Sketch and Refine: Towards Faithful and Informative Table-to-Text Generation

Peng Wang , Junyang Lin , An Yang ,
Chang Zhou , Yichang Zhang , Jingren Zhou , Hongxia Yang†

DAMO Academy, Alibaba Group
{zheluo.wp, junyang.ljy, ya235025, ericzhou.zc,
yichang.zyc, jingren.zhou, yang.yhx }@alibaba-inc.com

Abstract

Table-to-text generation refers to generating a descriptive text from a key-value table. Traditional autoregressive methods, though can generate text with high fluency, suffer from *low coverage* and *poor faithfulness* problems. To mitigate these problems, we propose a novel **S**keleton-based two-stage method that combines both **A**utoregressive and **N**on-**A**utoregressive generation (**SANA**). Our approach includes: (1) skeleton generation with an autoregressive pointer network to select key tokens from the source table; (2) edit-based non-autoregressive generation model to produce texts via iterative insertion and deletion operations. By integrating hard constraints from the skeleton, the non-autoregressive model improves the generation's coverage over the source table and thus enhances its faithfulness. We conduct experiments on both the WikiPerson and WikiBio datasets. Experimental results demonstrate that our method outperforms the previous state-of-the-art methods in both automatic and human evaluation, especially on coverage and faithfulness. In particular, we achieve PARENT-T recall of 99.47 in WikiPerson, improving over the existing best results by more than 10 points.

1 Introduction

Table-to-text generation is a challenging task which aims at generating a descriptive text from a key-value table. There have been a broad range of applications in this field, such as the generation of weather forecast (Mei et al., 2016), sports news (Wiseman et al., 2017), biography (Lebret et al., 2016; Wang et al., 2018), etc. Figure 1 illustrates a typical input and output example of this task.

Previous methods (Liu et al., 2018; Nie et al., 2018; Bao et al., 2018) are usually trained in an end-to-end fashion with the encoder-decoder architecture (Bahdanau et al., 2015). Despite generating text with high fluency, their lack of control in the generation process leads to *poor faithfulness* and *low coverage*. As shown in Figure 1, the case of *poor faithfulness* hallucinates the occupation "singer" not entailed by the source table, and the case of *low coverage* misses the information of the place of birth. Even if trained with a cleaned dataset, end-to-end methods still encounter these problems as it is too complicated to learn the probability distribution under the table constraints (Parikh et al., 2020).

key	value
Name_ID	Thaila Ayala
Place of birth	Brazil
Date of birth	April 14 1986
Occupation	Actress , Model

Poor Faithfulness: Thaila Ayala (born April 14, 1986 in Brazil) is an actress , model and singer .

Low Coverage: Thaila Ayala (born April 14, 1986) is an actress and model .

Figure 1: An example of table-to-text generation. The case of *poor faithfulness* hallucinates content not entailed by the table (marked in red color). The case of *low coverage* misses the information of the person's birth place (marked in blue color).

To alleviate these problems, recent studies (Shao et al., 2019; Puduppully et al., 2019; Ma et al., 2019) propose two-stage methods to control the generation process. In the first stage, a pointer network selects the salient key-value pairs from the table and arranges them to form a content-plan. In the second stage, an autoregressive seq2seq model generates text conditioned on the content-plan. However, such methods can cause the following problems: (1) since the generated content-plan may contain errors, generating solely on the content-

†Corresponding author.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4831–4843

plan leads to inconsistencies; (2) even if a perfect content-plan is provided, the autoregressive model used in the second stage is still prone to hallucinate unfaithful contents due to the well-known *exposure bias* (Wang and Sennrich, 2020) problem; (3) there is no guarantee that the selected key-value pairs can be described in the generated text. As a result, these methods still struggle to generate faithful and informative text.

In this paper, we propose a **S**keleton-based model that combines both **A**utoregressive and **N**on-**A**utoregressive generation (**SANA**). SANA divides table-to-text generation into two stages: *skeleton construction* and *surface realization*. At the stage of skeleton construction, an autoregressive pointer network selects tokens from the source table and composes them into a skeleton. We treat the skeleton as part of the final generated text. At the stage of surface realization, an edit-based non-autoregressive model expands the skeleton to a complete text via insertion and deletion operations. Compared with the autoregressive model, the edit-based model has the following advantages: (1) the model generates text conditioned on both the skeleton and the source table to alleviate the impact of incomplete skeleton; (2) the model accepts the skeleton as decoder input to strengthen the consistency between the source table and generated text; (3) the model generates texts with the hard constraints from the skeleton to improve the generation coverage over the source table. Therefore, SANA is capable of generating faithful and informative text.

The contributions of this work are as follows:

- We propose a skeleton based model **SANA** which explicitly models skeleton construction and surface realization. The separated stages helps the model better learn the correlation between the source table and reference.
- To make full use of the generated skeleton, we use a non-autoregressive model to generate text based on the skeleton. To the best of our knowledge, we are the first to introduce non-autoregressive model to table-to-text generation task.
- We conduct experiments on WikiPerson and WikiBio datasets. Both automatic and human evaluations show that our method outperforms previous state-of-the-art methods, especially on faithfulness and coverage. Specially, we obtain a near-optimal PARENT-T recall of 99.47 in the WikiPerson dataset.

