Point Precisely: Towards Ensuring the Precision of Data in Generated Texts Using Delayed Copy Mechanism

Liunian Li **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{liliunian, wanxiaojun}@pku.edu.cn

Abstract

The task of data-to-text generation aims to generate descriptive texts conditioned on a number of database records, and recent neural models have shown significant progress on this task. The attention based encoder-decoder models with copy mechanism have achieved state-of-the-art results on a few data-to-text datasets. However, such models still face the problem of putting incorrect data records in the generated texts, especially on some more challenging datasets like ROTOWIRE. In this paper, we propose a two-stage approach with a delayed copy mechanism to improve the precision of data records in the generated texts. Our approach first adopts an encoder-decoder model to generate a template text with data slots to be filled and then leverages a proposed delayed copy mechanism to fill in the slots with proper data records. Our delayed copy mechanism can take into account all the information of the input data records and the full generated template text by using double attention, position-aware attention and a pairwise ranking loss. The two models in the two stages are trained separately. Evaluation results on the ROTOWIRE dataset verify the efficacy of our proposed approach to generate better templates and copy data records more precisely.

1 Introduction

One important task in the natural language generation (NLG) area is to generate a natural language description for some structured data (i.e., a number of database records), a.k.a., the data-to-text generation task. Applicable to wide areas such as weather forecasting, financial reporting, game broadcasting, data-to-text generation systems have been studied for decades (Gatt and Krahmer, 2018). Traditionally this task is addressed by using templates or statistically learned shallow models with hand-crafted features (van Deemter et al., 2005; Belz, 2008; Konstas and Lapata, 2012). Recently, neural generation systems have shown significant progress on this task. The attention based encoder-decoder models with the copy mechanism have achieved state-of-the-art results on a few data-to-text datasets (Wiseman et al., 2017; Sha et al., 2017).

In this paper, we try to address the problem of faithfully describing the data in the generated texts, i.e., how to generate a descriptive text that contains correct entities and numeric values. Intuitively speaking, when a human writer writes weather forecasts or game summaries, he may not consider the exact entities or numbers right away. Rather, he may have finished the whole sentence (without the actual entities or numbers) in his mind, then look up the data table and fill in those exact entities and numbers at last.

Likewise, we don't have to train our models to construct sentences and fill in the data at the same time. If we take out the entities and numeric values from the text, what is left can be considered some kind of templates with data slots waiting to be filled in. Therefore, the process of generating descriptions can be naturally divided into two stages: template generation and slot filling. An encoder-decoder based generator can be in charge of generating the templates with entities and numeric values left blank. A delayed copy network can be in charge of filling in those data slots with proper data records. Compared to traditional copy mechanisms, our approach has the following advantages:

- Traditional copy mechanisms generate and copy words at the same time, which involves merging a score distribution over the input words and a score distribution over the words in the vocabulary. It remains unclear exactly how different merging strategies affect performance. We avoid this uncertainty by leaving the task of generating words to the template generator, the task of deciding which word to copy to the delayed copy network. The two-stage approach also allows us to use a more effective pairwise ranking loss function.

- Relieved of the burden of having to both generate and copy words, the template generator can now concentrate on generating better templates. This can be addressed by using a typical encoder-decoder model, which has proven to be easy to train and tune.

- The delayed copy mechanism can now concentrate on slot filling, and it can utilize all the information in the input data records and the full generated template text by using double attention and position-aware attention, making the delayed copy mechanism point more precisely.

We evaluate our approach on the public ROTOWIRE dataset and evaluation results verify the efficacy of our approach, which makes a boost on the data precision of generated texts and brings a noticeable increase on BLEU score.

We organize the paper as follows. We introduce some background knowledge in Section 2 and describe our approach in Section 3. In Section 4 we present the experiments and have discussion. In Section 5 we note some additional related works. In Section 6 we conclude this paper.

