

Syntax-guided text generation via graph neural network

Qipeng GUO¹, Xipeng QIU^{1*}, Xiangyang XUE¹ & Zheng ZHANG²¹Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai 200433, China;²NYU Shanghai and AWS Shanghai AI Lab, Shanghai 200335, China

Received 15 May 2019/Accepted 26 December 2019/Published online 31 March 2021

Abstract Text generation is a fundamental and important task in natural language processing. Most of the existing models generate text in a sequential manner and have difficulty modeling complex dependency structures. In this paper, we treat the text generation task as a graph generation problem exploiting both syntactic and word-ordering relationships. Leveraging the framework of the graph neural network, we propose the word graph model. During the process, the model builds a sentence incrementally and maintains syntactic integrity via a syntax-driven, top-down, breadth-first generation process. Experimental results on both synthetic and real text generation tasks show the efficacy of our approach.

Keywords text generation, deep learning, graph neural network, dependency parsing

Citation Guo Q P, Qiu X P, Xue X Y, et al. Syntax-guided text generation via graph neural network. *Sci China Inf Sci*, 2021, 64(5): 152102, <https://doi.org/10.1007/s11432-019-2740-1>

1 Introduction

Generating coherent text sequences is crucial to a wide range of important natural language processing applications, such as language modeling [1], machine translation [2], and dialogue generation [3]. Some of the most popular text generation approaches train an autoregressive recurrent neural network (RNN) to maximize the conditional probabilities of next tokens based on the ground-truth histories, which are sequential in nature. Despite these encouraging results, the generated sentences are often unsatisfactory [4–6].

RNN experiences difficulties in remembering and distinguishing rich and often long dependencies embedded in a flattened sequence. This is true even for advanced variants such as LSTM and GRU [7, 8]. For this reason, the generated texts often lack consistency in long-term semantics and are not coherent syntactically.

The structural dependence relations among words in a sentence are not only sequential but also syntactical and hierarchical. Figure 1 groups word dependence relations into two sets: the first is the sequential dependence **seq-dep** relationship that governs word order in a sentence; the second is the syntactical dependence **syn-dep** relationship, which implies that, for each word in a sentence, there exists another word to govern it. Such a relationship is produced by word-based grammar; therefore, sequential models typically explore only the first relationship.

Taking both types of relationships into account, a sentence can be formulated as a directed cyclic graph (DCG), in which nodes are the words and edges are the unions of the two relationships. Evidently, text generation can be formulated as a graph generation problem; however, this perspective raises two challenges: (1) inference modeling over the graph and (2) making generation process respect the graph topology.

We address the first challenge by borrowing the idea from recent studies on graph-structured neural networks [9–12] and propose the word graph model. In this framework, word representations are

* Corresponding author (email: xpqiufudan.edu.cn)

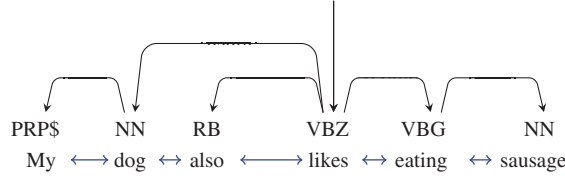


Figure 1 (Color online) Word graph of a sentence. In this paper, we consider both syntactical (black arrows) and sequential (blue arrows) dependencies in the text generation process. As a result, the sentence is modeled as a graph instead of a sequence or a tree.

propagated along the edges throughout the graph. As messages are aggregated, nodes obtain updated representations, and such propagation-and-aggregation rounds can be performed whenever the graph topology is modified during the generation process.

We deal with the second challenge by constructing the word graph incrementally along the dependency edges in a straightforward top-down breadth-first fashion. Because a dependency tree is inherently hierarchical, this method leads to partial sentences that are semantically meaningful, and internally emulates a curriculum learning [13] process.

The contributions of this paper are briefly summarized as follows.

- We regard text generation as a graph generation problem. This new perspective alleviates the long distance dependency problem that plagues the sequential models.
- We adopt the message-passing neural network framework, propose the word graph model, and learn the representation of a complex graph produced by a sentence. The proposed algorithm has a time complexity of $O(n^2)$.
- The top-down breadth-first generation order is an importance-first strategy that generates words carrying more information. An additional advantage is that the intermediate generated result is a simplified but still relatively fluent sentence.
- Our preliminary results on two generation tasks demonstrate the promise of this approach.

