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 Cognitive Services Research, Redmond, WA
xdong2ps@gmail.com, wyu1@nd.edu
chezhu@microsoft.com, mjiang2@nd.edu

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. 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 based on a given list of entities. Our model has a multi-step decoder that *injects* the entity types into the process of entity mention generation. Experiments on two public news datasets demonstrate type injection performs better than existing type embedding concatenation baselines.

1 Introduction

Entity, as an important element of natural language, plays the key role of making the text coherent (Grosz et al., 1995). Recently, modeling entities into NLG methods has demonstrated great improvements by assisting to infer the summary topic (Amplayo et al., 2018) or to generate coherent content (Ji et al., 2017; Clark et al., 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 et al., 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.

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 based on a given list of entities. 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, leading to serious incompleteness (Feng et al., 2018).

Table 1: An example of generating news with a list of names of entities and their types. How to use entity type information in NLG models is an open question.

Input: [COUNTRY:US$_1$, PERSON:Dick_Cheney$_2$, COUNTRY:Afghanistan$_3$, WEEKDAY:Monday$_4$, COUNTRY:Afghan$_5$, PERSON:Hamid_Karzai$_6$, ORGANIZATION:NATO$_7$, CITY:Bucharest$_8$]
Target: "US$_1$ vice president Dick_Cheney$_2$ made a surprise visit to Afghanistan$_3$ on Monday$_4$ for talks with Afghan$_5$ president Hamid_Karzai$_6$, ahead of the NATO$_7$ summit early next month in Bucharest$_8$."

In this paper, we propose a novel method of utilizing the type information in NLG, called *InjType*. It keeps the same encoder as Seq2Seq models (Sutskever et al., 2014). During decoding, it first predicts the probability that each token is a contextual 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

* The first two authors have equal contributions.
§ Our code and datasets are available at https://github.com/DM2-ND/InjType.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 734–741
November 7–11, 2021.

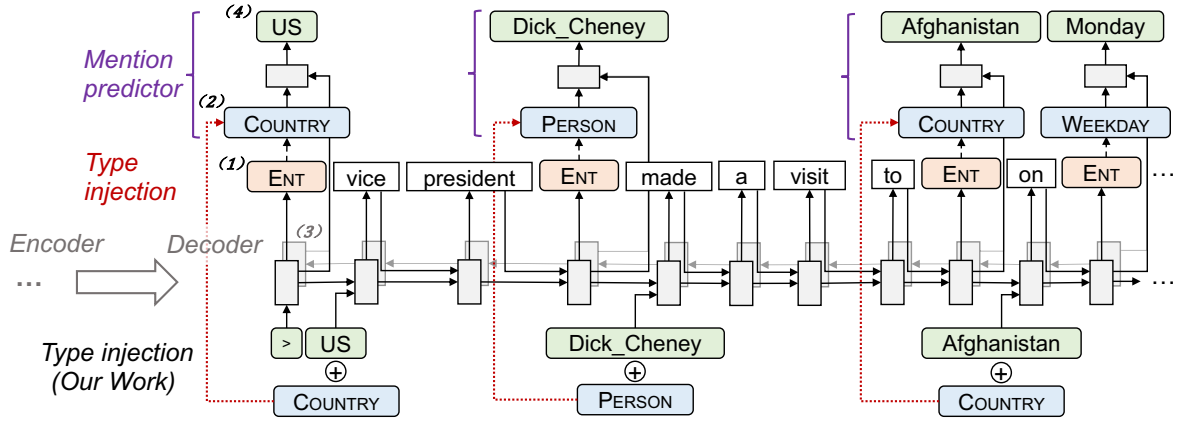

Figure 1: The **decoding** process of InjType has four steps: (S1) predicting the <Ent> token (i.e., entity indicator); (S2) injecting the entity types; (S3) combining an entity type enhanced NLU with backward information of target sequence; (S4) predicting the entity mention using the type embedding and hidden state by a mention predictor.

decoding hidden state. The type injection maximizes the likelihood of generating an entity indicator rather than the likelihood of sparse entity mentions. The hidden state is jointly optimized by predicting the role of token and predicting the entity mention so the entity's information is effectively embedded into the hidden states.

Experiments on two public news datasets GIGAWORDS and NYT demonstrate that InjType can generate more precise contextual words than the existing concatenation-based models.