2 Background

The Data-to-text Task Following the notations in Liang et al. (2009) and Wiseman et al. (2017), let $S = \{r_j\}$ be a set of records, where for each $r \in S$, $r.type$, $r.entity$ and $r.value$ are the record's type, entity and value, respectively. For example, there could be a record r in a basketball dataset such that $r.type$ = POINTS FIRST QUARTER, $r.entity$ = LEBRON JAMES, and $r.value = 10$, which means LeBron James scored 10 points in the first quarter. From these records, we are interested in generating an descriptive summary $\hat{\boldsymbol{y}} = \hat{y}_1, \ldots, \hat{y}_T$ of T words. A data-to-text dataset consists of pairs like $(S, \boldsymbol{y})$, where $\boldsymbol{y}$ is a gold (human generated) summary for the record set S. In this paper, we develop our system on the ROTOWIRE dataset (Wiseman et al., 2017), which consists of documents summarizing NBA basketball games and the corresponding game data. For every entity that appears in S, we add an additional record r^* to S, with $r^*.entity$ and the $r^*.value$ both set to the entity's name. For example, if LEBRON JAMES appears in the game, a record r^* with $r^*.entity$ = LEBRON JAMES, $r^*.value$ = LEBRON JAMES, $r^*.type$ = PLAYER NAME is added to S. This aims to copy entity in a neat and uniform way, which will be discussed later.

If a record is mentioned in the text, we consider the text as "faithfully describing the data" if the sentence in which $r.entity$ and $r.value$ are both present correctly reflects $r.type$. For example, for the fore-mentioned record r, if the sentence is "LeBron James got 10 rebounds in the second quarter", it is considered unfaithful to the record. Instead, a sentence correctly reflecting the record should be "LeBron James scored 10 points in the first quarter." Generating texts conditioned on the records is a non-trivial problem because a sentence could likely contain several records. However, there have been few works explicitly addressing this issue due to the lack of challenging datasets. Previous datasets (Liang et al., 2009; Chen and Mooney, 2008; Murakami et al., 2017) generally use relatively simple language and record structure, where some simple approaches suffice. For instance, some models (Mei et al., 2016) generate entities and numeric values as other normal words while some works (van Deemter et al., 2005; Liang et al., 2009; Murakami et al., 2017) employ a *"tag-then-replace"* method, where they train their models to generate special tags which will be replaced with true entities or numeric values using hand-written rules. However, these approaches may fail under more complicated situations. For example, in the ROTOWIRE dataset, there are over 40 *type*s of records and the entity names that appear in the *entity* portion of records change from game to game, making it hard, if not impossible, to devise hand-written rules for the *"tag-then-replace"* method.

Copy Mechanism Copy mechanism in encoder-decoder models (Vinyals et al., 2015; Gu et al., 2016; Gülçehre et al., 2016; Yang et al., 2017) provides a way to directly copy words from the input. At each time step of decoding, an attention-like function is used to produce an unnormalized score distribution over the inputs, denoted as P^*_{copy}. The decoder also produces an unnormalized distribution P^*_{word} over the words in the vocabulary. These two unnormalized score distributions are then merged into a normalized distribution P_{joint}. The loss function is defined as the negative log likelihood function of vanilla encoder-decoder models: $L = \sum_t - \log P_{joint}(w^*_t | w_{1:t-1}, S)$, where w^*_t is the target word at time step t, $w_{1:t-1}$ are previous words, and S is the inputs. We now illustrate how copy mechanism is used in this task. A copy means the $value$ portion (could be a numeric value or an entity name) of a record r is copied into the output text. To train a copy network, one has to specify which token in the output text is copied and from which input record it is copied. For some token y_t in the gold text, we assume y_t to be copied from some input record r only if the following constraint is met: $y_t = r.value$ and $r.entity$ is present in the same sentence with y_t. Therefore, for every token y_t, if we can find a non-empty set of records C, where every $r \in C$ meets the constraint, y_t is considered to be copied, and C is the set of valid records that is the source of y_t.

