

Sequence Model Principle and Application Analysis in Natural Language Processing

Yizhen Ma(2006703)^{1,2}

1.University of Wales Trinity St David, Swansea, Britain; 2.Wuhan University of Technology, Wuhan, China

Abstract: Recently, the application of deep neural networks including recurrent neural networks and their variants, have resulted in impressive results, and achieve better performance compared with traditional methods. In this assignment, We studied different deep models for processing time series data, analyzed their design considerations and problems they solved, and discussed their advantages and disadvantages. Based on this, we further analyze several sequence models in the field of Natural Language Processing, we mainly focus on the effectiveness of the deep models they used.

Key words: DNNs; sequence model, NLP

1. Introduction

Language is a talent of human beings, how to use natural language to communicate with computers has long been pursued by people. Natural language processing(NLP) is the realization of natural language communication between humans and machines. But natural language is a highly abstract symbolic system. There are discrete and sparse data between texts, and also problems such as polysemous words[1]. The deep learning methods have powerful feature extraction and learning capabilities, which can better process high-dimensional sparse data, and has made great progress in multiple tasks in NLP. In addition, since NLP tasks usually require processing of time series data, which is characterized by a certain correlation between the data at different time steps. The deep sequence model is proposed to learn this correlation, so as to provide guidance for classification, prediction or generation tasks.

In this assignment, We analyzed the development of deep sequence model, namely the original RNN, LSTM, GRU and other derivatives of RNN such as seq2seq and Transformer. In fact, the development of deep sequence models is closely related to the maturity of computer hardware technology. For example, models based on Transformer architecture own high computational efficiency and the ability to capture long-term dependencies in the sequence, but Transformers require huge amounts of computing resources. Of course, this is no longer a problem in the current computing environment. We critically evaluate each model and analyze their advantages and drawbacks and hope to identify the characteristics of each model thus capable to make appropriate selection of models under different scenarios. Furthermore, we analyze the application of deep sequence models in the context of NLP, we focus on the model selection, modification and architectural design in these work. We hope to summarize the paradigm model selection and design of different tasks, so as to provide us with guidance in specific tasks in future work.

2. Main Body

2.1 Recurrent Neural Network

Recurrent neural network(RNN)[2] was first proposed to compensate for some limitations in the structure of the Multilayer Perceptron(MLP)[3]. For example, when processing sequential data, this type of data has a characteristic that changes in multiple time steps, that is, every time steps in the entire sequence are not independent, but contains sequential information. For example, when

trying to predict the next word in a text generation task, the network must know its previous input and output information. MLP structure is obviously inappropriate when it comes to time series data since it is designed for static data[4]. RNN effectively uses the data correlation expansion calculation of continuous time steps, and at the same time performs the same calculation process for each element in the input sequence, that is, the output of the current step depends on the previous calculation. It can be seen that this network structure can obtain all input sequence information, and theoretically, RNN has the ability to process sequence information of any length.

The structure of computation units in RNN can be seen as Figure 1. The computation of hidden state of current unit in RNN is not only determined by the input of current time step, but also determined by the previous time step. The chain structure of RNN allows information continuously recirculated internally, so that it can ensure that the previous information is preserved in each step of the calculation.

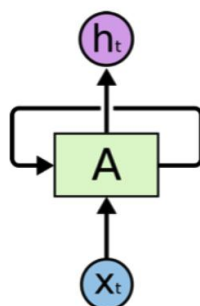


Figure 1. The structure of RNN unit

Although RNN have been successfully applied in the fields of speech recognition, text modeling, translation, subtitles, etc. It suffers from some serious problem in actual use, such as vanishing gradient, exploding gradient and lack the ability to capture long-term dependence which is very important for machine translation and sentence comprehension. The vanishing and exploding gradient problem are caused by continuous multiplication of gradient during the backpropagation process, the output of current time step requires the computation of hidden state of previous time step, and with the length of sequence increased, the gradient accumulation will lead to the problem of vanishing gradient and exploding gradient. RNN can solve the problem of short-term dependence to some extent, but when this kind of dependence become longer, RNN usually fail to capture this kind of dependence, this phenomenon is especially obvious in the process of sentence translation. The reason behind this is, in such case, the useful information is far away from the place where the information needs to be processed, it is easy for RNN to fail to learn useful information, and the final derivation task may fail.

