Refining Data for Text Generation

Qianying Liu[12], **Tianyi Li**[1], **Wenyu Guan**[1] **and Sujian Li**[1]
[1] Key Laboratory of Computational Linguistics, MOE, Peking University
[2] Graduate School of Informatics, Kyoto University
ying@nlp.ist.i.kyoto-u.ac.jp; litianyi01@pku.edu.cn ;
guanwy@pku.edu.cn; lisujian@pku.edu.cn

Abstract

Recent work on data-to-text generation has made progress under the neural encoder-decoder architectures. However, the data input size is often enormous, while not all data records are important for text generation and inappropriate input may bring noise into the final output. To solve this problem, we propose a two-step approach which first selects and orders the important data records and then generates text from the noise-reduced data. Here we propose a learning to rank model to rank the importance of each record which is supervised by a relation extractor. With the noise-reduced data as input, we implement a text generator which sequentially models the input data records and emits a summary. Experiments on the ROTOWIRE dataset verifies the effectiveness of our proposed method in both performance and efficiency.

1 Introduction

Recently the task of generating text based on structured data has attracted a lot of interest from the natural language processing community. In its early stage, text generation (TG) is mainly accomplished with manually compiled rules or templates, which are inflexible and mainly based on expert knowledge (Kukich, 1983; Holmes-Higgin, 1994; Reiter and Dale, 1997). With the development of neural network techniques, especially sequence-to-sequence (seq2seq) models, generating short descriptive texts from structured data has achieved great successes, including generating wikipedia-style biographies (Lebret et al., 2016; Sha et al., 2017) and restaurant introductions (Novikova et al., 2017).

However, the task of generating long text, such as generating sports news from data, still fails to achieve satisfactory results. The existing models often forge fake context, lose sight of key facts and display inter-sentence incoherence (Wiseman et al., 2017). For the sports news generation task, one challenging problem is that the input records are both large and noisy. Specifically, the inputted box scores, which contains hundreds of data records, belong to 40 different categories, such as fouls, three-pointer, starting position and so on. Meanwhile, not all of the inputted records are reflected in the sports news, and there exists a serious non-parallelism between data records and texts. According to our statistics for 3000 parallel sports news and its data records which is shown in Table 1 and Figure 1, an average of only 19.3 data records out of 670.6 are mentioned in the summaries on average, namely only less than 5% of the data records are reflected in the human written news and rest 95% of them may bring noise into the model. Such large and noisy input has also caused the parameter amount of the embedding and encoder layer to be enormous, which leads to massive memory usage and limits the computation speed. In such situation, it is essential to refine data records and choose those important information before generating the final text.

In addition, sport news is far more complex than short descriptive text in that they need to consider overall coherence (Bosselut et al., 2018). For example, it would be weird if there is an abrupt topic change between neighboring sentences. If we just pour all the data records with no order into a model, it would be difficult for the summarization model to learn content planning by itself. Thus, it is a good practice to order the data records before text generation.

As stated above, in this paper, we propose to refine data records for the data-to-text generation task by training a model to select an appropriate subset of data records, which carries the key facts of the

Object	Number
Average Data Records Mentioned	19.30
Average Data Records in Box Data	670.65
Average Summary Length	348.93
Types of Data Records	40

Table 1: Statistics of data records in 3000 sports news.

