Natural Language Processing: Survey on Generating Sequences with Recurrent Neural Networks

Xingming Li

January 10, 2024

1 Introduction

In the field of deep learning, neural networks have been employed to process all kinds of data. With the penetration of neural networks in various domains, the approaches to Natural Language Processing (NLP) tasks have gradually begun to adopt such deep learning methods. Thereof, Recurrent Neural Networks (RNNs) has been widely used in Speech Recognition, Machine Translation and other tasks because of its excellent performance in processing data sequences. In practice however, although RNNs have been remarkably successful, they still face some difficulties like capturing long-term dependencies in a sequence. In order to meet the challenges in different tasks, many variants and derivatives of RNNs have been proposed. This survey attempts to show the structure and principle of the RNN and the improvement of its variants including Long Short-Term Memory (LSTM) in Graves (2013) and Variational Autoencoder (VAE) which contains a Bi-directional RNN (BRNN) encoder in Ha and Eck (2017) and Tolosana et al. (2021) by demonstrating their applications in handwriting and sketch prediction and synthesis.

2 Recurrent Neural Networks

2.1 Standard RNN

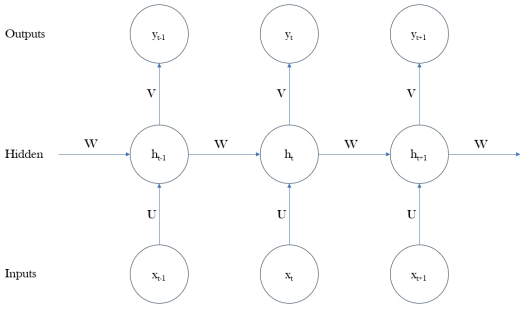

Figure 1: RNN Basic Structure

Figure 1 shows the standard structure of the RNN where each circle can represent a vector belonging to the input, output and hidden layers respectively. Data sequences enter into the network in chronological order. Chronology, in other words, represents the concept of time steps. U and V are the parameter matrices between the corresponding layers. W is the weight matrix between each time step. The reason why RNN can solve

sequence problems is that it can remember information from time to time. The hidden layer of each time step is determined not only by the input layer of that time, but also by the hidden layer of the previous time. According to the definition, the equations of the forward propagation process of RNN can be simply denoted as follows, where y_t represents the output of time step t, h_t represents the value of the hidden layer of time step t and b represents the bias vector which has been omitted for clarity in Figure 1:

$$h_t = f(Ux_t + Wh_{t-1} + b) \tag{1a}$$
$$y_t = g(Vh_t + b) \tag{1b}$$

2.2 Prediction Network

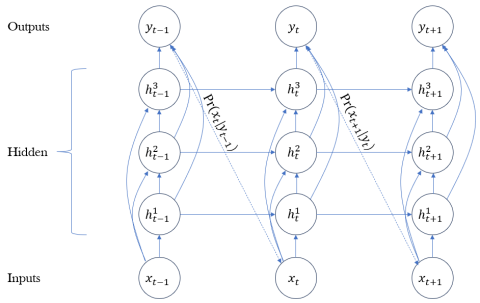

Figure 2: RNN Prediction Architecture

RNNs have extensible structures in both horizontal and vertical directions. The RNN prediction architecture used in Graves (2013) which is shown in Figure 2 is considered to be "deep" in both space and time. Unlike standard RNNs, the hidden layer can contain a stack of N recursively connected hidden layers. Hence, the Equation(1a) and Equation(1b) can be changed to:

$$h_t^1 = f(W_{ih^1}x_t + W_{h^1h^1}h_{t-1}^1 + b_h^1) \tag{2a}$$
$$h_t^n = f(W_{ih^n}x_t + W_{h^{n-1}h^n}h_t^{n-1} + W_{h^nh^n}h_{t-1}^n + b_h^n) \tag{2b}$$
$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^ny}h_t^n \tag{2c}$$
$$y_t = g(\hat{y}_t) \tag{2d}$$

Matrices W represent weights (e.g. W_{ih^n} weight matrix connects input to n^{th} hidden layer, $W_{h^1h^1}$ is the recurrent connection in first hidden layer, etc.). Each output vector y_t is used to parameterize a prediction distribution $Pr(x_{t+1}|y_t)$ for the possible next input x_{t+1}. The first element x_1 of each input sequence is always a null vector. Therefore, the network sends out a prediction for x_2. The number of processing steps between the bottom and top of the network is reduced by skipping connections from inputs to all hidden layers and from all hidden layers to outputs. This can, to a certain extent, overcome the difficulty of training deep networks by alleviating the "vanishing gradient" problem mentioned in Bengio et al. (1994).