2 Related Work

Table-to-text Generation Table-to-text generation has been widely studied for decades (Kukich, 1983; Goldberg et al., 1994; Reiter and Dale, 1997). Recent works that adopt end-to-end neural networks have achieve great success on this task (Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017; Sha et al., 2018; Nema et al., 2018; Liu et al., 2018, 2019a). Despite generating fluent texts, these methods suffer from poor faithfulness and low coverage problems. Some works focus on generating faithful texts. For example, Tian et al. (2019) proposes a confident decoding technique that assigns a confidence score to each output token to control the decoding process. Filippova (2020) introduces a "hallucination knob" to reduce the amount of hallucinations in the generated text. However, these methods only focus on the faithfulness of the generated text, they struggle to cover most of the attributes in the source table.

Our work is inspired by the recently proposed two-stage method (Shao et al., 2019; Puduppully et al., 2019; Moryossef et al., 2019; Ma et al., 2019; Trisedya et al., 2020). They shows that table-to-text generation can benefit from separating the task into content planing and surface realization stages. Compared with these methods, SANA guarantee the information provided by the first stage can be preserved in the generated text, thus significantly improving the the coverage as well as the faithfulness of the generated text.

Non-autoregressive Generation Although autoregressive models have achieved remarkable success in natural language generation tasks, they are time-consuming and inflexible. To overcome these shortcomings, Gu et al. (2018) proposed the first non-autoregressive (NAR) model that can generate tokens simultaneously by discarding the generation history. However, since a source sequence may have different possible outputs, discarding the dependency of target tokens may cause the degradation in generation quality. This problem also known as the "multi-modality" problem (Gu et al., 2018). Recent NAR approaches alleviate this problem via partially parallel decoding (Stern et al., 2019; Sun et al., 2019) or iterative refinement (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019). Specially, Stern et al. (2019) per-

forms partially parallel decoding through insertion operation. Gu et al. (2019) further incorporates deletion operation to perform iterative refinement process. These edit-based models not only close the gap with autoregressive models in translation task, but also makes generation flexible by allowing integrates with lexical constrains. However, the multi-modality problem still exists, making it difficult to apply NAR models to other generation tasks, such as table-to-text generation, story generation, etc. In this work, we use the skeleton as the initial input of our edit-based text generator. The skeleton can provide sufficient contexts to the text generator, thus significantly reducing the impact of multi-modality problem.

3 Methods

The task of table-to-text generation is to take a structured table T as input, and outputs a descriptive text $Y = \{y_1, y_2, ..., y_n\}$. Here, the table T can be formulated as a set of attributes $T = \{a_1, a_2, ..., a_m\}$, where each attribute is a key-value pair $a_i = \langle k_i, v_i \rangle$.

Figure 2 shows the overall framework of SANA. It contains two stages: skeleton construction and surface realization. At the stage of skeleton construction, we propose a Transformer-based (Vaswani et al., 2017) pointer network to select tokens from the table and compose them into a skeleton. At the stage of surface realization, we use an edit-based Transformer to expand the skeleton to a complete text via iterative insertion and deletion operations.

3.1 Stage 1: Skeleton Construction

3.1.1 Table Encoder

The source table is a set of attributes represented as key-value pairs $a_i = \langle k_i, v_i \rangle$. Here, the value of an attribute a_i is flattened as a token sequence $v_i = \{w_i^1, w_i^2, ..., w_i^l\}$, where w_i^j is the j-th token and l is the length of v_i. Following Lebret et al. (2016), we linearize the source table by representing each token w_i^j as a 4-tuple $(w_i^j, k_i, p_i^+, p_i^-)$, where p_i^+ and p_i^- are the positions of the token w_i^j counted from the beginning and the end of the value v_i, respectively. For example, the attribute of "$\langle$*Name_ID,* $\{$*Thaila Ayala*$\}\rangle$" is represented as "(*Thaila, Name_ID*, 1, 2)" and "(*Ayala, Name_ID*, 2, 1)". In order to make the pointer network capable of selecting the special token $\langle EOS \rangle$[1], we add a special tuple $(\langle EOS \rangle, \langle EOS \rangle, 1, 1)$ at the end of the table.

To encode the source table, we first use a linear projection on the concatenation $\left[\boldsymbol{w}_i^j; \boldsymbol{k}_i; \boldsymbol{p}_i^+; \boldsymbol{p}_i^-\right]$ followed by an activation function:

$$\boldsymbol{f}_i^j = \text{Relu}(\boldsymbol{W}_f[\boldsymbol{w}_i^j; \boldsymbol{k}_i; \boldsymbol{p}_i^+; \boldsymbol{p}_i^-] + \boldsymbol{b}_f) \quad (1)$$

where $\boldsymbol{W}_f$ and $\boldsymbol{b}_f$ are trainable parameters. Then we use the Transformer encoder to transform each $\boldsymbol{f}_i^j$ into a hidden vector and flatten the source table into a vector sequence $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_l\}$.