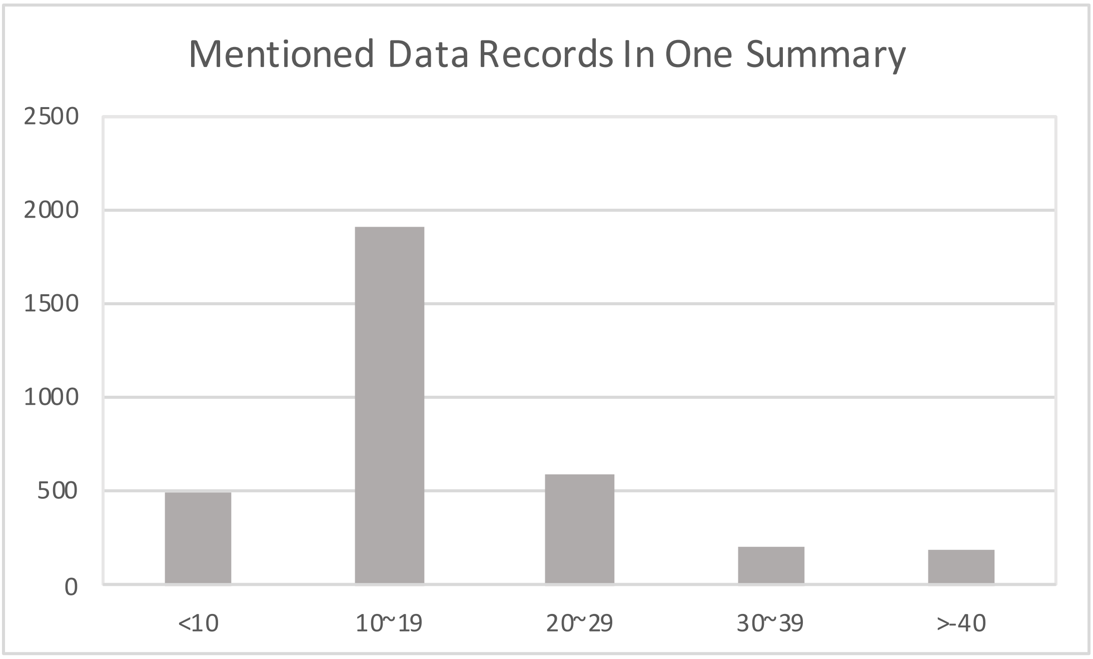

Figure 1: Statistics of data records mentioned in 3000 sports news. The horizontal axis stands for summary numbers and the vertical axis stands for data record numbers.

game, and further to plan an appropriate order for the selected records. This is also similar to the action of human writers who usually plan the important information to include before they write their articles.

Next, one key problem is to label the important records which would be time consuming and expensive. To solve this problem, inspired by Wiseman et al. (2017) which used an information extraction (IE) system for evaluation and Mintz et al. (2009) which used distance learning for relation extraction, we build an IE system based on distant supervision. The IE system extracts relations from gold text, matches them to the corresponding data records and its results can then be used to supervise the process of content selection and planning. Then, we design a ranking unit to learn which data records are selected and in what order they appear. Here we choose to use the learning-to-rank (L2R) method instead of a classifier, because there exists heavy imbalance between positive and negative instances. We also design a rule-based model to further help select the data records. We rank each data record by an overall score based on the two rankers and rule-based system. Finally, we feed the selected and ordered records, which not only the noise and the input size is reduced but also the content is planned, to the generator to obtain the summaries. In this way memory usage could be largely reduced, thus the training process could be accelerated.

We evaluate our method on the ROTOWIRE dataset (Wiseman et al., 2017). The results show how our system improves the model's ability of selecting appropriate context and ordering them. While we achieve comparable BLEU score, the efficiency of the model is greatly improved.

2 Related Work

Data-to-text generation has been an important topic of natural language generation for decades. Early approaches mainly use templates and rules to perform content selection and surface realization (Kukich, 1983; Holmes-Higgin, 1994; Reiter and Dale, 1997). These models have good interpretability and controllability, but the generated content often have problems in terms of diversity and consistency.

Recently, neural network techniques have greatly improved the results of generating short descriptive text from data. The E2E dataset (Lebret et al., 2016) stated the task of generating natural language descriptive text of the restaurants from structured information of the restaurants. The Wikibio dataset (Novikova et al., 2017) gives the infobox of wikipedia as the input data and the first sentence of the corresponding biography as output text. Various approaches have achieved good results on these two datasets which considered content selection and planning. Sha et al. (Sha et al., 2017) proposed a method that models the order of information via link-based attention between different types of data records. Perez-Beltrachini and Lapata (Perez-Beltrachini and Lapata, 2018) introduce a content selection method based on multi-instance learning.

Generating sport news summaries on the other hand,is more challenging because not only the output text is longer and more complex, but also the input data records are numerous and diversed. Wiseman et al. (Wiseman et al., 2017) proposed the ROTOWIRE data set and gave baselines model based on end-to-end neural networks with attention and copy mechanism, these models often overlook key facts, repeatedly output the same information and make up irrelevant content. Puduppully et al. (Puduppully et al., 2018) designed a system that uses gate mechanism and pointer network to select and plan the content. They only used the IE system to guide content planning, while we let the IE system guide both content selecting and planning. Meanwhile our system is lighter and has higher efficiency since we only feed the neural network with a small subset of the large set of data records.

