Author/s:
Trisedya, BD;Wang, X;QI, J;Zhang, R;Cui, Q

Title:
Grouped-Attention for Content-Selection and Content-Plan Generation

Date:
2021

Citation:
Trisedya, B. D., Wang, X., QI, J., Zhang, R. & Cui, Q. (2021). Grouped-Attention for Content-Selection and Content-Plan Generation. Moens, MF (Ed.) Huang, X (Ed.) Specia, L (Ed.) Yih, SWT (Ed.) Findings of the Association for Computational Linguistics: EMNLP 2021, pp.1935-1944. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-emnlp.166.

Persistent Link:
https://hdl.handle.net/11343/302142

Grouped-Attention for Content-Selection and Content-Plan Generation

Bayu Distiawan Trisedya[14*] **Xiaojie Wang**[2*] **Jianzhong Qi**[1] **Rui Zhang**[3] **Qingjun Cui**[2]
[1]The University of Melbourne [2]Amazon.com, Inc. [3]www.ruizhang.info [4]Universitas Indonesia
[1]{bayu.trisedya, jianzhong.qi}@unimelb.edu.au
[2]{xiojie, qingjunc}@amazon.com [3]rayteam@yeah.net

Abstract

Recent neural data-to-text generation models employ Pointer Networks to explicitly learn content-plan given a set of attributes as input. They use LSTM to encode the input, which assumes a sequential relationship in the input. This may be sub-optimal to encode a set of attributes, where the attributes have a composite structure: the attributes are disordered while each attribute value is an ordered list of tokens. We handle this problem by proposing a neural content-planner that can capture both local and global contexts of such a structure. Specifically, we propose a novel attention mechanism called **GSC-attention**. A key component of the GSC-attention is grouped-attention, which is token-level attention constrained within each input attribute that enables our proposed model captures both local and global context. Moreover, our content-planner explicitly learns content-selection, which is integrated into the content-planner to select the most important data to be included in the generated text via an attention masking procedure. Experimental results show that our model outperforms the competitors by 4.92%, 4.70%, and 16.56% in terms of *Damerau-Levenshtein Distance* scores on the WIKIALL, WIKIBIO, and ROTOWIRE datasets, respectively.

1 Introduction

Data-to-text generation (Reiter and Dale, 2000) is an important and challenging task in natural language processing. It aims to produce sentences given structured data. There are many downstream applications of data-to-text-generation, such as biography summarization (Lebret et al., 2016), automatic weather forecasting (Mei et al., 2016), etc.

Traditional approaches follow a pipeline framework consisting of three stages: content-selection, content-planning, and surface-realization. Content-selection selects the data to be expressed; content-planning determines the structure of the sentences to be generated; and surface-realization generates the output based on the content-planning. Recent neural data-to-text generation approaches integrate these stages into an end-to-end model, i.e., tasks of the stages are learned implicitly, as end-to-end training is becoming popular (Wang et al., 2018a). Despite their success, end-to-end models without proper content-planning may generate repetitive, incomplete, and incoherent sentences. Moreover, end-to-end models are less interpretable, making it difficult to perform error analysis for further improvement.

Input (**a set of attributes**)	⟨name; Barack Hussein Obama⟩ ⟨birth_place; Honolulu, Hawaii⟩ ⟨occupation; senator⟩ ⟨occupation; politician⟩ ⟨birth_date; August 4, 1961⟩ ⟨residence; Washington, D.C.⟩ ⟨citizenship; United States⟩
Example description	Barack Obama (born August 4, 1961, in Honolulu, Hawaii) is an American politician who lives in Washington, D.C., U.S.
Output (**content-plan**)	⟨Barack, Obama, August, 4, 1961, Honolulu, Hawaii, politician, Washington, D.C.⟩

Table 1: Input and output of content-plan generation.

It has been shown that explicitly learning content-planning improves the performance (e.g., reduce repetition or generate a coherent sentence) and the interpretability of neural data-to-text generation models (Trisedya et al., 2018). In this paper, we study the problem of neural content-plan generation. Given a set of attributes of an entity, we aim to select the salient attributes (i.e., the attributes mentioned) and reorder the selected attributes such that they follow the common attribute mentioning order in natural sentences. For example, in Table 1, the input is a set of attributes for the entity Barack Obama (in the form of key-value pairs): ⟨name; Barack Hussein Obama⟩, ⟨birth_place; Honolulu, Hawaii⟩, etc. Suppose that the target descrip-

* Equal contribution

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1935–1944
November 7–11, 2021.

tion is `Barack Obama (born August 4, 1961, in Honolulu, Hawaii) is an American politician who lives in Washington, D.C., U.S.` Our goal is to generate the content-plan of the target description, which is the attribute value mentioning order in the description. In this example, the content-plan is ⟨`Barack, Obama, August, 4, 1961, Honolulu, Hawaii, politician, Washington, D.C.`⟩.

