Operation-guided Neural Networks for High Fidelity Data-To-Text Generation

Feng Nie[1][*], Jinpeng Wang[2], Jin-Ge Yao[2], Rong Pan[1], Chin-Yew Lin[2]
[1]Sun Yat-Sen University [2]Microsoft Research Asia
fengniesysu@gmail.com, {jinpwa, jinge.yao, cyl}@microsoft.com, panr@sysu.edu.cn

Abstract

Recent neural models for data-to-text generation are mostly based on data-driven end-to-end training over encoder-decoder networks. Even though the generated texts are mostly fluent and informative, they often generate descriptions that are not consistent with the input structured data. This is a critical issue especially in domains that require inference or calculations over raw data. In this paper, we attempt to improve the fidelity of neural data-to-text generation by utilizing pre-executed symbolic operations. We propose a framework called **Op**eration-guided **Att**ention-based sequence-to-sequence network (OpAtt), with a specifically designed gating mechanism as well as a quantization module for operation results to utilize information from pre-executed operations. Experiments on two sports datasets show our proposed method clearly improves the fidelity of the generated texts to the input structured data.

1 Introduction

Data-to-text generation is a classic language generation task that takes structured data (e.g., a table of statistics or a set of event records) as input, aiming at automatically producing texts that informatively, correctly and fluently describe the data (Kukich, 1983; Reiter and Dale, 1997; Angeli et al., 2010; Konstas and Lapata, 2012; Perez-Beltrachini and Gardent, 2017). Traditionally, a data-to-text generation system should pay attention to the problem of content selection (i.e., *what to say*) and surface realization (i.e., *how to say*) (Reiter and Dale, 1997; Gatt and Krahmer, 2018). Modern neural generation systems avoid the distinction of these aspects by building over a standard encoder-decoder architecture (Sutskever et al., 2014) with the attention mechanism over input content (Bahdanau et al., 2015) and train the whole system in an end-to-end fashion. As a result, end-to-end neural text generation has drawn increasing attention from the natural language research community (Mei et al., 2016; Lebret et al., 2016; Wiseman et al., 2017; Kiddon et al., 2016).

[*]Contribution during internship at Microsoft.

Input Data				
Row	Team	Points	Rebound	City
1	Heat	94	44	Miami
2	Hawks	95	40	Atlanta
Generated Description				
Hawks edges the Heat with 95 - 94				

Table 1: An example of generated texts from structured data. In this example, the wining team is not indicated explicitly, but can be inferred from the scores for hte two teams. The words with underlining and wave lines are based on the facts from the input data and the results of inferring, respectively.

However, a critical issue for neural text generation has been largely overlooked. In domains such as sports, finance or medical care, language generation should adhere to facts which are supported by or can be derived from the input data through analysis or inference. For instance, the sentence "Hawks edges the Heat with 95-94" describing the result of a basketball game should always conform to the original data in team names and the scoreline. More importantly, the word "edges" in the description is an inferred fact that the scores between the two competing teams are rather close, while "Hawks" is the winner that scores the slightly higher point total of "95". Since current neural models do not have special treatment for such data analytics, they are likely to generate spurious and incorrect statements. This problem has already been pointed out in recent studies (Wiseman et al., 2017). Related studies on neural program induction have shown that current neural models have difficulties in learning arithmetic operations such as addition and comparisons (Joulin and Mikolov, 2015; Neelakantan et al., 2016).

A straightforward way to improve the fidelity of neural text generation is to separate symbolic operations out from the neural models. More specifically, it is viable to pre-execute a few symbolic operations before generation, and then use the results of execution to guide the whole generation process. However, there are two major challenges for incorporating pre-defined operations: (1) if we apply operations exhaustively on all fields with compatible value types in the table, it would create a huge search space in which mention worthy results are rare events and (2) it is difficult to establish the correspondences between specific spans of numeric results and lexical choices. For example, the word "edges" corresponds to the slight difference in score, i.e. 1, in Table. 1.

Inspired by recent work that separates neural representations and symbolic operations (Liang et al., 2017), we propose a framework for neural data-to-text generation that is able to utilize information from pre-computed operations on raw data. Based on a standard sequence-to-sequence model with an attention and copying mechanism, we design a gating mechanism for the neural model to decide which part of the execution results should be used for generation. To address the second challenge, we also design a quantization layer to map numerical execution results into bins to guide different lexical choices according to different quantities of values.

To examine the effectiveness of our proposed model, we collect a large dataset of sports headline generation for NBA basketball games[1]. We also evaluate the models on the ROTOWIRE dataset released by Wiseman et al. (2017) which targets at generating short paragraphs. Experiments show that our model outperforms current state-of-the-art neural methods in terms of both fluency and fidelity. In summary, we make the following contributions in this paper:

- We propose a neural data-to-text framework that generate texts by additional processing over input data. Based on a basic sequence-to-sequence model with attention and copying, we design a gating mechanism to enable the model to decide which part of the executed results should be utilized. We also propose a novel quantization layer to map specific numerical values onto different spans to affect lexical choices under different conditions.
- To focus our study on correct text generation, we collect a challenging dataset for NBA headline generation.
- We conduct experiments on the NBA headline dataset as well as the ROTOWIRE dataset from previous work. Results show improvements on both correctness and fluency from our proposed framework over baseline systems.