3 Model

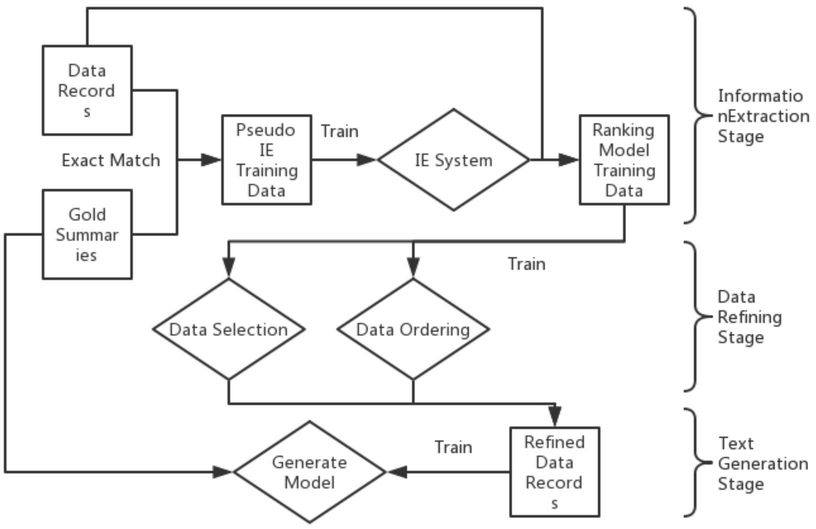

Figure 2: A brief flow graph of our model.

Our model consists of three modules: information extraction, data refining (record selection and planning) and text generation. Figure 2 is a brief flow chart showing the pipeline of our model, which illustrates the data flow and how the models are trained.

3.1 Information Extraction

This module aims to provide supervision for data refining and text generation, and is only used during training. We build a relation extractor similar to Wiseman et al. (2017), who used a relation extractor for automatic evaluation. We do not have human-annotated data for this specific domain, but this relation extractor can be trained by distance learning(Mintz et al., 2009), which uses exact match between candidate entity-value pairs and data records to build pseudo training data. For example, from a sentence *A scored 4 points and B scored 8 points*, which has two entities $\{A, B\}$ and two values $\{4, 8\}$, we can extract 4 candidate entity-value pairs $\{(A, 4), (A, 8), (B, 4), (B, 8)\}$. Then we compare them with the original data records and check whether these candidate pairs match with data records. In this example we can find *(A, 4, PTS)* and *(B, 8, PTS)* in the original data records, so we label the candidate pairs as $\{(A, 4, PTS), (A, 8, norel), (B, 4, norel), (B, 8, PTS)\}$, where *norel* is the label that stands for no relationship and form the pseudo data. To be noticed, there might be multiple data records that match with the candidate pair, so the training data here is multi-labeled. The reason why we use an IE system instead of using the pseudo data straight away is because with the help of context information, the IE system can make better decisions and generalize better than the exact-match method.

To train the IE system, we cast the relation extraction task into a classification problem by modeling whether an entity-value pair in the same sentence has relation or not (Zhang, 2004; dos Santos et al., 2015). We use neural network to train the relation extractor and ensemble various models to further improve the performance. Formally, given an input sentence $\mathbf{x} = \{x_t\}_{t=1}^{n}$ which contains an entity-value candidate pair *(r.E, r.M)*, we first embed each word into a vector e_t^W. The embedding is then concatenated with two position embedding vectors e_t^E and e_t^V, which stands for the distance between the word and the entity and the value. Then the final word embeddings $e_t = concat\left\{e_t^W, e_t^E, e_t^V\right\}$ are fed into a bi-directional long short-term memory network (BiLSTM) or a convolutional neural network (CNN) to model the sequential information.

$$\begin{aligned} h_t &= BiLSTM(e_t, h_{t-1}, h_{t+1}) \\ h_{LSTM} &= h_n \end{aligned} \tag{1}$$

$$h_{CNN} = CNN(concat\,\{e_t\}_{t=1}^{n}) \tag{2}$$

After encoding the sentence, we use multilayer perceptron network (MLP) with a rectified linear unit(ReLU) as active function to make classification decisions and maintain the model's prediction of the candidate pair $r.T$. To be minded, the output $r.T$ is a vector where each position indicates whether the candidate pair is aligned with the data record at this position. Since there could be multiple labels, the output vectors are not distributions.

$$r.T = ReLU(Wh + b) \tag{3}$$

Because the training data is multi-labeled, we use negative marginal log likelihood as the loss function, namely each position is optimized toward 1 if positive and 0 if negative. We then map the positive candidate pairs back to the data records as silver training labels for the next stage. If a positive candidate pair *(entity, value, r.T)*, which is extracted from the xth sentence, is also in the data records, we label this data record as *Appeared in the xth sentence of the summary.*

3.2 Data Refining

In this module, we use two ranking models to refine the data records. These two rankers have different targets to optimize and separately perform content selection and ordering.