Existing neural data-to-text generation models (Puduppully et al., 2019; Trisedya et al., 2020) employ Pointer Networks (Vinyals et al., 2015) as the content-planner. There are two limitations of such models. First, the oracle Pointer Networks used in Puduppully et al. (2019) and Trisedya et al. (2020) do not explicitly learn the content-selection. Moreover, the input to a data-to-text generation model is a set of attributes, which may be given in any order and do not have a sequential relationship. Using LSTM to encode and capture the ordering relationships in the input set may be sub-optimal. Second, Pointer Networks do not handle content-selection properly. Typically, a content-planner is applied at the token-level, which selects salient tokens for generating the description. For the example in Table 1, the tokens that are supposed to be selected for the attribute `name` are `Barack` and `Obama`, while the token `Hussein` is supposed to be filtered. In summary, the input of the task is attributes of an entity that have a composite structure (instead of a sequence): the attributes are disordered while each attribute value is an ordered list of tokens. To encode such a structure properly, the encoder should learn both the representation for each token of an attribute (i.e., *local context*) and the attribute as a whole (i.e., *global context*).

To address the limitations above, we propose a novel neural content-planner. Specifically, we propose a novel **GSC-attention** to capture the local and global contexts of the input set. The GSC-attention consists of three attention mechanisms. The first is *grouped-attention*, a token-level attention mechanism restricted within each attribute. The grouped-attention lets an attribute representation captures the relationship between tokens in an attribute (i.e., local context). The second is *self-attention* among attribute level representations. This attention updates the attributes' representations based on the overall attribute information (i.e., global context). The third is *cross-attention*, which updates token representations based on attribute representations to capture the attributes' composite relationships. We stack multiple layers of GSC-attention, and the updated token representations of a layer are used as input for the next layer. This way, GSC-attention ensures capturing the interaction between the local and global contexts.

We further propose a *content-selection masking* procedure to integrate content-selection into our content-planner. Content-selection aims to filter non-salient attribute tokens, which helps the content-planner arrange the selected attribute tokens properly. We integrate content-selection with our content-planner as follows. First, the content-selection module generates a *pseudo-content-selection*, which is a binary value that indicates whether an attribute token should be selected. Then, the pseudo-content-selection is used as a mask in the content-planning module to let content-planning focus on arranging the selected attribute tokens. The advantages of our masking procedure are twofold. First, it allows end-to-end joint training between content-selection and content-planning. Second, it explicitly learns content-selection from the content-planner, which improves the interpretability of the model, specifically in analyzing the error of the model.

The contributions of the paper are as follows.

- We propose a neural model for content-plan generation from a set of attributes that explicitly learns content-selection and content-planning.
- To properly encode a set of attributes, we propose a novel attention mechanism, GSC-attention, that effectively captures the local and the global contexts in an input set and their composite relationships.
- To integrate the content-selection and content-planning procedures, we propose a content-selection masking procedure.

2 Related Work

Content-planning is an essential part of data-to-text generation to determine the order of data mentioned in generated sentences. In early data-to-text generation approaches, content-planning is done by creating hand-crafted rules (Scott and de Souza, 1990; Hovy, 1993), using template-based models (McKeown, 1992; Reiter et al., 2000), or exploiting machine learning models (Duboue and McKeown, 2003; Barzilay and Lapata, 2005; Liang et al., 2009). Content-planning is coupled with

content-selection and surface-realization. Content-selection mainly relies on hand-built heuristics (Kukich, 1983; Ehud Reiter, 1997) and shallow statistical machine learning models (Duboue and McKeown, 2001, 2002; Kim and Mooney, 2010; Konstas and Lapata, 2012). For surface-realization, earlier studies exploit template-based models (McKeown et al., 1997; Deemter et al., 2005) and language models (Angeli et al., 2010).

As end-to-end training is becoming prevalent (Wang et al., 2019, 2021b), recent data-to-text generation approaches employ neural networks which can be trained end-to-end (Shen et al., 2020). These approaches use encoder-decoder frameworks (Cho et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017). The encoder is used to transform the input into some vector representation. The decoder takes the vector representation as context to generate the target sentence. In both the encoder and the decoder, sequence neural network models, such as LSTM (Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani et al., 2017), are used to process the data. Studies in this line of work include biography summarization (Lebret et al., 2016; Liu et al., 2018), weather forecasting (Mei et al., 2016), and game summarization (Wiseman et al., 2017). Despite their success, these models may generate incoherent sentences since they do not have a proper content-plan.

To improve the coherence of the generated text, recent studies re-introduce content-planning mechanism in neural approaches of data-to-text generation. Sha et al. (2018) propose a content-planning mechanism via link-based attention to model relationships among input data. It learns a matrix where each element indicates transition probability of two attributes. Goldfarb-Tarrant et al. (2020) employ the Aristotelian framework and a re-scoring model to find the best story plot in story generation. Other studies along this line propose two-stage models, i.e., learning the content-planning and the surface-realization consecutively. For example, Hua and Wang (2019) use LSTM-based content-plan, while Puduppully et al. (2019) exploit Pointer Networks (Vinyals et al., 2015) for content-planning. Both works use LSTM-based surface-realization. Trisedya et al. (2020) propose a content-planning-based attention model, where the content-plan is learned using Pointer Networks.

The aforementioned models use content-planning mechanisms with LSTM or shallow transformation networks as the input encoder. However, LSTM is sub-optimal to capture the relationships in the input. This is because the input is a set (e.g., a set of attributes), and each attributes may consist of multiple tokens, which form a hierarchical structure among the attributes instead of a sequence. Thus, applying LSTM on the input data may lead to improper representations of the input. In this paper, we propose a novel content-planner to capture the hierarchical structure of the input.