2 Proposed method

Our approach to sentence generation creates a word graph. In this section, we first explain the concepts and definitions of constructing such a graph and then describe the text generation process.

2.1 Word graph

Given a text sequence $w_{1:L}$ and its dependency tree T , its word graph is defined as $G = (V, E)$, where $V = \{w_1, w_2, \dots, w_L\}$ is the set of words and E describes the relations between the words. For each word w_i , there is an embedding x_i . In addition to encoding the usual semantic representation, x_i may contain other useful linguistic features such as its word form, part-of-speech tag, and others.

There are two types of edges: the ones of the first type run along the sequence itself capturing the word order (blue arrows in Figure 1) and ones of the second type signify hierarchical dependencies E_{seq} (black arrows in Figure 1). The subset of the former-type edges E_{seq} is reminiscent of an n-gram relationship in sequence learning and can be used to recover the sentence; although one direction suffices, in this study we maintain both directions.

The latter-type edges capture the syntactic structure of a sentence and play a key role in the graph generation process, as we will describe shortly.

For any word pair (w_i, w_j) in $w_{1:L}$, we denote its edge by the following:

$$e_{w_i w_j} = \begin{cases} 1, & \text{if } j - i = 1, \\ 2, & \text{if } i - j = 1, \\ 3, & \text{if } T(i, j) = 1, \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

where $T(w_i, w_j) = 1$ denotes that word w_i governs word w_j in the dependency tree T . $e_{w_i w_j} = 0$ means that no edge exists between w_i and w_j . A question arises when a neighboring edge is also a dependency edge; rather than dedicating a new edge type, we simply use both edge representations.

2.2 Message-passing neural network

The message-passing neural network (MPNN) [12] is a type of graph neural network. It is defined through a message-passing interface, and it has message as well as vertex update functions. The whole MPNN process comprises iterative message passing and vertex updating.

$$\mathbf{m}_u^{t+1} = \sum_{v \in N(u)} M_t(\mathbf{h}_u^t, \mathbf{h}_v^t, \mathbf{e}_{uv}), \quad (2)$$

$$\mathbf{h}_u^{t+1} = U_t(\mathbf{h}_u^t, \mathbf{m}_u^{t+1}), \quad (3)$$

where \mathbf{h} is the state of the vertex, \mathbf{e} is the edge, $N(u)$ represents the neighbors of the node u , M_t is the message function at step t , and U_t is the vertex update function at step t . Because the MPNN is a very general framework, we can use it to describe the min-sum or max-product belief propagation method with specific message and vertex update functions. However, the MPNN often works in a more “dirty” scenario where the goal of the MPNN is not only inference a given graph but also the estimation of the graph. For example, we may know the connectivity of the nodes but lack the details; therefore, it would need to be learned from the data. In addition, training a text generation model from scratch is a problem involving both estimation and inference, and we want the model can estimate the parameter of a graph from the data and inference it to sentences in an end-to-end framework.

2.3 Word graph encoding via message-passing

A word graph is represented by the message-passing framework, similar to the existing graph convolutional network [10]. Each node $v \in V$ is represented by a state vector $\mathbf{h}_v \in \mathbb{R}^d$, initialized with \mathbf{x}_v , and is updated by aggregating information from its neighbours. To do that, for each edge we compute a message:

$$\mathbf{m}_{u \rightarrow v} = f_{\mathbf{e}_{uv}}(\mathbf{h}_u, \mathbf{x}_u; \theta^{(\mathbf{e}_{uv})}), \quad \forall (u, v) \in E, \quad (4)$$

where $f_{\mathbf{e}_{uv}}(\cdot)$ is a fully-connected neural network. In our implementation, the parameters are shared between all edges of the same type. Note that, in propagating a message to other nodes, the word feature \mathbf{x}_u is always inserted to reinforce coherence.