3.1.2 Pointer Network

After encoding the source table, we use a pointer network to directly select tokens from the table and compose them into a skeleton. Our pointer network uses a standard Transformer decoder to represent the tokens selected at the previous steps. Let $\boldsymbol{r}_t$ denote the decoder hidden state of previous selected token $\hat{y}_t$. The pointer network predict the next token based on the attention scores, which are computed as follows:

$$\alpha_{ti} = \frac{e^{u(\boldsymbol{r}_t, \boldsymbol{h}_i)}}{\sum_{j=1}^{l} e^{u(\boldsymbol{r}_t, \boldsymbol{h}_j)}} \quad (2)$$

$$u(\boldsymbol{r}_t, \boldsymbol{h}_i) = \frac{(\boldsymbol{W}_q \boldsymbol{r}_t) \cdot (\boldsymbol{W}_k \boldsymbol{h}_i)}{\sqrt{d_r}} \quad (3)$$

where $\boldsymbol{W}_q$ and $\boldsymbol{W}_k$ are trainable parameters, d_r is the embedding dimension of $\boldsymbol{r}_t$. According to the calculated probability distribution $\boldsymbol{\alpha}$, we select token based on the following formula:

$$P_{copy}(w) = \sum_{w=w_i} \alpha_{ti} \quad (4)$$

$$\hat{y}_{t+1} = \arg\max_w P_{copy}(w) \quad (5)$$

where $\hat{y}_{t+1}$ represents the output at the next timestep, and $P_{copy}(w)$ represents the probability of copying token w from the source. There may be multiple identical tokens in the table, so we sum up the attention scores of their corresponding positions.

The pointer network needs target skeletons as supervision, which are not provided by the table-to-text datasets. In this paper, we obtain the skeleton by collecting tokens in both the table and description except the stop words. The token order in the skeleton remains the same as their relative positions in the description. More details are described in Appendix A.

[1] $\langle EOS \rangle$ denotes the end of the skeleton.

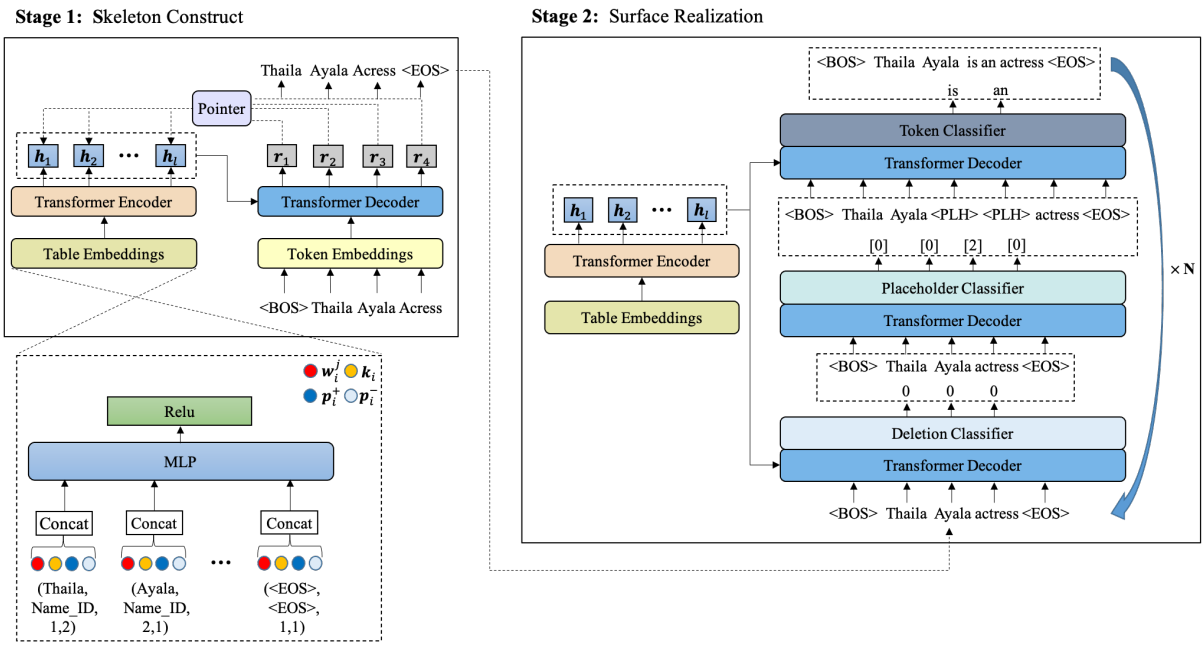

Figure 2: The overall diagram of SANA for generating description for *Thaila Ayala* in Fig 1.

Given the skeleton $S = \{s_1, s_2, ..., s_q\}$, the pointer network is trained to maximize the conditional log-likelihood:

$$\mathcal{L}_1 = -\sum_{t=1}^{q+1} \log P_{copy}(s_t | s_{0:t-1}, T), \quad (6)$$

where the special tokens $s_0 = \langle BOS \rangle$ and $s_{q+1} = \langle EOS \rangle$ denote the beginning and end of the target skeleton.

3.2 Stage 2: Surface Realization

At the surface realization stage, we use the same encoder as in the skeleton construction stage. The decoder is an edit-based Transformer decoder (Gu et al., 2019) that generates text via insertion and deletion operations. Different from the original Transformer decoder which predicts the next token in an left-to-right manner, the edit-based decoder can predict tokens simultaneously and independently. In this setting, we can use the full self-attention without causal masking.