3 Preliminary

Let $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ be a set of n attributes of an entity. Each attribute $a_i = \langle k_i; v_i \rangle$ is a pair of a key k_i and a value v_i ($i = 1, 2, ..., n$). Key k_i denotes the type of an attribute. Value $v_i = [v_{i,1}, v_{i,2}, ..., v_{i,m_i}]$ is a sequence of m_i tokens. The content-plan of a sentence specifies which attributes are selected to be included in the generated sentence and their order in the sentence. It consists of a sequence of tokens, where each token belongs to one of attributes. The content-plan can be represented by a sequence of index pairs $\mathcal{P} = [\langle i_1, j_1 \rangle, \langle i_2, j_2 \rangle, ..., \langle i_c, j_c \rangle]$. Here, i_k indicates the index of an attribute, and j_k indicates the index of a token within the attribute ($k = 1, 2, ..., c$).

Given a set of attributes $\mathcal{A}$, we aim to generate the content-plan $\mathcal{P}$. Note that our goal is *not* to generate a sentence given a set of attributes. Our work aims to organize the attributes in a way that enables generating better (i.e., non-repetitive and coherent) sentences for downstream textual generation models.

4 Proposed Model

Solution overview. We propose a novel end-to-end model for content-plan generation. The model consists of four modules: *an embedding and linear transformation module*, *an attribute-encoding module*, *a content-selection module*, and *a content-planning module*.

The embedding and linear transformation module (cf. Section 4.1) aims to obtain initial vector representations of attributes and tokens. The representations should maintain the order of tokens within an attribute and should not impose any order on the set of attributes.

The attribute-encoding module (cf. Section 4.2) aims to encode the attributes of an entity into fixed-length embeddings, which will be used by the

content-selection and the content-planning modules as inputs. The attributes of an entity have a composite structure: the attributes are disordered while each attribute value is an ordered list of tokens. To encode such a structure, we propose a novel attention mechanism capable of learning a representation for each token of an attribute and the attribute as a whole.

The content-selection module (cf. Section 4.3) aims to accurately predict both content-selection labels and attribute-selection labels. The content-selection labels help highlight the selected attributes, whereas the attribute-selection labels provide more supervision signals to train parameters of the content-selection module.

The content-planning module (cf. Section 4.4) integrates the content-selection into the Pointer Networks to generate a content-plan.

4.1 Embedding and Linear Transformation

We obtain representations of tokens and attributes by applying a linear transformation to their embeddings. To distinguish the same token in different attributes and maintain the internal order of tokens within an attribute, we represent a token of an attribute by a quadruple $t_{i,j} = [k_i, v_{i,j}, f_{i,j}, r_{i,j}]$ where k_i is the key of the attribute, $v_{i,j}$ is the token, $f_{i,j}$ and $r_{i,j}$ are the forward and backward positions of the token within the attribute, respectively. The representation of the token $\boldsymbol{t}_{i,j}$ is computed by applying a linear transformation on an embedding of the key $\boldsymbol{e}_{k_{i,j}}$, an embedding of the token $\boldsymbol{e}_{v_{i,j}}$, embeddings of the forward and backward positions $\boldsymbol{e}_{f_{i,j}}, \boldsymbol{e}_{r_{i,j}}$ as follows:

$$\boldsymbol{k}_{i,j} = \tanh(\mathbf{W}_{\boldsymbol{k}}[\boldsymbol{e}_{k_{i,j}}, \boldsymbol{e}_{f_{i,j}}, \boldsymbol{e}_{r_{i,j}}] + \boldsymbol{b}_{\boldsymbol{k}}), \quad (1)$$

$$\boldsymbol{v}_{i,j} = \tanh(\mathbf{W}_{\boldsymbol{v}}[\boldsymbol{e}_{v_{i,j}}, \boldsymbol{e}_{f_{i,j}}, \boldsymbol{e}_{r_{i,j}}] + \boldsymbol{b}_{\boldsymbol{v}}), \quad (2)$$

$$\boldsymbol{t}_{i,j} = \tanh(\boldsymbol{k}_{i,j} + \boldsymbol{v}_{i,j}). \quad (3)$$

The representation of an attribute is computed by:

$$\boldsymbol{a}_i = \tanh(\mathbf{W}_{\boldsymbol{a}}\boldsymbol{e}_{a_i} + \boldsymbol{b}_{\boldsymbol{a}}), \quad (4)$$

where $\boldsymbol{e}_{a_i}$ is an embedding of attribute a_i.

4.2 Attribute Encoding Module

We propose a novel attribute encoder to learn two types of embeddings: embeddings of tokens and embeddings of attributes. The attribute encoder consists of a stack of N identical layers (N is a system parameter). Each layer is composed of two sub-layers. The first is a Grouped-Self-Cross attention (GSC-attention) layer, a novel attention mechanism, and the second is a feed-forward layer.