The state vector \mathbf{h}_v is then updated in multiple rounds of message propagation. In each round, v aggregates all the incoming messages from its neighbors $N(v) \equiv \{u : (u, v) \in E\}$:

$$\mathbf{h}_v = f_h \left(\sum_{u \in N(v)} \mathbf{m}_{u \rightarrow v}, \mathbf{h}_v; \theta^h \right), \quad \forall v \in V, \quad (5)$$

where $f_h(\cdot)$ can be a fully-connected or recurrent neural network.

2.4 Word graph generation

Our algorithm incrementally inserts words (nodes) into an initially empty graph until completion, at which point all words are inserted and all edges are formed. This is achieved through a sequence of actions where each action generates a new word, modifies the graph topology, and updates its representation.

An identical word graph can be built with different traversal orders. We choose the most stable one by following the dependency edges E_{dep} in top-down and breadth-first order. We also impose a left-to-right order to generating the children. This procedure ensures a deterministic order when adding edges in the E_{dep} set. A partially completed sentence finds its sequential order by following an in-order walk of the tree, forming a dynamic E_{seq} set that precisely corresponds to those in the target graph at completion. This process is illustrated in Table 1.

Each step of the text generation process involves three actions: generating a new node, sampling its word, updating the graph. The process is described in Algorithm 1.

Generating a new node. Assuming the node u is the current operating node, the first action determines whether to generate its child node according to its representation \mathbf{h}_u :

$$p(a|u, G) = \text{softmax}(f_a(\mathbf{h}_u)), \quad (6)$$

where $a \in \{\text{LC}, \text{RC}, \text{STOP}\}$ is the action. LC, RC, STOP denote the actions of adding a child on the left, on the right, or not adding a child node for the current node, respectively. The choice between left

Table 1 (Color online) An illustration of our graph generation^{a)}

Step	Flatten sequence	Graph	Action		Surrounding
			Node	Word	
1	likes	likes	root	likes	-
2	dog <u>likes</u>		LC	dog	likes
3	dog also <u>likes</u>		LC	also	likes, dog
4	dog also <u>likes</u> eating		RC	eating	likes
5	My <u>dog</u> also likes eating		LC	My	dog
6	My dog also like <u>seating</u> sausage		RC	sausage	eating

a) “Graph” and “Sequence” illustrate the generated word graph and its flatten word sequence. The node in the black box is the current operating node, and the node in the red box is the newly generated node. “Action” can be “LC” or “RC” for generating a left or right child of the operating node, respectively, and “root” or “STOP” is for starting or terminating the process.

and right is dependent on the relative position of the child to the operating node in the sentence. As mentioned earlier, an in-order walk recovers the order and the dynamic edges E_{seq} . Some of these edges may need to be broken as the graph grows.

Sampling a word. Once a node v is generated, we set its initial representation vector $\mathbf{h}_v = \mathbf{0}$ because its word has not been generated yet. Then we update its representation via message passing, calculate the probability of the word choice, and sample an initial candidate w_v :

$$p(w|\mathbf{h}_v) = \text{softmax}(W\mathbf{h}_v + \mathbf{b}), \quad w_v \sim p(w|\mathbf{h}_v). \tag{7}$$

Graph update. After the word w_v is sampled, we re-initialize the representation $\mathbf{h}_v = \mathbf{x}_v$ and start updating the rest of the graph. Starting from v , this is done via message-passing in a broadcast manner using (4) and (5) until all nodes are reached.

Final sentence generation. After the words are added, we perform one more round of message-passing and pick words with argmax of word choice probability at each node.

3 Training

The process of graph generation involves three kinds of predictions: action and word per step, and all words at the final step. As such, our loss function is divided into three components:

$$\mathcal{L} = \sum_{t=1}^T \mathcal{L}^a(\mathbf{a}_t, \hat{\mathbf{a}}_t) + \sum_{t=1}^T \mathbf{I}(\mathbf{a}_t \neq \text{STOP}) \mathcal{L}^w(\mathbf{w}_t, \hat{\mathbf{w}}_t) + \sum_{u \in V} \mathcal{L}^{\text{tree}}(\mathbf{w}_u, \hat{\mathbf{w}}_u), \tag{8}$$

