Data-to-text Generation via Planning

Sen Yang¹, and Yang Liu*¹
¹Computer College, University of Defense Technology, Changsha, 410073, CN
*Corresponding author’s e-mail: liuyang_@nudt.edu.cn

Abstract. Planning based text generation from structural data is an emerging approach that manages to plan for “what to say.” Existing methods can not dynamically adjust the planning, because they generate complete plans statically without considering the result of surface realization. To address this issue, we propose a model that conducts dynamic text planning and surface realization alternately. In each sentence’s generation, our model first plans the records according to the generated text and the plan history, then realizes the sentence conditioned on the corresponding plan. Experimental results demonstrate that our model performs better on both text planning and text generation. Our model with the dynamic planner achieves impressive results on the E2E dataset.

1. Introduction
Data-to-text generation is to realize coherent and semantic consistent surface texts from structural data. Traditionally, data-to-text is addressed by splitting into several sub-tasks, including selecting and organizing specific input data, linguistic realization [6]. In short, these sub-tasks can be categorized as “what to say” and “how to say.” Recent neural attentional methods [2][7] cast all these sub-tasks into an end-to-end fashion model, which can generate fluent texts. Nevertheless, they still lack interpretability and struggle to generate long text that contains multiple sentences. To solve these shortcomings, various planning based methods have been proposed [1][3]. They try to separate the end-to-end neural generation model into a high-level planning module and a low-level surface realization module. The planning module is responsible for selecting the appropriate subset of input records for the generation of different granularities, while the realization module is to generate texts based on the planning result. Thanks to the planning module, it is easy to observe or even control the realization process. Besides, they also show the ability to model the inter-coherence of long generated text better.

However, there is one issue in existing planning based methods. Most of them require to conduct a complete plan in advance for text realization. It means that the planning process is static. The problem with this is that when the realization module produces unexpected results, the planner can not automatically adjust the following plan.

Targeting on this issue, we propose a dynamic planning model for data-to-text. Our model adopts encoder-decoder architecture. On the encoder side, the model embeds data records that contain slot and value into vector space. The decoder includes two modules: the planner and the realizer. The planner is used to plan the data records into different groups. Then the realizer generates sentences of which the content is limited to the given group of records. Unlike previous works, the decoder conducts planning and realization alternately. Since the planner also takes the realization result as input, it can flexibly adjust future planning. We evaluate our model on the E2E challenge [5].
2. Related works
Text generation from structural data is a classical task in the natural language generation. In recent years, neural attentional models have made distinguishable progress in data-to-text [2][7] due to their representation ability. Encoder-decoder architecture [8] is commonly used in these works. They perform well with well-designed encoder [2][9] and decoder that are usually equipped with attention mechanism [10] and copy mechanism [11]. Moreover, thanks to the large scale corpus and billions of parameters, pre-trained language models [12][13] have brought new state-of-the-arts.
More recent works introduce content selection and text planning into the decoder to compensate for the above disadvantages. Most of them use extracted alignment or order information to conduct supervised planning [1][3]. Another advantage of introducing text planning into neural attentional models is to enhance the diversity at a high-level [3] with the help of variational inference of handcrafted information. Our work is most alike with [3]. Both of us use a planner to select appropriate records for sentence generation. Nevertheless, our model adopts a dynamic planning mechanism.

3. Problem formulation and background
Given the source-target data pair \( X, Y \), data-to-text aims to generate \( Y \) from \( X \). \( X = r_1, r_2, \ldots, r_n \) denotes the input records set. \( Y = y_1, y_2, \ldots, y_m \) denotes the target text. Element \( x_i \) in \( X \) represents a record that is a tuple including a slot and a value. In E2E challenge dataset, an attribute could contain multiple words, so that we define \( x_i = (k_i, i_i, v_i) \), where \( k_i \) is the slot name, \( i_i \) is the index of word \( v_i \) in the value.

The neural attention approaches use encoder-decoder architecture to model data-to-text. Overall, such models try to maximize the probability of \( Y \) given the input \( X \): \( P(Y|X) = \prod_{i=1}^{m} P(y_i|y_{<i}, X) \) Specifically, the encoder, that can be a LSTM [14] or a Transformer [15], encode each input record \( x_i \) into a vector \( h_i \). The decoder generates text in an autoregressive way. In each time step, the decoder takes the previous word as input and outputs the current word’s probability.