3 Long Short-Term Memory

3.1 LSTM Memory Cell

In the standard structure of the RNN above, we can see that the hidden state of each time step is determined not only by the input at that time, but also by the value of the hidden layer at the previous time. If a long sequence has been processed by the end, the RNN may not be able to remember the information details from the beginning of the sequence. LSTM, as a variant of RNN, mitigates this problem with its "gates" helping store information selectively in the memory cell. In other words, the "gates", Input Gate, Output Gate and Forget Gate respectively, are used to control the memory and forgetting of information at every time step. Note that the structure diagram of the LSTM in Figure 3 is the internal structure of a single time step in the entire workflow.

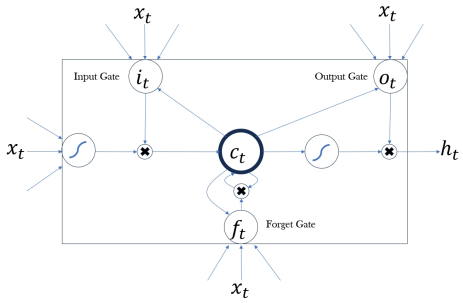

Figure 3: Long Short-Term Memory Cell

Unlike the standard RNN, the original input vector x_t appears three more times entering Input Gate, Output Gate and Forget Gate separately. We can simply observe that the four external inputs complement each other in the LSTM memory cell to create the value h_t. There are two activation functions commonly used in LSTM, one is tanh and the other is sigmoid. For the version of LSTM used in Graves (2013), h_t is implemented by the following composite function:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{3a}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{3b}$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3c}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{3d}$$
$$h_t = o_t \tanh(c_t) \tag{3e}$$

The weight matrix subscripts in the above equations have obvious intentions, such as W_{hi} is the hidden-input gate matrix, W_{xo} is the input-output gate matrix, etc. The bias vectors which have been added to i, f, c, and o in the composite function are omitted again in Figure 3 for clarity.

3.2 Handwriting Prediction and Synthesis

To test whether the LSTM-based prediction network can be used to generate convincing sequences of real values, Graves (2013) applied it to online handwriting data. In this case, online means that the writing is recorded as a sequence of pen-tip locations, while offline handwriting is only available with page images. Due to its low dimensionality (two real

numbers per data point) and ease of visualization, online handwriting is an attractive option for sequence generation.

For the handwriting experiments, the basic RNN structure and equations remain unchanged from Section 2.2. Each input vector x_t consists of a pair of real values x_1 and x_2, which defines the pen offset from the previous input, and a binary x_3, with a value of 1 if the pen is taken off the board and otherwise a value of 0. A bi-variate Gaussian distribution is used to predict x_1 and x_2, while a Bernoulli distribution is used for x_3.

Figure 4 shows the handwriting sample generated by the prediction network. The network has apparently learned to mimic strokes, letters and even short words, especially common words like "of" and "the". It also invented some words like "eald", "gagher" and "salet" which seem somewhat believable in English. Considering that the average character takes up more than 25 time steps, this demonstrates the network's ability to generate coherent long structures.

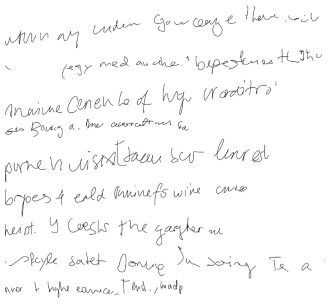

Figure 4: Prediction Sample

Handwriting synthesis is the generation of handwriting for a given text. Obviously, the prediction network we have described so far cannot do this, because there is no way to constrain the letters the network writes. Graves (2013) described an extension that allows the prediction network to generate data sequences based on some high-level annotation sequences which are strings in this case. The results shown in Figure 5 can make one believe that they are often indistinguishable from real handwriting.