Algorithm 1 Graph-based text generation

```

1: Initialize operating queue:  $Q \leftarrow \emptyset$ ;
2:  $Q.append(\text{root})$ ; //use special parameters for root
3: while  $Q$  is not empty do
4:    $u \leftarrow Q.pop()$ ; //pop the node from the queue
5:   for  $i=1$  to MAX_CHILDREN do
6:     Sampling an action  $a_i \sim p(a|u, G)$ ; //Eq. (6)
7:     if  $a_i = \text{stop}$  then break; //stop the generation of operating node
8:      $v \leftarrow \text{empty node}$ ; //if not break, there is a new node
9:     if  $a_i = \text{LC}$  then  $u.left\_children.append(v)$ ; //left child
10:    if  $a_i = \text{RC}$  then  $u.right\_children.append(v)$ ; //right child
11:    Modify the seq-dep edges related to  $v$ ;
12:    MP:  $N(v) \rightarrow v$ ; //update the new node  $v$ ; Eqs. (4) and (5)
13:    Sampling a word  $w_v \sim p(w|h_v)$ ; //Eq. (7)
14:    MP:  $v \rightarrow N(v) \rightarrow \dots \rightarrow G$ ; //global update from  $v$ ; Eqs. (4) and (5)
15:     $Q.append(v)$ ;
16:  end for
17: end while
18: for  $v \in V$  do //final round, re-sample words for the entire sentence
19:   MP:  $N(v) \rightarrow v$ ; //collect messages from its neighbors
20:   Pick the word with highest probability  $w_v = \text{argmax}_w p(w|h_v)$ ;
21: end for

```

where the indicator function $\mathbf{I}(x) = 1$ if x is true and zero otherwise; $\mathcal{L}^a(\mathbf{a}_t, \hat{\mathbf{a}}_t)$ and $\mathcal{L}^w(\mathbf{w}_t, \hat{\mathbf{w}}_t)$ are the losses per step, $\mathcal{L}^{\text{tree}}$ is the loss at the final round. \mathbf{a}_t and \mathbf{w}_t are the gold action and ground-truth words, respectively. All of the aforementioned loss functions are cross-entropy losses. In principle, each node receives two teaching signals: one when it is created and the other in the final round.

Maximum likelihood training is known to suffer from the exposure bias problem; therefore, we adopt schedule sampling [14] to bridge the gap between training (gold input) and inference (predicted input at test time). Specifically, when computing a message using (4), the input \mathbf{x}_u picks the ground-truth with probability ϵ or the model prediction otherwise.

Note that the top-down breadth-first generation order implicitly generates a simple sentence first and then makes it progressively more complex. In other words, the process implicitly embeds a sort of curriculum learning [13], and we observe that the model is easier to train than general graphs.

4 Experiments

To evaluate the effectiveness of our approach, we conducted two experiments on unconditional text generation tasks: one on a synthetic dataset and another on real-world texts. We compared the proposed method against similarly configured sequential generators based on the recurrent neural network: vanilla LSTM, SeqGAN, and LeakGAN [15,16]. SeqGAN and LeakGAN sidestep exposure bias by adopting an adversarial training paradigm, and LeakGAN optimizes further for long sequences. However, our results show that the GAN approach suffers from mode collapsing.

Implementation: All models had a hidden size of 32 and an embedding size of 128. We used the Adam optimizer with an initial learning rate of 0.01 and annealed it gradually after a few epochs until it reached $1\text{E} - 5$; scheduled sampling probability was annealed from 1.0 (fully ground-truth) to 0.5 (half-half).

4.1 Synthetic tree generation

Because it is difficult to measure the quality of a generated text with syntactic structure, we first conducted a simulated experiment with synthetic data. Following the idea in SeqGAN and LeakGAN, we chose a randomly initialized LSTM as our oracle. Instead of generating a sequence, we modified the oracle LSTM to output a linearized tree. To this end, we maintained several counters to determine the legal actions at each step. The possible actions included an opening and a closing bracket in addition to predicting (artificial) words and tags. The generation process had to obey some basic rules, such as tags and words should appear in pairs and all the brackets should be closed at the end. We added a constant negative bias to the unnormalized output of illegal actions to force the model to choose the legal ones. This mechanism affected both the generation and evaluation processes. As a result, the illegal action received a fixed penalty in the evaluation. We made sure that a depth-first walk of a tree is converted appropriately. We chose the vocabulary size for artificial tags and words to be 50 and 5000, respectively. The oracle allowed us to fabricate an arbitrary training set and evaluate the likelihood of generated sentences. This

Table 4 Results on the IMDB dataset^{a)}. The top half gives the results of generation quality. The bottom half shows the novelty measure by comparing the BLEU score and edit-distance of generated samples against the training set.

