Controllable Neural Natural Language Generation: comparison of state-of-the-art control strategies

Yuanmin Leng, François Portet, Cyril Labbé
Univ. Grenoble Alpes, CNRS, Grenoble INP,
LIG, F-38000 Grenoble, France
`yuanmin.leng@outlook.com`,
`francois.portet@imag.fr`,
`cyril.labbe@imag.fr`

Raheel Qader
Lingua Custodia, Paris, France
`raheel.qader@gmail.com`

Abstract

Most NLG systems target text fluency and grammatical correctness, disregarding control over text structure and length. However, control over the output plays an important part in industrial NLG applications. In this paper, we study different strategies of control in triple-to-text generation systems particularly from the aspects of text structure and text length. Regarding text structure, we present an approach that relies on aligning the input entities with the facts in the target side. It makes sure that the order and the distribution of entities in both the input and the text are the same. As for control over text length, we show two different approaches. One is to supply length constraint as input while the other is to force the end-of-sentence tag to be included at each step when using top-k decoding strategy. Finally, we propose four metrics to assess the degree to which these methods will affect a NLG system's ability to control text structure and length. Our analyses demonstrate that all the methods enhance the system's ability with a slight decrease in text fluency. In addition, constraining length at the input level performs much better than control at decoding level.

1 Introduction

If many researches have focused on end-to-end Neural NLG (NNLG), approaches depending on a pipeline architecture, where the generation can be divided into multiple steps and the steps are accomplished separately using either neural networks or other models, are still competitive (Nayak et al., 2017; Reed et al., 2018; Balakrishnan et al., 2019; Ferreira et al., 2019). Besides improvement in text fluency and grammatical correctness, succeeding in controlling different aspects of text such as style, structure and length is a key enabler for reliable systems fit for industrial applications and for better understanding of NNLG models.

In end-to-end models, control can be applied at various stages of the neural generation process such as input, hidden states and decoding (Prabhumoye et al., 2020). Plug and Play Language Model (PPLM) proposed by (Dathathri et al., 2019) takes an external input and performs computations on hidden states. It combines a pretrained LM with one or more simple attribute classifiers that guide text generation towards the desired topic or sentiment. Gehrmann et al., 2018 describe a training method based on diverse ensembling to encourage models to learn distinct text styles. Welleck et al., 2020 intervene during the decoding stage to relieve the problem of infinite generation. The end-to-end models can be equipped with the ability to control style and length.

In pipeline systems, control can be applied to various steps especially the step of text planning where controlling text content is the main objective. Mille et al., 2019 apply a series of rule-based graph-transducers and aggregation grammar while Moryossef et al., 2019 and Ferreira et al., 2019 allow RDF input triples to be rearranged in line with target.

To better understand the effect of such control in black-box models, it is necessary to look into state-of-the-art control strategies through independent quantitative analyses. In this paper, we focus on controlling text structure and text length for RDF triple-to-text generation. Text structure consists in sentence split and entity order which care about how the facts are distributed in the sentences and in what relative order the entities are stated. Text length is the word count of a produced text. In this work, we assume a two-step pipeline models which consist of text planning and realization (Moryossef et al., 2019). A text plan generated from the input facts in the first step is feeding the second step of realization. We make two main contributions.

First, a systematic study of how the method of

3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+),
Dublin, Ireland (Virtual), 18 December 2020, pages 34–39, ©2020 Association for Computational Linguistics

alignment affects control over text structure. The method aligns input with target in terms of sentence split and entity order. Furthermore, we propose three metrics to evaluate its effects.

Second, a study of the text length control using two different approaches. One approach, applied to sentence planning, is to add the length constraint into input while the other, applied to surface realization, is to force the end-of-sentence tag to be included at each step when using top-k decoding strategy (Welleck et al., 2020). We design one metric to evaluate the effects of the two different methods to control text length. None of the metrics we propose is used to optimize model training.

2 Method

In this section, we discuss the methods of controlling sentence split, entity order and text length.

2.1 Sentence split

Control of sentence split consists in determining the number of sentences and how the facts will be distributed in the sentences.

Similar to Moryossef et al., 2019; Ferreira et al., 2019, the method of controlling sentence split is to organize triples into sentences. However, RDF triples of training set are not provided with such syntactic features. For instance, let's take the following triples and target text.