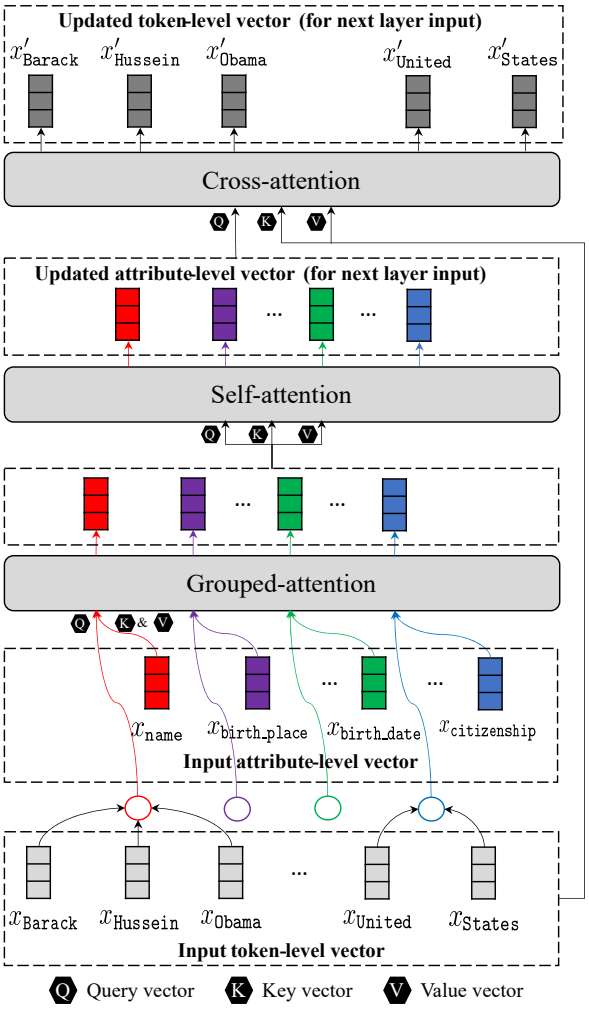

Figure 1: GSC-attention

We employ each of the two sub-layers to learn a residual function and then apply a batch normalization layer. Formally, the output from each sub-layer is $\text{BatchNorm}(\boldsymbol{x} + \text{SubLayer}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is an input to the sub-layer, $\text{SubLayer}(\cdot)$ is the residual function learned, and $\text{BatchNorm}(\cdot)$ denotes the batch normalization. We use a fixed dimensionality d for all layers throughout this paper to facilitate the residual connection.

As a key building block of the attribute encoder, the GSC-attention has three attention mechanisms: a grouped-attention, a self-attention, and a cross-attention. We illustrate them in Fig. 1 and describe the layers next.

Grouped-attention. The Grouped-attention aims to learn a representation for each attribute based on interactions among the tokens within the attribute. For simplicity, we use a one-dimensional index to represent a sequence of all tokens of all input attributes: $[t_1, t_2, ..., t_m]$, which is a simplified form of $[t_{1,1}, t_{1,2}, ..., t_{n,m_n}]$. In Grouped-attention, we require the tokens of the same attribute to appear

together in the sequence. The different attributes can be randomly ordered in the sequence. Let $\mathbf{G} \in \mathbb{R}^{n \times m}$ be a group mask matrix where each entry $g_{i,j} = 1$ if the value of an attribute a_i contains a token t_j, and $g_{i,j} = 0$ otherwise. We compute the Grouped-attention as follows.

$$\mathbf{Q}_g = [\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n]^\mathsf{T} \mathbf{W}_{\mathbf{Q}_g} \quad (5)$$

$$\mathbf{K}_g = [\boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_m]^\mathsf{T} \mathbf{W}_{\mathbf{K}_g} \quad (6)$$

$$\mathbf{V}_g = [\boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_m]^\mathsf{T} \mathbf{W}_{\mathbf{V}_g} \quad (7)$$

$$\text{GroupAtt} = \text{softmax}\left(\frac{(\mathbf{Q}_g \mathbf{K}_g^\mathsf{T}) \odot \mathbf{G}}{\sqrt{d}}\right)\mathbf{V}_g \quad (8)$$

where $\mathbf{Q}_g, \mathbf{K}_g$, and $\mathbf{K}_g$ are the query, key, and value matrices, respectively. Here, $\mathbf{Q}_g \in \mathbb{R}^{n \times d}$, while $\mathbf{K}_g, \mathbf{V}_g \in \mathbb{R}^{m \times d}$. $\odot$ denotes element-wise multiplication of two matrices, and $\mathbf{W}$ denotes a learned parameter. The use of the group mask $\mathbf{G}$ makes the attention weights focus on inter-attribute interactions, i.e., interactions among tokens within the same attribute.

Self-attention. The grouped-attention layer only considers intra-attribute interactions but not inter-attribute interactions, i.e., interactions among tokens from different attributes. To capture inter-attribute interactions, we employ a self-attention layer over the attribute embeddings

$$\text{SelfAtt} = \text{softmax}\left(\frac{\mathbf{Q}_s \mathbf{K}_s^\mathsf{T}}{\sqrt{d}}\right)\mathbf{V}_s \quad (9)$$

where $\mathbf{Q}_s, \mathbf{K}_s$, and $\mathbf{K}_s$ are the query, key, and value matrices that are computed from the updated attribute representation computed by Eq. 8.