			LSTM	SeqGAN	LeakGAN	Ours
Test/quality	↑	BLEU-2	0.652	0.683	0.809	0.876
	↑	BLEU-3	0.405	0.418	0.554	0.643
	↑	BLEU-4	0.304	0.315	0.358	0.415
	↑	BLEU-5	0.202	0.221	0.252	0.286
Train/novelty	↓	BLEU-2	0.915	0.997	0.987	0.941
	↓	BLEU-3	0.750	0.990	0.949	0.827
	↓	BLEU-4	0.545	0.980	0.892	0.613
	↓	BLEU-5	0.387	0.971	0.840	0.361
	↑	Edit dist	14.88	1.05	6.09	18.40
	↑	ED/LEN	71.2%	5.7%	27.4%	70.2%
Human evaluation	↑		0.494	0.535	0.644	0.552

a) “ED/LEN” means normalized over sentence length. ↑ means higher is better, and ↓ means lower is better. BLEU-N means the BLEU score average over 1–N grams.

Table 5 Samples from different models^{a)}

Method	Generated samples
LSTM	This is a heart but after the news reporter comes together of survival and lost and do you’d really simply work.
	But that and their is a great score just as a good start to entertain because it genuine.
	I think it would leave an impression which bad is the exact same formula pretty big .
SeqGAN	† This does not star Kurt Russell, but rather allows him what amounts to an extended cameo.
	This does good to make a movie and film relies a stupid sense of credibility for the genre or any movie not going to it.
	This also too hard at all, but is also a scene but I am not sure of anything, humor and I think you shouldn’t go
LeakGAN	† This movie is very creepy and has some good gory scenes that would be rather disturbing .
	† The story itself we may have seen a dozen times before but it doesn’t much matter .
	† This was a great family film and one of my new favorites from Disney and Pixar .
Ours	I guess I was some good elements for that attempt in the country movie .
	It ’s a movie ends up in this franchise and say this is a fan of his girlfriend.
	The beauty of the film, in this movie, I was a good man and that I don’t know.

a) † demonstrates a direct copy from training set.

whole test set for each generated sample. In addition, we add an informal measure of novelty by looking at the BLEU score and edit distance against the training set. There is a similar measurement proposed in [20]; they use the Jaccard distance, which is weaker than BLEU, and the edit distance for measuring sentence similarity. Intuitively, “novelty” encourages the model to generate unseen samples and patterns and punishes repetition. However, “novelty” does not measure the rationality of the unseen patterns. In this case, a good model should have high “novelty” and “quality” simultaneously. They are different measures that serve the same purpose: a higher BLEU score or a lower edit distance indicate that the model is more likely just memorizing and copying text from the training set.

Results. Because this is a pure generation task (i.e., there is no message to decode), the sentences are typically not meaningful. In general, our texts are more well-formed syntactically. On the other hand, while the texts produced by SeqGAN and LeakGAN appear to have high quality, usually they are just slight variations of those in the training set. See Table 5 for fully generated samples, and Table 6 for step-wise partial sentences produced by our model. More examples can be found in Appendix.

The quantitative results are shown in Table 4, which indicates our approach outperforms all the other methods. When generating text, it manages to keep both the quality as well as novelty high owing to a number of factors. Our model employs message passing along the dependency and word order edges, exploring structures much better than a vanilla LSTM generator. Mode collapsing is a known problem in models using adversarial training; however, our framework uses maximum likelihood and thus we reduce the problem of distribution drift with scheduled sampling.

We also provide a human evaluation (last row of Table 4). We sampled 50 unique sentences from different models and asked 20 people to score it; on average, each sentence was evaluated by 5 different people. The scores were between 0 and 1, where a higher score indicates the sentence is more realistic.

Table 6 Step-by-step examples^{a)}. Note how re-sampling helps to correct mistakes (c.f. the correction of “liked” in the first example).