3 Our Approach

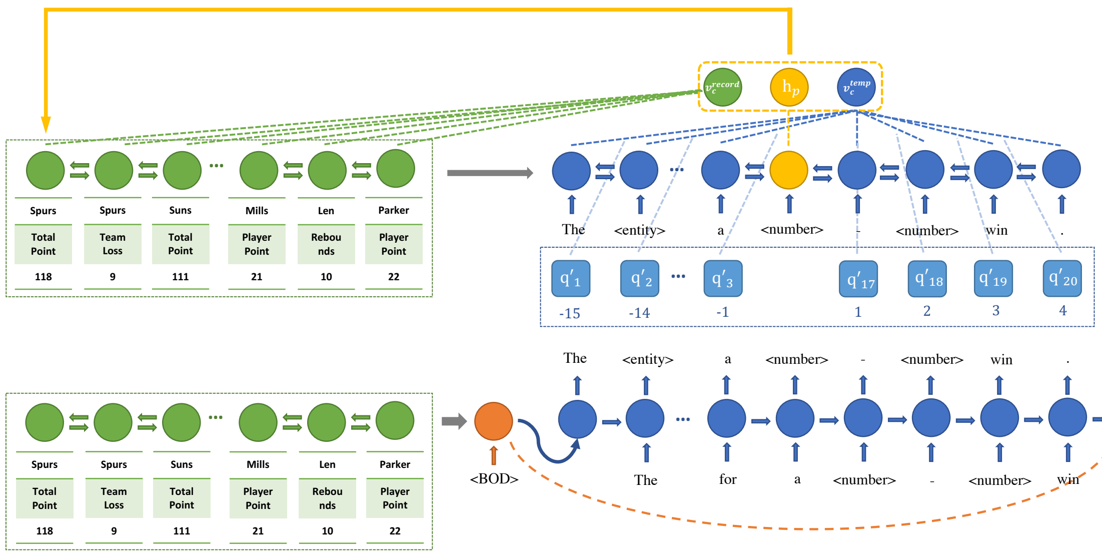

Figure 1: Our Framework: The template generator (below) generates templates and then the delayed copy network (above) fills in the slots.

Our system consists of two models, an encoder-decoder based template generator that constructs the whole sentence with data slots left blank and a delayed copy network that fills in those data slots informed by both the input and the generated sentence. From a computational perspective, this two-stage method has many practical advantages. The template generator is freed from the burden of having to both generate and copy words while the copy network could benefit from utilizing the information from the full templates. This is somewhat similar to Xia et al. (2017), where they observed performing a second decoding using information from the first decoding could boost performance. In addition, detaching the copy network from the text decoder allows room for alternative loss functions, opening up a new possibility for our model. We divided the task in a neat and intuitive way, avoiding the uncertainty in the merging procedure in traditional copy mechanisms. At last, we can reduce training time by training the template generator and the copy network concurrently.

Original Text: The San Antonio Spurs (47 - 9) held off the Phoenix Suns (14 - 42) for a 118 - 111 win. Spurs 's three - point shooting was the key , as they went 10 - for - 16 from long distance in the victory .
Target Template: The <entity> (<number> - <number>) held off the <entity> (<number> - <number>) for a <number> - <number> win. <entity> 's three - point shooting was the key , as they went <number> - for - <number> from long distance in the victory .

Table 1: An example of the original text and the corresponding target template.

3.1 Template Generation with Encoder-Decoder Model

Overview The template generator generates the major part of the text while leaving the task of actually copying to the delayed copy network. It is essentially an encoder-decoder model and any well-performing model would suffice. In this work, we use a hierarchical model similar to the one in Li et al. (2015).

While the template generator does not decide which input record to copy from, it is responsible for deciding whether to copy a token from the input records. At each time step t, if the template generator finds it necessary to copy a token, it outputs a special placeholder (a data slot), which will be replaced with actual entities and numeric values by the copy network. The special placeholder could be "<data>". However in this specific task, we make a distinction between entities and numeric values, and define two placeholders, "<entity>" and "<number>". We denote the templates as $\boldsymbol{y'}$. Now instead of learning to generate the original text $\boldsymbol{y}$, the template generator learns to generate $\boldsymbol{y'}$. An example is shown in Table 1.

The placeholders are added to the vocabulary of the template generator. The template generator outputs the placeholders just like other words. In this way, the process of deciding whether to copy a token is incorporated naturally into the text decoding process, cleverly avoiding the merging operation in traditional copy mechanisms. As mentioned in Section 2, traditional copy mechanisms have to merge P^*_{copy} and P^*_{word} into a normalized distribution P_{joint}. This was designed to allow the input records to "compete" with the words in the vocabulary but there is no clear conclusion about how different merging operations affect performance. In this task, we find that the template generator is fully capable of deciding whether to copy or not.

Structure An input encoder Enc_{input} first encodes every record $r \in S$ by embedding $r.type$, $r.entity$ and $r.value$ and applying a 1-layer MLP followed by an LSTM over the embedding vectors, i.e. $\{\tilde{r}_j\} = Enc_{\text{input}}(\{r_j\})$. We also note that in this paper, all embedding vectors and hidden states are in $\mathbb{R}^D$ unless stated otherwise, where D is the dimension. The input encoder is followed by a hierarchical decoder consisting of a sentence decoder Dec_{sent} and a word decoder Dec_{word}. The sentence decoder outputs sentence representations while the word decoder predicts words in a sentence sequentially. The hidden states of the word decoder Dec_{word} are denoted as $\{h^{word}_t\}$. At test time, beam search can be applied to the word decoder of the template generator. For more details, please refer to Li et al. (2015).

