{T} \log p(y_t | y_0 \cdots y_{t-1}) \tag{19}$$

where $p(y_t|\cdot)$ is computed by Equation 18. An ℓ_2 penalty is also added as in most other studies.

Since all the components described above are differentiable, our entire model can be trained end-to-end by backpropagation.

During inference, we use greedy search for simplicity, i.e., for each time step t, the word with the largest probability is chosen, given by $y_t = \mathrm{argmax}_w \, p_t(w)$. The decoding process terminates when a special symbol <eos> is generated, indicating the end of a sequence.

Experiments

Dataset

We used the newly published WikiBio dataset (Lebret, Grangier, and Auli 2016),[4] which contains 728,321 biographies from WikiProject Biography[5] (originally from English Wikipedia, September 2015).

[4] https://github.com/DavidGrangier/wikipedia-biography-dataset

[5] https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Biography

Group	Model	BLEU	ROUGE	NIST
Previous results	KN	2.21	0.38	0.93
	Template KN	19.80	10.70	5.19
	Table NLM^l	34.70	25.80	7.98
Our results	Content attention only	41.38	34.65	8.57
	Order planning (full model)	**43.91**	**37.15**	**8.85**

Table 2: Comparison of the overall performance between our model and previous methods. ^{l}Best results in Lebret, Grangier, and Auli (2016).

Each data sample comprises an infobox table of field-content pairs, being the input of our system. The generation target is the first sentence in the biography, which follows the setting in previous work (Lebret, Grangier, and Auli 2016). Although only the first sentence is considered in the experiment, the sentence typically serves as a summary of the article. In fact, the target sentence has 26.1 tokens on average, which is actually long. Also, the sentence contains information spanning multiple fields, and hence our order-planning mechanism is useful in this scenario.

We applied the standard data split: 80% for training and 10% for testing, except that model selection was performed on a validaton subset of 1000 samples (based on BLEU-4).

Settings

We decapitalized all words and kept a vocabulary size of 20,000 for content words and generation candidates, which also followed previous work (Lebret, Grangier, and Auli 2016). Even with this reasonably large vocabulary size, we had more than 900k out-of-vocabulary words. This rationalizes the use of the copy mechanism.

For the names of table fields, we treated each field name as a special token. By removing non-text fields whose content is "none" (e.g., *Image*) and grouping fields occurring less than 100 times as an "Unknown" field, we had 1475 different field names in total.

In our experiments, both words' and table fields' embeddings were 400-dimensional and LSTM layers were 500-dimensional. Notice that, a field (e.g., "name") and a content/generation word (e.g., also "name"), even with the same string, were considered as different tokens; hence, they had different embeddings. We randomly initialized all embeddings, which were tuned during training.

We used Adam (Kingma and Ba 2015) as the optimization algorithm with a batch size of 32; other hyperparameters were set to default values.

Baselines

We compared our model with previous results using either traditional language models or neural networks.

- KN and Template KN (Heafield et al. 2013): Lebret, Grangier, and Auli (2016) train an interpolated Kneser-Ney (KN) language model for comparison with the KenLM toolkit. They also train a KN language model with templates.

Component	BLEU	ROUGE	NIST
Content att.	41.38	34.65	8.57
Link att.	38.24	32.75	8.36
Hybrid att.	43.01	36.91	8.75
Copy+Content att.	41.89	34.93	8.63
Copy+Link att.	39.08	33.47	8.42
Copy+Hybrid att.	**43.91**	**37.15**	**8.85**

Table 3: Ablation test.

- Table NLM: Lebret, Grangier, and Auli (2016) propose an RNN-based model with attention and copy mechanisms. They have several model variants, and we quote the highest reported results.

We report model performance in terms of several metrics, namely BLEU-4, ROUGE-4, and NIST-4, which were computed by standard software, NIST mteval-v13a.pl (for BLEU and NIST) and MSR rouge-1.5.5 (for ROUGE). We did not include the perplexity measure from Lebret, Grangier, and Auli (2016) because the copy mechanism makes the vocabulary size vary among data samples, and thus the perplexity is not comparable among different approaches.

Results

Overall Performance. Table 2 compares the overall performance with previous work. We see that, modern neural networks are considerably better than traditional KN models with or without templates. Moreover, our base model (with content-attention only) outperforms Lebret, Grangier, and Auli (2016), showing our better engineering efforts. After adding up all proposed components, we obtain +2.5 BLEU and ROUGE improvement and +0.3 NIST improvement, achieving new state-of-the-art results.