2.2 LSTM

Long Short Term Memory network[5] (LSTM), a special RNN network which is designed to solve the long-term dependency problem and also the vanishing/exploding gradient problems. Since the day LSTM was proposed, many people have improved and popularized it. Their work has been used to solve a variety of problems and is still widely used until now.

The core of LSTMs is the cell state, which is represented by a horizontal line running through the cell. The cell state is like a conveyor belt. It runs through the entire cell but has only a few branches, which ensures that the information flows through the entire RNNs unchanged. LSTM controls the cell state by three gates, which are called forget gate, input gate and output gate. The first step of LSTM is to decide what information the cell state needs to discard. This part of the

operation is handled by a sigmoid unit called the forget gate. Through viewing the previous output and information of input, the forget gate learns to output a vector whose value is between 0 and 1. The value in the vector indicates which information in the cell state is retained or how much is discarded. The next step is to decide what new information to add to the cell state. This step is divided into two steps. First, use previous hidden state and current input to determine which information to update through an operation called an input gate. Then use it to get new candidate cell information through a tanh layer, which may be updated to the cell information. After updating the cell state, it is necessary to determine which state characteristics of the output cell are based on the input. Here, the input needs to pass through a sigmoid layer called the output gate to obtain the judgment condition, and then the cell state passes through the tanh layer to obtain a vector of intermediate values range from -1 to 1, which is multiplied by the judgment condition obtained by the output gate to obtain the final output of the RNN unit.

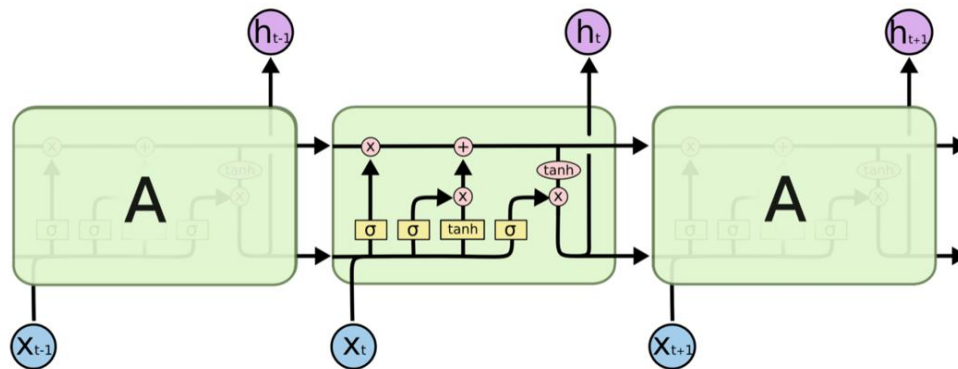


Figure 2. LSTM structure

According to the discussion in the previous section, the main reason for the vanishing and exploding gradient in RNN is that the gradient function contains a continuous multiplication term. If the continuous multiplication term can be removed, the problem of gradient disappearance can be overcome. LSTM uses the gate function to make the continuous multiplication term approximately equal to 0 or equal to 1, thereby removing the continuous multiplication term. When the gradient of the gate is close to 1, the continuous multiplication term can ensure that the gradient is well transmitted in the LSTM and avoid the disappearance of the gradient. When the gradient of the gate is close to 0, it means that the information at the previous moment has no effect on the current moment, and there is no need to return the gradient at this time. At the same time, the gate function in LSTM can bypass certain units to memorize longer step information, which also solve the problem of long-term dependency to some extent, similar idea has been used in ResNet[6] to overcome gradient vanishing problem in deep neural networks.