3.2 Slot Filling with Delayed Copy Network

Overview Our copy network fills in the data slots (placeholders) in the generated template. For every placeholder in the template $\boldsymbol{y'}$, the delayed copy network takes in the following inputs: an input vector h_p corresponding to the placeholder; a series of input-record-based vectors $\{h^{\text{record}}_j\}$, each corresponding to an input record r; another series of template-text-based vectors $\{h^{\text{temp}}_t\}$, each corresponding to a word in the template sentence that the placeholder is in. After applying the attention mechanism to both the input records and the template text, it generates a score distribution over the input records and selects the record with the highest score to copy from.

Let us assume the placeholder in question is the m-th word in the template sentence. We always let $h_p = h^{\text{temp}}_m$. We now discuss how we can obtain $\{h^{\text{record}}_j\}$ and $\{h^{\text{temp}}_t\}$. There are actually two ways to do this. We can reuse the hidden states from the template generator to get $\{h^{\text{record}}_j\}$ and $\{h^{\text{temp}}_t\}$[1]. Or we

[1]Concretely, the input-record-based vectors are from the encoder, i.e. $\{h^{\text{record}}_j\} = \{\tilde{r}_j\}$; the template-text-based vectors are from the word decoder, i.e. $\{h^{\text{temp}}_t\} = \{h^{word}_t\}$

can fully detach the delayed copy network from the template generator by using a template text encoder Enc_{temp} to get $\{h_t^{\text{temp}}\}$, a separate input encoder Enc_{input}^{*} to get $\{h_j^{\text{record}}\}^2$.

Although in the first method, models can be trained in an end-to-end fashion, the burden of actually copying tokens still falls back to the template generator as its hidden states are reused in the copy network. In the second method, which we call a "two-stage" method, the template generator and the copy network are trained separately. This allows the template generator to focus on generating templates. Moreover, since the copy happens after template generation, a bi-directional RNN (Schuster and Paliwal, 1997) can be used in Enc_{temp}, which enables the hidden state of each word to summarize not only the preceding words but also the following words.

In this specific task, we always fill in the <entity> placeholders first and when we do that, we only consider the additional records r^* mentioned in Section 2. When we fill in the <number> placeholders, we only consider the records whose *entity* portions have been copied into the sentence. In short, when we replace a certain placeholder, only a subset of S, which we denote as S^*, are considered. This small S^* trick ensures the copy network does not copy numeric values whose *entity* portion is not present in the same sentence. This trick is not applicable in traditional copy networks where the order in which we copy tokens can not be adjusted.

Double Attention Attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) can be described as a function mapping a query vector v_q and a series of source vectors $\{v_s\}$ to a context vector v_c, i.e. $v_c = Atten(v_q, \{v_s\})$. More concretely,

$$v_c = \sum a_s v_s \tag{1}$$

where

$$a_s = \frac{\exp(u_s)}{\sum_{s'} \exp(u'_s)}, \quad u_s = v_q^T \cdot W \cdot v_s \tag{2}$$

and W is a parameter matrix in $\mathbb{R}^{D \times D}$.

To point more precisely, attention mechanism is employed to the input records as well as to the generated template. Given the input vector h_p, the input-record-based vectors $\{h_j^{\text{record}}\}$ and the template-text-based vectors $\{h_t^{\text{temp}}\}$, an input-record-based context vector $v_c^{\text{record}} = Atten(h_p, \{h_j^{\text{record}}\})$ is computed to utilize information from the input records. Likewise, a template-text-based context vector $v_c^{\text{temp}} = Atten(h_p, \{h_t^{\text{temp}}\})$ is computed to utilize information from the template. In contrast, the traditional copy network only applies attention mechanism to the input records.

v_c^{record}, v_c^{temp} and the input vector h_p are then combined to produce a score for a i-th input record r_i:

$$Score(r_i) = [v_c^{\text{record}}; v_c^{\text{temp}}; h_p]^T \cdot W_1 \cdot \tilde{r}_i \tag{3}$$

where W_1 is a parameter matrix in $\mathbb{R}^{3D \times D}$.