Ablation Test. Table 3 provides an extensive ablation test to verify the effectiveness of each component in our model. The top half of the table shows the results without the copy mechanism, and the bottom half incorporates the copying score as described previously. We observe that the copy mechasnim is consistently effective with different types of attention.

We then compare content-based attention and link-based attention, as well as their hybrid (also Table 3). The results show that, link-based attention alone is not as effective as

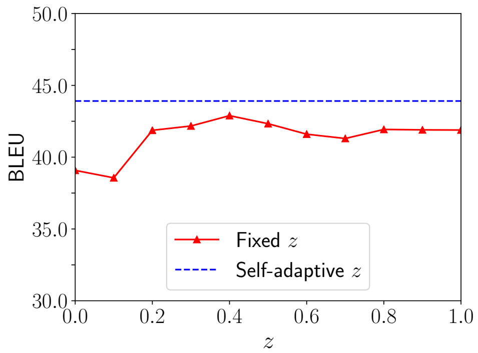


Figure 3: Comparing the self-adaptive gate with interpolation of content- and link-based attention. $z = 0$ is link-based attention, $z = 1$ is content-based attention.

content-based attention. However, we achieve better performance if combining them together with an adaptive gate, i.e., the proposed hybrid attention. The results are consistent in both halves of Table 3 (with or without copying) and in terms of all metrics (BLEU, ROUGE, and NIST). This implies that content-based attention and link-based attention do capture different aspects of information, and their hybrid is more suited to the task of table-to-text generation.

Effect of the gate. We are further interested in the effect of the gate z, which balances content-based attention $\boldsymbol{\alpha}^{\text{content}}$ and link-based attention $\boldsymbol{\alpha}^{\text{link}}$. As defined in Equation 11, the computation of z depends on the decoding state as well as table information; hence it is "self-adaptive." We would like to verify if such adaptiveness is useful, and thus designed a controlled experiment where the gate z was manually assigned in advance and fixed during training. In other words, the setting was essentially a (fixed) interpolation between $\boldsymbol{\alpha}^{\text{content}}$ and $\boldsymbol{\alpha}^{\text{link}}$. Specifically, we tuned z from 0 to 1 with a granularity of 0.1, and plot BLEU scores as the comparison metric in Figure 3.

As seen, interpolation of content- and link-based attention is generally better than either single mechanism, which again shows the effectiveness of hybrid attention. However, the peak performance of simple interpolation (42.89 BLEU when $z = 0.4$) is worse than the self-adaptive gate, implying that our gating mechanism can automatically adjust the importance of $\boldsymbol{\alpha}^{\text{content}}$ and $\boldsymbol{\alpha}^{\text{link}}$ at a particular time based on the current state and input.

Different Ways of Using Field Information. We are curious whether the proposed order-planning mechanism is better than other possible ways of using field information. We conducted two controlled experiments as follows. Similar to the proposed approach, we multiplied the attention probability by a field matrix and thus obtained a weighted field embedding. We fed it to either (1) the computation of content-based attention, i.e., Equations 5–6, or (2) the RNN decoder's input, i.e., Equation 13. In both cases, the last step's weighted field embedding was concatenated with the embedding of the generated word $\boldsymbol{y}_{t-1}$.

Feeding field info to . . .	**BLEU**	**ROUGE**	**NIST**
None	41.89	34.93	8.63
Computation of $\boldsymbol{\alpha}^{\text{content}}$	40.52	34.95	8.57
Decoder RNN's input	41.96	35.07	8.61
Hybrid att. (proposed)	**43.91**	**37.15**	**8.85**

Table 4: Comparing different possible ways of using field information. "None": No field information is fed back to the network, i.e., content-based attention computed by Equation 7 (with copying).

From Table 4, we see that feeding field information to the computation of $\boldsymbol{\alpha}^{\text{content}}$ interferes content attention and leads to some performance degradation, and that feeding it to decoder RNN slightly improves model performance. However, both controlled experiments are worse than the proposed method. The results confirm that our order-planning mechanism is indeed useful in modeling the order of fields, outperforming several other approaches that use the same field information in a naïve fashion.