Cross-attention After obtaining the attribute embeddings that capture both intra-attribute and inter-attribute interactions, we update the token embeddings by a cross-attention layer

$$\text{CrossAtt} = \text{softmax}\left(\frac{\mathbf{Q}_c \mathbf{K}_c^\mathsf{T}}{\sqrt{d}}\right)\mathbf{V}_c \quad (10)$$

where $\mathbf{Q}_c$ is a query matrix computed from the token embedding $\boldsymbol{t}$, $\mathbf{K}_c$ and $\mathbf{V}_c$ are the key and value matrices, respectively, which are computed based on the attribute representation from the self-attention (cf. Eq. 9). Unlike the grouped-attention, we do not employ the group mask in the cross-attention, such that the token embeddings consider both intra-attribute and inter-attribute interactions among attribute tokens.

4.3 Content-Selection Module

We define content-plan as a sequence of attribute-token in the order that they are mentioned in a sentence. For example, the content-plan for the example in Table 1 is ⟨`Barack, Obama, August, 4, 1961, Honolulu, Hawaii, politician, Washington, D.C.`⟩. The content-selection label is a set of binary variables $F = \{l_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m_i\}$ where subscripts i and j indicate the attribute index and the token index, i.e., $l_{i,j}$ indicates whether token $t_{i,j}$ in the value of attribute a_i appears in the content-plan ($l_{i,j} = 1$) or not ($l_{i,j} = 0$).

Given a sequence of token embeddings outputted by the attribute encoding module $[\boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_m]$, we use a fully-connected layer on top of each token embedding to compute the probability that the content-plan includes the token.

$$p_j = \text{sigmoid}(\mathbf{W}_{\boldsymbol{t}} \boldsymbol{t}_j + b_{\boldsymbol{t}}), \quad (11)$$

where $\mathbf{W}$ is a trainable parameters of a fully-connected layer, and $b_{\boldsymbol{t}}$ is the bias. We refer to such probabilities as *content-selection probabilities*. Since the content-selection labels are binary, we use a binary cross-entropy loss:

$$\mathcal{L}_{\text{CS}} = \sum_{j=1}^{m} l_j \log p_j + (1 - l_j) \log(1 - p_j). \quad (12)$$

4.4 Content-Planning Module

Given the token embeddings $t_j = [\boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_m]$ and the content-selection probabilities $p_j = [p_1, p_2, ..., p_m]$, we aim to generate a sequence of pointers $\mathcal{P} = [j_1, j_2, ..., j_c]$ where each pointer j_k is an index corresponding to the j_k-th token. We adapt Pointer Networks (Vinyals et al., 2015) to incorporate content-selection probabilities into generating the content-plan via content-selection masking procedure as follows.

We first transform the content-selection probabilities into pseudo content-selection, a binary value (i.e., $p_j = 1$ if the corresponding probability score > 0.5, otherwise $p_j = 0$) that indicates whether an attribute token is selected or not. Then, we use the Pointer Networks to produce a vector that modulates a content-based attention mechanism over tokens at each step. Let $[\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_c]$ be a sequence of hidden states of the networks. At each step k, we incorporate the pseudo content-selection to compute pointer attention over all tokens:

$$\boldsymbol{c} = \boldsymbol{v} \tanh(\mathbf{W}_{\boldsymbol{t}}[\boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_m] + \mathbf{W}_{\boldsymbol{h}} \boldsymbol{h}_k), \quad (13)$$

$$\boldsymbol{u} = \text{softmax}(\boldsymbol{c} \odot [p_1, p_2, ..., p_m]). \quad (14)$$

Here, $\boldsymbol{u}$ is a probability distribution over the tokens, where u_j is the probability for token $\boldsymbol{t}_j$,

$\boldsymbol{v}, \mathbf{W_t}, \mathbf{W_h}$ are parameters. We train the content-planner by maximizing a cross-entropy loss:

$$\mathcal{L}_{\text{CP}} = \sum_{k=1}^{c} \sum_{j=1}^{m} \mathbb{1}(j = j_k) \log u_{j_k}, \tag{15}$$

where $\mathbb{1}(\cdot)$ is an indicator function that returns 1 if the proposition in the argument is true, and 0 otherwise. The overall objective function is a sum of the objective functions of the binary classifiers and the pointer generator's objective function.

$$\mathcal{L} = \mathcal{L}_{\text{CS}} + \mathcal{L}_{\text{CP}}. \tag{16}$$

We optimize the overall objective function by back-propagation algorithm (Wang et al., 2018b, 2021a)

5 Experiments

5.1 Dataset

We evaluate our proposed model over three real-world datasets, **WIKIALL** (Trisedya et al., 2020), **WIKIBIO** (Lebret et al., 2016), and **ROTOWIRE** (Wiseman et al., 2017). The WIKIALL and WIKIBIO datasets contain attributes-description pairs. The description is a sentence that describes an entity extracted from the first sentence of the corresponding Wikipedia page. The attributes comprise a set of attributes that belongs to the entity. The WIKIALL dataset obtains the attributes from Wikidata (Vrandecic and Krötzsch, 2014), while the WIKIBIO dataset obtains the attributes from the Wikipedia infobox. The ROTOWIRE dataset contains pairs of data-records and NBA basketball game summary. The data-record is a table of statistics about an NBA game.