4. Approach
4.1. Overview
Our model adopts an encoder-decoder style. The encoder is a bidirectional long short memory networks (LSTM) that embed records into vector space. The decoder contains two modules, the planner and the realizer. Here we use a simple multi-layer perceptron (MLP) as our planner. While the realizer is also a LSTM.

The text generation is decomposed into multiple sentence generation. For each sentence, the planner first selects a record group according to the realized text and unselected records. Then the realizer generates the tokens sequentially with respect to the given group of records. The model conducts these two procedures alternatively so that both of them can adjust the future actions flexibly.

4.2. Record Encoding
The encoder embeds the input data records into vector representations. Since there is no natural order of records, we order them according to their slot names (and index if it exists) in preprocessing. For a given record \( r_i \), its embedding is the sum of the slot embedding and the value embedding. In the E2E challenge, it also includes the index embedding. Especially, to make the model sensitive to the value tokens, we scale the value embedding with the square root of embedding size. For example, \( E(r_i) = E(k_i) + \sqrt{d_v}E(v_i) \), where \( E \) denotes embedding, \( d \) indicates the size of embedding vectors.

We then use a bidirectional LSTM to encode records. Therefore, the final context-aware vector representations of record \( r_i \) is \( h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}] \). Note that LSTM is not necessary for encoding records. Other encoders such as Transformer [32] can also play the same.
4.3. Dynamic planning

The planner is to produce plans to guide the generation of the realizer. We define a plan as a sequence of record groups: \( P = g_1, g_2, \ldots, g_k \). Each group is a set of input records, corresponding to a sentence. Traditional planners formulate complete plans before the text generation. Therefore, if the realizer generates unexpected sentences that inconsistent with planned records, such a planner is insufficient to revise the existing plan. Inspired by this intuition, we feed the planner with the realizer state to enable it to plan dynamically. Besides, to make the planner aware of those unselected records, we also feed the planner with their representations. As a result, before generating the \( k \)-th sentence (denoted as \( S_k \)), the planner computes the probability of each input record been selected into \( g_k \) conditioned on the history groups and previous \( k-1 \) sentences \( S_{\leq k} \).

4.4. Surface realization

The realizer is a standard LSTM, with the init hidden state being the encoder’s last state. It decomposes long text generation into multiple sentence generation. To make the model distinguish the sentence separation, we insert a special token “<sep>“ between sentences during preprocessing the target text. When encounters this token, the planner will select a new group of records, and then the realizer begins to generate a new sentence, of which the content is restricted to the given record group. This token will be removed in the postprocessing for evaluation.

For each sentence generation, the realizer generates tokens sequentially. Specifically, in the generation of \( S_k \), the realizer is trained to predict the probability of \( t \)-th token in (denoted as \( y_{k,t} \)) conditioned on the current realizer hidden state \( h_{k,t}^{r} \) and the vector representations of records in \( g_k \).

5. Experiments

5.1. Datasets

We conduct the experiments on the E2E challenge [5], a crowdsourced dataset that generates restaurant descriptions with corresponding restaurant data. Each input could have multiple references. Its train/dev/test split includes 4862/547/630 unique inputs and 42061/4672/4693 text references.

5.2. Main result

Since there are no gold plans in E2E, we manage to extract the plan with a simple keyword based method. Specifically, we first split the sentence with punctuations “.?!”. Then for each record, if its slot name, value, or other related keywords appears in one sentence, we classify it into the corresponding plan groups of this sentence. The keywords for a record are tightly closed to the slot name. For example, given the record “rating: [1 out of 5]”, its keywords includes “rated”, “ratings”, “starts”, “respected”, etc.

We compare our model against a set of strong baselines, including SLUG [17], an ensemble model winning the E2E challenge; TUDA [18], the best template based model; NTEMP [19], a neural templated based model; Shen’s Model [4], the latest work on the E2E dataset we know.