[1]Available at https://github.com/janenie/espn-nba-data

2 Background: Attention-Based Neural Sequence-to-Sequence Model

In this section, we briefly introduce the architecture of the attention-based sequence-to-sequence (Seq2Seq) (Cho et al., 2014b; Bahdanau et al., 2015) model with a copy mechanism (See et al., 2017), which is the basis of our proposed model.

2.1 RNN Encoder-Decoder

The goal of data-to-text generation is to generate a natural language description for a given set of data records $S = \{r_j\}_{j=1}^K$. Usually, a Seq2Seq model consists of an encoder and a decoder with recurrent neural networks (RNN). First, each input record r_j is encoded into a hidden vector $\mathbf{h}_j$ with $j \in \{1, ..., K\}$ using a bidirectional RNN. The decoder generates the description word by word using another RNN.

In the training phase, given a record set and its corresponding natural language description (S, y), the Seq2Seq model maximizes the conditional probability as follows:

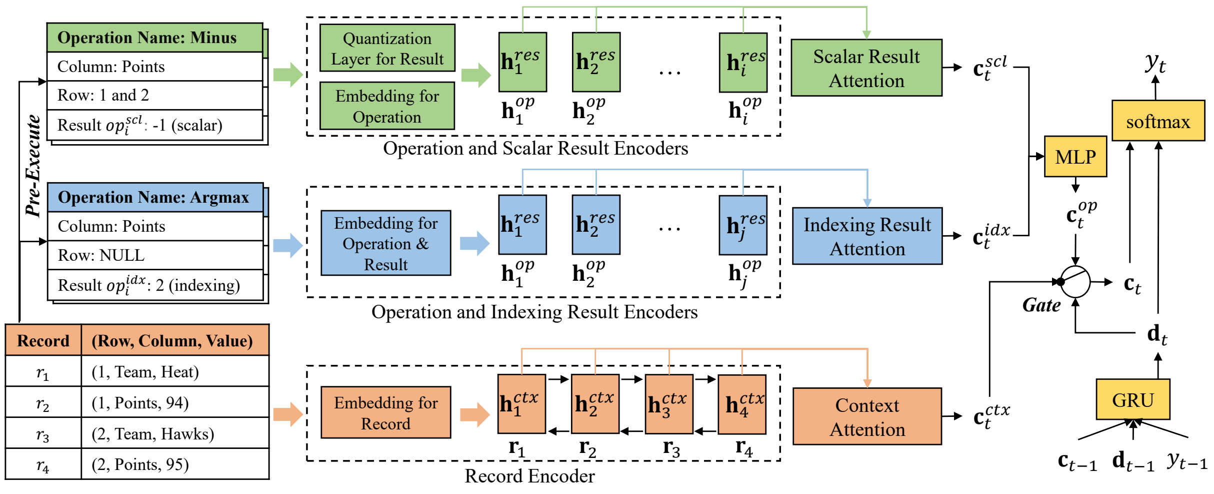
$$P(y|S) = \prod_{t=1}^{T} P(y_t|y_{<t}, S) \tag{1}$$

where y_t is the t-th word in the description and T is the length of the description. The conditional probability $P(y_t|y_{<t}, S)$ is computed as:

$$P(y_t|y_{<t}, S) = \text{softmax}(f(\mathbf{d}_t, y_{t-1}, \mathbf{c}_t)) \tag{2}$$

where $f(\cdot)$ is a non-linear function and $\mathbf{d}_t$ is the hidden state of the decoder at step t:

$$\mathbf{d}_t = g(\mathbf{d}_{t-1}, y_{t-1}, \mathbf{c}_{t-1}) \tag{3}$$



Operation Name: Minus
Column: Points
Row: 1 and 2
Result op_i^{scl}: -1 (scalar)
Pre-Execute
Operation Name: Argmax
Column: Points
Row: NULL
Result op_i^{idx}: 2 (indexing)
Record
(Row, Column, Value)
r_1 (1, Team, Heat)
r_2 (1, Points, 94)
r_3 (2, Team, Hawks)
r_4 (2, Points, 95)
Quantization Layer for Result
Embedding for Operation
$\mathbf{h}_1^{res}$ $\mathbf{h}_2^{res}$ … $\mathbf{h}_i^{res}$
$\mathbf{h}_1^{op}$ $\mathbf{h}_2^{op}$ $\mathbf{h}_i^{op}$
Operation and Scalar Result Encoders
Scalar Result Attention
$\mathbf{c}_t^{scl}$
Embedding for Operation & Result
$\mathbf{h}_1^{res}$ $\mathbf{h}_2^{res}$ … $\mathbf{h}_j^{res}$
$\mathbf{h}_1^{op}$ $\mathbf{h}_2^{op}$ $\mathbf{h}_j^{op}$
Operation and Indexing Result Encoders
Indexing Result Attention
$\mathbf{c}_t^{idx}$
Embedding for Record
$\mathbf{h}_1^{ctx}$ $\mathbf{h}_2^{ctx}$ $\mathbf{h}_3^{ctx}$ $\mathbf{h}_4^{ctx}$
$\mathbf{r}_1$ $\mathbf{r}_2$ $\mathbf{r}_3$ $\mathbf{r}_4$
Record Encoder
Context Attention
$\mathbf{c}_t^{ctx}$
MLP
$\mathbf{c}_t^{op}$
Gate
$\mathbf{c}_t$
$\mathbf{d}_t$
softmax
y_t
GRU
$\mathbf{c}_{t-1}$ $\mathbf{d}_{t-1}$ y_{t-1}

