arXiv:2009.13401v1 [cs.CL] 28 Sep 2020

Injecting Entity Types into Entity-Guided Text Generation

Xiangyu Dong[1*], **Wenhao Yu**[1*], **Chenguang Zhu**[2], **Meng Jiang**[1]
[1]University of Notre Dame, Notre Dame, IN, [2]Microsoft Research, Redmond, WA
{xdong2, wyu1, mjiang2}@nd.edu, chezhu@microsoft.com

Abstract

Recent successes in deep generative modeling have led to significant advances in natural language generation (NLG). Incorporating entities into neural generation models has demonstrated great improvements by assisting to infer the summary topic and to generate coherent content. In order to enhance the role of entity in NLG, in this paper, we aim to model the entity type in the decoding phase to generate contextual words accurately. We develop a novel NLG model to produce a target sequence (i.e., a news article) based on a given list of entities. The generation quality depends significantly on whether the input entities are logically connected and expressed in the output. Our model has a multi-step decoder that *injects* the entity types into the process of entity mention generation. It first predicts the token of being a contextual word or an entity, then if an entity, predicts the entity mention. It effectively embeds the entity's meaning into hidden states, making the generated words precise. Experiments on two public datasets demonstrate type injection performs better than type embedding concatenation baselines.

Introduction

Recent years have witnessed the rapid development of natural language generation (NLG) with a variety of applications such as machine translation, summarization, and news generation (Iqbal and Qureshi 2020; Garbacea and Mei 2020). End-to-end neural generation models (e.g., Seq2Seq) learn from training data to generate fluent text and thus have been successfully applied to the NLG tasks (Sutskever, Vinyals, and Le 2014; Vaswani et al. 2017; Radford et al. 2019).

Entity, as an important element of natural language, plays the key role of making the text coherent (Grosz, Joshi, and Weinstein 1995). Recently, modeling entities into NLG methods has demonstrated great improvements by assisting to infer the summary topic (Amplayo, Lim, and Hwang 2018) or to generate coherent content (Ji et al. 2017; Clark, Ji, and Smith 2018). To enhance the representation of entity, entity type is often used in existing work – represented as a separate embedding and concatenated with the embedding of entity mention (i.e., surface name) in the encoding/decoding phase (Zhao et al. 2019; Puduppully, Dong, and Lapata 2019; Yu et al. 2018; Chan et al. 2019). Although the concatenation performed better than using the entity mention embedding only, the relationship between entity mention and entity type was not reflected, making the signal from entity type undermined in the NLG.

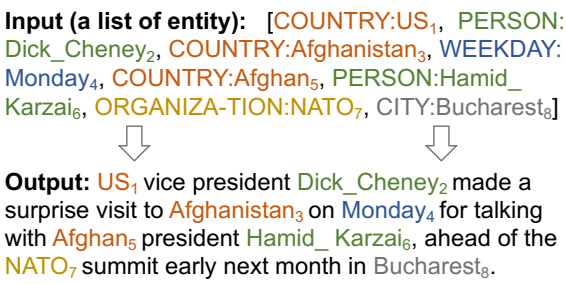

Figure 1: An example of generating a news article from a list of names of entities and their types. How to effectively use the type information in the NLG model is an open question.

To address the above issue, our idea is to model the entity type carefully in the decoding phase to generate contextual words accurately. In this work, we focus on developing a novel NLG model to produce a target sequence (i.e., a news article) based on a given list of entities. We assume the list has entity mentions as well as their types. Figure 1 presents an example of this task. We name the task as entity-guided text generation (EGTG). Compared to the number of words in the target sequence, the number of given entities is much smaller. Since the source information is extremely insufficient, it is difficult to generate precise contextual words describing the relationship between or event involving multiple entities such as person, organization, and location. Besides, since input entities are important prompts about the content in the target sequence (Yao et al. 2019), the quality of generated sequence depends significantly on whether the input entities are logically connected and expressed in the output. However, existing generation models may stop halfway and fail to generate words for the expected entities (Feng et al. 2018). They are not able to use all the information in the given input and thus lead to serious incompleteness.

In this paper, we propose a new strategy of utilizing the type information in NLG, called *type injection*, and a novel entity-guided text generation model. Besides the concatenation in the encoder, our model has a multi-step decoder. It first predicts the probability that each token is a contex-

* The first two authors have equal contributions.

tual word in the vocabulary or an entity from a given list. If the token is an entity, the model will directly inject the embedding of the entity type into the process of generating the entity mention by using a mention predictor to predict the entity mention based on the type embedding and current decoding hidden state. The type injection maximizes the likelihood of generating an entity indicator rather than the likelihood of sparse entity mentions, ensuring entities to appear at correct positions. And the hidden state is jointly optimized by predicting the role of token and predicting the entity mention so that the entity's information is effectively embedded into the states. To better utilize all the given entities, we further add an entity type-enhanced natural language understanding (NLU) module to predict entity mentions from the backward direction so that context from both directions are employed to predict the entity mentions. Besides, our model can fully utilize all input entities and strictly maintain the order of given entities in the target sequence.

We conduct experiments on two public news datasets GIGAWORDS and NYT. Results demonstrate that the type injection-based model generates more precise contextual words than the existing concatenation-based models.

