Fine-Grained Control of Sentence Segmentation and Entity Positioning in Neural NLG

Kritika Mehta **Raheel Qader** **Cyril Labbé** **François Portet**
Univ. Grenoble Alpes, LIG
38000 Grenoble, France
kritika.mehta@grenoble-inp.org
raheel.qader@univ-grenoble-alpes.fr
{cyril.labbe,francois.portet}@imag.fr

Abstract

The move from pipeline Natural Language Generation (NLG) approaches to neural end-to-end approaches led to a loss of control in sentence planning operations owing to the conflation of intermediary micro-planning stages into a single model. Such control is highly necessary when the text should be tailored to respect some constraints such as which entity to be mentioned first, the entity position, the complexity of sentences, etc. In this paper, we introduce fine-grained control of sentence planning in neural data-to-text generation models at two levels - realization of input entities in desired sentences and realization of the input entities in the desired position among individual sentences. We show that by augmenting the input with explicit position identifiers, the neural model can achieve a great control over the output structure while keeping the naturalness of the generated text intact. Since sentence level metrics are not entirely suitable to evaluate this task, we used a metric specific to our task that accounts for the model's ability to achieve control. The results demonstrate that the position identifiers do constraint the neural model to respect the intended output structure which can be useful in a variety of domains that require the generated text to be in a certain structure.

1 Introduction

Typical NLG models are characterized by a pipeline of stages (Walker et al., 2007; Barzilay and Lapata, 2006; Walker et al., 2002; Stent, 2002; Barzilay and Lee, 2002; Langkilde and Knight, 1998; Reiter and Dale, 1997). This approach can be conceptually divided into solving two questions: *what to say?* aka content determination and planning, and *how to say it?* aka text realization (Gatt and Krahmer, 2018). In contrast, end-to-end NLG systems combine these stages in a single end-to-end learning framework. Recently, there has been a lot of interest in combining sentence planning and realization stage into a single neural model (Nayak et al., 2017; Dušek and Jurčíček, 2016; Lampouras and Vlachos, 2016; Wen et al., 2015; Mei et al., 2015). Although this resulted in some improvement at the grammatical level, in neural natural language generation this led to a loss of control that was otherwise possible in the pipeline approaches.

Neural NLG systems struggle to produce a consistent order of entities and are sometimes not faithful to the input by either hallucinating, omitting or repeating the entities (Moryossef et al., 2019). They do not allow control over the output structure and while they exhibit impressive levels of fluency, they are less equipped to deal with higher levels of text structuring in a consistent manner. They are also unable to generalize sentence planning operations beyond what is seen in the training. It is therefore important to introduce explicit control in neural NLG so that the output is faithful to the input. In this way, the system would be able to generate diverse realizations making way for explicit control over the output text structure.

By controlling the facts in the generated text, different variations can be produced that emphasize a particular fact which is more important than others. For example, if the focus should be on *"a cheap italian place"*, then *"There is a cheap italian place called The Sorrento. It is located in the city center."* will be more appropriate than *"The Sorrento is located in the city center. It is an Italian Restaurant. It is cheap too"*. This is particularly helpful in different domains, for instance, when generating hotel review summaries, it is important to put the elements important for the user in front (e.g., family, bathroom etc), when generating company descriptions, it is important to put the

Proceedings of the 1st Workshop on Discourse Structure in Neural NLG, pages 18–23
Tokyo, Japan, November 1, 2019.

(1) **Alignment information**	"name[Blue Spice], eatType[coffee shop], area[city centre]", A coffee shop in the city centre area called Blue Spice.,(2: eattype) (21: area) (45: name)
(2) **Annotated reference text**	a (eattype)coffee shop in the (area)city centre area called (name)blue spice.
(3) **MR with sentence id.**	1_name[blue spice], 1_eattype[coffee shop], 1_area[city centre]
(4) **MR with sentence and slot id.**	1_3_name[blue spice], 1_1_eattype[coffee shop], 1_2_area[city centre]

Table 1: Example of an MR augmented with sentence and slot position identifiers.

main company selling points in front, and when generating messages for user with low literacy it is important to break sentences in small pieces, etc.