Soundplate foundedIn **2010** **Soundplate** products **website** **Soundplate** products **record label** **Soundplate** foundedBy **Matt Benn**	**Soundplate** is a London-based **record label** and music platform. Originally founded in **2010** by **Matt Benn**, **Soundplate** started as a **website** covering all aspects of dance music and all genres of the global scene therein.

In this example, the triples should be split into two groups, and each group of entities should be consistent with those in the corresponding sentence. To recover the information about sentence boundaries, the entities should first be identified. The identification works by generating all n-grams of the text and comparing each of them with each entity based on Levenshtein distance. The n-gram whose distance from the entity is the smallest is seen as the appearance of the entity in the text. This method has been tested with a spare dataset of 15370 entities and is able to recover 99.18% of the entities (15245). The input after this step is as follow where ⟨SNT⟩ is the end-of-sentence token:

Soundplate products **record label** ⟨SNT⟩ **Soundplate** foundedIn **2010** **Soundplate** products **website** **Soundplate** foundedBy **Matt Benn** ⟨SNT⟩	**Soundplate** is a London-based **record label** and music platform ⟨SNT⟩ Originally founded in **2010** by **Matt Benn**, **Soundplate** started as a **website** covering all aspects of dance music and all genres of the global scene therein ⟨SNT⟩

We propose two metrics to evaluate the effects of this method. The first metric takes into account consistency between input and output in sentence count. It calculates the percentage of the produced texts which contain the same sentence count as that of inputs.

$$P = \frac{1}{N}\sum_{i=1}^{N} bool(s\_count(input_i) == s\_count(output_i)) \tag{1}$$

where $s\_count(x)$ returns sentence count of x.

The second metric calculates the percentage of the entities that are distributed into the right sentences, irrespective of the order.

$$P = \frac{1}{N}\sum_{i=1}^{N} \frac{|entities\ correctly\ distributed|}{|entities|} \tag{2}$$

Let's take the example of input facts and output text below.

Andrews County Airport location **Texas** ⟨SNT⟩ **Texas** language **Spanish** ⟨SNT⟩	**Texas** is the location of **Andrews County Airport**, where **Spanish** is spoken ⟨SNT⟩

In this example, the facts are described in two sentences in the input but the system only produced one sentence as output. Indeed, four entities of the two RDF triples in the input must be expressed, each in one sentence. But in the output only three entities are realised in only one sentence. The result of the second metric is therefore 0.5 since only one 2 entities are correctly realised over the four of the input.

2.2 Entity Order

Control of entity order involves defining in what relative order the entities will be produced.

The method of controlling entity order is to align input with target, ensuring that entity order in input is consistent to that in target. For our example mentioned in 2.1, the text plan will be:

Soundplate [> products > **record label**] ⟨SNT⟩ [**2010** < founded in <] [**Matt Benn** < founded by <] **Soundplate** [> products > **website**] ⟨SNT⟩

For datasets where no triples overlap, each group of triples is considered as an undirected graph and each entity as a node. Following Moryossef et al., 2019, we use depth-first search from each node to traverse the whole graph. For datasets where some triples share a same entity, the common entity is first located and then we make it direct towards the rest of the entities.

We propose one metric to evaluate effects of the alignment constraint. The metric is based on calculation of similarity between two lists of entities using edit distance. Edit distance is the minimum number of operations on entities (including deleting an entity, inserting an entity or swapping the positions of two entities) required to transform one list into the other.

$$similarity = 1 - \frac{edit\_distance(l_1, l_2)}{max(|l_1|, |l_2|)} \quad (3)$$

l_1 and l_2 represent lists of entities extracted respectively from text plan and its corresponding text.

Final scores of the metric are mean similarities.

$$P = \overline{similarity} = \frac{1}{N}\sum_{i=1}^{N}(similarity) \quad (4)$$

where N means the total number of text plans. For example, given the text plan and the output:

Agnes Kant [> nationality > **Netherlands** [> leader name > **Mark Rutte**]] ⟨SNT⟩ **Netherlands** [> currency > **euro**] ⟨SNT⟩	**Agnes Kant** has the nationality of **Netherlands** where the leader is **Mark Rutte** and the currency is the **euro** ⟨SNT⟩

The second *Netherlands* is omitted in the output, so the edit distance between input and output is 1 and the similarity 0.8 since one of the four input entities is omitted.