Related Work

Entity-related Text Generation

Entities in a natural language carry useful contextual information (Nenkova 2008) and therefore play an important role in different NLG tasks such as summarization (Sharma et al. 2019; Amplayo, Lim, and Hwang 2018), image caption (Lu et al. 2018), table description (Puduppully, Dong, and Lapata 2019) and story generation (Clark, Ji, and Smith 2018). In summarization, entity mentions have been used to extract non-adjacent yet coherent sentences, link to existing knowledge bases, and infer the summary topic (Sharma et al. 2019; Amplayo, Lim, and Hwang 2018). In table description, entity mentions have been used to achieve discourse coherence (Puduppully, Dong, and Lapata 2019). Our task is relevant to (Chan et al. 2019) that generates product description from a list of product entities. Different from above work, we aim to leverage entity type into the decoding phase for better predicting entities and contextual words.

Word-to-text Generation

Generating natural language from topic words and keywords is an important yet challenge task. It not only has plenty of practical applications, e.g., benefiting intelligent education by assisting in essay writing (Feng et al. 2018; Yang et al. 2019) and automated journalism by helping news generation (Zheng et al. 2017; Kanerva et al. 2019), but also serves as an ideal test bed for controllable text generation (Wang and Wan 2018; Yao et al. 2019). The main challenge of word-to-text lies in that the source information is extremely insufficient compared to the target output, leading to poor topic consistency in generated text (Yang et al. 2019). Some existing work improved word-to-text generation quality by incorporating external knowledge such as knowledge graph (Yang et al. 2019) and pre-trained language models (Fan, Lewis, and Dauphin 2018; Guan et al. 2020), which can provide additional commonsense knowledge and background information. Two other related scenarios with our task are (i) using words (e.g., sentiment, style) as constrains in decoding (Hu et al. 2017; Li et al. 2018) and (ii) word ordering (Dinu et al. 2019; Miao et al. 2019). In difference, we aim to use entity type to improve text generation from a list of entity mentions.

Entity-Guided Text Generation

In this section, we first give definition of the entity-guided text generation (EGTG) task. We introduce type embedding concatenation for EGTG. Then we present a novel model called InjType. It injects type information into the process of entity mention generation. InjType has a multi-step decoder. The decoder includes a entity indicator predictor and a mention predictor. It is further augmented by a entity type enhanced natural language understanding (NLU) modeule.

Task Definition

Given a list of entities $X = (x_1, \ldots, x_n)$, where $x_i = (x_i^M \in \mathcal{M}, x_i^T \in \mathcal{T})$ consists of the mention and type of the i-th entity, where $\mathcal{M}$ is the set of entity mentions and $\mathcal{T}$ is the set of entity types. The expected output sequence is $y = (y_1, \ldots, y_m)$ containing all the entity mentions in order, i.e., $y_{t_i} = x_i^M, 1 \leq t_1 \leq t_2 \leq ... \leq t_n \leq m$. We denote the vocabulary of contextual words by $\mathcal{V}$. So $y_j \in \mathcal{M} \cup \mathcal{V}$, $j \in \{1, \ldots, m\}$. The task is to learn a predictive function $f : X \rightarrow Y$, mapping a list of entities to a target sequence.

Background: Type Embedding Concatenation

A natural way to incorporate type information into a Seq2Seq generation framework is to concatenate entity mention embeddings and type embeddings (Yu et al. 2018; Zhao et al. 2019; Chan et al. 2019). In the encoding phase, we take both entity mention and entity type as input, and learn contextual representation of each entity in the given list through a bi-directional GRU encoder. In the decoding phase, we adopt a standard attention mechanism (Bahdanau, Cho, and Bengio 2015) to generate output sequence.

Mention Level Generation In the *encoding* phase, we concatenate the embedding of entity mention x_i^M with the embeddings of its corresponding type x_i^T extracted by CoreNLP (Finkel, Grenager, and Manning 2005). So the input embedding of the i-th entity is defined as:

$$\mathbf{x}_t = \mathbf{x}_t^M \oplus \mathbf{x}_t^T, \tag{1}$$

where $\oplus$ denotes vector concatenation. We adopt bi-directional gated recurrent unit (Bi-GRU) (Cho et al. 2014) as encoder to capture contextualized representation for each entity given in the input list. The encoder has a forward GRU which reads input entities X from x_1 to x_n, where n is the length of input list. It also has a backward GRU to learn from the backward direction. We then concatenate hidden states from both directions:

$$\mathbf{h}_i = \overrightarrow{GRU}(\mathbf{x}_i) \oplus \overleftarrow{GRU}(\mathbf{x}_i). \tag{2}$$

In the *decoding* phase, another GRU serves as the model decoder to generate entity mention and contextual words to

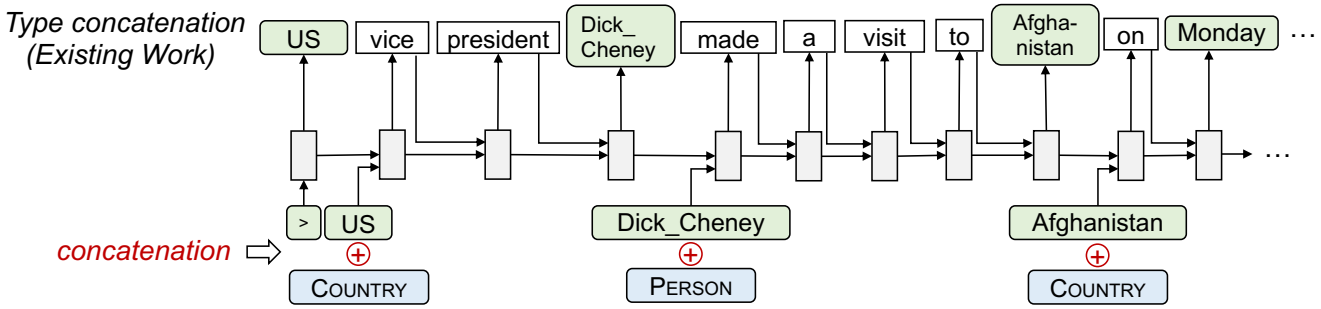

Figure 2: Concatenating entity mention embeddings and type embeddings is a straightforward strategy to use the type information. However, it may not be effective due to lack of contextual information in the hidden states. ***Note*** that the figure only highlights the decoder. A bi-GRU encoder and attention mechanism are not included since they are widely used in NLG.