Both RNN and LSTM can only predict the output at the next time based on the timing information of previous time steps. However, in some problems, the output at the current time is not only related to the previous state, but may also be related to the future state. For example, predicting the missing words in a sentence not only needs to be judged based on the previous text, but also needs to consider the content behind it, and truly judge based on the context. Bi-LSTM[7] was proposed to solve this problem to store two values in hidden layer to ensure both forward and backward information are under consideration, in some tasks, Bi-LSTM performs better than one directional LSTM.

2.3 GRU

Gate Recurrent Unit(GRU)[8] is also a type of Recurrent Neural Network (RNN). Similar to

LSTM, it is also proposed to solve problems such as long-term memory and gradients in back propagation. In the original paper, GRU achieves a similar performance compared with LSTM but more cheaper computational cost. A GPU unit also accept input from the hidden state of previous unit and input from current node. A GPU unit contains two gates, namely reset gate and update gate. The reset gate accept from previous node and current input, and decide how much of previous state are added to the current hidden state. The update gate blended both forget and selective memory function, which can be performed at the same time when the same gate is used. Compared with LSTM, there is one less gate inside GRU, which has fewer parameters than LSTM, but it can also achieve functions equivalent to LSTM.

2.4 seq2seq

Sequence to sequence(seq2seq) model belongs to a kind of encoder-decoder structure, it is used to solve the problem of unequal length of input and output sequence, for example, in machine translation, the source sentence and target sentence in different language also have different lengths. The basic idea of seq2seq model is to use two RNNs, the first RNN serves as the encoder and the other RNN as the decoder. The encoder is responsible for compressing the input sequence into a vector of a specified length. This vector can be regarded as the semantics of the sequence. This process is called encoding. As shown in the Figure 3., the simplest way to obtain the semantic vector is to directly use the hidden state of the last input as semantic vector. It is also possible to perform a transformation on all the hidden states of the input sequence to obtain a semantic variable.

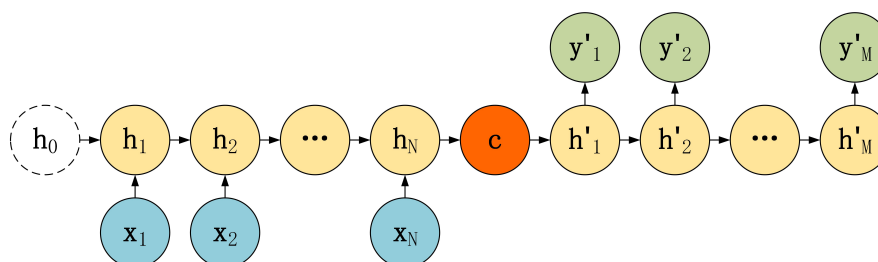


Figure 3. A typical seq2seq architecture

The decoder is responsible for generating the specified sequence based on the semantic vector. This process is also called decoding. The simplest way is to input the semantic variable obtained by the encoder into the RNN of the decoder as the initial state to obtain the output sequence. It can be seen that the output at the previous moment will be used as the input at the current moment, and the semantic vector is only used as the initial state to participate in the operation.

Seq2seq model has a lot of derivatives, the main difference between them is the structure of decoder. One kind of derivative is that the decoder no longer regards the context vector as the initial hidden state of the RNN, but as the input of each neuron of the RNN, that is, each neuron in the decoder has the same context vector input. Another structure of decoder is similar to the structure described above, but with the output of the previous neuron in the input part. That is, the input of each neuron includes: the hidden layer vector of the previous neuron, the output of the previous neuron, and the current input context vector.

2.5 Transformer

Transformer architecture is first proposed by Google in their paper[9]. It has beaten many

SOTA methods in various tasks of NLP since the day it was proposed. The biggest feature in Transformer is that the traditional CNN and RNN are abandoned, and the entire network structure is entirely composed of the attention mechanism. Due to its excellent performance and friendliness to downstream tasks, it has been expanded to the field of computer vision. Recently, there have been some good attempts in the combination of Transformer and vision tasks, such as object detection[10] and image recognition[11].