For content selection, we use both ListNet(Cao et al., 2007) and rule-based methods to select data records. The training data of this stage is seriously imbalanced: more than 95% of the input data records do not appear in the summaries and are labeled as negative. This makes it difficult for classification models to achieve good results. So here we use the L2R method to perform content selection. Instead of a point-wise loss function, which looks at a single example at a time, pair-wise and list-wise loss functions try to come up with the optimal ordering of a pair or a list of examples. In this stage we use

Feature	Type	Explanation
Record Type	One hot	The one-hot representation of record type (i.e. PTS)
Is Team	Value	Boolean of team or player
Home Visit	Value	Boolean of home or visit team
Win Lose	Value	Boolean of win or loss
Win ratio	Value	The win ratio of previous matches
Lose ratio	Value	The lose ratio of previous matches
Team Performance	Values	All values of the team (i.e. PTS, PTS_QTR1, FG_PCT)
Player Performance	Values	All values of the player. Zeros if it is team record
Start Position	One hot	The start position of player. Zeros if it is team record
Pair Value	Value	The value of **f**, if not a number then 0
N/A	Value	Whether the value is N/A
Team Rank	Values	Whether the team value is larger that the other
Player Rank	Values	The rank of each record type of this player

Table 2: The details of features used for the ranking unit.

Type	Rule	Threshold
TEAM-PTS	all	$\setminus$
TEAM-WINS	all	$\setminus$
TEAM-LOSSES	all	$\setminus$
AST	bar	9
PTS	bar	11
REB	bar	9
TEAM-FG3_PCT	bar	45
TEAM-FG_PCT	bar	10

Table 3: The details of rules for the ranking unit. 'all' stands for choosing all records of this type of data. 'bar' stands for choosing the data records which value is larger than the threshold.

ListNet, which optimizes a list-wise loss function, so the data imbalance problem can be relieved. Given a list of data records $\mathbf{r} = \{r_k\}_{k=1}^{n} = \{r.E_k, r.M_k, r.T_k\}_{k=1}^{n}$, we design hand-craft features and form a feature vector f_k for each data record as the input of the ranking model. We give the details of the features in the Table 2. Then the ranking model assigns a score s_k^S to each data record.

$$s_k^S = ListNet(f_k) \tag{4}$$

During inference stafe, we use a hyper-parameter threshold α tuned on the validation set to choose data records.

The rules are designed based on common sense and statistics of basketball news. We observe that several types of data records are chosen mainly according to whether the data record's value is larger than a specific threshold. Some other type of data records always appear in pairs, such as FTA and FTM. We give a table of details of the rules in the Table 3.

For content ordering, we use a pair-wise L2R method RankBoost(Freund et al., 2003) to reorder the selected data records. While training, we use the subset of data records $\mathbf{r} = \{r_k | r_k.t \neq negative\}$ to train this model. When we perform inference, the output of the content selecting unit is used as the input. We similarly embed r_k into a feature vector f_k and then use RankBoost to assign a score s_k^O to each r_k.

$$s_k^O = RankBoost(f_k) \tag{5}$$

We use s_k^O to reorder $\{r\}$ into $\{r^O\}$ and feed this ordered list of data records to the text generation module.

3.3 Text Generation

In the text generation module, we use a sequence-to-sequence encoder-decoder system to generate the summaries (Sutskever et al., 2014). Given a list of data records $r^{O}= [r^{O}_{k}]^{n}_{k=1}$. We map these data records to a feature vector e_k by embedding $r.E$, $r.T$ and $r.M$ and concatenate the three embedding vectors and then use one layer of MLP to merge them into the final embedding vector.

The embeddings are then fed into the encoder, which is a BiLSTM to sequentially model the input and maintain the encoder output vectors hidden states h_t.

$$h_t = [h^f_t ; h^b_t] = BiLSTM(e_t, h^f_{t-1}, h^b_{t+1}) \tag{6}$$

The decoder is built based on the Gated Recurrent Network (GRU). At each time step the decoder receives an input e^d_t and calculates the output vector s^d_t. Meanwhile it updates its own hidden state h^d_t.

$$s^d_t, h^d_t = GRU(e^d_t, h^d_{t-1}) \tag{7}$$

Here we implement the attention mechanism, conditional copy mechanism and coverage mechanism to further improve the model's performance.