Recently there has been some work on controlling outputs of neural NLG models. In (Reed et al., 2018), authors use token supervision to reproduce sentence planning and discourse operations where a sentence scoping operation controls the number of sentences in the generated output which is measured using the period operator. This method does not provide any information about the word order in a particular sentence. While in (Moryossef et al., 2019) an explicit and symbolic text planner is proposed which determines the information structure and expresses it in the form of ordered trees. The plan structures in this work take the form of ad-hoc explorations for specific tasks and does not evolve into general-purpose plan structures. Moreover, this work is dataset dependent and does not generalize to datasets other than graph-based ones. To improve over the work in the literature of controlling neural NLG systems, in this paper, we propose an approach to explicitly control the realization of input entities in the desired sentences and in the desired position among individual sentences.

2 Overview of the Approach

Our method focuses on the control of sentence planning at two levels - 1) realization of input facts in the desired sentences and 2) realization of input facts in the desired position in the individual sentences. The idea is to directly attach sentence identifiers and slot identifiers to each slot, that indicate the sentence number and the position of the slot within that sentence respectively. The next step is to feed the modified Meaning Representation (MR) as an input to the seq2seq model and test if the model is able to learn to realize the slots in the correct positions.

2.1 Data Preparation

We used the E2E dataset for the experiments which provides information about restaurants and consists of about 50k combinations of a dialogue-act-based MR and 8.1 text references on an average. Each MR consists of up to 8 slots/attributes and their corresponding values.

As the dataset does not already contain a sentence plan, we modified it in a way that the MRs contain sentence and slot position identifiers. More specifically, given a reference text, two position identifiers were attached to each slot of the MR representing the sentence number in which the slot is found and the location of the slot in that specific sentence. The alignment information is extracted using a script[1] provided by the authors of Juraska et al. (2018). Table 1 provides an example of different stages of aligning the reference text with the MR. Initially, we annotate the reference text to identify the beginning of each slot value in the text (*line 2*), then using this annotation, we first attach the sentence identifiers (*line 3*), and finally the MRs are augmented with the position of each slot within a sentence (*line 4*). Owing to faults in the alignment information, in some cases the slot values are not detected in the reference text despite being present in the MR and for other cases the sentences do not contain some slots present in the MR. For such cases, a position token in the format 0_0_Slottype[Slot value] is attached.

Ideally, we expect a human to assign the position identifiers to each slot in the MR based on the desired output text. However, doing so for 4000 samples of the test set for validating our work would be very exhaustive. Therefore, we experiment with two different strategies to attach the position identifiers. Firstly, we directly use the test set reference text to extract the position identifiers. However, this will lead to biased results when computing the automatic metrics scores. Second

[1] https://github.com/jjuraska/slug2slug/tree/master/data/rest_e2e

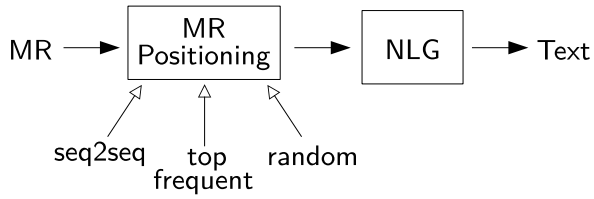

Figure 1: The three proposed approaches to automatically assign position identifiers to an MR.

strategy is to predict the position identifiers automatically. For this, we propose three different approaches as shown in Figure 1 and described in detail below:

- **Random:** The position identifiers are randomly chosen and assigned to the slots in the MR by taking care of a few rules.
- **Seq2seq:** The position identifiers are learned using a sequence-to-sequence (seq2seq) model. The train set MR without the position identifiers and the corresponding train set MR with the position identifiers are fed as training data to the model described in the Section 3. The test set MR without the position identifiers are then fed to the trained model, which outputs the test set MR with learned position identifiers.
- **Top Frequent:** The position identifiers for the test set are obtained from the most frequent combination of position identifiers in the train set. For each entry in the test set, the slot types (e.g., name) and slot values (e.g, blue spice) are separated and then the training entries which have slots with the same values are identified. Then, the most commonly occurring position identifier combination is picked for the test set entry.