Figure 1: A diagram of the operation guided neural data-to-text generation. The input record table is converted from the first 3 columns of Table1. First, a set of operations are applied to the input records. Then, the records, operations and pre-excuted operation results are encoded. Finally, an attention-equipped GRU decoder with a gating mechanism decides which part of the execution results and context should be used for generation.

where $g(\cdot)$ is a non-linear function. We adopt the Gated Recurrent Unit (GRU) (Cho et al., 2014a) as the recurrent unit for the encoder and decoder. $\mathbf{c}_t$ in Eq. 2 is the context vector at timestep t, computed as a weighted hidden vectors $\mathbf{h}_j$:

$$\mathbf{c}_t = \sum_{j=1}^{K} \alpha_{t,j} \mathbf{h}_j \tag{4}$$

where $\alpha_{t,j}$ is computed by an attention scheme, typically implemented as a softmax distribution over scores calculated with a multi-layer perceptron (Bahdanau et al., 2015).

2.2 Copy Mechanism

Recent work augments Seq2Seq models to copy words directly from the source information on which they are conditioned (Gu et al., 2016; See et al., 2017). These models usually introduce an additional binary variable z_t into per-timestep target word distribution, which indicates whether the target word y_t is copied from the source or is generated from the recurrent hidden states. We use the pointer-generator network (See et al., 2017) for the copy mechanism. Specifically, the binary variable z_t is calculated from the context vector $\mathbf{c}_t$, the decoder state $\mathbf{d}_t$ and the decoder input y_{t-1}:

$$p_{gen} = \sigma(\mathbf{w}_c^\top \mathbf{c}_t + \mathbf{w}_d^\top \mathbf{d}_t + \mathbf{w}_y^\top y_{t-1} + b_{ptr}) \tag{5}$$

where vectors $\mathbf{w}_c$, $\mathbf{w}_d$, $\mathbf{w}_y$ and the scalar b_{ptr} are learnable parameters, and σ is the sigmoid function. The joint probability for generating y_t is formulated as follows:

$$P_{copy}(y_t|y_{<t}, S) = p_{gen} P(y_t|y_{<t}, S) + (1 - p_{gen}) \sum_{i:r_i = y_t} \alpha_{t,i} \tag{6}$$

3 The Proposed Model

In this paper, we propose to utilize information from pre-executed operations on the input data to guide the generation. As shown in Fig. 1, our model consists of a record encoder, an operation encoder and an operation result encoder, and an attention-equipped GRU decoder with a gating mechanism. First, a set of operations are applied to all valid records in the input data, yielding their corresponding pre-executed results. The pre-executed results act as facts inferred from input data to guide the generation. Then, the records, operation and pre-executed operation results are encoded into corresponding representation. Finally, we design a gating mechanism for the GRU decoder to decide which part of the inferred facts should be used for generation. Moreover, to address the challenge in establishing correspondences between specific numeric results and lexical choices, a quantization layer maps the results into several segmentations to guide the lexical choices.

3.1 Notation

Given the input data and description pair (S, y), where each target description $y = y_1, ..., y_T$ consists of T words, and each input data is stored in a table (e.g., Table 1), where each row is an entity and each column is a field of this entity. The input data can be transferred into K records $S = \{r_i\}_{i=1}^{K}$, where each record r is a triple $(r.idx, r.f, r.v)$. For r_4 in the table of Fig. 1, $r.idx$ $r.f$ and $r.v$ refer to the row index (e.g., row 2), the field name (e.g., column Points) and value (e.g., cell value 95), respectively. We also define a set of operations $\{op_i\}$, and the operations are applied to the input records S to produce corresponding results at the preprocessing stage. The results of operations can be categorized into two types: op_i^{scl} denotes results with a type of scalar value and op_i^{idx} denotes results with a type of indexing value.

3.2 Encoding Records

We map each record $r \in S$ into a vector $\mathbf{r}$ by concatenating the embedding of $r.idx$ (e.g., row 2), $r.f$ (e.g., column Points) and $r.v$ (e.g., cell value 95), denoted as $\mathbf{r} = [\mathbf{e}^{idx}, \mathbf{e}^{f}, \mathbf{e}^{v}]^{\top}$, where $\mathbf{e}^{idx}$, $\mathbf{e}^{f}$, $\mathbf{e}^{v}$ are trainable word embeddings of $r.idx$, $r.f$ and $r.v$ respectively, similar to (Yang et al., 2017). We feed a set of record vectors $\mathbf{r}_1, ..., \mathbf{r}_K$ to a bidirectional GRU and yield the final record representations $\mathbf{h}_1^{ctx}, ..., \mathbf{h}_K^{ctx}$ as introduced in Section 2. We leave the exploring of different encoding methods as future work, as it would affect the performance.