The WIKIALL dataset contains $152,231$ attributes-description pairs. It includes 53 entity types with an average of 15 attribute types (and up to 100 attribute tokens) per entity and an average of 20 tokens per description. The WIKIBIO dataset focuses on biography (i.e., this dataset only contains one entity type: `PERSON`). It contains $728,321$ attributes-description pairs. Its average number of attribute types per entity is 19 (and up to 100 attribute tokens), and its average number of tokens per description is 26. The ROTOWIRE dataset contains $4,900$ record-summary pairs with 39 record types. Its average number of attribute tokens per game record is 600, and its average number of tokens per summary is 337.

Our primary goal is to generate a content-plan from a set of attributes. Our proposed model includes content-selection learning to obtain a better content-plan. We need content-selection and content-plan labels for each attributes-description pair in all three datasets to train such a model. For the WIKIALL and WIKIBIO datasets, we use the data pre-processed by Trisedya et al. (2020). For the ROTOWIRE dataset, we use the data pre-processed by Puduppully et al. (2019). The pre-processed data have content-plan labels, but not content-selection labels. To obtain content-selection labels, we give label 1 for input tokens that appear in the target content-plan, and label 0 for the rest of the input tokens.

5.2 Training Details

We implement our model with Tensorflow and train it on NVIDIA GeForce RTX 2080 Ti. We use grid search to tune the hyperparameters. We select the embedding size in $[8, 128]$, the dropout rate in $[0.1, 0.5]$, and the learning rate in $[1e^{-2}, 1e^{-4}]$. The best hyper-parameter settings are as follows. We use 128 hidden units for the networks. We use 32, 16, and 8 dimensions of word embeddings (attribute value token), type embeddings (attribute key), and position embeddings, respectively. We use a 0.1 dropout rate. We use Adam (Kingma and Ba, 2015) with a learning rate of $1e^{-4}$. The memory cost to store the model is $2,475$ MB, and the average running time for training and testing the model is 350 minutes on the WIKIALL dataset.

5.3 Tested Models

We compare our proposed model (**GSC-attention**) with the following models.

- **Enc-Dec** (Wiseman et al., 2017), which employs an encoder-decoder framework with LSTM in both the encoder and the decoder. It also uses conditional copy (Gulcehre et al., 2016) on the decoder side.
- **NCP** (Puduppully et al., 2019), which uses LSTM in the encoder and Pointer Networks in the decoder.
- **Transformer**, which is a direct adaptation of the Transformer model (Vaswani et al., 2017). For this model, we used the Transformer encoder and coupled it with Pointer Networks to generate content-plans.

It is worth noting that we only take the content planner part from the existing models (Enc-Dec and NCP) since the main goal in this paper is to generate a content-plan from a set of attributes. For ablation tests, we run three variants for each compared model as follows.

Model	WIKIALL			WIKIBIO			ROTOWIRE		
	Precision	Recall	DLD	Precision	Recall	DLD	Precision	Recall	DLD
Enc-Dec	80.90 ±.27	68.79 ±.23	71.33 ±.28	70.01 ±.29	55.51 ±.29	58.42 ±.20	18.22 ±.15	27.12 ±.19	8.44 ±.18
+ Sorted	81.92 ±.21	72.30 ±.11	73.33 ±.24	69.88 ±.27	57.81 ±.24	59.07 ±.17	18.39 ±.19	27.79 ±.28	9.44 ±.10
+ CS loss	82.75 ±.10	70.98 ±.30	73.73 ±.25	71.00 ±.28	58.71 ±.24	59.82 ±.26	18.63 ±.30	28.66 ±.13	9.83 ±.13
+ CS mask	83.03 ±.21	70.56 ±.28	75.95 ±.14	71.88 ±.18	55.85 ±.29	60.70 ±.21	20.19 ±.18	26.51 ±.29	10.46 ±.20
NCP	85.72 ±.30	73.46 ±.23	73.96 ±.17	77.80 ±.11	61.76 ±.16	61.97 ±.25	33.88 ±.27	50.94 ±.26	18.27 ±.27
+ Sorted	86.48 ±.24	77.49 ±.22	77.16 ±.20	77.73 ±.12	62.60 ±.17	61.75 ±.30	34.65 ±.10	**51.37** ±.13	19.04 ±.19
+ CS loss	86.99 ±.10	75.82 ±.26	76.15 ±.14	78.42 ±.14	62.99 ±.20	62.26 ±.21	34.63 ±.19	51.30 ±.14	19.75 ±.24
+ CS mask	88.22 ±.30	73.97 ±.25	78.08 ±.19	79.00 ±.28	61.15 ±.20	63.54 ±.26	35.72 ±.15	48.09 ±.13	21.37 ±.14
Transformer	90.19 ±.17	83.21 ±.11	80.57 ±.30	78.82 ±.29	64.40 ±.11	62.86 ±.21	33.16 ±.10	33.97 ±.19	22.14 ±.21
+ Sorted	89.72 ±.15	83.18 ±.29	80.24 ±.10	79.27 ±.21	64.13 ±.20	63.48 ±.27	34.58 ±.23	37.08 ±.14	24.35 ±.21
+ CS loss	90.05 ±.14	83.63 ±.19	81.40 ±.23	80.31 ±.20	66.75 ±.21	63.90 ±.28	35.04 ±.19	39.45 ±.28	25.96 ±.15
+ CS mask	90.30 ±.18	81.99 ±.14	81.76 ±.25	79.00 ±.19	64.11 ±.14	64.41 ±.13	37.39 ±.10	37.42 ±.25	28.07 ±.12
GSC-attention	90.70 ±.27	84.12 ±.24	82.71 ±.20	80.88 ±.27	67.20 ±.29	65.12 ±.24	44.97 ±.29	41.25 ±.22	27.64 ±.14
+ Sorted	90.48 ±.29	84.42 ±.22	82.72 ±.21	80.21 ±.23	66.19 ±.28	63.79 ±.30	47.59 ±.19	42.34 ±.11	27.84 ±.25
+ CS loss	90.99 ±.28	**84.49** ±.11	82.81 ±.15	81.23 ±.19	**69.65** ±.28	65.38 ±.26	49.07 ±.12	44.39 ±.10	29.87 ±.27
+ CS mask	**92.15** ±.11	83.74 ±.12	**84.69** ±.13	**82.28** ±.13	68.91 ±.15	**65.98** ±.23	**51.68** ±.12	43.60 ±.12	**32.72** ±.27