2 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 et al., 2018), concept generation (Zeng et al., 2021), table description (Puduppully et al., 2019) and news generation (Yu et al., 2021). 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 et al., 2018). In table description, entity mentions have been used to achieve discourse coherence (Puduppully et al., 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 class into the decoding phase for better predicting entities and contextual words.

Words-to-text Generation. It is also referred as constraint text generation (Zhang et al., 2020; Qin et al., 2019). Generating text from topic words and keywords is a popular task in NLG. 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; Zhang et al., 2020), but also serves as an ideal test bed for controllable text generation (Wang and Wan, 2018; Yao et al., 2019). The main challenge of words-to-text lies in that the source information is extremely insufficient compared to the target output, leading to poor topic consistency in generated text (Yu et al., 2020).

3 Proposed Method: *InjType*

In this section, we first give the task definition, then introduce our proposed type injection method. **We note that** *InjTyp* users the same encoder as in Seq2Seq models (Sutskever et al., 2014), i.e., a bi-directional GRU. So, in Figure 1 and the following sections, we only describe the decoding process.

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 type. The expected output sequence is $y = (y_1, \ldots, y_m)$ containing all the entity mentions. 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.

3.1 Entity Indicator Predictor

At each step, the decoder predicts either an entity indicator or a contextual word. An entity indicator, denoted as <Ent>, 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 <Ent> generated in the output sequence. So the first-step output sequence is:

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

Each block has one or multiple contextual words, and it ends with an entity indicator (<Ent>). In each block, the generation process is the same as the auto-regressive decoding process. When the auto-regressive decoder generates an <Ent>, the generation process of the current block ends. When the decoder generated the $(n+1)$-th entity indicator <Ent>, the entire generation terminates.

Suppose the ground truth of entity indicator output y'' is the target sequence with entity mentions replaced with entity indicators <Ent>. Now, the loss function with entity indicator is defined as:

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

3.2 Mention Predictor

Since each block's generation is ended with the entity indicator token (<Ent>), the representations of the last hidden states in different blocks are assimilated, which may lose contextual information in previous generated tokens. In order to let the decoding hidden states carry rich contextual information and better predict the entity mention, we present a novel mention predictor by *injecting* the entity type embedding into current hidden state, and feed the combined embedding into a mention classifier. In i-th block, the predicted entity mention is $x^M_{i''} = \text{softmax}(\mathbf{W}_m \cdot [\mathbf{s}_t \oplus \mathbf{x}^T_i])$, where $\mathbf{s}_t$ is the hidden state of the t-th token in the generated text, and $\mathbf{x}^T_i$ is the i-th entity type embedding. In this way, the last hidden state in each block not only has to be classified as an entity indicator (<Ent>), but also carries both entity type and entity mention information in order to make precise generation. The classification loss $\mathcal{L}_{MP}$ can be written as:

$$\mathcal{L}_{\text{MP}} = -\sum_{i=1}^{n} x^M_i \cdot \log(x^M_{i''}),$$

3.3 Entity type-Enhanced NLU

Inspired by the UniLM (Dong et al., 2019), 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 during training. The entity type enhanced NLU task asks the decoder to predict entity mentions corresponding to the types in the 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 $\overleftarrow{\text{GRU}}$. We reuse the original NLG decoder without attention, denoted by $\overrightarrow{\text{GRU}'}$ for the NLU task. This module generates the prediction as follows:

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

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 as $x^M_{i''} = \text{softmax}(\mathbf{W}_r \cdot \mathbf{s}'_t)$. So the NLU loss is only calculated at the entity positions:

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

3.3.1 Joint Optimization

InjType jointly optimizes the following loss:

$$\mathcal{L} = \mathcal{L}_{Ent} + \lambda_1 \cdot \mathcal{L}_{MP} + \lambda_2 \cdot \mathcal{L}_{NLU}, \quad (1)$$

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

4 Experiments

Table 2: Statistics of two datasets. Additional information is in Section 6.1 and Table 6 in Appendix.

	GIGAWORDS-6K	NYT-8K
#Articles	7,903	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

4.1 Experimental Settings

Datasets. We conduct experiments on two news datasets: GIGAWORDS (Graff et al., 2003), NYT (Sandhaus, 2008). Statistics can be found in Table 6 in Appendix. To obtain entity types, we first tokenize the article then apply the Stanford NER extraction method in CoreNLP (Finkel et al., 2005). In total, there are 14 entity types: COUNTRY, LOCATION, PERSON, WEEKDAY, YEAR, MONTH,

Table 3: Our *InjType* can outperform various baseline models enhanced by type embedding concatenation.