3.3 Encoding Operations

As shown in Fig. 1, each operation op_i consists of: a) the name of the operation $op_i.t$ (e.g., minus); b) the column $op_i.c$ to which the operation applies (e.g., Points); and c) the row to which the operation applies, denoted as $op_i.arg = \{r_i.idx\}_{i=1}^{A}$, where A is the count of arguments. We then encode each operation op_i by concatenating the representation of these three components and feed them into a nonlinear layer to represent each operation as follows:

$$\mathbf{h}_i^{op} = \tanh(\mathbf{W}_{op}[\mathbf{op}_i^{t}, \mathbf{op}_i^{c}, \mathbf{op}_i^{arg}]^{\top}{}_i + b_{op}), \quad (7)$$

where $\mathbf{op}_i^{t}$ is the embedding of $op_i.t$; $\mathbf{op}_i^{c}$ is the embedding of column $op_i.c$ which shares the same parameters of embedding with record column $r.f$. For $op_i.arg$, it may contain multiple arguments, so we apply a nonlinear layer to get a fixed length representation as follows:

$$\mathbf{op}_i^{arg} = \tanh(\sum_{k \in arg_i} \mathbf{W}_k^{arg} \mathbf{e}_k^{idx} + b_{arg}), \quad (8)$$

where $\mathbf{e}_k^{idx}$ is the same embedding as used to encode the row index $r.idx$, and $\mathbf{W}_k^{arg}$ and b_{arg} are learnable parameters. For operations which are applied in the entire column (e.g., argmax) without specific rows, the representation of arguments is a special vector which stands for ALL.

3.4 Encoding Operation Results

The operations produce two types of results, one is scalar results (e.g., the minus operation returns -1), the other is indexing results (e.g., the argmax operation returns the row number 2), and two encoders are designed to encode these results respectively.

Scalar Results Representation In Table. 1, the word "edges" is generated based on the fact that the points gap of the two teams is -1. In fact, other value likes -2 or -3 is close to -1, and the word "edges" is also applicable to them. However, directly establishing the lexical choices on various sparse numeric values is not easy (Reiter et al., 2005; Smiley et al., 2016; Zarrieß and Schlangen, 2016). Reiter et al. (2005) use consistent data-to-word rules for time-series weather forecast summary generation. In this paper, we aim to capture the data-to-word mapping automatically by a simple quantization unit. A *quantization layer* is designed to map the scalar values into several bins, namely quantization units. Specifically, we feed each scalar value op_i^{scl} to a softmax layer, and its representation $\mathbf{h}_i^{res}$ is computed as the weighted sum of all quantization embeddings:

$$\mathbf{q}_i = \mathbf{W}_q op_i^{scl} + b_q, \quad (9)$$

$$\mu_{i,l} = \frac{\exp(q_{i,l})}{\sum_{j=1}^{L} \exp(q_{i,j})}, \quad (10)$$

$$\mathbf{h}_i^{res} = \sum_{l=1}^{L} \mu_{i,l}\, \mathbf{e}_l^{scl} \quad (11)$$

where $\mathbf{W}_q$ and b_q are trainable parameters, $\mathbf{e}^{scl}$ is the quantization embedding and L is the size of quantization units. Note that L is much smaller than the unique number of scalar results. We set L to 5 in this paper.

Indexing Results Representation Some operations produce the row number of records (denoted as idx_i) as a result. For instance, the argmax operation in Fig. 1 returns row 2. We then look up the row embedding of the selected record defined in Section 3.2 to represent the result. Defined as $\mathbf{h}_i^{res} = \mathbf{e}_i^{idx}$.

3.5 Decoder

Comparing with the Seq2Seq model described in Section 2 and our model, the main difference is in the context vector $\mathbf{c}_t$. Different from Eq. 4, our model has both records and operations as input. We design two attention layers to summarize information from both parts respectively, the overall context vector $\mathbf{c}_t$ is balanced by a dynamic gate λ_t.

$$\mathbf{c}_t = (1-\lambda_t)\mathbf{c}_t^{op} + \lambda_t \mathbf{c}_t^{ctx}, \tag{12}$$

$$\lambda_t = \sigma(\mathbf{W}_g \mathbf{d}_t + b_g), \tag{13}$$

where $\mathbf{c}_t^{op}$ and $\mathbf{c}_t^{ctx}$ are the context vector of operation results and records, respectively.