Step	Case 1	Case 2
1	<u>get</u>	<u>is</u>
2	<u>I</u> get	<u>This</u> is
3	I get <u>racing</u>	This is <u>one</u>
4	I get racing <u>because</u>	This is <u>a</u> one
5	I get racing because <u>I</u>	This is a <u>good</u> one
6	I get racing because I <u>liked</u>	This is a good one <u>.</u>
7	I get racing because I liked <u>.</u>	This is a good one , <u>and</u>
8	I get <u>the</u> racing because I liked .	This is a good one , and <u>excellent</u>
9	I get the racing because I <u>would</u> liked .	This is a good one , and excellent <u>.</u>
10	I get the racing because I would like .	This is a good one , and excellent <u>job</u> .
11		This is a good one , and excellent job <u>of</u> .
12		This is a good one , and excellent job of <u>NUM</u> .
13		This is a good one , and excellent job of <u>the</u> NUM .
14		This is a good film , and excellent job of the film .

a) word is the new word at a step and **final sentence** is the result of re-sampling at the final round.

LeakGAN received the highest score, likely because it was copying from the training set.

5 Discussion

One drawback of adopting the message passing paradigm is the computational overhead. Because multiple passes are needed, the computation overhead is $O(N^2)$, N being the length of a sentence. When the outer loop inserts a new node, the inner loop processes the propagation. In addition, the trees within a mini-batch are usually unaligned in their structures, and thus we have to process one tree at a time and aggregate gradients over a mini-batch of trees before updating.

We applied a number of optimizations. For example, we maintain a flag to avoid redundant updates when the representations of a node’s neighbors have not changed. Because the graphs are often sparse, this optimization brings significant computational saving. We also tried stopping updates for nodes that are too far away or are leaves. Without these optimizations, training was almost 10 times slower than that for a similarly configured LSTM; however, with this optimization it is five times slower, which is still very significant. Making the training process more efficient is one of our future research directions.

6 Related work

Although deep neural networks have made great progress in text generation, most of them employ sequential models performing auto-regression directly on the text sequence. For example, Ref. [21] linearized parsing trees to bracket expression form and used an attention-enhanced sequence-to-sequence model to parse a sentence. Although this method can generate a tree-structured text with slight modifications, the linearization process only exacerbates the long-standing challenge of dealing with long-distance dependencies in sequential models.

While structures for sentence representation have been explored in studies such as Tree-LSTM [22], text generation models attempting to explicitly incorporate syntactic structures have started to appear only recently. In contrast, we adopt the MPNN framework and explicitly treat text generation as a graph generation problem. Some of the other related studies are as follows.

Ref. [23] used a transition-based method to generate phrase-structure trees. A parser works from the bottom-up, consuming the sentence while building the hierarchy. This mechanism can be converted to a generative model, by replacing the SHIFT operator with an action that generates a word. We observe two difficulties here. One is that the bottom-up parsing actions can have difficulties dealing with partially complete texts; second, the depth-first nature ignores the opportunity to have global planning with future contexts.

Ref. [24] used a doubly recurrent neural network model comprised of separate width and depth recurrences to generate tree structured sentence directly. Same as [21], they used an encoder-decoder

framework to generate a tree structure from a given sentence.

Ref. [25] also proposed a sequence-to-tree approach, which transformed trees into equivalent ternary trees, and then, used a tree-structured search to generate the text in the form of dependency trees.

The graph neural network [26] and its more recent variant, the message passing neural network, have been applied to a variety of graph problems [9–12, 27, 28]. To the best of our knowledge, our method is the first to directly adopt MPNN for text generation.

7 Conclusion

Text generation has been considered as a sequential problem for a long time. The conventional approach is to decompose a sentence into a series of words via chain rule and generate the words in that order. This is an effective strategy, but it only covers a part of word relationships and ignores the structure of language. In this paper, we introduced a graph-based text generation approach that considers the syntactic dependency relationships between words. Experimental results on synthetic and real text generation tasks show that our approach outperforms the sequential methods.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2018YFC0831103), Shanghai Municipal Science and Technology Major Project (Grant No. 2018SHZDZX01), and Zhejiang Lab.