Methods	GIGAWORDS			NYT		
	ROUGE-2	ROUGE-L	BLEU-4	ROUGE-2	ROUGE-L	BLEU-4
Seq2Seq	8.83±0.15	31.43±0.13	12.21±0.30	8.83±0.15	31.43±0.13	12.21±0.30
SeqAttn	9.10±0.13	36.62±0.11	16.17±0.28	5.95±0.15	29.67±0.06	11.86±0.15
CopyNet	9.44±0.11	36.96±0.10	16.40±0.24	6.25±0.14	30.58±0.09	11.96±0.14
GPT-2	9.04±0.20	31.30±0.16	15.66±0.40	5.86±0.20	24.19±0.14	10.89±0.22
UniLM	11.77±0.18	36.54±0.15	17.66±0.35	7.47±0.15	30.66±0.13	12.90±0.20
InjType	**13.37±0.12**	**41.16±0.31**	**18.55±0.09**	**8.55±0.09**	**31.53±0.17**	**13.14±0.03**
⊢ w/o MP	9.39±0.16	38.34±0.10	16.36±0.25	6.52±0.09	30.10±0.08	12.19±0.10
⊢ w/o NLU	12.85±0.18	40.65±0.37	18.24±0.26	8.13±0.10	30.80±0.36	13.10±0.09

Table 4: Human Evaluations on GIGAWORD: *InjType* (ours) v.s. Seq2Seq attention with type concatenation.

	Win	Tie	Lose
Grammar	**25.2%**	50.4%	24.4%
Fluency	**24.8%**	52.6%	22.6%
Coherence	**48.0%**	21.4%	30.6%
Informativeness	**50.6%**	21.2%	28.2%

DAY, ORGANIZATION, TIMEUNIT, DIGIT, DIGITRANK, DIGITUNIT, TIMEUNIT, LENGTHUNIT.

Baselines. We compare with three Seq2Seq methods (i.e., Seq2Seq (Sutskever et al., 2014), SeqAttn (Bahdanau et al., 2015), CopyNet (Gu et al., 2016)), and two pre-trained language models (i.e., GPT-2 (Radford et al., 2019), UniLM (Dong et al., 2019)). **It should be noted that** all Seq2Seq baselines are implemented with the concatenation of entity mention and entity types. We fine-tune GPT-2 and UniLM on the training set for 20 epochs. Since our model is not pre-trained on large corpora and has much less model parameters, competing with GPT-2 and UniLM is very challenging.

Implementation Details We take 256 as dimension size of hidden states for GRU encoder and decoder. The word embedding size is 300. We use Adam optimizer (Kingma and Ba, 2015) with learning rate of 1e-4. We trained our model for 60 epochs on an NVIDIA 2080-Ti GPU. We did grid search on the hyperparameters in loss Eq.(1) and got the best performance at $\lambda_1 = \lambda_2 = 2$.

Evaluation metrics. The performance is measured by standard corpus-level metrics, including BLEU (Papineni et al., 2002), ROUGE (Lin, 2004).

4.2 Experimental Results

Tables 3 compares our *InjType* with competitive baseline methods on three datasets. As shown in the table, *InjType* performs the best among all methods. This demonstrates that our proposed multi-step decoder is a more effective strategy than simple entity type embedding concatenation.

Table 5 shows a case in the test set to compare the generated results from different models. We observe our model can generate *more precise contextual words* than baseline methods.

We compare our model with its variants in Table 3. We observe that the mention predictor (MP) contributes more than NLU modules. Adding MP improves BLEU-4 by +1.65%, while adding NLU improves BLEU-4 by +0.26%. 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.

4.3 Human Evaluation

We sample 50 examples from GIGAWORD test. Every generated news is presented to 5 annotators on Amazon Mechanical Trunk (AMT). Annotators were presented with two news articles and asked to decide which one was better and which one was worse in order of grammar and fluency, coherence, and informativeness. The result is "win", "lose" or "tie". Table 4 demonstrates the human evaluation results. *InjType* can significantly outperform Seq2Seq attention with type concatenation on coherence and informativeness. So, entity type injection is a more effective way to leverage entity type information than a simple concatenation way.