Position-aware Attention In the task of relationship classification, which involves determining the relationship of a pair of entities, it has been observed that the relative distances from other words in the sentence to the entities could be informative (Zeng et al., 2014; dos Santos et al., 2015). Inspired by Zhang et al. (2017), we employ a modified position-aware attention mechanism to the template text. When replacing some placeholder using the copy network, the distances between every word in the generated text and the placeholder are incorporated into the calculation of the template-text-based context vector v_c^{temp}. Compared to simply using $Atten(h_p, \{h_t^{\text{temp}}\})$ to calculate v_c^{temp}, this method allows the network to pay attention to words according to their relative locations to the placeholder.

Formally, let's assume $\{y'_1, y'_2, ..., y'_n\}$ is the generated template sentence and the placeholder we are replacing is the m-th token. For every word in the sentence other than the placeholder itself, we calculate its distance to the placeholder to obtain a position sequence $\{q_1, q_2, ..., q_{m-1}, q_{m+1}, ..., q_n\}$,

[2]Concretely, $\{h_t^{\text{temp}}\} = Enc_{\text{temp}}(\{y'_t\})$, where $\{y'_t\}$ is the sentence the placeholder is in; the separate input encoder has identical settings with the input encoder in template generator, so $\{h_j^{\text{record}}\} = Enc_{\text{input}}^{*}(\{r_j\})$; the hidden state of the Enc_{temp} is initialized using the hidden state of Enc_{input}^{*}.

where $q_t = t - m$. Using a position embedding matrix Q, we can then obtain a positional embedding sequence$\{q'_1, q'_2, ..., q'_{m-1}, q'_{m+1}, ..., q'_n\}$. Combining $\{q'_t\}$ and $\{h_t^{\text{temp}}\}$, we can calculate the template-text-based context vector v_c^{temp} as:

$$v_c^{\text{temp}} = \sum_{t \in [1,n]; t \neq m} a_t h_t^{\text{temp}} \tag{4}$$

where

$$a_t = \frac{\exp(u_t)}{\sum_{t'} \exp(u'_t)}, \quad u_t = h_p^T \cdot W_2 \cdot [h_t^{\text{temp}}; q'_t] \tag{5}$$

and W_2 is a parameter matrix in $\mathbb{R}^{D \times 2D}$.

Pairwise Ranking Loss As mentioned in Section 2, in traditional copy mechanisms, the unnormalized distribution over input records P^*_{copy} and the unnormalized distribution over words in the vocabulary P^*_{word} are merged into a normalized distribution P_{joint}, which involves somehow applying a softmax function to P^*_{copy}[3]. This method limits the possibility of alternative loss functions for the copy network. In fact, choosing the right input record to copy can also be viewed as ranking the records according to their relative appropriateness, which justifies the use of a ranking loss function. It has been observed in the task of relationship classification that a pairwise ranking loss function is better than a log-likelihood loss function following the softmax function (dos Santos et al., 2015). Following the notation in Section 2, for any y_t that is considered to be copied from the input, C is the set of valid input records that is the source of y_t while S^* is the record set worth considering. A score can be calculated for every record in S^* using Equation 3. We reward the scores of the valid input records and penalize the highest score among all invalid input records by minimizing a pairwise ranking loss function, defined as:

$$L = log(1 + exp(\gamma(m^+ - \sum_{r_i \in C} Score(r_i))) + log(1 + exp(\gamma(m^- + \max_{r_i \in S^*; r_i \notin C} Score(r_i))) \tag{6}$$

where m^+ and m^- are margins and γ is a scaling factor that magnifies the difference between the score and the margin, and helps to penalize more on the prediction errors.

4 Experiments

4.1 Setup

Dataset We experiment our method on the ROTOWIRE dataset (Wiseman et al., 2017). The average text length is 337.1, while the average text lengths of previous datasets (Chen and Mooney, 2008; Liang et al., 2009; Lebret et al., 2016) do not exceed 30. The average number of input records is 628, while those of other datasets do not exceed 200. The text in this dataset also contains a decent percentage of entities and numeric values. This is indeed a challenging enough dataset to test our model on.

Implementation We perform an additional step in preprocessing called "relexicalization"[4]. We used PyTorch (Paszke et al., 2017) for implementation. For encoders and decoders, we use two layers of LSTM. The hidden states and the embedding vectors are all in $\mathbb{R}^{600}$. General-style attention and input-feeding (Luong et al., 2015) are employed. The models are trained using Adam (Kingma and Ba, 2014). The hyper parameters for pairwise ranking loss are set to those provided in dos Santos et al. (2015) without further tuning.