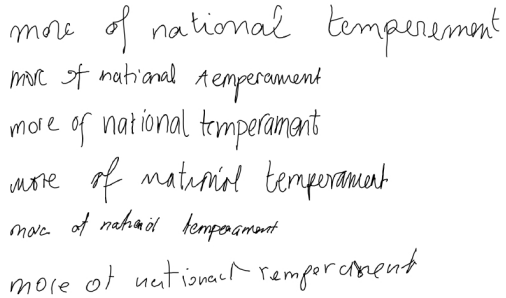

Figure 5: Synthesis Sample (The top line is real handwriting, and the rest are unbiased samples from the synthetic network.)

4 Variational Autoencoder with Bi-directional RNN

BRNN, proposed by Schuster and Paliwal (1997), is an extended form of standard RNN. Standard RNNs focus only on the preceding part of the sequence, while BRNNs focus on the preceding and following context at the same time because structurally, they consist of two RNNs in opposite directions which are connected to the same output layer. That

is, it can use more information to make predictions. BRNNs are slightly different with RNNs in details such as the steps of training as well.

In Ha and Eck (2017) and Tolosana et al. (2021), VAE with BRNN is used in the synthesis of sketches and handwriting. First, an Autoencoder (AE) consists of an encoder and a decoder. It is usually used for feature representation by dimensionality reduction and data recovery of the input high-dimensional feature data such as images. The encoder is used to compress the data ,that is, to map a high-dimensional data sample to a numerical code in a low-dimensional space, while the decoder is used to restore data ,that is, to map the code to a high-dimensional space. Both the encoder and decoder here are nonlinear mapping functions. In deep learning, this can be achieved by building neural networks. VAE is an improvement on AE which the major difference is that a feature distribution is applied to the latent space rather than some fixed discrete code points mapped to the latent space. The encoder is not just used to encode the input data, but to learn the mean and variance for each feature of each sample based on the input sample information. And it makes the distribution $P(Z|X)$ for these means and variances as close as possible to the standard normal distribution. The decoder uses data sample $[z_1, z_2, z_3, ..., z_N]$ of the latent space feature distribution to generate data sample X', which approximates the distribution of the input data sample X.

In the work of Ha and Eck (2017), the authors examined lower dimensional, vector-based representations ground on the way humans draw. Their model, Sketch-RNN, is a Sequence-to-Sequence VAE which uses BRNNs as repeatable neural network units. The purpose of using the Sequence-to-Sequence VAE is to train the neural network to encode the input statement as a floating point vector, a latent vector, and the decoder uses this latent vector to generate an output sequence that reconstructs the input sequence as closely as possible. Specifically, the decoder obtains the latent vector and generates sequences for drawing new sketches.

In the study of Tolosana et al. (2021), the authors proposed DeepWriteSYN as a novel online handwriting synthesis method by deep short-term representation. It consists of two modules: i) Optional and interchangeable temporal segmentation that divide handwriting into short segments which form single or multiple consistent strokes; ii) Online synthesis of these short sequences of handwriting. The synthesis is based on Sequence-to-Sequence VAE as well. As the authors' experiments show, DeepWriteSYN can generate real handwriting variations for a given handwriting structure, corresponding to natural variations within a given population or a given subject. This could be useful for the development of fully synthesized long-term realistic handwriting generation models or the natural variations of a given handwriting sample.

5 Conclusion

This survey set out to explore the structure and working principle of RNN. The paper has also shown that variants and derivative models of RNNs such as LSTM and VAE can be used to generate very realistic handwriting and sketches. We could be excited about the possibilities for the future of generative models in the area of NLP. LSTM, VAE and other generative models will enable many interesting and innovative applications in many tasks. They can also be used as tools to enhance our understanding of our own creative thought processes.

References

Bengio, Y., Simard, P. and Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE transactions on neural networks* **5**(2), 157–166.

Graves, A. (2013), 'Generating sequences with recurrent neural networks', *arXiv preprint arXiv:1308.0850* .

Ha, D. and Eck, D. (2017), 'A neural representation of sketch drawings', *arXiv preprint arXiv:1704.03477* .

Schuster, M. and Paliwal, K. K. (1997), 'Bidirectional recurrent neural networks', *IEEE transactions on Signal Processing* **45**(11), 2673–2681.

Tolosana, R., Delgado-Santos, P., Perez-Uribe, A., Vera-Rodriguez, R., Fierrez, J. and Morales, A. (2021), Deepwritesyn: On-line handwriting synthesis via deep short-term representations, *in* 'Proceedings of the AAAI Conference on Artificial Intelligence', Vol. 35, pp. 600–608.