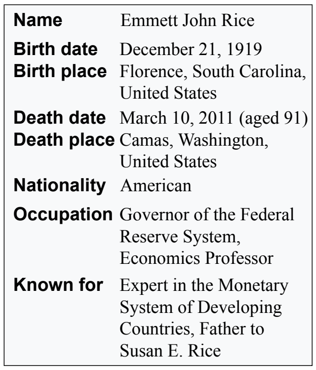
Case Study and Visualization

We showcase an example in Table 5. With only content-based attention, the network is confused about when the word *American* is appropriate in the sentence, and corrupts the phrase *former governor of the federal reserve system* that appears in the reference. However, when link-based attention is added, the network is more aware of a natural order between fields "Nationality" and "Occupation" (although necessarily appearing in the particular reference), and generates the nationality *American* before the occupation *economist*. This process can also be visualized in Figure 4. Here, we plot our model's content-based attention, link-based attention, and their hybrid. (The content- and link-based attention probabilities may be different from those separately trained in the ablation test.) After generating "*emmett john rice (december 21, 1919 – march 10, 2011) was*," content-based attention skips the nationality and focuses more on the occupation. Link-based attention, on the other hand, provides a strong clue suggesting to generate the nationality first and then occupation.

Related Work

Text generation has long aroused interest in the NLP community due to is wide applications including automated navigation (Dale, Geldof, and Prost 2003) and weather forecasting (Reiter et al. 2005). Traditionally, text generation can be divided into several steps (Stent, Prassad, and Walker 2004): (1) *content planning* defines what information should be conveyed in the generated sentence; (2) *sentence planning* determines what to generate in each sentence; and (3) *surface realization* actually generates those sentences with words.

In early years, surface realization was often accomplished by templates (Van Deemter, Theune, and Krahmer 2005) or statistically learned (shallow) models, e.g., probabilistic context-free grammars (Belz 2008) and language mod-



Name	Emmett John Rice
Birth date	December 21, 1919
Birth place	Florence, South Carolina, United States
Death date	March 10, 2011 (aged 91)
Death place	Camas, Washington, United States
Nationality	American
Occupation	Governor of the Federal Reserve System, Economics Professor
Known for	Expert in the Monetary System of Developing Countries, Father to Susan E. Rice

Reference	emmett john rice (december 21 , 1919 – march 10 , 2011) was a former governor of the federal reserve system , a Cornell university economics professor , expert in the monetary systems of developing countries and the father of the current national security advisor to president barack obama , susan e . rice .
Content-based attention	emmett john rice (december 21 , 1919 – march 10 , 2011) was an economist , author , public official and the former american governor of the federal reserve system , the first african american UNK .
Hybrid attention	emmett john rice (december 21 , 1919 – march 10 , 2011) was an american economist , author , public official and the former governor of the federal reserve system , expert in the monetary systems of developing countries .

Table 5: Case study. **Left**: Wikipedia infobox. **Right**: A reference and two generated sentences by different attention (both with the copy mechanism).

els (Angeli, Liang, and Klein 2010), with hand-crafted features or rules. Therefore, these methods are weak in terms of the quality of generated sentences. For planning, researchers also apply (shallow) machine learning approaches. Barzilay and Lapata (2005), for example, model it as a collective classification problem, whereas Liang, Jordan, and Klein (2009) use a generative semi-Markov model to align text segments and assigned meanings. Generally, planning and realization in the above work are separate and have difficulty in capturing the complexity of language due to the nature of shallow models.

Recurrent neural networks (RNNs) are now playing a more important role in natural language generation. As RNNs can automatically capture highly complicated patterns during end-to-end training, they have successful applications including machine translation (Bahdanau, Cho, and Bengio 2015), dialog systems (Shang, Lu, and Li 2015), and text summarization (Tan, Wan, and Xiao 2017).

Researchers are also beginning to use RNNs for text generation from structured data. Mei, Bansal, and Walter (2016) propose a coarse-to-fine grained attention mechanism that selects one or more records (e.g., a piece of weather forecast) by a precomputed but fixed probability and then dynamically attends to relevant content during decoding. Lebret, Grangier, and Auli (2016) incorporate the copy mechanism (Gu et al. 2016) into the generation process. However, the above approaches do not explicitly model the order of content. It is also nontrivial to combine traditional planning techniques to such end-to-end learned RNN.

Our paper proposes an order-planning approach by designing a hybrid of content- and link-based attention. The model is inspired by hybrid content- and location-based addressing in the Differentiable Neural Computer (Graves et al. 2016), where the location-based addressing is defined heuristically. Our model is also similar to a hybrid content and temporal analysis of multi-party conversations in one author's previous work (Meng, Mou, and Jin 2017); however, the temporal information in Meng, Mou, and Jin (2017) is modeled by a vector. Instead, this paper proposes a transition-like link matrix that models how likely a field is mentioned after another, which is more suited in our scenario. Our entire model is differentiable, and thus the *planning* and *realization* steps in traditional language generation can be learned end-to-end.