3.2.1 Model Structure

To perform the insertion and deletion operations, we remove the softmax layer at the top of the Transformer decoder and add three operation classifiers: *Deletion Classifier*, *Placeholder Classifier* and *Token Classifier*. We denote the outputs of the Transformer decoder as $(\boldsymbol{z}_0, \boldsymbol{z}_1, ..., \boldsymbol{z}_n)$, details of these three classifiers are as follows:

1. *Deletion Classifier* which predicts for each token whether they should be "deleted"(1) or "kept"(0):

$$\pi_\theta^{\text{del}}(d|i, Y) = \text{softmax}(\boldsymbol{W}_{\text{del}} \boldsymbol{z}_i) \quad (7)$$

2. *Placeholder Classifier* which predicts the number of placeholders [PLH] to be inserted at each consecutive pair:

$$\pi_\theta^{\text{plh}}(p|i, Y) = \text{softmax}(\boldsymbol{W}_{\text{plh}}[\boldsymbol{z}_i; \boldsymbol{z}_{i+1}]) \quad (8)$$

3. *Token Classifier* which replaces each [PLH] with an actual token:

$$\pi_\theta^{\text{tok}}(t|i, Y) = \text{softmax}(\boldsymbol{W}_{\text{tok}} \boldsymbol{z}_i) \quad (9)$$

During decoding, we use the skeleton predicted from the first stage as the initial input of the decoder. We also use the full table information from encoder side to mitigate the impact of incomplete skeleton. As shown in Figure 2, the skeleton will pass through the three classifiers sequentially for several iterations. Benefiting from the full self-attention, each operation is allowed to condition on the entire skeleton, and thus reduces the probability of hallucinating unfaithful contents in the final text.

3.2.2 Training

Following Gu et al. (2019), we adopt imitation learning to train our model and simplify their training procedure. The iterative process of our model

will produce various of *intermediate sequences*. To simulate the iterative process, we need to construct the intermediate sequence and provide an optimal operation $\boldsymbol{a}^*$ (either oracle insertion $\boldsymbol{p}^*$, $\boldsymbol{t}^*$ or oracle deletion $\boldsymbol{d}^*$) as the supervision signal during training. Given an intermediate sequence Y, the optimal operation $\boldsymbol{a}^*$ is computed as follows:

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} \mathcal{D}(Y^*, \mathcal{E}(Y, \boldsymbol{a})) \tag{10}$$

Here, $\mathcal{D}$ denotes the Levenshtein distance (Levenshtein, 1965) between two sequences, and $\mathcal{E}(Y, \boldsymbol{a})$ represents the output after performing operation $\boldsymbol{a}$ upon Y.

To improve the training efficiency, we construct the intermediate sequence via a simple yet effective way. Given a source table, skeleton and reference (T, S, Y^*), We first calculate the longest common subsequence X between S and Y^*, and then construct the intermediate sequence Y' by applying random deletion on Y^* except the part of X. We use Y' to learn the insertion and deletion operations. The learning objective of our model is computed as follows:

$$\mathcal{L}_{\text{edit}} = \mathcal{L}_{\text{ins}} + \lambda \mathcal{L}_{\text{del}} \tag{11}$$

$$\begin{aligned}\mathcal{L}_{\text{ins}} = &- \sum_{p_i^* \in \boldsymbol{p}^*} \log \pi_\theta^{\text{plh}}(p_i^* | i, T, Y') \\ &- \sum_{t_i^* \in \boldsymbol{t}^*} \log \pi_\theta^{\text{tok}}(t_i^* | i, T, Y'')\end{aligned} \tag{12}$$

$$\mathcal{L}_{\text{del}} = - \sum_{d_i^* \in \boldsymbol{d}^*} \log \pi_\theta^{\text{del}}(d_i^* | i, T, Y''') \tag{13}$$

where Y'' is the output after inserting placeholders $\boldsymbol{p}*$ upon Y', Y''' is the output by applying the model's insertion policy π_θ^{tok} to Y''.[2] λ is the hyper parameter.[3]

3.2.3 Inference

As mentioned above, at the inference stage, we use the generated skeleton as the initial input of the decoder. The insertion and deletion operations will perform alternately for several iterations. We stop the decoding process when the current text does not change, or a maximum number of iterations has been reached.

In order to completely retain the skeleton in the generated text, we follow Susanto et al. (2020) to enforce hard constraints through forbidding the deletion operation on tokens in the skeleton. Specially, we compute a *constraint mask* to indicate the positions of constraint tokens in the sequence and forcefully set the deletion classifier prediction for these positions to "keep". The constraint masks are recomputed after each insertion and deletion operation.

[2]We do argmax from Equation (9) instead of sampling.

[3]In our experiment, $\lambda = 1$ gives a reasonable good result.

4 Experiment Setups

4.1 Datasets

We conduct experiments on the **WikiBio** (Lebret et al., 2016) and **WikiPerson** (Wang et al., 2018) datasets. Both datasets aim to generate a biography from a Wikipedia table, but they have different characteristics. Their basic statistics are listed in Table 1.

Dataset	Train	Valid	Test	Avg Len
WikiBio	582,657	72,831	72,831	26.1
WikiPerson	250,186	30,487	29,982	70.6

Table 1: Statistics of WikiBio and WikiPerson datasets. **Avg Len** means the average target length of the datasets.

WikiBio This dataset aims to generate the first sentence of a biography from a table. It is a particularly noisy dataset which has 62% of the references containing extra information not entailed by the source table (Dhingra et al., 2019).

WikiPerson Different from the WikiBio whose reference only contains one sentence, the reference of WikiPerson contains multiple sentences to cover as many facts encoded in the source table as possible. In addition, WikiPerson uses heuristic rules to remove sentences containing entities that do not exist in the Wikipedia table, making it cleaner compared to the WikiBio dataset.