The architecture of Transformer is showed in Figure 4. it's worth noting that for NLP tasks, the input words are first pass through an embedding layer to transfer to vectors(which usually called Word2Vec). Two key components in Transformer are positional embedding and multi-head attention. The former is proposed to solve the problem of sequence information lost during parallel computation, since transformer do not contain a chain structure or sequence encoding manner, so it is vital to provide a distance measurement between input of different time steps. Two typical methods to represent position are network automatic learning and define a rule. Multi-head attention is an expansion of tradition attention mechanism, the main goal of attention in Transformer is to ensure the encoding output of each input will introduce the encoding information of the remaining input through the attention mechanism, thus achieve the purpose of short-term or long-term dependence extraction. Further, Transformer uses multiple heads to extends the similarity of the different feature levels that the model focuses on the input. In an Transformer based model, different heads in a the same layer usually have different attention patterns.

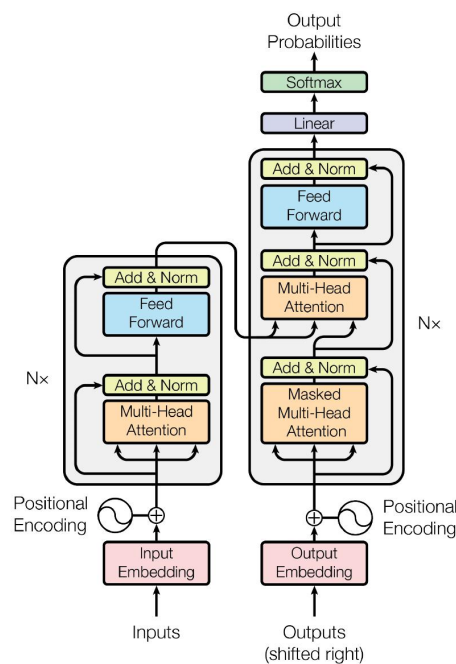


Figure 4. Transformer architecture

Transformer provides parallel computing capabilities and shows stronger performance than other sequence models when it comes to long-term dependence. Whether for RNN, LSTM or GRU, it is not conducive to parallel training and inference, because related algorithms can only be calculated from left to right or from right to left. As for long-term dependence extraction, the long-range dependence information will lost in the sequential calculation process of tradition sequence model. Although LSTM claims to be alleviated, it cannot be completely resolved.

2.6 DNNs in NLP tasks

In this section, we review several methods which take advantage of DNNs in the domain of NLP, we mainly focus on the deep neural networks they use and analyze the effectiveness of them.

DPCNN: Deep Pyramid Convolutional Neural Networks for Text Categorization (DPCNN)[12] is proposed for solving the problem of previous methods that long-distance dependence of the text cannot be obtained through convolution. In DPCNN, by continuously deepening the network, long-distance text dependencies can be extracted and without increasing the computational cost too much, the best accuracy can be obtained by increasing the network depth.

To capture the long-distance dependencies in sequence, they proposed a method that downsampling with the number of feature maps fixed, which is showed in Figure 5. They chose an appropriate two-layer equal-width convolution to increase the richness of the representation of lexical embedding then downsampling. After each convolution block, they use a fix size and stride for maxpooling. The length of the sequence is compressed to half of the original length. The text segment it can perceive is twice as long as before. In order to reduce the calculation time, they fixed the number of feature maps. Whenever a fixed size and stride layer is used for maxpooling, the calculation time of each convolutional layer is halved (the data size is halved), thus forming a pyramid.