Attention and Coverage The attention at each step is calculated similar to See et al.(See et al., 2017), which is called perception attention. To calculate the attention weight between the hidden state of the decoder h^d_t and one output of the encoder h_i, we map the two vectors to fix size vectors seperately by two MLPs W_a and U_a with trainable bias b_a as h_{a_i}. Then we use a trainable vector v_a and dot multiply it with $tanh(h_{a_i})$ as the attention score s_{t_i}. At last we calculate the softmax over attention scores $\{s_{t_i}\}^n_{i=0}$ as the attention weights $\{a_{t_i}\}^n_{i=0}$. We finally dot-multiply the attention weights $\{a_{t_i}\}^n_{i=0}$ with the encoder outputs $\{h_i\}^n_{i=0}$ and sum them as the final attention vector $h_{t_{attn}}$.

$$h_{a_i} = W_a h^d_t + U_a h_i + b_a \tag{8}$$

$$s_{t_i} = v^T_a tanh(h_{a_i}) \tag{9}$$

$$a_{t_i} = softmax(s_{t_i}) = \frac{exp(s_{t_i})}{\sum_j exp(s_{t_j})} \tag{10}$$

$$h_{t_{attn}} = \sum^{n}_{i=0} a_{t_i} h_i \tag{11}$$

We also found that model often tends to repeatedly write about the same information, so we introduce coverage mechanism here to relief this problem. The key idea of coverage is to reduce the probability of paying attention to the information that is already generated.

If the sum of the previous attention weights is very high, there is a high probability that the information of this position is already generated. So in coverage model, we maintain a coverage score c_{t_i} for each encoder position at each decoder timestep, which is the sum of the attention weight of the previous timesteps $\{a_{t'_i}\}^{t-1}_{t'=0}$.

$$c_{t_i} = \sum^{t-1}_{t'=0} a_{t'_i} \tag{12}$$

We then modify the previous attention score with this coverage score. We assign a trainable weight vector w_c to c_{t_i} and sum it with h_{a_i} to maintain the adapted attention score.

$$s_{t_i} = v^T_a tanh(h_{a_i} + w_c c_{t_i}) \tag{13}$$

Conditional Copy The copy mechanism has shown great effectiveness as an augmentation of encoder-decoder models recently. At each step the model uses an additional variable z_t to choose to copy or generate a word. The model either copies a word from the input sequence or generates a word from the vocabulary at step t.

Although both $r_k.E$ and $r_k.M$ may appear in the summaries, we only consider the probability of copying $r_k.M$. Instead of directly marginalizing out the latent-variable z_t, when we train the model we assume that any word y_t that appears both in the source data records and the summary is copied, so that we can jointly optimize the negative log-likelihood of y_t and z_t. To be noticed, there might be not only one $r_k.M$ that matches with y_t. Because our input data shares the same sequential order with the information mentioned in the summaries, we map the values from the start of the data records and skip the ones that are already mapped to align the records and copied values.

$$y = \begin{cases} p_{copy}(y_t|z_t; y_{1:t-1}; h_{1:n})p(z_t|y_{1:t-1}; h_{1:n}); \\ \qquad z_t = 1 \\ p_{generate}(y_t|z_t; y_{1:t-1}; h_{1:n})p(z_t|y_{1:t-1}; h_{1:n}); \\ \qquad z_t = 0 \end{cases}$$

We use the attention weights explained previously as the distribution $p_{copy}(y_t|z_t; y_{1:t-1}; h_{1:n})$. We concatenate the decoder input e_t^d, the decoder output s_t^d and the attention vector $h_{t_{attn}}$ and feed them into one MLP layer with sigmoid to model $p(z_t|y_{1:t-1}; h_{1:n})$.

4 Experiments and Results

4.1 Dataset

Here we use the ROTOWIRE dataset(Wiseman et al., 2017), which contains 3378 data-text pair in the training data. In addition to BLEU, this data set provides three automatic evaluation metrics, which are content selection (CS), relation generation (RG), and content ordering (CO). The first primarily targets ”what to say” while the latter two metrics target ”how to say”. These three metrics are calculated based on an information extraction system that serves to align entity-mention pairs in the text with data records. We use the code released by Wiseman et al. (2017) to maintain the evaluation scores of our model.