As there are two types of operation results which have quite different meanings, their context vectors are calculated separately and then put together by a nonlinear layer. The context vectors $\mathbf{c}_t^{scl}$ of operation results with scalar value at timestep t are constructed as (Luong et al., 2015):

$$\mathbf{c}_t^{scl} = \sum_{j=1}^{N} \alpha_{t,j}^{scl} * \mathbf{h}_j^{res} \tag{14}$$

$$\beta_{t,j}^{scl} = \text{MLP}(\mathbf{d}_{t-1}, \mathbf{h}_j^{op}), \tag{15}$$

$$\alpha_{t,j}^{scl} = \frac{\exp(\beta_{t,j}^{scl})}{\sum_k \exp(\beta_{t,k}^{scl})} \tag{16}$$

where MLP stands for standard 1-layer perceptron (with $\tanh$ nonlinearity), and $\alpha_{t,j}^{scl}$ refers to the importance of j-th operations at the current timestep t. Eq. 14 is based on the attention mechanism which can be treated as mapping a query and a set of key-value pairs to an output. The output $\mathbf{c}_t^{scl}$ is computed as a weighted sum of the values $\mathbf{h}_j^{res}$, where the weight assigned to each value is computed by a compatibility function of the query $\mathbf{d}_{t-1}$ with the corresponding key $\mathbf{h}_j^{op}$. In this way, we also construct $\mathbf{c}_t^{idx}$. Then the context vector of operation results at time step t is computed by putting these two context vectors together:

$$\mathbf{c}_t^{op} = \text{MLP}([\mathbf{c}_t^{scl}, \mathbf{c}_t^{idx}]^\top) \tag{17}$$

The context vector representation $\mathbf{c}_t^{ctx}$ for records is constructed by replacing $\mathbf{h}_j^{res}$ with $\mathbf{h}_j^{ctx}$ in Eq. 14 and replacing $\mathbf{h}_j^{op}$ with $\mathbf{h}_j^{ctx}$ in Eq. 15.

After obtaining $\mathbf{c}_t$, the word distribution for generation can be calculated by substituting the $\mathbf{c}_t$ in Eq. 2. For the copy probability defined in Eq. 6, to copy words based on the information of both operations and records at current time step t, we need to update the attention weights for Eq. 6 based on the newly computed context vector $\mathbf{c}_t$ and decoding state $\mathbf{d}_t$:

$$\beta_{t,j}^{new} = \text{MLP}(\mathbf{h}_j^{ctx}, [\mathbf{d}_{t-1}, \mathbf{c}_t]^\top) \tag{18}$$

$$\alpha_{t,j}^{new} = \frac{\exp(\beta_{t,j}^{new})}{\sum_k \exp(\beta_{t,k}^{new})} \tag{19}$$

	ESPN	ROTOWIRE	WIKIBIO
Vocab	3.3K	11.3K	400K
Tokens	114.3K	1.6M	19M
Examples	15.1K	4.9K	728K
Avg Len	9.5	337.1	26.1
Input facts	62.7%	61.2%	72.1%
Inferred facts	29.1%	11.7%	7.4%
Unsupported	8.2%	27.1%	20.5%

Table 2: Dataset statistics. For each dataset, we also manually label the source for the facts mentioned in 20 descriptions, and report the percentage of facts based on the input data, inferred facts and unsupported facts.

3.6 Training

As the results of operations are pre-computed in an offline stage, our proposed model is fully differentiable and can be optimized in an end-to-end manner using back propagation. Given the batches of records $\{S\}_N$ and the standard natural language descriptions $\{Y\}_N$, the objective function is to minimize the negative log-likelihood:

$$L = -\frac{1}{N}\sum_{k=1}^{N}\sum_{t=1}^{T_k} \log p(y_t^k | y_{<t}^k, S^k) \tag{20}$$

where the superscript k indicates the index of the records-description pair, and T_k is the length of the k-th description.

4 Experiments

4.1 Datasets

Several benchmark datasets have been used in recent years for data-to-text generation (Liang et al., 2009; Chen and Mooney, 2008; Lebret et al., 2016). For instance, Lebret et al. (2016) have built a biography generation dataset from Wikipedia. However, a recent study by Perez-Beltrachini and Gardent (2017) shows that existing datasets have a few missing properties such as lacking syntactic and semantic diversity. To check whether the facts mentioned in the descriptions are based on input data, we identify the text spans which contain facts (e.g., in table 1, “Hawks” is a span contain fact) from the descriptions and divide each span into three categories: a) input facts (facts that can be directly found from the input), b) inferred facts (facts that can not be directly found from the input but can be derived), c) unsupported facts (facts that can not be found or derived from input data). Table 2 shows that WikiBio dataset requires inference on only 5.4% of its data. To better demonstrate the effectiveness of our approach, we adopt the following datasets which require substantially more inference based on the input data:

ROTOWIRE We use the dataset and its standard splits released by Wiseman et al. (2017), which consists of 4,853 human written NBA basketball game summaries aligned with their corresponding game statistics. Table 2 shows that 11.7% of facts in the game summaries can be inferred based on the input data. However, this dataset focuses on generating long text and 27.1% of facts are unsupported[2], which brings difficulties to the analysis of fidelity for the generated text.

ESPN We collect 15,054 NBA game result headlines during 2006-2017 from the ESPN website, paired with their corresponding game statistics. These headlines are professional and concise, e.g., the description in Fig. 1. The percentage of inferred facts is 29.1% while unsupportive facts is only 8%, so we can focus on generation for the inferred facts. We split the dataset into 12,043 (80%) for training, 1,505 (10%) for development and 1,506 (10%) for testing respectively.

