How I Did It

Victor von Frankenstein

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2018

Abstract

This doctoral thesis will present the results of my work into the reanimation of lifeless human tissues.

Acknowledgements

Many thanks to my mummy for the numerous packed lunches; and of course to Igor, my faithful lab assistant.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Victor von Frankenstein*)

Table of Contents

Chapter 1

Introduction

In this chapter, we will first briefly introduce the Recurrent Neural Network (RNN) and the branch prediction, which reflect the motivation of this project. Then we will discuss the objectives and contribution of the project. Finally, we also describe the structure of this document.

1.1 Motivation of the Project

In the area of Machine Learning, RNN is a type of neural network, which is good at capturing the tenporal features of sequential data. Therefore, RNN is widely used to predict and label sequential data in the area of Natural Language Processing(NLP), for example recognising sequential speech data (Graves, Mohamed, & Hinton, 2013) and language modeling (Mikolov, Karafiát, Burget, Černockỳ, & Khudanpur, 2010). Specifically, different types of RNN were used in these works, the most popular types are the Long Short-Term Memory network (LSTM) (Gers, Schmidhuber, & Cummins, 1999), bidirectional RNN (Graves et al., 2013) and the Gated Recurrent Unit network (GRU) (Cho et al., 2014).

Furthermore, a branch predictor predicts the direction of a branch given the program history. A mispredicted branch could cause the waste of program work (Michaud, Seznec, & Uhlig, 1996). Therefore, it is important to improve the accuracy of the branch prediction. A higher prediction accuracy could offset the misprediction penalties and improve the performance of the processor (Lee, Malishevsky, Beck, Schmid,

& Landry, n.d.). The state-of-the-art branch predictor uses the partial matching compression algorithm and achieves the accuracy of 99% (Seznec, 2006).

Finally, because program branches are time-based, branch prediction could benefit from the usage of RNN. This project aims at predicting branch outcomes given the branch history using RNN. The primary purpose of this project is to build the RNN branch predictor, which is a new attempt in the area of branch prediction. Moreover, to make the RNN predictor more competitive with the state-of-the-art brach predictors, different RNN types and ML techniques are used, which will be described in subsequent chapters.

1.2 Objective and Contribution of the Project

The primary objective of the project is to build an RNN branch predictor, which could predict the future outcome of a branch given the branch history. More specifically, instead of building a predictor whose performance exceeds the existing best predictor, we focus on attempting different techniques in ML and find a relatively better model configuration. Besides, the second objective is exploring the influence of varying branch history length, and various training set size has on the model's performance.

Our contributions could be...

1.3 Structure of the Dissertation

Chapter 2

Background

In this chapter, we will provide the basic knowledge of the techniques used in this project, including the RNN, the branch prediction, the dataset, and the methods used during training. We will first present the basic concept of each technique, and then the relevant literature will be provided.

2.1 Recurrent Neural Network

2.1.1 Basic Knowledge

RNN is a type of neural network, different from the feed-forward neural network, RNN could remember the information from the previous input data, which benefits from the recurrent pattern inside the RNN units. Figure 2.1 shows the structure of the vanilla RNN and the unrolling version. We could see that the output o_t depends on both the input at time t (x_t) and the previous hidden state s_{t-1}. The output at time t could be calculated by the euations below, where f could be a nonlinear function.

$$\begin{aligned} s_t &= f(Ux_t + Ws_{t-1}) \\ o_t &= softmax(Vs_t) \end{aligned} \tag{2.1}$$