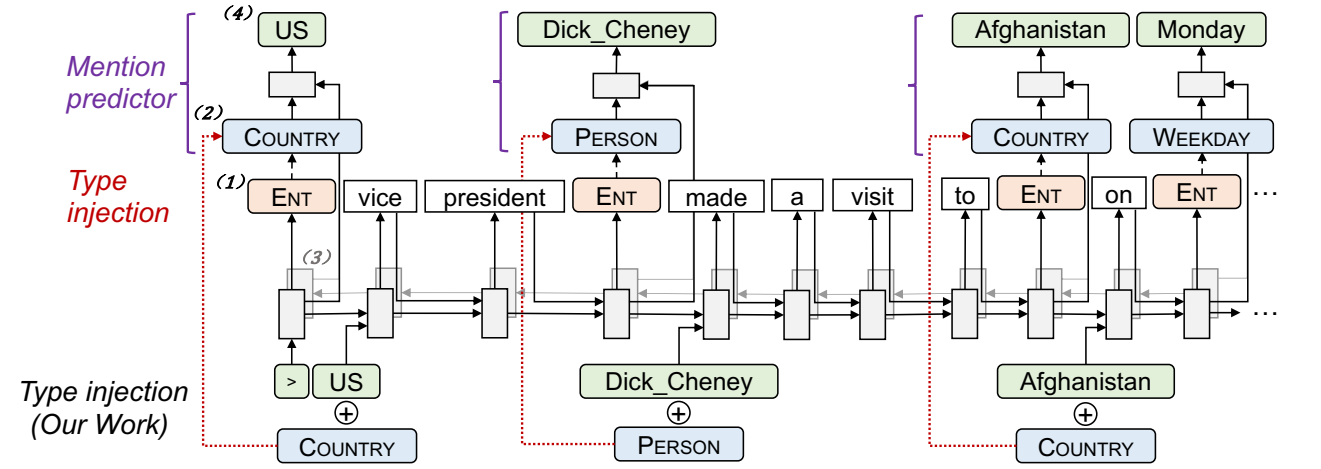

Figure 3: ***Note*** that, same as Figure 2, the figure only highlights the decoder. Our idea is to *inject* the type information into the process of entity mention generation. The process has four steps: (1) predicting the ENT token (i.e., entity indicator) to maintain the order of entities; (2) injecting the entity types; (3) combining an entity type enhanced NLU with backward information of target sequence; (4) predicting the entity mention using the injected type and hidden state by a mention predictor.

generate the target sequence. Therefore, given the current decoding hidden state $\mathbf{s}_t$ and the source-side attentive context vector $\mathbf{c}_t$, the readout hidden state $\mathbf{r}_t$ is given as follows:

$$\mathbf{r}_t = \tanh(\mathbf{W}_c \cdot [\mathbf{s}_t \oplus \mathbf{c}_t]), \tag{3}$$

where the source-side attentive context vector $\mathbf{c}_t$ at the current decoding step t is computed through attention mechanism which matches the last decoding state $\mathbf{s}_{t-1}$ with each encoder hidden state $\mathbf{h}_i$ to get an importance score $\alpha_{t,i}$. Then, all importance scores are then normalized to get the current context vector $\mathbf{c}_t$ by weighted sum:

$$\mathbf{e}_{t,i} = \tanh(\mathbf{W}_a \cdot \mathbf{s}_{t-1} + \mathbf{U}_a \cdot \mathbf{h}_i), \tag{4}$$

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \cdot \mathbf{h}_i, \text{ where } \alpha_{t,i} = \frac{exp(\mathbf{e}_{t,i})}{\sum_{i=1}^{n} exp(\mathbf{e}_{t,i})}, \tag{5}$$

where $\mathbf{W}_a$ and $\mathbf{U}_a$ are trainable parameters. Then the readout state $\mathbf{r}_t$, is passed through a multilayer perceptron (MLP) to predict the next word with a softmax layer over the decoder vocabulary (all entity mentions and contextual words):

$$p(y_t|y_{<t}, X) = \text{softmax}(\mathbf{W}_r \cdot \mathbf{r}_t). \tag{6}$$

The loss function of mention-level generation is:

$$\mathcal{L}_{Mention} = -\sum_{t=1}^{m} \log\left(p(y_t \in \mathcal{M} \cup \mathcal{V}|y_{<t}, X)\right). \tag{7}$$

Figure 2 is a reference to the model design.

Type Level Generation Directly generating entity mentions is challenging because the decoder vocabulary is too large and many entity mentions do not frequently appear. In order to address the above challenge, we perform a type-level generation as follows. For encoding phase, we take the same input as the mention level generation. However, for the decoding phase, we replace entity mentions with entity types in the decoder vocabulary. Meanwhile, we use entity types as supervisory signals instead of entity mentions. Once the

type-level generation finished, we would *put the entity mention at the position of the generated type* according to the mention order in the input entity list. Suppose the ground truth of typed output $y\prime$ is the target sequence in which entity mentions are replaced by entity types. So, the loss of type-level generation is defined as:

$$\mathcal{L}_{Type} = -\sum_{t=1}^{m} \log\left(p(y'_t \in \mathcal{T} \cup \mathcal{V} | y'_{<t}, X)\right). \quad (8)$$

Because $|\mathcal{T}| \ll |\mathcal{M}|$, the decoder vocabulary is much smaller than that in mention level generation.