References

- 1 Sordoni A, Galley M, Auli M, et al. A neural network approach to context-sensitive generation of conversational responses. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL), Denver, 2015. 196–205
- 2 Bahdanau D, Cho K H, Bengio Y. Neural machine translation by jointly learning to align and translate. In: Proceedings of the 5th International Conference on Learning Representations, 2015
- 3 Serban I V, Sordoni A, Bengio Y, et al. Building end-to-end dialogue systems using generative hierarchical neural network models. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence, 2016. 3776–3784
- 4 Ranzato M A, Chopra S, Auli M, et al. Sequence level training with recurrent neural networks. In: Proceedings of the 4th International Conference on Learning Representations, 2016
- 5 Wiseman S, Rush A M. Sequence-to-sequence learning as beam-search optimization. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Austin, 2016. 1296–1306
- 6 Bowman S R, Vilnis L, Vinyals O, et al. Generating sentences from a continuous space. In: Proceedings of the SIGNLL Conference on Computational Natural Language Learning, Berlin, 2016. 10–21
- 7 Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*, 1997, 9: 1735–1780
- 8 Chung J Y, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: Proceedings of the Advances in Neural Information Processing Systems Deep Learning Workshop, 2014
- 9 Henaff M, Burna J, LeCun Y. Deep convolutional networks on graph-structured data. 2015. ArXiv:1506.05163
- 10 Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations, Toulon, 2017
- 11 Battaglia P W, Pascanu R, Lai M, et al. Interaction networks for learning about objects, relations and physics. In: Proceedings of the Thirtieth Conference on Neural Information Processing Systems, 2016. 4502–4510
- 12 Gilmer J, Schoenholz S S, Riley P F, et al. Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning, Sydney, 2017. 1263–1272
- 13 Bengio Y, Louradour J, Collobert R, et al. Curriculum learning. In: Proceedings of the 26th International Conference on Machine Learning, Montreal, 2009. 41–48
- 14 Bengio S, Vinyals O, Jaitly N, et al. Scheduled sampling for sequence prediction with recurrent neural networks. In: Proceedings of the 29th Conference on Neural Information Processing Systems, Montréal, 2015. 1171–1179
- 15 Yu L T, Zhang W N, Wang J, et al. SeqGAN: sequence generative adversarial nets with policy gradient. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017. 2852–2858
- 16 Guo J X, Lu S D, Cai H, et al. Long text generation via adversarial training with leaked information. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018. 5141–5148
- 17 Fedus W, Goodfellow I J, Dai A M. MaskGAN: better text generation via filling in the _____. In: Proceedings of the 6th International Conference on Learning Representations, Vancouver, 2018
- 18 Diao Q M, Qiu M H, Wu C Y, et al. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014. 193–202

- 19 Papineni K, Roukos S, Ward T, et al. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 2002. 311–318
- 20 Wang K, Wan X J. SentiGAN: generating sentimental texts via mixture adversarial networks. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18), 2018. 4446–4452
- 21 Vinyals O, Kaiser L, Koo T, et al. Grammar as a foreign language. In: Proceedings of the Neural Information Processing Systems, 2015. 2773–2781
- 22 Tai K S, Socher R, Manning C D. Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, 2015. 1556–1566
- 23 Dyer C, Kuncoro A, Ballesteros M, et al. Recurrent neural network grammars. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, 2016. 199–209
- 24 Alvarez-Melis D, Jaakkola T S. Tree-structured decoding with doubly-recurrent neural networks. In: Proceedings of the International Conference on Learning Representations, 2017
- 25 Zhou G B, Luo P, Cao R Y, et al. Tree-structured neural machine for linguistics-aware sentence generation. In: Proceedings of 32nd AAAI Conference on Artificial Intelligence, 2018. 5722–5729
- 26 Scarselli F, Gori M, Ah Chung Tsoi M, et al. The graph neural network model. *IEEE Trans Neural Netw*, 2009, 20: 61–80
- 27 Wu S Z, Zhang D D, Yang N, et al. Sequence-to-dependency neural machine translation. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017. 698–707
- 28 Li Y J, Tarlow D, Brockschmidt M, et al. Gated Graph Sequence Neural Networks. In: Proceedings of the 4th International Conference on Learning Representations, San Juan, 2016