2.3 Text length

Control of text length is to constrain the output to contain len words. To implement this functionality two methods have been used.

The first method works by inserting the text length constraint len into the input. At test time, len is predicted by a linear regression model. This model was trained using triple count and word count of text plan as input. The second method, proposed by Welleck et al., 2020, forces the end-of-sentence tag to be included at each step of decoding after the generated text already reaches the given length.

To evaluate performances of the two approaches, we use mean squared error to compute difference between expected output length and actual output length:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(|tokens\ in\ output| - len)^2 \quad (5)$$

3 Experiments

Dataset	Train	Dev	Test
Wikipedia Company	34043	3785	4167
WebNLG	34411	4325	1589

Table 1: Statistics about the two datasets used in the study

3.1 Datasets

The approaches were experimented on two datasets: the Wikipedia Company corpus (Qader et al., 2018) and WebNLG 2020 dataset (Ferreira et al., 2020). Table 1 shows statistics about the two datasets. Wikipedia Company corpus is collected from Wikipedia without manual annotation, where some of input facts are not described in target while target contains facts which are not mentioned in input. To denoise the corpus, we delete the triples having entities that do not appear in target and drop the sentences without any entity in them. However, a large quantity of useless information still exists in target. As a result, we do not use it to perform experiments on control over text length. Since only the training set of WebNLG 2020 Challenge was released at the time of writing we used the test set from WebNLG 2017 dataset (Gardent et al., 2017) for system evaluation. We made sure that none of the instances in the test set was seen while training. Hence, when we report results on WebNLG it means WebNLG 2020 as training set and WebNLG 2017 as test set.

Control of sentence split & entity order We propose two systems: one system (aligned system) uses the train set processed by the methods described in Section 2.1 and 2.2, while the other (baseline system) is trained on the dataset where triples are combined together without considering

System	BLEU	ROUGE-L	METEOR	Sentence count	Entity distribution	Entity order
Aligned system	0.0681	0.3162	0.2689	**0.9921**	**0.7639**	**0.6915**
Baseline	**0.1023**	**0.3329**	**0.3076**	0.5731	0.6922	0.5348

Table 2: Control of sentence split and entity order: Wikipedia Company corpus

System	BLEU	ROUGE-L	METEOR	Sentence count	Entity distribution	Entity order
Aligned system	0.5116	0.6145	0.6291	**0.9937**	**0.9301**	**0.906**
Aligned system (random plan)	0.5021	0.5927	0.616	0.8232	0.8443	0.8117
Baseline	**0.5696**	**0.7035**	**0.6774**	0.6891	0.8181	0.4556

Table 3: Control of sentence split and entity order: WebNLG dataset (2020 as training and 2017 as testing)

consistency to target. Input of the test set for both systems is produced by the same way as we generate input of the train set for the baseline system. Additionally, with the aim of further investigating the aligned system's ability, we also generated a another test set with the same examples but where input triples are randomly split into several groups and each group of triples are randomly merged (e.g., random plan).

Control of text length We propose three systems: the first system (System 1) uses the train set where text length is prepended into input; in the second system (System 2), the end-of-sentence tag is forced to be included at each step when using top-k strategy; the last system is the baseline system.

3.2 Models

The model of all systems consists of a 4-layer Transformer with 4-head attention (Vaswani et al., 2017) on both the encoder/decoder. All experiments were performed using the OpenNMT toolkit (Klein et al., 2017).

3.3 Results

Besides the metrics presented in Section 2, standard automatic measures BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004) and METEOR (Denkowski and Lavie, 2014) were also used to evaluate the variations of performance when applying various controls. We use the program proposed by E2E NLG Challenge[1] to compute scores of the automatic metrics.