InjType: A Novel Type Injection Model

Although the concatenation has demonstrated better performance in previous work (Zhao et al. 2019; Yu et al. 2018; Chan et al. 2019), the relationship between entity mention and entity type was not reflected, making the signal from entity type undermined. Therefore, we propose a novel type injection method (called *InjType*) with a multi-step decoder of four steps. First, it generates either an entity indicator or a contextual word. Second, if an entity indicator is generated, it inject entity type at the corresponding position in the input entity list into the next step of generation. Third, it combines an entity type enhanced NLU module with backward information of target sequence. Fourth, it builds up a module of mention predictor to predict entity mentions from the current hidden state and the injected entity type.

Entity Indicator Predictor At each step, the decoder predicts either an entity indicator or a contextual word. An entity indicator, denoted as $\langle \text{ENT} \rangle$, indicates that the current decoding step should generate an entity in the output sequence. If the input has n entities, there will be n entity indicator $\langle \text{ENT} \rangle$ generated in the output sequence. Traditional auto-regressive decoder stops when it generates an end-of-sentence token ($\langle \text{EOS} \rangle$). It prevents the model from directly interfering with the generation process. Instead, we split the generation process into multiple blocks by the $\langle \text{ENT} \rangle$ tokens. So the first-step output sequence is as follows:

$$\text{block}_1, \text{ENT}_1, \text{block}_2, \ldots, \text{ENT}_n, \text{block}_{n+1}.$$

Each block has one or multiple contextual words. And it ends with an entity indicator ($\langle \text{ENT} \rangle$). Within each block, the generation process is the same as the auto-regressive decoding process. When the auto-regressive decoder generates an $\langle \text{ENT} \rangle$, the generation process of the current block ends. When the decoder generated the $(n+1)$-th entity indicator $\langle \text{ENT} \rangle$, the entire generation terminates. Then we fill $\langle \text{ENT} \rangle$ with the corresponding entity mention at the same position given in the input entity list. In this way, we can retain the same order of the entities in the generated sequence:

$$\text{block}_1, x_1^M, \text{block}_2, \ldots, x_n^M, \text{block}_{n+1}.$$

Suppose the ground truth of entity indicator output y'' is the target sequence with entity mentions replaced with entity indicators $\langle \text{ENT} \rangle$. Now, the loss function of type-level generation with entity indicator is defined as:

$$\mathcal{L}_{Ent} = -\sum_{t=1}^{m} \log\left(p(y''_t \in \{\text{ENT}\} \cup \mathcal{V} | y''_{<t}, X)\right). \quad (9)$$

The size of decoder vocabulary ($|\mathcal{V}|+1$) is even smaller than that in concatenation-based type level generation ($|\mathcal{V}|+|\mathcal{T}|$).

Mention Predictor Entity indicator helps retain the order of entities, making the generated sequence fluent and logical. However, because each block's generation is ended with the entity indicator token ($\langle \text{ENT} \rangle$), the representations of the last hidden states in different blocks are assimilated, which may lose contextual information in previous generated blocks.

We employ a module of mention predictor to apply entity mention classification so that the hidden states can carry the rich contextual information. When the decoder generates an entity indicator $\langle \text{ENT} \rangle$, it indicates that the current step should be fulfilled with an entity. Thus, for each generated entity indicator, we feed the concatenation of injected entity type and current hidden state into a mention classifier. In i-th block, the predicted entity mention is:

$$x_{i'}^M = \text{softmax}(\mathbf{W}_m \cdot [\mathbf{s}_t \oplus \mathbf{x}_i^T]), \quad (10)$$

where $\mathbf{s}_t$ is the hidden state of the t-th token in the generated text. In this way, the last hidden state in each block not only has to be classified as an entity indicator ($\langle \text{ENT} \rangle$), but also carries both entity type and entity mention information in order to make precise generation. The classification loss $\mathcal{L}_{MP}$ is defined as follows:

$$\mathcal{L}_{MP} = -\sum_{i=1}^{n} x_i^M \cdot \log(x_{i'}^M), \quad (11)$$

where m_i is the one-hot representation of entity mention.

Entity Type-Enhanced NLU Inspired by the recent advanced language generation model UniLM (Dong et al. 2019) that unifies natural language understanding (NLU) and natural language generation (NLG) tasks within a single model, we let our decoder complete a *type enhanced NLU task* along with its original generation task. We borrow the decoder from the NLG task to conduct an NLU task on the ground truth articles including entity types from the training set. The entity type enhanced NLU task asks the decoder to predict entity mentions corresponding to the types in the typed ground truth based on context words. If the decoder is able to correctly predict the entity mention given contextual information, it should be capable of generating good context words that can help predict entity mentions as well. Since the decoder used for generation is naturally one-way (left-to-right), in order to complete the NLU task more reasonably, we train a GRU module in a reversed direction, represented as $\overleftarrow{GRU}$. Additionally, we reuse the original NLG decoder without attention, denoted by $\overrightarrow{GRU'}$ for the NLU task. This module generates the prediction as follows:

$$\mathbf{s}'_t = [\overrightarrow{GRU'}(y''_t) \oplus \overleftarrow{GRU}(y''_t)], \quad (12)$$

$$y''_t = \begin{cases} y_t, & y_t \in \mathcal{V}, \\ y_t^T, & y_t \in \mathcal{M}, \end{cases} \quad (13)$$

Table 1: With SeqAttn, type embedding concatenation performs better than using mention embedding only.