[3]In some models (Gu et al., 2016) P^*_{copy} is not directly normalized. It is mixed with P^*_{word} and then normalized jointly. But generally, we can say that P^*_{copy} is somehow "normalized" using a softmax function.

[4]An entity may have different aliases. For example, "Phoenix Suns" can be referred to as "Suns", "Phoenix" or "Phoenix Suns"; a player can be referred to using his first name or second name or full name. In this work, we didn't make our model learn this naming strategy for processing simplicity. We replace all different alias of the same entity with a uniform name. However, there might be need for diverse naming strategies in a real world situations and we leave that to future work.

	Development					
	RG		*CS*		*CO*	BLEU
Model	*P%*	#	*P%*	*R%*	*DLD%*	
Gold	94.39	16.72	100	100	100	100
Joint Copy with Rec & TVD	62.83	16.76	26.99	**39.62**	11.96	12.50
Conditional Copy	74.62	16.53	30.74	38.23	14.69	14.47
Our model	**85.81**	19.39	31.02	39.45	**16.93**	**16.44**
-double attention	83.04	18.22	**31.20**	38.83	16.70	16.23
-pos attention	84.80	19.17	30.88	39.32	16.73	16.18
-ranking loss	77.62	17.93	30.29	39.36	15.71	15.79
-S^* *trick*	83.17	**19.69**	29.25	39.06	16.19	16.30
	Test					
Gold	94.50	16.99	100	100	100	100
Joint Copy with Rec & TVD	60.82	16.31	26.78	**39.90**	14.15	13.47
Conditional Copy	74.70	16.19	**32.32**	38.96	15.24	14.16
Our Model	**84.86**	**19.31**	30.81	38.79	**16.34**	**16.19**

Table 2: Comparison results. All results are tested after "relexicalization".

Evaluation Metrics For a long time, the automatic evaluation metrics for the data-to-text task have been limited to ones like BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004). However, Wiseman et al. (2017) proposed to use relationship classification techniques for evaluation. Given a sentence and a pair of an entity and a numeric value in the sentence, a relationship classification system can predict the relationship between the entity and the numeric value. Therefore, such a system can be used to assess the quality of automatic generations by extract correctly described records from the generated text. We follow Wiseman et al. (2017) and evaluate our models on these three automatic metrics:

Relation Generation (RG): precision ($P\%$) and number ($\#$) of unique records correctly reflected in the generated text.

Content Selection (CS): precision ($P\%$) and recall ($R\%$) of unique records correctly reflected in the generated text that are also reflected in the gold text.

Content Ordering (CO): normalized Damerau-Levenshtein Distance ($DLD\%$) (Brill and Moore, 2000) between the sequences of records extracted from the generated text and those from the gold text.

Among the three metrics, we argue that the RG metrics is the most crucial one as it evaluates the data fidelity of the system. In this task, it could be considered okay if the system decides to report the game from a different angle and describe some data that are not present in the gold text or describe them in a different order, as the saying goes, "There are a thousand Hamlets in a thousand people's eyes." However, it would be intolerable if it got the scores wrong. We also note that these automatic metrics provided in Wiseman et al. (2017) can achieve over 94% accuracy on held-out data, making it practical for evaluation.

4.2 Comparison with the State-of-the-art Models

Model	Dev Set BLEU	Test Set BLEU
Joint Copy with Rec & TVD	11.53	11.83
Conditional Copy	13.08	12.59
Our Model	**14.66**	**14.33**

Table 3: Template quality

Here we present the comparison results against models with state-of-the-art traditional copy mechanisms[5] in Table 2, including the Joint Copy with Rec & TVD model, which is an augmented version of

[5]We also tried to train a Conditional Copy model with a hierarchical structure, however the results are not satisfying. We argue that this could be because the hierarchical structure is interfering with the copy network, let alone the fact that this Conditional Copy model has been observed to require extra consideration in training procedure.

the Joint Copy model (Gu et al., 2016), and the Conditional Copy model (Gülçehre et al., 2016). Here we used trained models provided by Wiseman et al. (2017) and performed "relexicalization" on the results to keep things fair.

It is clear that our model outperformed other models by a large margin on data fidelity. The precision of record generation (RG) boosted by more than 10%, achieving 85%. The number of correctly generated records also went up by more than 2.5.