Our random approach could be considered as a very naive baseline as randomly assigned identifiers might not even make sense in some cases. For example, having only one single slot in the first sentence (which never happens in the training set) would not be enough to form a grammatically correct sentence. Thus, in such cases, the model will fail to follow the assigned identifiers. The seq2seq model should perform much better than the random approach since it learns how to put position identifiers directly from the training data. However, this model has to re-generate the whole sequence including the slot type, slot value, and the newly added position identifiers. This means that any errors introduced during the generation process will significantly effect the text generated by the NLG model. Lastly, the top frequent approach is expected to perform better than the other two approaches. It extracts the most common sentence plan for some given values directly from the training set. Thus, the chosen sentence plan is among the ones that the NLG model was most exposed to during the training process.

2.2 Evaluation Metrics

In most NLG problems, sentence level metrics such as BLEU, ROUGE and METEOR are used to evaluate the results. However, these metrics are not entirely suitable to evaluate slot positioning since they measure performance at sentence level and do not provide precision in terms of sentence planning accuracy. In addition to using these automatic metrics to compare the results of the position augmented dataset with the original dataset, we evaluate the sentence planning accuracy using a modified version of the word error rate called the Slot Error Rate (SER) which is computed as:

$$SER = \frac{S + D + I}{N},$$

where S refers to number of substitutions, D refers to number of deletions, I refers to number of insertions and N refers to the total number of slots in the input MR. If a slot is realized in the wrong sentence it is counted as *wrong-sentence* substitution whereas if a slot appears in the wrong position but the correct sentence, it is counted as a case of *wrong-slot-position* substitution. It is important to note that the slot error rate compares two sequences of different nature (MR vs text). The generated text is pre-processed to extract the expressed slots and position of each slot using a script with some heuristics that involves lots of E2E dataset-specific handwritten rules. It is worth mentioning that our definition of SER is slightly different from the ones in the literature, particularly (Reed et al., 2018). In our case we take into account the exact position of each slot in the generated text, thus, the positioning of the realized slot in the generated text plays has a huge impact on the SER. Because of this difference, our metric can also be interpreted as slot position error rate in the realized text.

MR: 1_1_name[the cricketers], 1_2_eattype[restaurant], 1_3_food[chinese], 1_4_pricerange[20-25], 4_1_customer_rating[high], 2_1_area[riverside], 0_0_familyfriendly[no], 3_1_near[all bar one] **Output:** the cricketers is a restaurant providing chinese food in the 20-25 price range. it is located in the riverside. it is near all bar one. its customer rating is high.
MR: 1_1_name[the cricketers], 1_3_eattype[restaurant], 1_2_food[chinese], 1_7_pricerange[cheap], 1_6_customer_rating[5 out of 5], 1_5_area[city centre], 1_8_familyfriendly[yes], 1_4_near[all bar one] **Output** the cricketers is a chinese restaurant near all bar one in the city centre with a customer rating of 5 out of 5 and is cheap and family friendly.

Table 2: Output examples: position identifiers added from text references.

Experiment Type	SER
MR with sentence identifier	27%
MR with sentence and slot identifier	7%

Table 3: Results: SER on the test set prepared from test set alignment information.

	SER	BLEU	ROUGE	METEOR
Random	32%	0.22	0.39	0.34
Seq2Seq	5%	0.20	0.40	0.27
Top Frequent	0.7%	0.30	0.49	0.35

Table 4: Results: SER and sentence-level metrics on different versions of the test set with sentence and slot identifiers

3 Model Architecture

We use a standard seq2seq model with attention (Bahdanau et al., 2014; Luong et al., 2015). The seq2seq model consists of an encoder and a decoder based on Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). The encoder reads the position augmented MR tokens one by one and feeds each token into an embedding layer and then to an LSTM layer. Finally the LSTM-based decoder takes the last hidden state from the encoder and starts generating output tokens one by one. Our decoder uses the dot attention mechanism as described in (Luong et al., 2015).

4 Experiments and Results

We begin by comparing the SER obtained using the test set with only sentence identifier and the test set with both sentence and slot identifiers. The position identifiers are attached based on the alignment information obtained from the test set. Table 3 shows that the SER for the model with both sentence and slot identifiers is significantly lower. This is probably because the model in this case has more information to learn from and the slot identifier proves to be very important. In order to analyze the results better, in Table 2 some of the output examples are shown. As it can be seen, even though both examples have quite complex combination of position identifiers, all the slots are realized in the correct position as indicated in the MR.