Dataset	Entity level	ROUGE-2		ROUGE-L		BLEU-4		METEOR	
		dev.	test	dev.	test	dev.	test	dev.	test
GIGAWORDS-6K	Mention level	10.65	9.88	32.22	31.60	13.43	12.59	26.54	25.72
	Type level	9.42	9.20	38.50	38.39	13.88	13.61	26.02	25.69
	Mention $\oplus$ Type	**10.73**	**10.76**	**38.53**	**38.73**	**17.22**	**16.94**	**29.40**	**29.09**
NYT-8K	Mention level	5.12	4.70	22.09	21.90	5.40	4.88	18.88	18.70
	Type level	**7.19**	**6.86**	29.92	29.72	11.39	11.37	20.65	20.52
	Mention $\oplus$ Type	6.77	6.57	**30.60**	**30.71**	**11.85**	**11.91**	**21.24**	**21.36**

Table 2: With type injection, our model performs significantly better than state-of-the-art models enhanced by type embedding concatenation. Our model which was not pre-trained on large data can win over the pre-trained models.

On GIGAWORDS-6K:	ROUGE-2		ROUGE-L		BLEU-4		METEOR	
	dev.	test	dev.	test	dev.	test	dev.	test
Seq2Seq (Sutskever, Vinyals, and Le 2014)	10.30	9.83	31.74	31.43	12.90	12.21	25.72	25.14
SeqAttn (Bahdanau, Cho, and Bengio 2015)	10.73	10.76	38.53	38.73	17.22	16.94	29.40	29.09
CopyNet (Gu et al. 2016)	11.65	11.17	35.64	35.63	16.34	15.90	28.97	28.65
GPT-2 (Radford et al. 2019)	3.94	4.67	14.37	24.22	11.77	9.49	9.92	15.32
UniLM (Dong et al. 2019)	12.36	11.77	36.05	35.54	18.00	17.66	21.10	20.22
InjType (Ours)	**12.72**	**12.68**	**39.72**	**39.74**	**19.01**	**18.93**	**30.95**	**31.05**

where $\mathbf{s}'_t$ is the concatenated hidden state of the original hidden state $\mathbf{s}_t$ and new hidden state derived from added GRU module. The entity mention is then predicted:

$$x^M_{i''} = \text{softmax}(\mathbf{W}_r \cdot \mathbf{s}'_t). \tag{14}$$

So the NLU loss is only calculated at the entity positions:

$$\mathcal{L}_{NLU} = -\sum_{i=1}^{n} x^M_i \cdot \log(x^M_{i''}). \tag{15}$$

Joint Optimization InjType jointly optimizes the following loss:

$$\mathcal{L} = \mathcal{L}_{Ent} + \lambda_1 \cdot \mathcal{L}_{MP} + \lambda_2 \cdot \mathcal{L}_{NLU}, \tag{16}$$

where λ_1 and λ_2 are hyperparameters to control the importance of different tasks.

Experiments

Datasets

We create two datasets from public sources: GIGAWORDS (Graff et al. 2003) and NYT (Sandhaus 2008). To make the entity-guided setting, we filter out the articles whose lengths are bigger than 100 and 200 on the two datasets, respectively; we also filter out those in which the number of entities is fewer than 8. Statistics can be found in Table 7 (in Appendix). For both datasets, we use 1,000 articles for development, 1,000 for test, and the remaining for training. On average, the models are expected to predict more than 70 contextual words *precisely* from a list of about 10 entities.

Baseline Methods

Seq2Seq (Sutskever, Vinyals, and Le 2014) Seq2Seq is the basic encoder-decoder model widely used in NLG tasks.

SeqAttn (Bahdanau, Cho, and Bengio 2015) SeqAttn adds a soft attention mechanism to the input sequence where the most relevant information is concentrated.

CopyNet (Gu et al. 2016)**.** This model introduces copy mechanism to decoder to alleviate the out-of-vocabulary issue caused by infrequent words.

GPT-2 (Radford et al. 2019) GPT-2 is a pre-trained Transformer language model. It excels at generating convincing articles with initial prompts.

UniLM (Dong et al. 2019) UniLM is also a pre-trained language model. It unifies three tasks: Seq2Seq generation, conditional generation, and NLU.

We applied the concatenation of entity mention and type in Seq2Seq and SeqAttn. We used mention level generation for CopyNet, GPT-2, and UniLM, because CopyNet was designed to copy rare words (like entity names), GPT-2 and UniLM were pre-trained mention level corpora. Note that our InjType model was not pre-trained on large corpora, so competing with GPT-2 and UniLM would be challenging.

Implementation Details

We take 512 as dimension size of hidden states. The sizes of mention embeddings and type embeddings are 300. All parameters are initialized by sampling from a uniform in $[-1, 1]$. We use Adam optimizer (Kingma and Ba 2014) with an initial learning rate of 1e-4. Our model takes about 12 hours to train 60 epochs on an NVIDIA 2080-Ti GPU. We

Table 3: Similar observations on NYT-8K as in Table 2. The type injection model performs the best.

On NYT-8K:	ROUGE-2		ROUGE-L		BLEU-4		METEOR	
	dev.	test	dev.	test	dev.	test	dev.	test
Seq2Seq (Sutskever, Vinyals, and Le 2014)	4.60	4.06	21.65	21.14	5.15	4.61	17.55	17.09
SeqAttn (Bahdanau, Cho, and Bengio 2015)	6.77	6.57	30.60	**30.71**	11.85	11.91	21.24	21.36
CopyNet (Gu et al. 2016)	6.47	6.25	26.31	26.58	9.47	9.36	20.52	20.66
GPT-2 (Radford et al. 2019)	2.25	2.22	17.02	17.27	7.05	7.25	10.67	10.89
UniLM (Dong et al. 2019)	6.32	6.47	24.72	24.93	12.85	12.90	14.10	14.24
InjType (Ours)	**8.03**	**7.46**	**30.66**	30.55	**13.75**	**13.41**	**24.16**	**23.51**

Table 4: Ablation study. MP stands for mention predictor. On both datasets, the type injection model performs the best when trained on both the type-to-mention classification and NLU tasks.