We also observed an increase in BLEU score by about two points. We argue that it is not only because the entities and numbers are filled in more precisely, but also because the templates are generated better. In fact, we compared the template quality of different models in Table 3. We take out all the entities and numeric values in the generated text as well as the gold text and calculate their BLEU score. Results show that our model indeed generates better templates.

With the precision up, we also observed a decent increase on the content ordering (CO) metrics. It is worth noting that an increase on RG does not guarantee improvement on content selection (CS). This is perhaps because models that learn to generate records more accurately may not learn to generate records more like human writers. In fact, all models achieved close scores on the recall of CS. It is worth noting that the Joint Copy model, worst on RG, achieved the highest recall on CS.

4.3 Model Validation

We conduct experiments to show how the model's performance is affected by removing some components in the delayed copy network. We used the same template generator for all these models. Results (Table 3) reveal some interesting characteristics about our models.

The biggest boost comes from the pairwise ranking loss we employed. Replacing the ranking loss with a log-likelihood loss in traditional copy mechanisms ("*-ranking loss*") would result in a drop in RG. The BLEU suffered less, which we argue is due to the well-performing template generator. The "*-ranking loss*" model still outperforms the Conditional Copy model, with higher precision of record generation, more correct records and higher BLEU score. Removing the template-text-based attention ("*-double attention*") or replacing the template-text-based position-aware attention with a vanilla attention ("*-pos attention*") also results in a decline on mostly RG and BLEU, showing the efficacy the double attention and position-aware attention. We also tested a model without the S^* trick ("*-S^* trick*") mentioned in Section 2. It can be seen that this trick makes a small improvement to our model yet it is not a defining factor.

4.4 Case Study

Our Model: The Magic (25 - 53) defeated the Bulls (46 - 32) 105 - 103 on Wednesday at the Amway Center in Orlando . The Magic got off to a quick start in this one , out - scoring the Bulls 29 - 21 in the first quarter alone . The Magic were the superior shooters , going 46 percent from the field and 35 percent from the three - point line , while the Bulls went 43 percent from the floor and just 46 percent from beyond the arc . The Magic were also able to force the Bulls into 25 turnovers , while committing only 15 of their own , which may have been the difference in this one . The Magic were led by Tobias Harris , who posted 8 points (2 - 13 FG , 4 - 4 FT) , 3 rebounds and three steals ...
Conditional Copy: The Magic (25 - 53) defeated the Magic (25 - 53) 105 - 105 on Wednesday at the Amway Center in Orlando . The Bulls got off to a quick start in this one , out - scoring the Bulls 29 - 21 right away in the first quarter . The Magic were the superior shooters in this game , going 46 percent from the field and 46 percent from the three - point line , while the Magic went just 43 percent from the floor and 35 percent from deep . The Bulls were also able to force the Magic into 16 turnovers , while committing only 16 of their own . The Bulls were led by the duo of Nikola Vucevic and Nikola Vucevic . Nikola Vucevic went 9 - for - 16 from the field and 3 - for - 4 from the three - point line to score a game - high of 22 points , while also adding two rebounds and two assists ...
Gold: The Magic (25 - 53) defeated the Bulls (46 - 32) 105 - 103 on Wednesday at the Amway Center in Orlando . Down two with just over six seconds left in the game , it was Pau Gasol who was able to force his way to the free throw line and hit a pair of free throws to tie the game up . Victor Oladipo then came up clutch for the Magic , driving for a layup with just a second left in the game , therefore giving them a two point lead and eventually the victory . With the loss , the Bulls move into a tie with the Toronto Raptors for the third projected playoff seed in the Eastern Conference . With just four games left in the regular season , it will be a battle for future playoff positioning from here on out . The Magic were superior shooters in this one , going 46 percent from the field , while the Bulls finished at 43 percent ...

Table 4: Example texts generated by different models. Only a part of the texts are showed due to space limitation. The erroneous text has been marked red. Unmarked text correctly reflects records.