Table 2: Main results: comparison of different encoders

+ **Sorted**. This variant does not change the model but takes a sorted input (alphabetically ordered by the attribute keys). The intuition is that sorted input may be easier to learn.
+ **CS loss**. This variant jointly learns content-selection and content-planning but does not use the masking strategy described in Section 4.4.
+ **CS mask**. This variant uses the masking strategy described in Section 4.4.

5.4 Main Results

We use the following metrics to evaluate the models Wiseman et al. (2017). To measure model performance on extracting salient attributes (i.e., content-selection), we use **precision** and **recall**. To measure how well a model orders the selected attributes (i.e., content-planning), we use *Damerau-Levenshtein Distance* (**DLD**) between the generated content-plan and the gold standard.

Table 2 shows the results. From these results, we see that our proposed GSC-attention achieves the best performance for generating content-planning, indicated by the highest DLD score on all three datasets. We also see that the Transformer adaptation outperforms two existing models, Enc-Dec and NCP. This is because both existing models employ LSTM in the encoder side, which is suboptimal to encode the attribute set. Our model further outperforms the direct Transformer adaptation since our model can capture both local and global relationships among the attributes. In contrast, the Transformer adaptation linearizes the input set, which omits the local relationships between tokens within the same attributes. In general, all models achieve higher DLD scores in WIKIALL and WIKIBIO datasets but lower scores in the ROTOWIRE dataset. This is because the ROTOWIRE dataset contains a larger (i.e., 600 records per game compared to 100 attributes per entity in WIKIALL and WIKIBIO) and homogeneous (i.e., mainly, it contains numbers related to a game statistics) input.

Sorting the attributes in the input set (i.e., the **+ Sorted** variant) gives a deterministic order to the model input. However, this strategy does not ensure that the encoder (especially the LSTM encoders) can capture the relationships among the attributes. The alphabetically ordered attributes may not reflect the correct attribute relationships. These results verify that capturing the relationships of the input set is non-trivial.

Ablation test results. In the ablation tests, we aim to evaluate the effectiveness of our proposed content-selection integration. We apply our proposed integration to all models. In general, the content-selection integration improves the performance of the content-planner. All models benefit from the content-selection integration. The variants that use joint learning of content-selection and content-planning (i.e., the **+ CS loss** variants) gain 1 to 2 points in DLD comparing with the respective models without the integration. The **+ CS mask** variants further improve the content-planner's performance by 1 to 3 points in DLD. It is worth noting that the masking strategy substantially improves the precision and DLD score, but it may lower the recall. This is because the masking strategy narrows the output selection to the attribute selected by the content-selector. However, in content-plan gener-

Model	BLEU-4 / ROUGE		
	WIKIALL	WIKIBIO	ROTOWIRE
Enc-Dec	54.21 / 48.789	38.32 / 34.488	14.24 / 12.816
NCP	59.87 / 53.883	40.21 / 36.189	16.45 / 14.805
Transformer	63.48 / 57.132	41.56 / 37.404	17.12 / 15.408
GSC-attention	**64.59 / 58.131**	**43.57 / 39.213**	**19.21 / 17.289**

Table 3: Text generation results

Model	Repetition	Completeness	Coherence
Original input	1.25	1.35	1.43
Enc-Dec	2.32	2.19	1.45
NCP	2.52	2.23	1.58
Transformer	2.54	2.27	1.63
GSC-attention	**2.68**	**2.45**	**2.02**

Table 4: Manual evaluation results

ation, we argue that precision and DLD are more important than recall because we want to generate accurate planning. Take an example in the basketball summary generation. The content-planner should make an accurate content-plan prediction to generate an accurate game summary.

5.5 Evaluation with Text Generation

The primary goal of this paper is content-plan generation, which is an intermediate goal of text generation. In this experiment, we further evaluate the quality of the generated content-plans by using them for text generation. We train a text generation model using an encoder-decoder framework. Both the encoder and the decoder use the LSTM model. We train the model over the gold standard content-plan–target-sentence pairs of the dataset (cf. Section 5.1). For testing, we use the content-plan generated by the tested models as input, and we compute the BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) scores of the sentence generated by the text generation model.