Dataset	MP	NLU	ROUGE-2		ROUGE-L		BLEU-4		METEOR	
			dev.	test	dev.	test	dev.	test	dev.	test
GIGAWORDS-6K	-	-	10.30	10.03	38.44	38.59	16.65	16.16	28.73	28.00
	-	✔	10.58	10.22	38.73	38.72	17.09	16.50	29.05	28.18
	✔	-	11.77	11.08	38.92	38.69	18.41	17.84	30.71	30.44
	✔	✔	**12.72**	**12.68**	**39.72**	**39.74**	**19.01**	**18.93**	**30.95**	**31.05**
NYT-8K	-	-	5.99	5.43	29.46	29.27	12.61	12.07	22.51	21.94
	-	✔	6.59	5.84	29.39	28.97	12.93	12.29	22.73	22.21
	✔	-	7.36	6.79	29.96	29.98	13.66	13.12	**24.27**	**23.78**
	✔	✔	**8.03**	**7.46**	**30.66**	**30.55**	**13.75**	**13.41**	24.16	23.51

did grid search on the hyperparameters in loss Eq. (16) and got the best performance at $\lambda_1 = 2$ and $\lambda_2 = 1.5$.

Evaluation Metrics

We use four kinds of standard metrics: *i)* BLEU-4 measures the average 4-gram precision on a set of reference texts; *ii)* ROUGE-2 computes the recall of bigrams; *iii)* ROUGE-L computes the overlap of the longest common subsequence between the hypothesis and the reference; *iv)* METEOR uses a weighted F-score based on mapping unigrams and a penalty function for incorrect word order.

Experimental Results

Type embedding concatenation vs. no type information Table 1 shows that type level generation improves most of the metrics such as BLEU-4 and ROUGE-L by +1.02% and +6.28% on GIGAWORDS-6K, +6.49% and +7.83% on NYT-8K over mention level generation, respectively. This is because the size of decoder vocabulary drops from $|\mathcal{V}| + |\mathcal{M}|$ to $|\mathcal{V}| + |\mathcal{T}|$, making the prediction task easier. Since the entity mention has a long tail distribution, the representation of entity mention will not be effective enough for the low-frequency ones. We observe that concatenating type embeddings with entity mention embeddings outperforms the two methods that used no type embeddings. Comparing with type level generation, it improves BLEU-4 by +3.33% and +0.54% on the two datasets, respectively. So type information is a useful addition to the representations of entities.

Type injection vs. type embedding concatenation only Tables 2 and 3 compare the proposed InjType with competitive baselines on two datasets. The best non-pretrained baseline models are SeqAttn and CopyNet, which have been enhanced by type embedding concatenation for the task. Compared with SeqAttn, InjType improves BLEU-4 by +1.99% on GIGAWORDS-6K, +1.50% on NYT-8K, respectively. This demonstrates that type injection is a more effective strategy than simple type embedding concatenation. Compared with pre-training language models such as GPT-2 and UniLM, our model is still winning though with a smaller margin. It improves BLEU-4 by +9.43% and +1.27% on GIGAWORDS-6K, +6.16% and +0.49% on NYT-8K.

Ablation study on decoder components We compare InjType with its variants in Table 4. We observe that the mention predictor (MP) contributes more than NLU modules. Adding MP improves BLEU-4 by +1.68% and +1.05% on GIGAWORDS-6K and NYT-8K, while adding NLU improves BLEU-4 by +0.34% and +0.22%, respectively. The combination the two performs the best. The NLU module has a positive impact but not being claimed as a core contribution. It reuses the decoder to predict entity mentions and aligns seamlessly with our goal of effectively embedding entity meaning into hidden states. The bi-directional GRU used in the NLU brings extra coherency into the model since it considers entities and contexts after the current token.

Maintaining the order of entities Entity indicator predictor addresses the issue that existing models cannot maintain the entity order. Given a generated text, we extract the sequence of entities and compare it with the given list. We measure the sequence difference (error) by edit distance, a.k.a *Levenshtein* distance. It is the minimum number of

Table 5: InjType maintains the order of entities and generates more precise contextual words than baselines. The blue texts are entity mentions generated in the output; the green texts highlight the snippets generated by InjType that match ground truth.