We show an example of the text generated by our model, the Conditional Copy model and the corresponding gold text in Table 4. It is clear that our model made fewer mistakes when it comes to accurately describing the data. One interesting finding is that the Conditional Copy model suffers from a common issue found in language-model based models, repetition. The repetition is not shown in repeated sentence

Model	Correct	Wrong	Repeated
Conditional Copy	3.34	1.58	0.35
Our Model	**3.78**	**1.03**	**0.10**

Table 5: Average number of correct , wrong and repeated records in the generated text per sentence.

patterns like in language-model based models but in the repeatedly copied entities and numbers. There are more than 7 repetitions in this short text. However, in our model, the delayed copy network avoided such problem since it is actually not based on language models. We conducted human evaluations to support our assumption. We manually checked 100 sentences randomly selected from summaries for 8 randomly selected games. We checked for correct records, wrong records and repeated records in the text. A record is considered repeated if it is mentioned twice in a short segment. Results (Table 5) show that our model generates records more precisely and makes much fewer repetitions.

We also inspected the attention heat map of the decoding-side attention. The attention weights seem to concentrate heavily on a single word, rather than spread across words. We argue that this is probably because the attention mechanism in our delayed copy network works somehow differently from the traditional attention mechanism. In machine translation (Bahdanau et al., 2014; Luong et al., 2015), the attention mechanism is applied between a language-model based decoder and an encoder. However, in our model, the template-text-based attention mechanism is applied within a bi-directional text encoder. So the bi-directional encoder in our model could have learned to condense the information into the hidden state of a single word.

5 Related Work

Traditionally, data-to-text tasks are decomposed into two subproblems (Kukich, 1983; Goldberg et al., 1994): *content selection*, which involves choosing a subset of relevant records to talk about, and *surface realization*, which is concerned with generating natural language descriptions for this subset. There is also an alternative decomposition in Reiter and Dale (1997), where they break the problem down to three modules: *content selection*, *micro planning*, and *surface realization*. There is also a corresponding work (Mellish et al., 2006) that challenges this decomposition. In early stages, *surface realization* is often realized using templates (van Deemter et al., 2005) or statistically learned models with hand-crafted features (Belz, 2008; Konstas and Lapata, 2012). Machine learning approaches have also been applied to *content selection*. Barzilay and Lapata (2005) models it as a classification problem, whereas Liang et al. (2009) uses a generative semi-Markov model.

With the rise of neural networks, some recent work has focused on generating text from data using neural networks, for example, generating weather forecasts as well as game descriptions (Mei et al., 2016), generating short biographies from Wikipedia Tables (Lebret et al., 2016; Hachey et al., 2017; Sha et al., 2017; Liu et al., 2017), and generating market comments from stock prices (Murakami et al., 2017). With these models achieving excellent results on traditional evaluation metrics such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004), Wiseman et al. (2017) proposed to evaluate data-to-text systems from three aspects using relationship classification systems: data fidelity, content selection and content ordering. While there has been some work (Liu et al., 2017; Sha et al., 2017) related to the last two aspects, we find little work explicitly tackling the data fidelity problem.

Here we point out a parallel and independent work (Lu et al., 2018) on image caption that also employs a similar framework. They also generate slotted templates first and then fill in those slots with a separate model, though the exact models used in their work is quite different from ours.

6 Conclusion and Future Work

In this paper, we present a novel two-stage approach with a delayed copy mechanism to improve the precision of data in generated texts. Experiments show that our model points more precisely than other state-of-the-art models and also generates better templates. There is lots of future work we can do. The content selection ability of the model has the potential to be improved and we would like to further improve the precision of copy mechanism. Another appealing direction is to adopt our delayed copy mechanism to other NLG domains.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036, 61331011) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank Jiwei Tan, Hongyu Zang and the anonymous reviewers for their helpful comments. We also thank the authors of Wiseman et al. (2017) for releasing their code. Xiaojun Wan is the corresponding author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 331–338.

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

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, October 1-8, 2000.*

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 128–135.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 626–634.

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.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ben Hachey, Will Radford, and Andrew Chisholm. 2017. Learning to generate one-sentence biographies from wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 633–642.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761. Association for Computational Linguistics.

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

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1106–1115.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 91–99.

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

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *CoRR*, abs/1711.09724.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2018. Neural baby talk. *CoRR*, abs/1803.09845.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 720–730.

Chris Mellish, Donia Scott, Lynne J. Cahill, Daniel S. Paiva, Roger Evans, and Mike Reape. 2006. A reference architecture for natural language generation systems. *Natural Language Engineering*, 12(1):1–34.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1374–1384.

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, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

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

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

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2017. Order-planning neural text generation from structured data. *CoRR*, abs/1709.00155.

Kees van Deemter, Mariët Theune, and Emiel Krahmer. 2005. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1782–1792.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1850–1859.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 2335–2344.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 35–45.