4.2 Evaluation Metrics

Automatic Metrics For automatic evaluation, we apply BLEU (Papineni et al., 2002) as well as PARENT (precision, recall, F1) (Dhingra et al., 2019) to evaluate our method. Different from BLEU which only compare the outputs with the references, PARENT evaluates the outputs that considers both the references and source tables. Following Wang et al. (2020), we further use their proposed PARENT-T metric to evaluate our method in WikiPerson dataset. PARENT-T is a variant of

PARENT which only considers the correlation between the source tables and the outputs.

Human Evaluation Human ratings on the generated descriptions provide more reliable reflection of the model performance. Following Liu et al. (2019b), we conduct comprehensive human evaluation between our model and the baselines. The annotators are asked to evaluate from three perspectives: fluency, coverage (how much table content is recovered) and correctness (how much generated content is faithful to the source table). We hire 5 experienced human annotators with linguistic background. During the evaluation, 100 samples are randomly picked from the WikiPerson dataset. For each sample, an annotator is asked to score the descriptions generated by different models without knowing which model the given description is from. The scores are within the range of $[0, 4]$.

4.3 Implementation Details

We implement SANA using fairseq (Ott et al., 2019). The token vocabulary is limited to the 50K most common tokens in the training dataset. The dimensions of token embedding, key embedding, position embedding are set to 420, 80, 5 respectively. All Transformer components used in our methods adopt the base Transformer (Vaswani et al., 2017) setting with $d_{\text{model}} = 512, d_{\text{hidden}} = 2048, n_{\text{heads}} = 8, n_{\text{layers}} = 6$. All models are trained on 8 NVIDIA V100 Tensor Core GPUs.

For the skeleton construction model, the learning rate linearly warms up to 3e-4 within 4K steps, and then decays with the inverse square root scheduler. Training stops after 15 checkpoints without improvement according to the BLEU score. We set the beam size to 5 during inference.

For the surface realization model, the learning rate linearly warms up to 5e-4 within 10K steps, and then decays with the inverse square root scheduler. Training stops when the training steps reach 300K. We select the best checkpoint according to the validation BLEU.

4.4 Baselines

We compare SANA with two types of methods: end-to-end methods and two-stage methods.

For end-to-end methods, we select the following methods as baselines: (1) **DesKnow** (Wang et al., 2018), a seq2seq model with a table position self-attention to capture the inter-dependencies among related attributes; (2) **PtGen** (Pointer-Generator, See et al. (2017)), an LSTM-based seq2seq model with attention and copy mechanism; (3) **Struct-Aware** (Liu et al., 2018), a seq2seq model using a dual attention mechanism to consider both key and value information; (4) **OptimTrans** (Wang et al., 2020), a Transformer based model that incorporates optimal transport matching loss and embedding similarity loss. (5) **Conf-PtGen** (Tian et al., 2019), a pointer generator with a confidence decoding technique to improve generation faithfulness; (6) **S2S+FA+RL** (Liu et al., 2019b), a seq2seq model with a force attention mechanism and a reinforce learning training procedure; (7) **Bert-to-Bert** (Rothe et al., 2019), a Transformer encoder-decoder model where the encoder and decoder are both initialized with BERT (Devlin et al., 2019).

For two-stage methods, we select the following methods as baselines: (1) **Pivot** (Ma et al., 2019), a two stage method that first filter noisy attributes in the table via sequence labeling and then uses the Transformer to generate text based on the filter table; (2) **Content-Plan** (Puduppully et al., 2019), a two stage method that first uses a pointer network to select important attributes to form a content-plan and then uses a pointer generator to generate text based on the content-plan.

5 Results

5.1 Comparison with End-to-End Methods

We first compare SANA with end-to-end methods, Table 2 shows the experimental results. From Table 2, we can outline the following statements: (1) For WikiPerson dataset, SANA outperforms existing end-to-end methods in all of the automatic evaluation metrics, indicating high quality of the generated texts. Specially, we obtain a near-optimal PARENT-T recall of 99.47, which shows that our model has the ability to cover all the contents of the table. (2) For the noisy WikiBio dataset, SANA outperforms previous state-of-the-art models in almost all of the automatic evaluation scores except the PARENT precision, which confirms the robustness of our method. Although Conf-PtGen achieves the highest PARENT precision, its PARENT recall is significantly lower than any other method. Different from Conf-PGen, SANA achieves the highest recall while maintaining good precision. (3) It is necessary to prohibit deleting tokens in the skeleton. After removing this restriction (– *hard constrains*), our method has different degrees of decline in various automatic metrics. (4) SANA

Model	WikiPerson			WikiBio	
	BLEU	PARENT(P / R / F1)	PARENT-T(P / R / F1)	BLEU	PARENT(P / R / F1)
DesKnow	16.20	63.92 / 44.83 / 51.03	41.10 / 84.34 / 54.22	-	- / - / -
PtGen	19.32	61.73 / 44.09 / 49.52	42.03 / 81.65 / 52.62	41.07	77.59 / 42.12 / 52.10
Struct-Aware	22.76	51.18 / 46.34 / 46.47	35.99 / 83.84 / 48.47	44.93	74.18 / 43.50 / 52.33
OptimTrans	24.56	62.86 / 48.83 / 53.06	43.52 / 85.21 / 56.10	-	- / - / -
Conf-PtGen	-	- / - / -	- / - / -	38.10	**79.52** / 40.60 / 51.38
S2S+FA+RL	-	- / - / -	- / - / -	45.49	76.10 / 43.66 / 53.08
Bert-to-Bert	-	- / - / -	- / - / -	45.62	77.64 / 43.42 / 53.54
SANA	**25.23**	**65.69 / 56.88 / 59.96**	**44.88 / 99.47 / 61.34**	**45.78**	76.93 / **46.01 / 55.42**
− *hard constrains*	24.97	64.72 / 56.42 / 59.29	43.75 / 98.97 / 60.17	45.31	76.32 / 45.26 / 54.64
− *skeleton*	19.55	61.80 / 44.29 / 50.29	40.80 / 84.03 / 53.97	42.58	74.29 / 41.32 / 50.41

Table 2: Comparison with end-to-end methods. **P, R, F1** represent precision, recall and F1 score, respectively. "− *hard constrains*" means removing the restriction of forbidding the deletion operation on tokens in the skeleton, "− *skeleton*" means removing the skeleton construction stage.