Input entities: British (COUNTRY, C), Gordon_Brown (PERSON, P), United_States (C), US (C), George_W._Bush (P), White_House (ORGANIZATION, O), Thursday (WEEKDAY, W), Brown (P), Camp_David (LOCATION, L), Sunday (W), Bush (P), Tony_Snow (P).
Ground truth: British prime minister Gordon_Brown is to make his first official trip as premier to the United_States , for two days of talks with US president George_W._Bush , the White_House said Thursday . Brown was to arrive at Camp_David late Sunday , have dinner with Bush , " then there will be a pretty full meeting schedule the following day , " spokesman Tony_Snow said .
Seq2Attn: British prime minister Gordon_Brown has recalled the United_States for the highest United_States since two days of US president George_W._Bush 's close to the White_House , the White_House said Thursday . Brown , who on his cabinet reshuffle , arrived at the White_House later this week before returning to a dinner at Sunday 's session .
CopyNet: British prime minister Gordon_Brown returned to the the United_States head of two days before he will meet with US president George_W._Bush , the White_House said Thursday . Brown won the grand whereabouts at the weekend inauguration and his opponent was taking the g8 summit in this year 's retreat Sunday , said he would make his inauguration .
InjType (Ours): British prime minister Gordon_Brown will make his first official visit to the United_States two days after talks with new US president George_W._Bush , the White_House said Thursday . Brown has arrived at Camp_David on Sunday as spokesman Bush said in a brief statement . " today , something how long , " spokesman Tony_Snow said .
Input entities: China (COUNTRY, C), Russia (C), WTO (ORGANIZATION, O), Chinese (C), Wen_Jiabao (PERSON, P), Friday (WEEKDAY, W), Moscow (L), Interfax_news_agency (O), Wen (P), Russia (C), WTO (O), Russian (C), Vladimir_Putin (P), China (C).
Ground truth: China has agreed to back Russia's entry into the WTO, Chinese prime minister Wen Jiabao said Friday after talks in Moscow, the Interfax news agency reported . Wen said that an accord ending bilateral negotiations on Russia's WTO membership bid was expected to be signed next month when Russian president Vladimir Putin visits China.
Seq2Attn: China has lodged a veto global WTO plan to WTO the global trade body at its senior Chinese counterpart Wen Jiabao on Friday, Interfax news agency reported, as saying from Moscow, Interfax news agency said, quoted by the Interfax news agency. "It's very important to be injured by the (lower) ," he said, as he met with Putin by his Russian counterpart Vladimir Putin, saying the refusal .
CopyNet: China and Russia want to sell WTO trade sanctions against the World_Trade_Organisation , warning the proposed sanctions against the WTO agreement are at the WTO agreement we are concerned , giving Chinese energy issues , the report said after meeting with his meeting counterpart Chretien on Friday , while in Moscow , the Interfax_news_agency reported . it will be expected to attend the event , adding that he would be due to attend this week . " we are ready to hold any full decision to have made it very much , " he said .
InjType (Ours): China has agreed Russia's bid to join a WTO at the WTO opening, Chinese foreign minister Wen Jiabao said Friday Moscow, report quoted the state official from Interfax news agency. Wen will call on Russia backed a member of the WTO, in particular, he said. "I hope that the Russian government is capable," Vladimir Putin said in a statement from the China press conference.

Table 6: Our model can maintain the order of entities. The sequences of entities in the input (entity list) and output (news text) are consistent with *zero* difference.

Model	Entity level	*Levenshtein* error	Error rate
SeqAttn	Mention only	7.204 ± 7.857	98.3%
CopyNet	Mention only	4.800 ± 5.376	97.7%
SeqAttn	Mention $\oplus$ Type	2.683 ± 3.220	91.8%
InjType	Mention $\oplus$ Type	$\mathbf{0.000 \pm 0.000}$	**0.0%**

edits (addition, deletion, replacement) to make the two sequences consistent. As shown in Table 6, SeqAttn models and CopyNet have high error rate (i.e., percentage of inconsistent pairs). In contrast, InjType makes *zero* error.

Case Study and Limitation

Table 5 shows two cases in the test set to compare the generated results from different models. First, we observe that InjType has better *entity coverage* than baseline methods. In Seq2SeqAttn and CopyNet, some input entities are not generated in the output sequence such as "Camp_David" and "Tony_Snow", leading to a semantically incomplete sequence. InjType can retain the same order of the entities in the generated sequence. Second, InjType can generate *more precise contextual words* than baseline methods. For example, InjType generates "Gordon_Brown will make his first official visit to the United_States", which is more reasonable than "Gordon_Brown will return/recall the United_States" and is closer to the ground truth sequence. However, "as premier" is also very important in the sentence but InjType fails to prescribe a limit to the action of "his first official visit". This is mainly due to the lack of commonsense knowledge. Furthermore, InjType, as well as Seq2SeqAttn and CopyNet, often fails to generate a sentence containing complex entity relationships. In the second case, for example, each sentence contains six to eight entities, making the models hard to capture the interdependency when generating sequentially.

Conclusions

Entity plays the key role of making the text coherent in different NLG tasks. In order to enhance the role of entity in NLG, in this paper, we aim to model the entity type in the decoding phase to generate contextual words accurately. We propose a novel model (called *InjType*) that injects the entity types into the process of mention generation. The multi-step decoder first predicts the token of being a contextual word or an entity, then if an entity, predicts the entity mention. So it effectively embeds the entity's meaning into hidden states, making the generated words precise. Experiments on two public datasets demonstrate that type injection performs better than conventional type embedding concatenation.

References

[Amplayo, Lim, and Hwang 2018] Amplayo, R. K.; Lim, S.; and Hwang, S.-w. 2018. Entity commonsense representation for neural abstractive summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

[Bahdanau, Cho, and Bengio 2015] Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. *International Conference for Learning Representation*.

[Chan et al. 2019] Chan, Z.; Chen, X.; Wang, Y.; Li, J.; Zhang, Z.; Gai, K.; Zhao, D.; and Yan, R. 2019. Stick to the facts: Learning towards a fidelity-oriented e-commerce product description generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[Cho et al. 2014] Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing*.

[Clark, Ji, and Smith 2018] Clark, E.; Ji, Y.; and Smith, N. A. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2250–2260.

[Dinu et al. 2019] Dinu, G.; Mathur, P.; Federico, M.; and Al-Onaizan, Y. 2019. Training neural machine translation to apply terminology constraints. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3063–3068.