Table 1 shows performances of all models. We can see that our model with the static planner has achieved competitive performance with others. While our model with the dynamic planner can surpass strong baselines on most automatic evaluation metrics. It indicates the effectiveness of our model.

|                  | BLEU  | METEOR | R-L   | CIDEr |
|------------------|-------|--------|-------|-------|
| SLUG             | 0.662 | 0.445  | 0.677 | 2.262 |
| TUDA             | 0.566 | 0.453  | 0.661 | 1.821 |
| N_TEMP           | 0.598 | 0.388  | 0.650 | 1.950 |
| Shen’s Model     | 0.651 | **0.455** | 0.682 | 2.241 |
6. Conclusion and discussion
Planning based approaches perform well in data-to-text generation. Nevertheless, most of the planning procedure is static, making the planner unable to adjust the plan timely according to the generated text. Besides, although the planning result has an influence on the text generation, little attention has been paid to evaluating the quality of the plan. Target on these two issues, in this paper, we first propose a simple but effective dynamic planning based model for data-to-text, and then we propose a planning evaluation metric based on set similarity, which is computed by dynamic programming. Experiments on the E2E dataset shows the effectiveness of our proposed method. By evaluating the plan's score with our proposed metric, we can conclude that our model achieves improvement via dynamic planning.

References
[1] Puduppully, R., Dong, L., & Lapata, M. (2019) Data-to-text generation with content selection and planning. Proceedings of the AAAI Conference on Artificial Intelligence, 33, 6908-6915.
[2] Koncel-Kedziorski, R., Bekal, D., Luan, Y., Lapata, M., & Hajishirzi, H. (2019) Text generation from knowledge graphs with graph transformers.
[3] Shao, Z., Huang, M., Wen, J., Xu, W., & Zhu, X. (2019) Long and diverse text generation with planning-based hierarchical variational model.
[4] Shen, X., Chang, E., Su, H., Niu, C., & Klakow, D. (2020) Neural Data-to-Text Generation via Jointly Learning the Segmentation and Correspondence. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.
[5] J. Novikova, O. Dusek, and V. Rieser. (2017) The E2E dataset: New challenges for end-to-end generation. in Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Saarbrucken, Germany.
[6] Gatt, A., & Krahmer, E. (2017) Survey of the state of the art in natural language generation: core tasks, applications and evaluation. Journal of Artificial Intelligence Research, 61(45), 1-16.
[7] E. Chang, D. I. Adelani, X. Shen, and V. Dembirst. (2020) Unsupervised pidgin text generation by pivoting english data and self-training.
[8] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, (2014) Learning phrase representations using rnn encoder–decoder for statistical machine translation, in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734.
[9] T. Wang, X. Wan, and S. Yao, (2020) Better amr-to-text generation with graph structure reconstruction," in Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, 7, pp, 3919-3925.
[10] D. Bahdanau, K. Cho, and Y. Bengio, (2015) Neural machine translation by jointly learning to align and translated, in 3rd International Conference on Learning Representations.
[11] J. Gu, Z. Lu, H. Li, and V. O. Li, (2016) Incorporating copying mechanism in sequence-to-sequence learning, in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 1631-1640.
[12] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, (2020) BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7871–7880.
[13] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., & Matena, M., et al. (2019) Exploring the limits of transfer learning with a unified text-to-text transformer.
[14] S. Hochreiter and J. Schmidhuber, (1997) Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780.
[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, (2017) Attention is all you need, in Advances in neural information processing systems, pp. 5998–6008.

[16] Kingma, D., & Ba, J. (2014). Adam: a method for stochastic optimization. Computer Science.

[17] Juraska, J., Karagiannis, P., Bowden, K. K., & Walker, M. A. (2018) A Deep Ensemble Model with Slot Alignment for Sequence-to-Sequence Natural Language Generation. North American Chapter of the Association for Computational Linguistics (NAACL), pp.152-162.

[18] Puzikov, Y., & Gurevych, I. (2018) E2E NLG Challenge: Neural Models vs. Templates. Proceedings of the 11th International Conference on Natural Language Generation, pp. 463–471.

[19] S. Wiseman, S. Shieber, and A. Rush, (2018) Learning neural templates for text generation,” in Proceedings of the 2018 Conference on Empirical, Methods in Natural Language Processing, pp. 3174–3187.