4.2 Implementation Details

We tune all the hyper-parameters according to the model performance on the validation set. The rules of the ranking unit are chosen according to their performance on the training set. We use grid search to tune parameters of the rankers. We use the implementation of RankLib to train the rankers. The embedding size, hidden size of both the encoder and decoder are all 1200. The layer number of the encoder and decoder are both two. The batch size is 12. We set dropout of 0.1 and use Adagrad to optimize the text generator with a learning rate of 0.01.

For the ranking units and text generator, we use the data records that the IE system extracts directly to reduce noise while training. During validation and test, we use the ranking units to extract the input of the generator.

We re-tokenize the original training data by separating numbers connected by ’-’ and ’:’. We also delete one Latin summary from the training data.

4.3 Performance

The results of our model and other baseline systems are shown in Table 4.

From the results we can see the effectiveness of our model, since it has significantly improved all the content evaluation metrics. Thus we can say refining the input data can help the model to be faithful to the input (RG), select good content (CS) and order them considering overall coherence (CO). We can see the BLEU score of our model is slightly lower than the baseline models. We think this is acceptable in trade of the great improvement of other evaluation scores.

Model	RG		CS		CO	BLEU
	P	#	P	R	DLD	
Validation Set						
Template(Wiseman et al., 2017)	**99.35**	**49.7**	18.28	**65.52**	12.2	6.87
CC(Wiseman et al., 2017)	71.07	12.61	21.90	27.27	8.70	**14.46**
JC + TVD + Rec(Wiseman et al., 2017)	57.51	11.41	18.28	25.27	8.05	12.04
CC + R	76.86	16.43	**31.20**	38.94	**14.98**	13.27
Test Set						
Template(Wiseman et al., 2017)	**99.30**	**49.61**	18.50	**64.70**	8.04	6.78
CC (Wiseman et al., 2017)	71.82	12.61	21.90	27.16	8.68	**14.49**
JC + TVD +Rec (Wiseman et al., 2017)	60.27	9.18	23.11	23.69	8.48	12.96
CC + R	75.12	16.90	**32.79**	39.93	**15.62**	13.46

Table 4: The results of text generation on validation set and test set. CC stands for conditional copy, JC stands for joint copy, TVD stands for the total variation distance loss, Rec stands for reconstruction losses, R stands for ranking.

5 Analysis

5.1 Content Selection

The results of our content selection and ordering models on the valid set are shown in Table 5. The results can prove our models' ability of refining data. We can see, because of imbalanced training data, ranking models with a threshold can significantly out perform classification models.

Model	P	R	F1
ListNet	18.08	26.93	21.63
SVM	10.76	21.27	14.29
Random Forest	9.63	55.36	16.41
ListNet + Rule	59.02	59.98	59.50

Table 5: The results of content selection and data ordering on the valid set.

#	select	emb	hid	bs	GPU	time
1	False	600	600	2	9275	214
2	True	600	600	2	2163	45
3	True	600	600	16	10375	8
4	True	1200	1200	12	10525	16

Table 6: The results of original input and refined order input. 'emb' and 'hid' stands for embedding and hidden dimensions. 'bs' stands for batch size. 'GPU' stands for the maximum memory used on GPU. 'time' stands for the time used for every epoch, the unit is minute.

5.2 Model Efficiency

Our model also significantly improves the efficiency of the model. We show the comparison of our model and CC(Wiseman et al., 2017) model in Table 6 and Figure 3. Our model significantly reaches convergence faster and uses less memory and time to train. The parameter in the embedding and encoder layer is greatly reduced due to the refining of the input. For case #1 and #2, we can see the GPU memory usage and the time for each epoch is greatly reduced, which leads to faster convergence of the model. In case #3 and #4, we show that by refining the input, we can allow larger batch size, embedding size and