4.2 Instantiation

In the following experiments, we define two operations, the minus operation which returns the scalar result and the argmax operation which returns a id of a row. These operations are applied to all columns and rows whose record values are numeric numbers. The number of pre-executed results increases with the number of operations, arguments and the size of input data, which will impact the efficiency of our model. The unnecessary operation arguments can be pruned, e.g., only apply operations to the arguments co-mentioned in descriptions on the training set. We will leave this part of research for our future work.

4.3 Experiment Setup

In the main experiments, we compare our model with the following methods: (a) Template: a problem-specific template-based generator which fills structured data into corresponding placeholders to generate texts[3], (b) Seq2Seq+copy: Seq2Seq model with pointer network copy mechanism introduced in Section 2. It is one of the state-of-the-art methods, (c) Seq2Seq+op: Seq2Seq+copy plus the results of operations, where results are directly treated as extra records and fed to the record encoder introduced in Section 3.2 with the original input together, (d) Seq2Seq+op+quanti: We apply the quantization layer Eq. 9-11 to the results of minus operation on the basis of Seq2Seq+op. For completeness, we also report the results of Wiseman et al. (2017) on the ROTOWIRE dataset. The difference between this baseline and Seq2Seq+copy is that the former uses an LSTM rather than GRU for decoding and an additional copying loss. All the experiments use a beam size of 5 in decoding[4].

For model training, we use the stochastic gradient descent algorithm and the AdaDelta optimizer (Zeiler, 2012). The dimension of trainable word embeddings are set to 256 except for the dimension of input record row embedding, which is set to 32; and the dimension of hidden units in GRUs are all set to 512. All the parameters are initialized using a normal distribution with zero mean and a variance of $\sqrt{6/(d_{in}+d_{out})}$, where d_{in} is the dimension of the input layer and d_{out} is the dimension of the output layer (Glorot and Bengio, 2010). Training converges after 40 epochs.

4.4 Main Results

We adopt both automatic evaluation and human evaluation to evaluate the proposed model.
Automatic Evaluation We employ BLEU-4 as the metric for automatic evaluation. Table 4 gives the automatic evaluation results for generation on two datasets. Our proposed model OpAtt outperforms neural network baselines (See et al., 2017; Wiseman et al., 2017). The results show that our method which incorporates the operations enables generating texts that are fidelity to facts and therefore yields the best performance. Seq2Seq+op+quant outper-

[2]e.g., injuries, rankings in the league, team schedule, etc.

[3]For the ROTOWIRE dataset, we adopt Wiseman et al. (2017)'s templates. For the ESPN dataset, we use Dou et al. (2018)'s system to extract templates. The template is constructed by emitting teams and players information in a sentence: `<team1>` beats `<team2>` with `<point1>`-`<point2>`.

[4]The authors have updated the dataset to fix some mistakes recently, so we cannot use the result which is reported in their paper and rerun this baseline with the authors' code.

	ESPN			ROTOWIRE		
	#Cont./#Supp. (input facts)	#Cont./#Supp. (inferred facts)	#Cont. (unsupported)	#Cont./#Supp. (input facts)	#Cont./#Supp. (inferred facts)	#Cont. (unsupported)
Ref	0.00 / 4.90	0.00 / 1.12	0.51	0.00 / 12.87	0.00 / 3.07	3.20
Seq2Seq+copy	0.44 / 4.61	0.16 / 1.25	0.25	3.75 / 14.44	0.89 / 2.20	2.82
Seq2Seq+op	0.24 / 3.97	0.07 / 1.08	0.76	5.55 / 18.13	0.42 / 2.53	1.93
Seq2Seq+op+quant	0.21 / 4.88	0.03 / 1.10	0.32	3.47 / 16.02	0.53 / 2.02	2.13
OpAtt	0.04 / 5.00	0.02 / 1.27	0.19	2.24 / 16.56	0.18 / 2.84	2.07

Table 3: Average annotators judgment for the count of facts contradicting (#Cont.) and supporting (#Supp.) on facts based on input data, inferred facts and unsupported facts respectively.

	ESPN		ROTOWIRE	
	Dev	Test	Dev	Test
Template	13.75	14.27	8.97	8.93
Wiseman's	-	-	13.57	13.62
Seq2Seq+copy	15.63	15.30	13.72	13.47
Seq2Seq+op	14.07	13.74	13.52	13.44
Seq2Seq+op+quant	15.68	15.49	14.05	13.88
OpAtt	**17.19***	**18.00***	**14.96***	**14.74***

Table 4: BLEU scores (%) over two datasets. Statistical significant is indicated with *($p < 0.05$) with respect to Seq2Seq+copy.

forms the baseline method Seq2Seq+copy, but is not as good as our method. The result confirms that our proposed method with specialized operation encoder and gating mechanism utilizes the information of operations more effectively. Moreover, Seq2Seq+op+quant outperforms Seq2Seq+op showing the effectiveness of the quantization layer.