Model	WikiPerson			WikiBio	
	BLEU	PARENT(P / R / F1)	PARENT-T(P / R / F1)	BLEU	PARENT(P / R / F1)
Pivot	24.71	62.24 / 50.02 / 52.99	41.78 / 89.68 / 56.35	44.39	76.35 / 41.90 / 51.85
+ *Oracle*	25.08	62.34 / 50.63 / 53.47	42.08 / 89.71 / 56.59	45.38	75.98 / 42.57 / 52.45
Content-Plan	25.07	58.56 / 53.86 / 54.52	38.63 / 91.18 / 54.01	43.21	74.69 / 43.53 / 52.71
+ *Oracle*	28.50	59.31 / 56.02 / 55.96	39.64 / 91.62 / 55.07	50.57	76.32 / 47.33 / 56.45
SANA	25.23	65.69 / 56.88 / 59.96	44.88 / 99.47 / 61.34	45.78	76.93 / 46.01 / 55.42
+ *Oracle*	**30.29**	**69.27 / 67.89 / 68.28**	**45.13 / 99.79 / 61.54**	**54.51**	**80.03 / 51.02 / 61.01**

Table 3: Comparison with two-stage methods. **P, R, F1** represent precision, recall and F1, respectively. "+ *Oracle*" means using oracle information (i.e., oracle skeleton or content-plan) as input.

performs poorly after removing the skeleton construction stage (− *skeleton*). This shows that the edit-based non-autoregressive model is difficult to directly apply to table-to-text generation tasks. The skeleton is very important for the edit-based model, which can significantly reduce the impact of the multi-modality problem. Combining both autoregressive and non-autoregressive generations, SANA achieves state-of-the-art performance.

5.2 Comparison with Two-Stage Methods

We further compare SANA with the two-stage methods. As shown in Table 3, there is an obvious margin between SANA and the two baselines, which shows that SANA can more effectively model the two-stage process. In order to prove that SANA can make use of the information provided by the first stage, we use the gold standard (i.e., the oracle skeleton or content-plan extracted from heuristics methods) as the input of the models used in the second stage. With this setup, SANA has made significant improvements in multiple automatic metrics while other methods have limited improvements. Specially, the improvements of Pivot are limited because its gold standard does not model the order of the attributes. Although the first stage of Content-Plan is similar to SANA, its PARENT scores (either precision, recall and F1) has not been obvious improved, especially on WikiPerson dataset. This shows that the edit-based decoder of SANA can make use of the oracle skeleton to produce high quality descriptions.

Model	Fluency ↑	Coverage ↑	Correctness ↑
Pivot	3.40	3.58	2.89
Content-Plan	3.39	3.70	2.98
Struct-Aware	3.31	3.60	2.94
DesKnow	3.45	3.42	3.07
SANA	**3.46**	**3.72**	**3.11**

Table 4: Human evaluation on WikiPerson for SANA and baselines. The scores (higher is better) are based on fluency, coverage and correctness, respectively.

5.3 Human Evaluation

We report the human evaluation result on the WikiPerson dataset in Table 4. From the demonstrated results, it can be found that SANA outperforms the other end-to-end or two-stage models on all the human evaluation metrics. This is consistent with our model's performance in the automatic evaluation. In the evaluation of fluency, though the

(1) Name_ID: William Edward Ayrton **(2) Place_of_burial**: Brompton Cemetery **(3) Place_of_birth**: London
(4) Educated_at: University College London **(5) Date_of_birth**: 14 September 1847 **(6) Occupation**: Physicist
(7) Date_of_death: 8 November 1908 **(8) Aware_received**: Fellow of the Royal Society **(9) Child**: Barbara Ayarton-Gould

Model	Miss	Output
DesKnow	**(9)**	William Edward Ayrton Fellow of the Royal Society (14 September 1847 – 8 November 1908) was a British Physicist . Brompton Cemetery he was born in London the son of Sir Thomas and his wife Mary (nee Fleming) . he was educated at University College School and University College London .
Struct-Aware	**(1) (2) (9)**	William Edward Keeler Fellow of the Royal Society (14 September 1847 – 8 November 1908) was a British Physicist and Physicist . he was elected a Fellow of the Royal Society in 1889 and was a member of the Royal Society of London and the Royal Society of London and the Royal Society of London . he was educated at the University College London and at the University College London where he was a pupil of the chemist William.
OptimTrans	**None**	William Edward Ayrton Fellow of the Royal Society (14 September 1847 – 8 November 1908) was an English Physicist . William was born in London and educated at University College London. he is buried in Brompton Cemetery London . he was elected a Fellow of the Royal Society in 1901. he was the father of Barbara Ayrton-Gould .
SANA	**None**	William Edward Ayrton Fellow of the Royal Society (14 September 1847 – 8 November 1908) was an English Physicist . he is buried in Brompton Cemetery London . he studied physics at University College London . Ayrton was born in London . he was the father of Barbara Ayrton-Gould.