[Dong et al. 2019] Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified language model pre-training for natural language understanding and generation. In *In proceedings of Advances in Neural Information Processing Systems*.

[Fan, Lewis, and Dauphin 2018] Fan, A.; Lewis, M.; and Dauphin, Y. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

[Feng et al. 2018] Feng, X.; Liu, M.; Liu, J.; Qin, B.; Sun, Y.; and Liu, T. 2018. Topic-to-essay generation with neural networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 4078–4084.

[Finkel, Grenager, and Manning 2005] Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, 363–370. Association for Computational Linguistics.

[Garbacea and Mei 2020] Garbacea, C., and Mei, Q. 2020. Neural language generation: Formulation, methods, and evaluation. *arXiv preprint arXiv:2007.15780*.

[Graff et al. 2003] Graff, D.; Kong, J.; Chen, K.; and Maeda, K. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

[Grosz, Joshi, and Weinstein 1995] Grosz, B. J.; Joshi, A.; and Weinstein, S. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2):203–225.

[Gu et al. 2016] 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*.

[Guan et al. 2020] Guan, J.; Huang, F.; Zhao, Z.; Zhu, X.; and Huang, M. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*.

[Hu et al. 2017] Hu, Z.; Yang, Z.; Liang, X.; Salakhutdinov, R.; and Xing, E. P. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1587–1596.

[Iqbal and Qureshi 2020] Iqbal, T., and Qureshi, S. 2020. The survey: Text generation models in deep learning. *Journal of King Saud University-Computer and Information Sciences*.

[Ji et al. 2017] Ji, Y.; Tan, C.; Martschat, S.; Choi, Y.; and Smith, N. A. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

[Kanerva et al. 2019] Kanerva, J.; Rönnqvist, S.; Kekki, R.; Salakoski, T.; and Ginter, F. 2019. Template-free data-to-text generation of finnish sports news. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Li et al. 2018] Li, J.; Jia, R.; He, H.; and Liang, P. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1865–1874.

[Lu et al. 2018] Lu, D.; Whitehead, S.; Huang, L.; Ji, H.; and Chang, S.-F. 2018. Entity-aware image caption generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 4013–4023.

[Miao et al. 2019] Miao, N.; Zhou, H.; Mou, L.; Yan, R.; and Li, L. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[Nenkova 2008] Nenkova, A. 2008. Entity-driven rewrite for multi-document summarization. In *Proceedings of the Third International Joint Conference on Natural Language Processing*.

[Puduppully, Dong, and Lapata 2019] Puduppully, R.; Dong, L.; and Lapata, M. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

[Radford et al. 2019] Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1(8):9.

[Sandhaus 2008] Sandhaus, E. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6(12):e26752.

[Sharma et al. 2019] Sharma, E.; Huang, L.; Hu, Z.; and Wang, L. 2019. An entity-driven framework for abstractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3271–3282.

[Sutskever, Vinyals, and Le 2014] Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

[Wang and Wan 2018] Wang, K., and Wan, X. 2018. Sentigan: generating sentimental texts via mixture adversarial networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 4446–4452.

[Yang et al. 2019] Yang, P.; Li, L.; Luo, F.; Liu, T.; and Sun, X. 2019. Enhancing topic-to-essay generation with external commonsense knowledge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2002–2012.

[Yao et al. 2019] Yao, L.; Peng, N.; Weischedel, R.; Knight, K.; Zhao, D.; and Yan, R. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[Yu et al. 2018] Yu, T.; Li, Z.; Zhang, Z.; Zhang, R.; and Radev, D. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 588–594.

[Zhao et al. 2019] Zhao, G.; Fu, H.; Song, R.; Sakai, T.; Chen, Z.; Xie, X.; and Qian, X. 2019. Personalized reason generation for explainable song recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10(4):1–21.

[Zheng et al. 2017] Zheng, H.-T.; Wang, W.; Chen, W.; and Sangaiah, A. K. 2017. Automatic generation of news comments based on gated attention neural networks. *IEEE Access* 6:702–710.

Appendix

Statistics of Dataset

Our experiments are conducted on two public news datasets: Gigawords (Graff et al. 2003) and NYT (Sandhaus 2008). The Gigawords dataset contains around 4 million human-written news articles from various famous news publishers such as the New York Times and the Washington Posts from 1994 to 2010. The NYT dataset contains news articles written and published by New York Times from January, 1987 to June, 2007. It also contains 650,000 article summaries.

Since generating news articles from entity lists is a challenging task, we removed articles with length greater than 100 words from Gigawords and used news summaries with length smaller than 200 words from the NYT corpus. Besides, each news article has at least 8 entities. Statistical information is shown in Table 7. To obtain entity types, we first tokenize the article then apply the Stanford NER extraction method in CoreNLP (Finkel, Grenager, and Manning 2005). In total, there are 14 entity types: COUNTRY, LOCATION, PERSON, WEEKDAY, YEAR, MONTH, DAY, ORGANIZATION, TIMEUNIT, DIGIT, DIGITRANK, DIGITUNIT, TIMEUNIT, LENGTHUNIT. To ensure there's enough information from entities to produce a reasonable news article, we put a limit on the minimum number of entities and maximum length of news. We keep the articles with entity density greater than 5% and at least 5 entities from Gigawords.

Table 7: Statistics of two datasets.

	GIGAWORDS-6K	NYT-8K
#Articles	6,510	8,371
#Mentions $\lvert\mathcal{M}\rvert$	14,645	15,300
#Types $\lvert\mathcal{T}\rvert$	14	14
#Words $\lvert\mathcal{V}\rvert$	30,121	38,802
Len. Input/Output	14.0 / 86.1	10.6 / 78.6