Human Evaluation Because of the approximate nature of the automated metric BLEU, we also conduct human evaluation to examine the fidelity of the generated texts. We randomly select some games from testing set, and entrust a professional crowdsourcing company to annotate the generated texts[5]. Specifically, three native English workers who are familiar with NBA games are hired. They are first required to identify the text spans which contain facts from the generated texts, then categorize the text spans into one of three facts listed in Table 2, and finally judge whether the span is supported or contradicted by the input data.

Table 3 shows the annotation results. Our method talks more about the inferred facts in the generated texts while includes less contradictions. In addition, all methods produce some unsupported facts which affect the fidelity of the generated texts. We leave this issue for future work.

[5]The Fleiss' kappa score of the annotation is 0.782 for ESPN and 0.761 for ROTOWIRE respectively. For the ESPN dataset, we select 50 games and each with one generated sentence. For ROTOWIRE, by following (Wiseman et al., 2017), we select 15 games and each with 3 randomly selected sentences.

	Dev	Test
Seq2Seq + copy	15.63	15.30
OpAtt	**17.19**	**18.00**
OpAtt w/o argmax op	15.71	15.97
OpAtt w/o quantization	16.35	16.70
OpAtt w/o gate	16.35	16.15

Table 5: BLEU scores (%) of model ablation.

Reference	horford 's dunk helps hawks edge nets , 114 - 111
Seq2Seq +copy	nets **rally** from 17 down to **top** nets 111 - 111
OpAtt w/o argmax op	hawks **rally** from 17 down to beat nets 114 - 111
OpAtt	horford scores 24 as hawks beat nets 114 - 111

Table 6: The generated texts by introducing different operations. The words with underline, wavy line and dot line are input facts, inferred facts and unsupported fact, respectively. And the **bold** words are contradicted facts.

4.5 Analysis

As discussed in Section 4.1, the ESPN dataset is rich in inferred facts. Therefore, the model analysis is based on this dataset, and all case studies are made on the development set.

4.5.1 Effect of Operations

We examine the necessity and the benefit of introducing operations by removing the argmax operation (see "OpAtt w/o argmax op" in Table 5). Comparing to Seq2Seq+copy, the results show that our full model and "OpAtt w/o argmax op" which incorporates results of operations both work well in terms of BLEU, and the improvements increase with the number of operations.

To better illustrate that our model can generate factually correct text, we show the texts generated by different models in Table 6. The game results mentioned in the text generated by the Seq2Seq+copy model are wrong, which shows the inability for existing neural models on inferring facts from the structured data. After adding the minus operation, "OpAtt w/o argmax op" is able to infer the game result by applying the minus operation on the points of the two competing teams, therefore its generated text conforms to the game results. The results confirm the necessity of introducing operations to ensure factually correct generation. Furthermore, our full model generates text with the correct point leader and game result based on the results of operation argmax and operation minus respectively.

4.5.2 Effect of Quantization

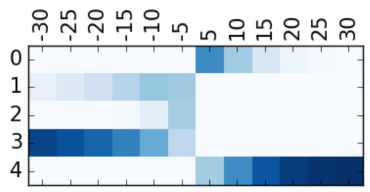


Figure 2: Weights of the quantization softmax layer when mapping the points gap of two competing teams to five bins. X-axis is points gap and Y-axis is quantization bin.

The quantization layer maps the numerical execution results into several bins to enable different lexical choices according to different quantities of values. Compared to our full model, "OpAtt w/o quantization" in Table 5 which removes the quantization layer decreases the BLEU performance, which shows the effectiveness of the quantization layer in the lexical choices during generation.

In Fig. 2, we visualize the weights of quantization softmax layer $\mu_{i,l}$ produced by Eq. 10 when mapping the points gap of two competing teams to five bins. We can see that the points gaps with close numerical values are mapped to the same bin, so the decoder can choose similar words for them in generation. When the absolute value of the points gap is small, the weights distribution over the points gap is dispersive. At this time, the decoder tends to generate general words. This distribution becomes more centralized with the increase of the absolute value of the points gap, to generate more unique words. Moreover, we show the distribution of words that describes the winning relationship of games over different intervals of game points gap. As shown in Table 7, we can clearly see that apart from three most common

Points Gap	Words describes winning relationship
[0, 5)	beat, past, win over, edge, hold off, survive
[5, 10)	beat, past, win over, out last, hold off
[10, 20)	beat, past, win over, blow out, top, pull away, rout
>= 20	beat, past, win over, power, rout, easy win over, roll past

Table 7: The words that describing the winning relationship of games over different intervals of game points gap.

word "beat", "past", "win over", our proposed quantization layer can choose specific words according to the points gap. The word "edge" and "hold off" will only be chosen when the points gap is small, while the word "rout" and "blow out" will appear when the points gap is larger than 10.

4.5.3 Effect of Gating Mechanism

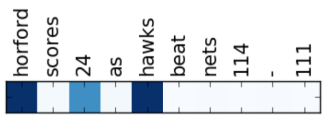


Figure 3: The gating weights at different time steps.