5 Conclusions

Entity plays the key role of making the text coherent in news generation. In order to enhance the role

Table 5: Case study. Our proposed *InjType* can generate more precise contextual words than baseline methods.

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)
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** .
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. [The second "WTO" should be "Russia". "Interfax" appeared multiple times.]
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 , the report said after meeting with his meeting counterpart Wen_Jiabao on Friday in Moscow , the Interfax_news_agency reported .
***InjType* (Our method):** China **has agreed** Russia's bid to join the WTO opening, Chinese foreign minister Wen Jiabao said Friday Moscow, **report quoted the state official from** Interfax news agency.

of entity, we propose a novel multi-step decoder that can effectively embed the entity's meaning into decoding hidden states, making the generated words precise. Experiments on two news datasets demonstrate that our method can perform better than conventional type embedding concatenation.

Acknowledgements

This work is supported by National Science Foundation IIS-1849816 and CCF-1901059.

References

Reinald Kim Amplayo, Seonjae Lim, and Seung-won Hwang. 2018. Entity commonsense representation for neural abstractive summarization. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference for Learning Representation (ICLR)*.

Zhangming Chan, Xiuying Chen, Yongliang Wang, Juntao Li, Zhiqiang Zhang, Kun Gai, Dongyan Zhao, and Rui Yan. 2019. Stick to the facts: Learning towards a fidelity-oriented e-commerce product description generation. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 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 (NAACL-HLT)*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems (Neruips)*.

Xiaocheng Feng, Ming Liu, Jiahao Liu, Bing Qin, Yibo Sun, and Ting Liu. 2018. Topic-to-essay generation with neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Annual meeting on association for computational linguistics (ACL)*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Barbara J Grosz, Aravind Joshi, and Scott Weinstein. 1995. A framework for modeling the local coherence of discourse. *Computational Linguistics*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Annual Meeting of the Association for Computational Linguistics*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference for Learning Representation (ICLR)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.

Ani Nenkova. 2008. Entity-driven rewrite for multi-document summarization. In *International Joint Conference on Natural Language Processing*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Annual meeting on association for computational linguistics (ACL)*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lianhui Qin, Michel Galley, Chris Brockett, Xiaodong Liu, Xiang Gao, William B Dolan, Yejin Choi, and Jianfeng Gao. 2019. Conversing by reading: Contentful neural conversation with on-demand machine reading. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*.

Eva Sharma, Luyang Huang, Zhe Hu, and Lu Wang. 2019. An entity-driven framework for abstractive summarization. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (Neruips)*.

Ke Wang and Xiaojun Wan. 2018. Sentigan: generating sentimental texts via mixture adversarial networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Pengcheng Yang, Lei Li, Fuli Luo, Tianyu Liu, and Xu Sun. 2019. Enhancing topic-to-essay generation with external commonsense knowledge. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020. A survey of knowledge-enhanced text generation. *arXiv preprint arXiv:2010.04389*.

Wenhao Yu, Chenguang Zhu, Tong Zhao, Zhichun Guo, and Meng Jiang. 2021. Sentence-permuted paragraph generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Qingkai Zeng, Jinfeng Lin, Wenhao Yu, Jane Cleland-Huang, and Meng Jiang. 2021. Enhancing taxonomy completion with concept generation via fusing relational representations. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*.

Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. 2020. Pointer: Constrained text generation via insertion-based generative pre-training. *In Empirical Methods in Natural Language Processing (EMNLP)*.

Guoshuai Zhao, Hao Fu, Ruihua Song, Tetsuya Sakai, Zhongxia Chen, Xing Xie, and Xueming Qian. 2019. Personalized reason generation for explainable song recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

Hai-Tao Zheng, Wei Wang, Wang Chen, and Arun Kumar Sangaiah. 2017. Automatic generation of news comments based on gated attention neural networks. *IEEE Access*.

6 Appendix

Table 6: Statistics of two datasets.

	GIGAWORDS	NYT
#Articles	7,903	8,371
# Training	5,903	6,371
# Validation	1,000	1,000
# Test	1,000	1,000
#Mentions $\|\mathcal{M}\|$	14,645	15,300
#Types $\|\mathcal{T}\|$	14	14
#Words $\|\mathcal{V}\|$	30,121	38,802
Len. Input/Output	14.0 / 86.1	10.6 / 78.6
Output Sentences	4.1	3.8

6.1 Additional Dataset Information

We create two datasets from public sources: 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. 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.