For each model, we take the best variant (i.e., the CS mask variant) to generate the content-plan. Table 3 shows the text generation results. These results confirm that our proposed model achieves the best performance. The BLEU scores of the generated text from the content-plan generated by our model on WIKIALL, WIKIBIO, and ROTOWIRE as input are 64.59, 43.57, and 19.21 respectively. Note that the upper-bound performances are 67.21, 46.32, and 24.06 in terms of BLEU score in WIKIALL, WIKIBIO, and ROTOWIRE datasets, respectively.

Manual evaluations. We further perform manual evaluations on the generated text. We define three metrics for the manual evaluations: *repetition*, *completeness*, and *coherence*. *Repetition* measures whether there is repeating information (or tokens) in the generated text. *Completeness* measures whether there is a missing attribute in the text. *Coherence* measures correctness of the attribute order in the generated text.

We randomly select 100 input sets of the WIKIALL dataset along with the generated text. The manual evaluation is done by giving a score from 1 to 3 for the generated text in each metric. For each metric, score 3 is given to generated text with no error; score 2 is given to generated text with a single error; and score 1 is given to generated text with more than one error.

Table 4 shows the manual evaluation results. From these results, we can see that exploiting the content-plan helps a text generation model to produce a better output. A text generation model that takes the original attribute set as input generates text that contains many repetition errors and missing information, i.e., it achieves a low score on the repetition and completeness metric compared to the text generation models that receive a content-plan as input. This is because the generated content-plan has been filtered from unnecessary information. Among the compared content-planner, our proposed GSC-attention achieves the best score in all manual evaluation metrics. This result confirms the automatic evaluation where our proposed model outperforms the competitors.

6 Conclusions and Future Works

We presented a model for generating a content-plan from a set of entity attributes. To capture the local and global contexts from the input set, we proposed a novel GSC-attention. This attention mechanism consists of three attention schemes, which combine the intra-attention among tokens in an attribute and the inter-attention among attributes. Our content-planner further integrates a content-selection mechanism via a masking strategy. Experimental results on real-world datasets confirm the effectiveness of our model to generate a content-plan.

Despite outperforming all competitors, our generated content-plan can be further improved, especially for a large and homogeneous input set (e.g., the ROTOWIRE dataset). A further decoding strategy will be explored for future work. Another interesting direction is to design a text generation model that exploits the proposed content-planner.

Acknowledgments

This work is partially supported by Australian Research Council (ARC) Discovery Project DP180102050 and Google Faculty Research Award.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 502–512.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 331–338.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Kees van Deemter, Mariët Theune, and Emiel Krahmer. 2005. Real versus template-based natural language generation: A false opposition? *Computational linguistics*, 31(1):15–24.

Pablo Duboue and Kathleen McKeown. 2001. Empirically estimating order constraints for content planning in generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 172–179.

Pablo Duboue and Kathleen McKeown. 2002. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of the International Natural Language Generation Conference*, pages 89–96.

Pablo Duboue and Kathleen McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 121–128.

RD Ehud Reiter. 1997. Building applied natural language generation systems. *Natural Language*.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 4319–4338.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 140–149.

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

Eduard H Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial intelligence*, 63(1-2):341–385.

Xinyu Hua and Lu Wang. 2019. Sentence-level content planning and style specification for neural text generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 591–602.

Joohyun Kim and Raymond Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the International Conference on Computational Linguistics*, pages 543–551.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 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 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 91–99.

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

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

Kathleen McKeown. 1992. *Text generation*. Cambridge University Press.

Kathleen McKeown, Desmond A Jordan, Shimei Pan, James Shaw, and Barry A Allen. 1997. Language generation for multimedia healthcare briefings. In *Proceedings of the Conference on Applied Natural Language Processing*, pages 277–282.

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 North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

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*, pages 6908–6915.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press.

Ehud Reiter, Roma Robertson, and Liesl Osman. 2000. Knowledge acquisition for natural language generation. In *Proceedings of the International Conference on Natural Language Generation*, pages 217–224.

Donia Scott and Clarisse Sieckenius de Souza. 1990. Getting the message across in rst-based text generation. *Current research in natural language generation*, 4:47–73.

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*.

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. 2020. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165.

Bayu Distiawan 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*, pages 9057–9064.

Bayu Distiawan Trisedya, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. Gtr-lstm: A triple encoder for sentence generation from rdf data. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1627–1637.

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 *Proceedings of the Conference on Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 2692–2700.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Xiaojie Wang, Jianzhong Qi, Kotagiri Ramamohanarao, Yu Sun, Bo Li, and Rui Zhang. 2018a. A joint optimization approach for personalized recommendation diversification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 597–609.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2018b. Kdgan: knowledge distillation with generative adversarial networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 783–794.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2019. Doubly robust joint learning for recommendation on data missing not at random. In *International Conference on Machine Learning*, pages 6638–6647.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2021a. Adversarial distillation for learning with privileged provisions. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 43(03):786–797.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. 2021b. Combating selection biases in recommender systems with a few unbiased ratings. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 427–435.

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