In the next part of the experiments, instead of using the reference text of the test set, we used 3 other approaches listed in Section 2.1 to attach the sentence and slot identifiers. The idea here is to i) not rely on the reference text, since it will bias the automatic metrics such as BLEU, ROUGE and METEOR and ii) to try more complex position identifier combinations and test the robustness of the model. Table 4 summarizes the BLEU, ROUGE, METEOR, and SER results obtained on the test set using the 3 approaches. It can be seen that the sentence-level metrics scores for the different versions of position augmented test set are quite variable. Top frequent seems to significantly outperform the other two approaches. This can be attributed to the fact that top frequent uses the most common sentence plans from the train set which are the easiest for the model to generate. This also shows that the test set and the training sets are extremely similar in their sentences' structures. Surprisingly the seq2seq approach performs significantly worse than the top frequent one. As described earlier, this is mainly because the seq2seq model introduces many errors in the slot types and values during the generation process which does not happen in the top frequent approach. These errors are also propagated to the NLG model, and hence, the performance is significantly impacted. It is important to note that these sentence-level metrics are computed on a single reference as opposed to the E2E challenge systems where multiple references where used, and as a matter of fact, their results there were much higher. The reason that we cannot use multiple references is because in the case of the random approach, each MR is changed and is assigned a unique set of identifiers, thus, distinguishing all of

the previously unique MRs. To make the results consistent, we report single reference scores for the other two approaches as well.

When it comes to SER, we can see that top frequent again achieves the lowest score of 0.7%. The seq2seq model was trained with the position augmented train set that mostly consists of text with just one sentence. As data with one sentence is relatively easier than the data with multiple sentences, the SER of 5% with seq2seq model based test set is justifiable. The test set prepared with random identifiers reports an SER of 32%, which can be explained by the fact that the identifiers are attached without significant rules, and hence, some of the MRs do not make a logical sense if realized as text. Nevertheless, the model still learns to produce logically and grammatically correct sentences.

To better assess the faithfulness of the model, we used human verification of the model's output. We randomly selected 50 generated outputs (from samples of Table 3, *line 2*) and 4 annotators manually annotated the MR to show deletion, insertion, substitution and hallucination of slots where hallucination refers to realization of a wrong *slot value*. The original MR is compared with the new annotated MR to obtain an SER. The different scores obtained were averaged and the final SER reported is 14.25%. This score is slightly higher than the 7% reported in Table 3 since human subjects were additionally annotating hallucinations too. Excluding hallucinations will lead to a similar score obtained using the SER metric. Thus we can say that the human verification scores is consistent with the scores obtained from the SER metric.

5 Conclusion

We presented an approach to explicitly control the output text structure by incorporating control at two levels of sentence planning- realization of input entities in desired sentences and in the desired position among individual sentences. We created a new data set with position identifiers designed specifically for controlling sentence planning operations and we investigated different ways of preparing such sentence plans. Our results show that the model learns from the extra position identifiers which provide the capability to control variation in the output and enables generalizing to unseen combinations without a significant loss of performance in terms of sentence-level metrics.

Acknowledgments

This project was partly funded by the IDEX Université Grenoble Alpes innovation grant (AI4I-2018-2019) and the Région Auvergne-Rhône-Alpes (AISUA-2018-2019).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Regina Barzilay and Mirella Lapata. 2006. Aggregation via set partitioning for natural language generation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 359–366. Association for Computational Linguistics.

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 164–171. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Juraj Juraska, Panagiotis Karagiannis, Kevin K Bowden, and Marilyn A Walker. 2018. Slug2slug: A deep ensemble model with slot alignment for sequence-to-sequence natural language generation.

Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1101–1112.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 704–710. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. *arXiv preprint arXiv:1904.03396*.

Neha Nayak, Dilek Hakkani-Tür, Marilyn A Walker, and Larry P Heck. 2017. To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. In *INTERSPEECH*, pages 3339–3343.

Lena Reed, Shereen Oraby, and Marilyn Walker. 2018. Can neural generators for dialogue learn sentence planning and discourse structuring? *arXiv preprint arXiv:1809.03015*.

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

Amanda J Stent. 2002. A conversation acts model for generating spoken dialogue contributions. *Computer Speech & Language*, 16(3-4):313–352.

Marilyn A Walker, Owen C Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3-4):409–433.

Marilyn A Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.