Control of sentence split & entity order Table 2 and Table 3 show the comparison between the aligned system and the baseline system on the same test set. The aligned system outperforms the baseline system. The former achieves nearly full marks regarding consistency in sentence count, which reveals its great capacity of controlling sentence count when there is only one sentence in text plan. For example, given the outputs of the aligned system and the baseline system,

Plan: **Canada** [< birth place < **Alex Plante** [> height > **1.9304**] [> club > **Anyang Halla**]] ⟨SNT⟩
Aligned system: **Canadian** born, **Alex Plante**, is **1.9304m** tall and has played for the club **Anyang Halla** ⟨SNT⟩
Baseline system: **Alex Plante** was born in **Canada** and is **1.9304** m. tall ⟨SNT⟩ **He** played for the club **Anyang Halla** ⟨SNT⟩

The aligned system succeeds in producing texts consistent to plans while the baseline system fails. The results of experiments performed on random plans in Table 3 make a stronger case for the ability of the aligned system. However, models learned with aligned data have lower scores in corpus-based metrics. The low scores in automatic metrics (BLEU, ROUGE-L and METEOR) can be explained by the fact that the baseline system – not being constrained by a plan during training – can learn to arrange output in a way that better fits the corpus on which it has been trained.

Control of text length Table 4 shows a gap among the three systems. Although both systems 1 & 2 improve the MSE metric, the method of prepending length constraint (System 1) clearly outperforms the method applied at decoding time. For example, here are three diversified outputs from the systems:

Plan of System 1: 10 **Eberhard van der Laan** [< leader < **Amsterdam**] ⟨SNT⟩
Plan of System 2 & Baseline system: **Eberhard van der Laan** [< leader < **Amsterdam**] ⟨SNT⟩

[1] https://github.com/tuetschek/e2e-metrics

System	BLEU	ROUGE-L	METEOR	Sentence count	Entity distribution	Entity order	MSE
System 1	0.4430	0.5692	0.6047	0.8301	0.8314	0.8102	**8.11**
System 2	0.4386	0.5652	0.5999	0.8464	0.8212	0.7916	25.12
Baseline	**0.4665**	**0.5801**	**0.6121**	**0.8471**	**0.846**	**0.8123**	25.90

Table 4: Control of text length: WebNLG dataset (random)

System 1: **Eberhard van der Laan** is the leader of **Amsterdam** ⟨SNT⟩
System 2: **Eberhard van der Laan** is the name of the leader , ⟨SNT⟩
Baseline system: **Eberhard van der Laan** is the name of the leader of **Amsterdam** ⟨SNT⟩

System 1 generates a fluent text of the given number of words which contains all facts. In contrast, System 2 produces a text unfaithful to input, stopping the generation ‘roughly’ when there are ten tokens in output.

4 Conclusion and further-work

In this study, we demonstrate that the alignment between the input and the target regarding sentence split and entity order leads to a substantial increase in the ability of NNLG models to control text structure. As for control of text length, we show that a control at the input level is highly preferable over a control at decoding level since it gives the model the opportunity to integrate the length constraint during the processing to avoid ending up generating an incomplete text. As shown in the study, different types of control do not seem independent (e.g. length influences number of sentences). The next step is to get more insight about these interdependencies.

References

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained Decoding for Neural NLG from Compositional Representations in Task-Oriented Dialogue. In *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussalem, and Anastasia Shimorina. 2020. Webnlg challenge 2020. https://webnlg-challenge.loria.fr/challenge_2020.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. 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)*, pages 552–562.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander M Rush. 2018. End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Simon Mille, Stamatia Dasiopoulou, and Leo Wanner. 2019. A portable grammar-based nlg system for verbalization of structured data. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1054–1056.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of NAACL-HLT*, pages 2267–2277.

Neha Nayak, Dilek Hakkani-Tur, Marilyn Walker, and Larry Heck. 2017. To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. In *Proc. of Interspeech*.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. 2020. Exploring controllable text generation techniques. *arXiv preprint arXiv:2005.01822*.

Raheel Qader, Khoder Jneid, François Portet, and Cyril Labbé. 2018. Generation of company descriptions using concept-to-text and text-to-text deep models: dataset collection and systems evaluation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 254–263.

Lena Reed, Shereen Oraby, and Marilyn Walker. 2018. Can Neural Generators for Dialogue Learn Sentence Planning and Discourse Structuring? In *Proc. of the 11th International Conference on Natural Language Generation (INLG)*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sean Welleck, Ilia Kulikov, Jaedeok Kim, Richard Yuanzhe Pang, and Kyunghyun Cho. 2020. Consistency of a recurrent language model with respect to incomplete decoding. *arXiv preprint arXiv:2002.02492*.