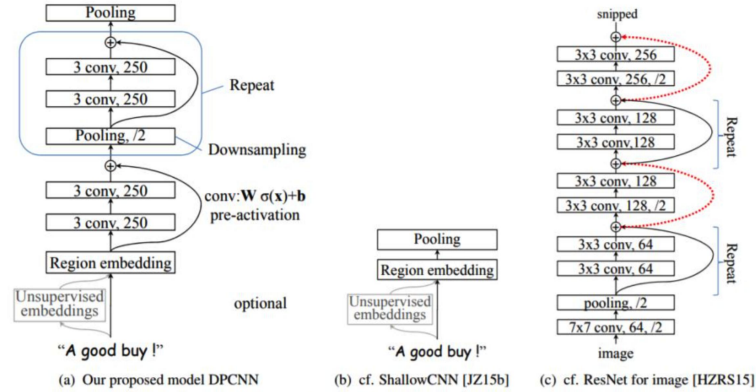


Figure 5. DPCNN architecture

Through previous method, DPCNN can capture longer dependencies in the sequence, but deeper network accompanied with the problem of vanishing gradient. Similar to ResNet, they proposed to use addition of shortcut connections for identity mapping, which make the training of deep networks possible.

BERT: BERT[13] was proposed to improve the Word2Vec representation, the backbone of BERT is Transformer which we have analyzed in the previous section. The essence of BERT is to learn a good feature representation for words by running a self-supervised learning method on the basis of massive corpus. The so-called self-supervised learning refers to supervised learning that runs on data that is not manually labeled. In future specific NLP tasks, researchers can directly use the feature representation of BERT as the word embedding feature of the task. Therefore, BERT provides a model for transfer learning of other tasks, which can be used as a feature extractor after being fine-tuned or fixed according to the task.

The input embedding of BERT is the sum of 3 embedding features, namely WordPiece Embedding, Position Embedding and Segment Embedding. WordPiece refers to dividing a word into a set of limited common sub-word units, which can be used in the validity of the word and the character. There is a compromise between flexibility. Position embedding refers to encoding the position information of a word into a feature vector, which is also widely used in Transformer architecture. Position embedding is a crucial part of introducing the position relationship of words into the model. Segment Embedding is used to distinguish two sentences, such as whether B is the following of A (dialogue scene, question-and-answer scene, etc.).

BERT is a multi-task model, and its task is composed of two self-supervised tasks, namely Mask Language Model (MLM) and Next Sentence Prediction (NSP). MLM refers to masking some words from the input expectation during training, and then predicting the word through the context. NSP is used to determine whether sentence B is the following of sentence A. If yes, output "IsNext", otherwise output "NotNext".

After training the BERT on a large number of corpus, it can be applied to various tasks of NLP. For other tasks, we can also make corresponding predictions based on the output information of BERT. We only need to add an output layer on the basis of BERT to complete the fine-tuning of specific tasks. In Figure 5. We show the pre-training and fine-tuning procedures for BERT.

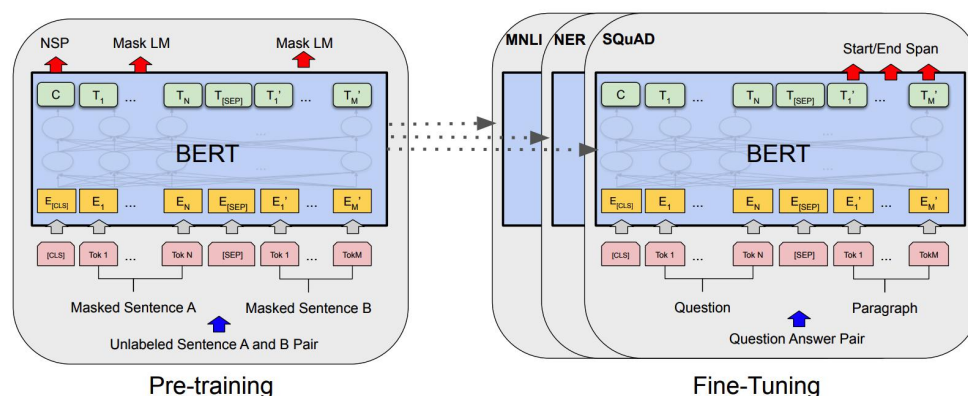


Figure 6. pre-training and fine-tuning procedures for BERT

There are several characteristics of BERT model. BERT Use Transformer as the main framework of the algorithm since Transformer can more thoroughly capture the two-way relationship in the sentence; BERT use multi-task training objectives by MLM and NSP; BERT use of more powerful machines to train larger-scale data and made the results of BERT reach a new height.