Table 5: Example outputs from different methods. The red text stands for the hallucinated content in each generated description. Given the table, all the models **except SANA** generate unfaithful content to varying degrees. Meanwhile, both DesKnow and Struct-Aware miss some table facts, while SANA recovers them all.

models except for Struct-Aware reach similar performances, SANA performs the best, which demonstrates that its generation has fewer grammatical and semantic mistakes. In the evaluation coverage, SANA outperforms the Content-Plan model and defeats the other models by a large margin. This result is consistent with our proposal that SANA can cover sufficient information in the source table, and it can ensure the informativeness of generation. As to correctness, the advantage of SANA over the other models indicates that our model generates more faithful content and suffers less from the hallucination problem. It should be noted that although Content-Plan and DesKnow are on par with SANA on coverage and correctness respectively, they fail to perform well on both metrics in contrast with SANA. This indicates that our model generates both informative and faithful content.

5.4 Case Study

Table 5 shows the descriptions generated by different methods from the test set of WikiPerson.[4] DesKnow and Struct-Aware miss some attributes and hallucinate unfaithful contents (marked in red). Although OptimTrans achieves better coverage, it hallucinates the unfaithful content "in 1901" not entailed by the table. Compared to these methods, our method can cover all the attributes in the table and does not introduce any unfaithful contents. In addition, the generation length of SANA is shorter than Struct-Aware and OptimTrans, which shows that SANA can use more concise text to cover the facts of the table. These results indicate our method is capable of generating faithful and informative text. We put more generation examples in Appendix B.

[4]For fair comparison, we use the generation examples of baselines provided by Wang et al. (2020)

6 Conclusion

In this paper, we focus on faithful and informative table-to-text generation. To this end, we propose a novel skeleton-based method that combines both autoregressive and non-autoregressive generations. The method divides table-to-text generation into skeleton construction and surface realization stages. The separated stages helps model better learn the correlation between the source table and reference. In the surface realization stage, we further introduce an edit-based non-autoregressive model to make full use of the skeleton. We conduct experiments on the WikiBio and WikiPerson datasets. Both automatic and human evaluations demonstrate the effectiveness of our method, especially on faithfulness and coverage.

Acknowledgements

We thank Tianyu Liu for his suggestions on this research and his providing of inference results of the baseline models. We also thank Yunli Wang for the insightful discussion.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.

Katja Filippova. 2020. Controlled hallucinations: Learning to generate faithfully from noisy data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 864–870.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. 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 6114–6123.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

V. I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710.

Tianyu Liu, Fuli Luo, Qiaolin Xia, Shuming Ma, Baobao Chang, and Zhifang Sui. 2019a. Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6786–6793.

Tianyu Liu, Fuli Luo, Pengcheng Yang, Wei Wu, Baobao Chang, and Zhifang Sui. 2019b. Towards comprehensive description generation from factual attribute-value tables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5985–5996.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shuming Ma, Pengcheng Yang, Tianyu Liu, Peng Li, Jie Zhou, and Xu Sun. 2019. Key fact as pivot: A two-stage model for low resource table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2047–2057.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1539–1550.

Feng Nie, Jinpeng Wang, Jin-ge Yao, Rong Pan, and Chin-Yew Lin. 2018. Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

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.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

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

Sascha Rothe, Shashi Narayan, and A. Severyn. 2019. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, et al. 2019. Long and diverse text generation with planning-based hierarchical variational model. 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 3248–3259.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97, pages 5976–5985. PMLR.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. *Advances in Neural Information Processing Systems*, 32:3016–3026.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with levenshtein transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3536–3543.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Bayu Trisedya, Jianzhong Qi, and Rui Zhang. 2020. Sentence generation for entity description with content-plan attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9057–9064.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 10–21.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

A Automatic Skeleton Annotation

Algorithm 1 describes the automatic skeleton annotation process. Given a table and its corresponding description, we first collect tokens appearing in both the table and description except the stop words, then these tokens are sorted based on their positions in the description in ascending order. In this way, we can obtain a sequence composed of the selected tokens. We regard this sequence as a skeleton.

B More Generation examples

We further provide a case study, using another two examples (including a very challenging example which needs to recover a large number of facts), to show the effectiveness of our method SANA. In the following pages, we show the example outputs in Table 6 and 7. In these examples, the SANA model shows much better capability of generating informative and faithful descriptions compared with the baselines.

Algorithm 1 Automatic Skeleton Annotation

Input: A stop word set W, a parallel dataset $D = \{(T_1, Y_1^*), (T_2, Y_2^*), ..., (T_{|D|}, Y_{|D|}^*)\}$;
Output: A skeleton list S;

Initial the skeleton list $S = []$
for $(T_i, Y_i^*) \in D$ **do**
  $T_i = ((k_1, v_1), (k_2, v_2), ..., (k_m, v_m))$
  $V_i = (v_1, v_2, ..., v_m)$
  $Y_i^* = (y_1^*, y_2^*, ..., y_n^*)$
  Initialize the skeleton list $S_i = []$
  for $y_j^* \in Y_i^*$ **do**
    if $y_j^* \in V_i$ and $y_j^* \notin W$ **then**
      Append token y_j^* to the skeleton list S_i
    end if
  end for
  collect the skeleton list S += S_i
end for