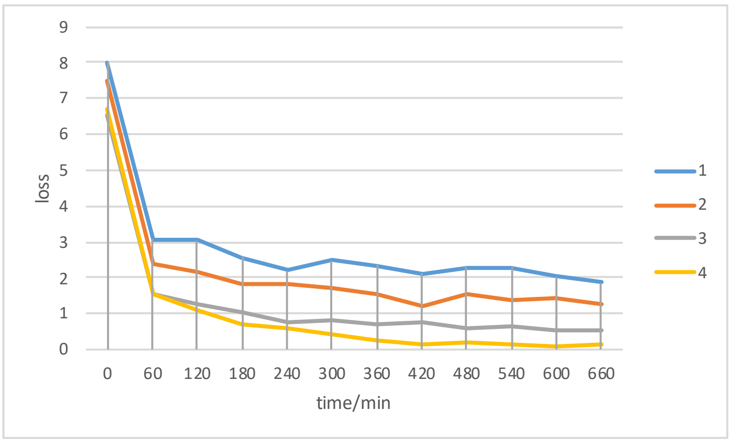

Figure 3: Statistics of how the loss changes over time. The number labels of the poly-lines match with the order in Table 4.

hidden state size for the model to further boost the performance. While the architecture of the generator of our model and CC is similar, we show refining the input can greatly improve the model's efficiency.

5.3 Case Study

Here we show one example of the pipeline on the validation set in Figure 4. We show the triples extracted by the IE system, triples extracted by the refining unit the gold text and the final generated text.

From this example we can see, the IE system has a strong ability of extracting relation pairs from the gold text. The IE system missed two information pairs which are (Pacers,35,TEAM-TEAM-PTS_QTR3) and (Knicks,12,TEAM-TEAM-PTS_QTR3), but succeeded in all other pairs, ending with an accuracy of 87.5% in this example.

The refining system shows a high precision comparing to the gold reference, covering 12 out of 16 triples.

The generated text is very faithful to the refined input at the first 5 sentences, but began making up false information when it tries to generate facts not given by the refined input. 2 - 3 3Pt , 3 - 3 FT are fake information about Jose Calderon where the corresponding information is not selected by the refining system. The following text contains more fake information. This shows the limitations in generating long text for seq2seq models and some shortages of pre-selected refined text. For further improvement, we should improve the ability of the model to generate long text, and also consider dynamically giving information that the model needs instead of feeding fixed triples.

6 Conclusion

In this paper we propose a data-to-text generating model which can learn data selecting and ordering from an IE system. Different from previous methods, our model learns what to say and how to say from the supervision of an IE system. To achieve our goal, we propose to use a ranking unit to learn selecting and ordering content from the IE system and refine the input of the text generator. Experiments on the ROTOWIRE dataset verifies the effectiveness of our proposed method.

Acknowledgement

We thank the anonymous reviewers for their helpful comments on this paper. This work was partially supported by National Key Research and Development Project (2019YFB1704002) and National Natural Science Foundation of China (61876009 and 61572049). The corresponding author of this paper is Sujian Li.

IE (New York Knicks, 82, TEAM-PTS), (New York Knicks, 9, TEAM-WINS), (Indiana Pacers, 31, TEAM-LOSSES), (Indiana Pacers, 17, TEAM-WINS), (Indiana Pacers, 103, TEAM-PTS), (New York Knicks, 38, TEAM-LOSSES), (Roy Hibbert, 18, PLAYER-PTS), (Roy Hibbert, 10, PLAYER-REB), (Carmelo Anthony, 7, PLAYER-FGM), (Carmelo Anthony, 16, PLAYER-FGA), (Carmelo Anthony, 18, PLAYER-PTS), (Rodney Stuckey, 22, PLAYER-PTS), (Rodney Stuckey, 13, PLAYER-FGA), (Rodney Stuckey, 8, PLAYER-FG)

Refine (Knicks, 38, TEAM-LOSSES), (Pacers, 31, TEAM-LOSSES), (Knicks, 9, TEAM-WINS), (Pacers, 17, TEAM-WINS), (Knicks, 42, TEAM-FG_PCT), (Pacers, 53, TEAM-FG_PCT), (Pacers, 33, TEAM-FG3_PCT), (Knicks, 31, TEAM-FG3_PCT), (Knicks, 82, TEAM-PTS), (Pacers, 103, TEAM-PTS), (Carmelo Anthony, 18, PLAYER-PTS), (Carmelo Anthony, 16, PLAYER-FGA), (Carmelo Anthony, 7, PLAYER-FGM), (Carmelo Anthony, 2, PLAYER-FG3M), (Carmelo Anthony, 4, PLAYER-FG3A), (Ian Mahinmi, 10, PLAYER-REB), (Carmelo Anthony, 25, PLAYER-MIN), (Lou Amundson, 17, PLAYER-PTS), (Rodney Stuckey, 22, PLAYER-PTS), (Jose Calderon, 9, PLAYER-PTS), (Jose Calderon, 28, PLAYER-MIN), (Jose Calderon, 4, PLAYER-FGM), (Jose Calderon, 7, PLAYER-FGA), (Carmelo Anthony, 2, PLAYER-FTM), (Carmelo Anthony, 2, PLAYER-FTA), (Carmelo Anthony, 1, PLAYER-REB), (Carmelo Anthony, 1, PLAYER-AST), (Roy Hibbert, 10, PLAYER-REB), (Roy Hibbert, 18, PLAYER-PT)