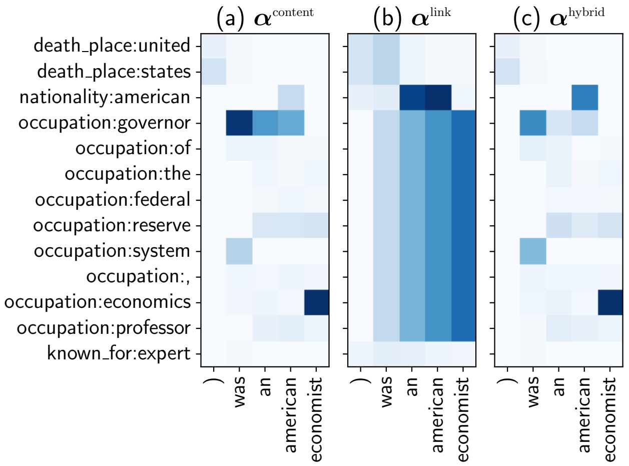


Figure 4: Visualization of attention probabilities in our model. x-axis: generated words "...) was an american economist ..."; y-axis: ⟨field : content word⟩ pairs in the table. (a) Content-based attention. (b) Link-based attention. (c) Hybrid attention. Subplot (b) exhibits strips because, by definition, link-based attention will yield the same score for all content words with the same field. Please also note that the columns do not sum to 1 in the figure because we only plot a part of the attention probabilities.

Conclusion

In this paper, we propose an order-planning neural network that generates text from a table (Wikipedia infobox). The text generation process is built upon an RNN with attention to table content. Different from traditional content-based attention, we explicitly model the order of content by a link matrix, based on which we compute a link-based attention. Then a self-adaptive gate balances the content- and link-based attention mechanisms. We further incorporate a copy mechanism to our model to cope with rare or unseen words.

We evaluated our approach on a newly proposed large-scale dataset, WikiBio. Experimental results show that we outperform previous results in terms of BLEU, ROUGE, and NIST scores. We also performed extensive ablation tests showing the effectiveness of the copy mechanism, as well as the hybrid attention of content and linking information. We compared our order-planning mechanism with other possible ways of modeling fields; the results confirm that the proposed method is better than feeding field embedding to the network in a naïve fashion. Finally we provide a case study and visualize the attention scores so as to better understand our model.

Acknowledgments

We thank Jing He from AdeptMind.ai for helpful discussions on different ways of using field information. We also thank reviewers for helpful advice. The research is supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002101, and the National Natural Science Foundation of China under Grant No. 61772040. The contact authors are Baobao Chang and Zhifang Sui.

References

Angeli, G.; Liang, P.; and Klein, D. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 502–512.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Barzilay, R., and Lapata, M. 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*, 331–338.

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

Dale, R.; Geldof, S.; and Prost, J.-P. 2003. CORAL: Using natural language generation for navigational assistance. In *Proceedings of the 26th Australasian Computer Science Conference*, volume 16, 35–44.

Graves, A.; Wayne, G.; Reynolds, M.; et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471–476.

Green, N. 2006. Generation of biomedical arguments for lay readers. In *Proceedings of the 4th International Natural Language Generation Conference*, 114–121.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 1631–1640.

Heafield, K.; Pouzyrevsky, I.; Clark, J. H.; and Koehn, P. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 2, 690–696.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Karlin, S. 2014. *A First Course in Stochastic Processes*. Academic Press.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Lebret, R.; Grangier, D.; and Auli, M. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1203–1213.

Liang, P.; Jordan, M. I.; and Klein, D. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 91–99.

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

Meng, Z.; Mou, L.; and Jin, Z. 2017. Towards neural speaker modeling in multi-party conversation: The task, dataset, and models. *arXiv preprint arXiv:1708.03152*.

Reiter, E.; Sripada, S.; Hunter, J.; Yu, J.; and Davy, I. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167(1-2):137–169.

Shang, L.; Lu, Z.; and Li, H. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 1577–1586.

Stent, A.; Prassad, R.; and Walker, M. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, 79–86.

Tan, J.; Wan, X.; and Xiao, J. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 1171–1181.

Turner, R.; Sripada, S.; and Reiter, E. 2010. Generating approximate geographic descriptions. In *Empirical Methods in Natural Language Generation*, 121–140.

Van Deemter, K.; Theune, M.; and Krahmer, E. 2005. Real versus template-based natural language generation: A false opposition? *Computational Linguistics* 31(1):15–24.

William, S., and White, E. B. 1999. *The Element of Style*. Pearson, 4th edition.