(1) Name_ID: Aaron Miller **(2) Member_of_sports_team**: Colorado Avalanche **(3) Member_of_sports_team**: Quebec Nordiques **(4) Date_of_birth**: August 11 1971 **(5) Place_of_birth**: Buffalo, New York **(6) Country_of_citizenship**: United States **(7) Participant_of**: 2006 Winter Olympics **(8) Participant_of**: 2002 Winter Olympics **(9) Sport**: Ice hockey **(10) Position_played_on_team / speciality**: Cornerback

Model	Miss	Output
DesKnow	**7, 8**	Aaron Miller (born August 11 1971) is an United States former professional Ice hockey Defenceman who played in the National Hockey League (NHL) for the Quebec Nordiques and the Colorado Avalanche . he was born in Buffalo, New York and played for the Quebec Nordiques and the Ottawa Senators .
Struct-Aware	**7, 8**	Aaron Miller (born August 11 1971 in Buffalo, New York New York) is a retired United States professional Ice hockey Defenceman who played in the National Hockey League (NHL) for the Quebec Nordiques Colorado Avalanche Colorado Avalanche Colorado Avalanche Colorado Avalanche and Quebec Nordiques. he was drafted in the 2nd round of overall of the 2002 NHL Entry Draft.
OptimTrans	**None**	Aaron Miller (born August 11 1971 in Buffalo, New York) is an United States former professional Ice hockey Defenceman who played in the National Hockey League (NHL) for the Quebec Nordiques and Colorado Avalanche . he was a member of the United States men's national Ice hockey team at the 2002 Winter Olympics and 2006 Winter Olympics.
SANA	**None**	Aaron Miller (born August 11 1971 in Buffalo, New York) is a retired United States professional Ice hockey Defenceman . he also played for the Quebec Nordiques and the Colorado Avalanche . Miller was also a member of United States 's ice hockey in the 2002 Winter Olympics and 2006 Winter Olympics .

Table 6: Example outputs from different methods. The red text stands for the hallucinated content in each generated description. Compared with DesKnow and Struct-Aware, SANA recovers all the table facts without generating any unfaithful content.

(1) Name_ID: Émile Mbouh **(2) Member_of_sports_team**: Le Havre AC **(3) Member_of_sports_team**: Perlis FA **(4) Member_of_sports_team**: Sport Benfica e Castelo Branco **(5) Member_of_sports_team**: Qatar SC **(6) Member_of_sports_team**: Vitória S.C. **(7) Member_of_sports_team**: Tanjong Pagar United FC **(8) Member_of_sports_team**: CS Chênois **(9) Member_of_sports_team**: Cameroon national football team **(10) Number_of_matches_played**: 46 **(11) Number_of_points / goals / set_scored**: 2 **(12) Member_of_sports_team**: Union Douala **(13) Member_of_sports_team**: Diamant Yaoundé **(14) Member_of_sports_team**: Ettifaq FC **(15) Member_of_sports_team**: Liaoning Whowin F.C. **(16) Member_of_sports_team**: Sabah FA **(17) Place_of_birth**: Douala **(18) Date_of_birth**: 30 May 1966 **(19) Sport**: Association football **(20) Participant_of**: 1994 FIFA World Cup **(21) Participant_of**: 1990 FIFA World Cup **(22) Position_played_on_team / speciality**: Midfielder

Model	Miss	Output
DesKnow	**2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 15, 16, 21, 22**	Émile Mbouh (born 30 May 1966) is a former Cameroon national football team Association football . he was born in Douala and played for the Tanjong Pagar United FC in the 1994 FIFA World Cup .
Struct-Aware	**2, 3, 4, 5, 6, 8, 10, 11, 12, 13, 14, 15, 16, 17, 21**	Émile Mbouh, (born 30 May 1966) is a Cameroonian retired Association football who played as a Midfielder . Le represented Cameroon national football team at the 1994 FIFA World Cup and 1994 FIFA World Cup . he played for Le FC Sport Yaoundé, United Yaoundé and Tanjong Pagar United FC
OptimTrans	**2, 3, 5, 6, 8, 12, 13, 14**	Émile Mbouh (born 30 May 1966) is a Cameroonian retired Association football who played as a Midfielder . born in Douala Émile began his career with Sport Benfica e Castelo Branco and Tanjong Pagar United FC . he also represented Cameroon national football team at the 1994 FIFA World Cup and 1990 FIFA World Cup . he also played for Sabah FA and Liaoning Whowin F.C. in the Malaysia Super League . he also played for Tanjong Pagar United FC and Liaoning Whowin F.C. in the Chinese Super League.
SANA	**3, 12, 13, 21**	Émile Mbouh (born 30 May 1966) is a retired Cameroonian Association football who played as a Midfielder . born in Douala Mbouh played club football in France for Sport Benfica e Castelo Branco Le Havre AC CS Chênois Vitória S.C. Tanjong Pagar United FC Qatar SC Ettifaq FC Tanjong Pagar United FC Qatar SC Sabah FA and Liaoning Whowin F.C. . Mbouh played for the Cameroon national football team (46 caps and scoring 2 goals) and two games at the 1994 FIFA World Cup .

Table 7: Example outputs from different methods. The red text stands for the hallucinated content in each generated description. This table contains a large number of facts to recover, which makes the case very challenging. In contrast with the other models, SANA misses much fewer facts and does not produce unfaithful content.