Gold The Indiana Pacers (17 - 31) throttled the New York Knicks (9 - 38) 103 - 82 Thursday at the Bankers Life Fieldhouse . The Knicks held a five - point deficit at halftime before having a complete meltdown in the third quarter . The Pacers outscored the Knicks 33 - 12 in the third to all but ice the game . Roy Hibbert scored 10 of his 18 points in the third and added 10 rebounds in the contest to notch his 12th double - double on the season . New York earned an impressive 100 - 92 win over Oklahoma City on Wednesday but was unable to carry that momentum into Thursday 's contest against Indiana . The Knicks were unable to get much of anything going in the second half . Carmelo Anthony had n't scored less than 20 points in any game this month , but was held to 18 points on 7 - of - 16 shooting against the Pacers . The Knicks have two days off before hosting the Lakers on Sunday . Indiana was able to rebound nicely after its 104 - 91 loss to Toronto on Tuesday . George Hill and C.J Miles were both able to play , but were on minutes restrictions and only saw 12 and 13 minutes respectively . Rodney Stuckey led the team in scoring , putting up 22 points on 8 - of - 13 shooting . The Pacers will look to secure their second straight win Saturday when they host the Kings .

Generated The Pacers (17 - 31) held on for a 103 - 82 win over the Knicks (9 - 38) on Friday . Carmelo Anthony led the way with a game - high 18 points on 7 - of - 16 shooting , to go along with one rebound , one assist and one block in 25 minutes . Jose Calderon was the only other starter to score in double figures , scoring 10 points on 4 - of - 7 shooting in 28 minutes off the bench . The Knicks shot 53 percent from the field and 33 percent from three - point range . The Knicks were led by Carmelo Anthony , who finished with 18 points (7 - 16 FG , 2 - 4 3Pt , 2 - 2 FT) , one rebound , one assist and one block in 25 minutes . Jose Calderon added 10 points (4 - 7 FG , 2 - 3 3Pt , 3 - 3 FT) , six assists , one rebound and one steal in 33 minutes . The Knicks shot 53 percent from the field and 33 percent from three - point range . Carmelo Anthony was the high - point man for the Knicks , with 18 points (7 - 16 FG , 2 - 4 3Pt , 2 - 2 FT) , to go with one rebound , one assist , one steal and one block . Jose Calderon chipped in 10 points (4 - 7 FG , 2 - 3 3Pt , 3 - 3 FT) , six assists , one rebound and one steal in 33 minutes . The Knicks shot 53 percent from the field and 31 percent from three - point range . The Knicks have now lost four of their last five games , as they continue to battle for the eighth seed in the Eastern Conference . They will look to get back on track as they play host to the Bucks on Sunday . The Pacers have now lost four of their last six games , as they continue to battle for a playoff spot in the Eastern Conference . They will look to bounce back as they play host to the Nets on Sunday .

Figure 4: One example of our system. IE stands for triples extracted by the IE system, Refine stands for triples selected by the refining system, Gold stands for the reference text and Generated stands for the generated result.

References

Antoine Bosselut, Asli Celikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. 2018. Discourse-aware neural rewards for coherent text generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 173–184.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 626–634.

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969.

Paul Holmes-Higgin. 1994. Text generation—using discourse strategies and focus constraints to generate natural language text by kathleen r. mckeown, cambridge university press, 1992, pp 246,£ 13.95, isbn 0-521-43802-0. *The Knowledge Engineering Review*, 9(4):421–422.

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

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.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. Association for Computational Linguistics.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany. arXiv:1706.09254.

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1516–1527.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2018. Data-to-text generation with content selection and planning. *arXiv preprint arXiv:1809.00582*.

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

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)*, volume 1, pages 1073–1083.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2017. Order-planning neural text generation from structured data. *arXiv preprint arXiv:1709.00155*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 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.

Zhu Zhang. 2004. Weakly-supervised relation classification for information extraction. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 581–588. ACM.