The BERT algorithm still has a lot of room for optimization. For example, how to make the model have the ability to capture the Token sequence relationship in Transformer, instead of simply relying on position embedding. Another problem of BERT is that, BERT contains a huge amount of parameters to optimize, which makes it hard to use, especially the training process of BERT is basically impossible to complete for normal researchers with current computing resources.

XLNet: XLNet[14] is a generalized autoregressive language model based on Transformer-XL [15]. In their work, they summarize the disadvantages of Transformers: (1) The maximum dependent distance between characters is limited by the length of the input. (2) When

the length of the input text exceeds 512 characters, each segment is trained separately from the beginning, which reduces the training efficiency and affects the performance of the model. Aiming at the above two shortcomings, Transformer-XL introduced two solutions: Division Recurrence Mechanism and Relative Positional Encoding. Transformer-XL tests faster and can capture longer context lengths. Unsupervised representation learning has achieved great success in the field of NLP. Under this concept, many researchers have explored different unsupervised pre-training goals. Autoregressive language modeling and auto-encoding language are the two most successful pre-training goals. XLNet is a generalized autoregressive method that integrates autoregressive and autoencoding. XLNet does not use the fixed forward or backward factorization order in the traditional autoregressive model, but uses a random arrangement of natural language to predict the words that may appear in a certain position. This method not only makes every position in the sentence can learn contextual information from all locations, and it can also construct bidirectional semantics to better acquire contextual semantics. Since XLNet uses Transformer-XL, the performance of the model is better, especially in tasks that include long text sequences. After the language model is trained through XLNet, it can be used for downstream related tasks, such as reading comprehension. The results obtained based on XLNet have far exceeded the human level, and excellent results have been achieved in tasks such as text classification and machine translation.

3. Conclusion

In this assignment, we review the deep sequence models in the domain of NLP, we give a brief description of each sequence model and critically analyze their strengths and shortcomings, besides, we evaluate some sequence models used in landmark papers of NLP, especially some important architectural design and modification of current mainstream sequence model. Until now, Transformers have fully exceeded other sequence model and become the most preferred choice for tasks dealing with time series data, due to its parallel computing capabilities and stronger sequential dependence modeling capabilities. Furthermore, the success of Transformers in NLP tasks have enlighten researchers in other domains such as Computer Vision(CV). Various kinds of Transformers have been proposed to handle tasks in CV such as object detection, video prediction etc. so far. It can be seen with the development of modern computation capability, more and more powerful sequence model will be proposed and improve the computer's ability to analyze and process time series data.

References

- [1] Russell, S. and Norvig, P. (2009) Artificial Intelligence: A Modern Approach (3rd edition), Prentice Hall
- [2] Elman, Jeffrey L. "Finding structure in time." *Cognitive science* 14.2 (1990): 179-211.
- [3] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961
- [4] Goodfellow, I; Bengio, Y; Courville, A; Bach, F (2017), Deep Learning, MIT Press
- [5] Hochreiter, S., & Schmidhuber, J. (1997). "Long short-term memory". *Neural computation*, 9(8), 1735-1780.
- [6] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016 pp. 770-778.
- [7] Huang, Zhiheng, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging." *arXiv preprint arXiv:1508.01991* (2015).
- [8] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).
- [9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
- [10] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. "End-to-end object detection with transformers". In *European Conference on Computer Vision (ECCV)*, 2020 pp. 213-229.
- [11] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Houlsby, N. "An image is worth 16x16 words: Transformers for image recognition at scale". *arXiv preprint arXiv:2010.11929*(2020).
- [12] Johnson, Rie, and Tong Zhang. "Deep pyramid convolutional neural networks for text categorization." *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers). 2017.
- [13] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
- [14] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. "Xlnet: Generalized autoregressive pretraining for language understanding." *arXiv preprint arXiv:1906.08237* (2019).
- [15] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. "Transformer-xl: Attentive language models beyond a fixed-length context." *arXiv preprint arXiv:1901.02860* (2019).