To obtain entity types, we first tokenize the article then apply the Stanford NER extraction method in CoreNLP (Finkel et al., 2005). In total, there are 14 entity types: COUNTRY, LOCATION, PERSON, WEEKDAY, YEAR, MONTH, DAY, ORGANIZATION, TIMEUNIT, DIGIT, DIGITRANK, DIGITUNIT, TIMEUNIT, LENGTHUNIT.

6.2 Entity 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 et al., 2015) to generate output sequence. Specifically, 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 et al., 2005). So the input embedding of the i-th entity is defined as:

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

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:

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

In the *decoding* phase, another GRU serves as the model decoder to generate entity mention and contextual words to generate the target sequence $\mathbf{s}_t = \overrightarrow{\mathrm{GRU}}(\mathbf{y}_t, \mathbf{s}_{t-1}, \mathbf{c}_t)$, where $\mathbf{y}_t$ is the concatenation of entity mention embedding $\mathbf{y}_i^M$ with the embedding of its corresponding type $\mathbf{y}_i^T$ (as shown in Figure 2). So, given the current decoding hidden state $\mathbf{s}_t$ and the source-side attentive context vector $\mathbf{c}_t$, the readout hidden state is defined as $\mathbf{r}_t = \tanh(\mathbf{W}_c \cdot [\mathbf{s}_t \oplus \mathbf{c}_t])$, 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 $\mathbf{e}_{t,i}$ are then normalized to get the current context vector $\mathbf{c}_t$ by weighted sum:

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

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

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):

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

The loss function is defined as:

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

Figure 2 is a reference to the model design.

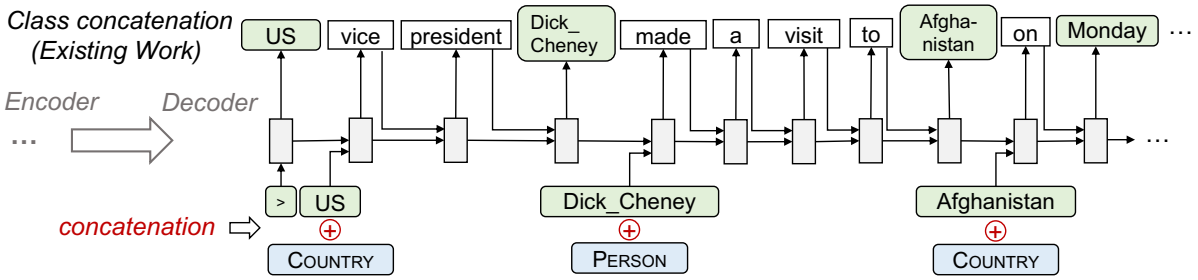

Figure 2: Concatenating entity mention embeddings and type embeddings is a straightforward strategy to use the type information. However, it may not be effective due to lack of contextual information in the hidden states. ***Note*** that the figure only highlights the decoder. Attention mechanism is also employed, but is not shown in the figure.

Table 7: Case study. Our proposed *InjType* can generate more precise contextual words than baseline methods.

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* (Our method):** 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 .

6.3 Baseline Methods

Seq2Seq (Sutskever et al., 2014) It is the basic encoder-decoder model widely used in NLG tasks.

SeqAttn (Bahdanau et al., 2015) It adds a soft attention mechanism to the input sequence where the most relevant information is concentrated.

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

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

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

6.4 Evaluation Metrics

We use four kinds of standard metrics: (1) BLEU-4 measures the average 4-gram precision on a set of reference texts; (2) ROUGE-2 computes the recall of bigrams; (3) ROUGE-L computes the overlap of the longest common subsequence between the hypothesis and the reference;

6.5 Human Evaluation Settings

We sample 50 inputs from Gigaword test set, and generate news articles by type embedding concatenation and our proposed *InjType*. Annotators were presented with two news articles and asked to decide which one was better and which one was worse in order of grammar and fluency (is the news written in well-formed English?), coherence (does the news describe the whole event coherently?), and informativeness (does the news relate to the input entities?). The result is "win", "lose" or "tie". We provided good and bad examples and explain how they succeed or fail to meet the defined criteria. Every generated news is presented to 5 annotators on Amazon Mechanical Trunk (AMT).