We design a gating mechanism to decide when to incorporate the results of operations to guide the process of generation. From Table 5, "OpAtt w/o gate" stands for the method which replaces the balancing weight λ in Eq. 12 to 0.5, which is a special case of our proposed gating mechanism. The performance of this ablation is worse than our full model, which demonstrates that the gating mechanism is an essential component. Fig. 3 shows an example of the gating weights at each time step in generation, where a darker cell means the incorporation of more information from operation results for decoding corresponding word. We can see that the gate weights are reasonable, as the gate values are large when deciding the team leader "horford" and the winner of the game "hawks".

5 Related Work

Data-to-text generation is a task of natural language generation (NLG) (Gatt and Krahmer, 2018). Previous research has focused on individual content selection (Kukich, 1983; Reiter and Dale, 1997; Duboué and McKeown, 2003; Barzilay and Lapata, 2005) and surface realization (Goldberg et al., 1994; Soricut and Marcu, 2006; Wong and Mooney, 2007).

Recent work avoids the distinction of the content selection and sentence realization. Chen and Mooney (2008) use an SMT based approach to learn alignments between comments and their corresponding event records. Angeli et al. (2010) transform the problem into a sequence of local decisions using a log-linear model. Konstas and Lapata (2012) employ a PCFG to simultaneously optimize the content selection and surface realization problem.

In the field of neural text generation, Mei et al. (2016) uses a neural encoder-decoder approach for end-to-end training. Some have focused on conditional language generation based on tables (Yang et al., 2017), short biographies generation from Wikipedia tables (Lebret et al., 2016; Chisholm et al., 2017) and comments generation based on stock prices (Murakami et al., 2017). However, none of these methods consider incorporating the facts that can be inferred from the input data to guide the process of generation. Murakami et al. (2017) post-process the price by extending the copy mechanism and replacing numerical values with defined arithmetic operations after generation. While our model, OpAtt utilizes information from pre-computed operations on raw data to guide the generation.

Our work is related to research areas on deep learning models for program induction and question answering from a knowledge base (Neelakantan et al., 2016; Liang et al., 2017; Ling et al., 2017). Neelakantan et al. (2016) solve the problem of semantic parsing from structured data and generate programs using pre-defined arithmetic operations. Liang et al. (2017) design a set of executable operators and obtain the answers by the generated logic forms. Ling et al. (2017) design a set of operators to generate the latent program for math problem solving. However, data-to-text is a different task. The operations for these methods are designed to find the answers, while we use the operations to guide the process of generation.

6 Conclusion and Future Work

In this work, we address the problem of generating consistent text from structured data in a neural data-to-text generation framework, where we extract facts that can be inferred in the given data by applying several executable symbolic operations to guide the generation. Moreover, we design a special quantization layer to operations whose result type is numeric value and establish the correspondence between the numeric values and lexical choice in generation. Experiments show that our method, OpAtt, outperforms existing state-of-the-art neural methods, in both fluency and fidelity evaluations.

As applying operations on a large number of records greatly increases the search space for the attention mechanism, we will extend our model to automatically detect the relevant operations to reduce computing complexity. We will also extend the set of operations to accommodate historical data, graph data and detect the unsupported facts in the generation within the single framework.

7 Acknowledgement

We thank the anonymous reviewers for their helpful comments. We also thank Zhirui Zhang, Shuangzhi Wu and Yu Wu for helpful conversations and comments on the work. The contact author of this paper, according to the meaning given to this role by Sun Yat-Sen University, is Rong Pan.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 502–512.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 141–148.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 128–135.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR*, abs/1702.06235.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Longxu Dou, Guanghui Qin, Jinpeng Wang, Jin-Ge Yao, and Chin-Yew Lin. 2018. Data2text studio: Automated text generation from structured data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, Brussels, Belgium, October 11- November 4, 2018*.

Pablo Ariel Duboué and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2003, Sapporo, Japan, July 11-12, 2003*.

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

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

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, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 190–198.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 329–339.

Ioannis Konstas and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 369–378.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, June 15-17, 1983.*, 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, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 91–99.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

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 *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 720–730.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1374–1384.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. *ICLR*.

Laura Perez-Beltrachini and Claire Gardent. 2017. Analysing data-to-text generation benchmarks. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 238–242.

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

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artif. Intell.*, 167(1-2):137–169.

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, Vancouver, Canada. Association for Computational Linguistics.

Charese Smiley, Vassilis Plachouras, Frank Schilder, Hiroko Bretz, Jochen Leidner, and Dezhao Song. 2016. When to plummet and when to soar: Corpus based verb selection for natural language generation. In *Proceedings of the 9th International Natural Language Generation conference*.

Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using widl-expressions and its application in machine translation and summarization. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

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, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263.

Yuk Wah Wong and Raymond J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, April 22-27, 2007, Rochester, New York, USA*, pages 172–179.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1850–1859.

Sina Zarrieß and David Schlangen. 2016. Towards generating colour terms for referents in photographs: Prefer the expected or the unexpected? In *Proceedings of the 9th International Natural Language Generation conference*.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.