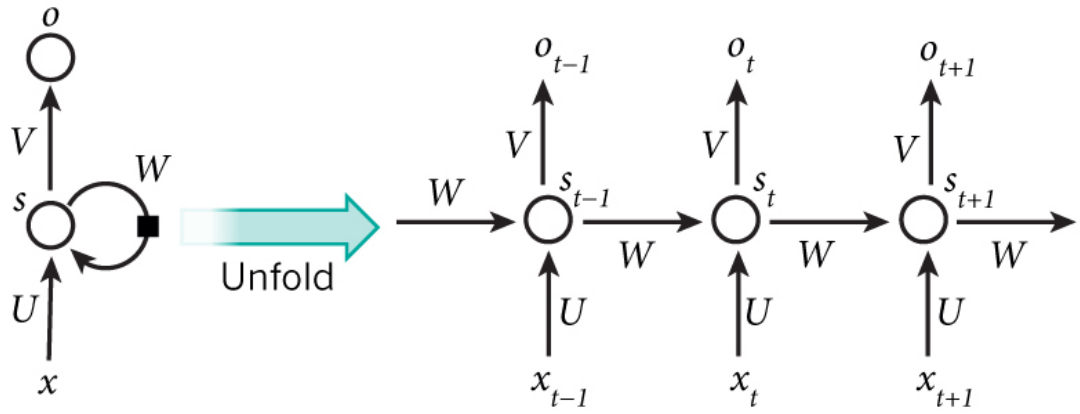
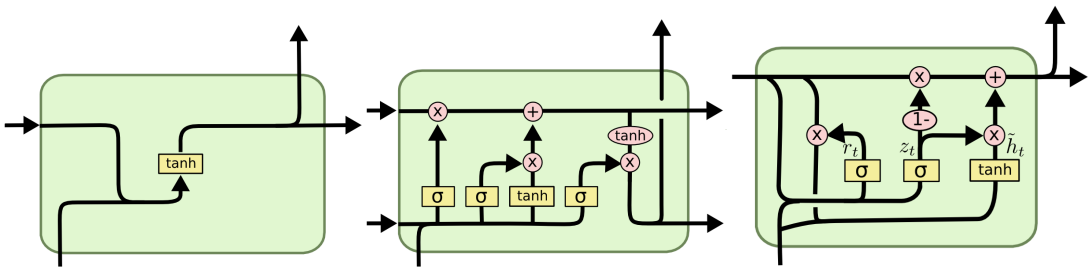


Figure 2.1: Unrolling an RNN unit. Left is the original RNN, right is the unrolled version. x_t is the input at time t, s_t is the hidden state of the RNN at time t, o_t is the output of the RNN at time t. W and V are the weight matrix.

Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

There are three popular types of RNN units, including the vanilla RNN, the LSTM (Hochreiter & Schmidhuber, 1997), and the GRU (Cho et al., 2014). The main difference among these three types of RNNs is the choice of the function f in the Equation 2.1. As shown in Figure 2.2, in the vanilla RNN unit, the f is a simple *tanh* function. While, in the LSTM and GRU, the f is a combination of several functions. The complex structures inside the LSTM and GRU unit could solve the long-term dependency problems caused by the simple *tanh* function in the vanilla RNN unit (Hochreiter & Schmidhuber, 1997).



(a) Vanilla RNN unit. (b) LSTM unit. (c) GRU unit.

Figure 2.2: Three types of RNN units.

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Bidirectional RNNs are similar to the 2-layered RNNs, except that the first RNN layer recieves the input sequence word by work from left to right, while the second layer gets the input sequence from right to left (Schuster & Paliwal, 1997). Therefore, bidirectional RNNs could capture the information before and after the current input data. The RNN unit inside the bidirectional RNNs could be the vanilla RNN unit, the LSTM unit, or the GRU.

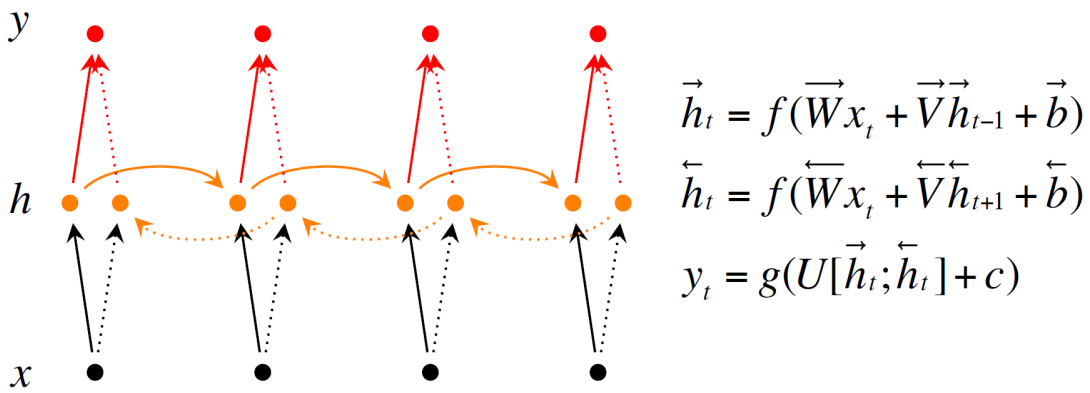


Figure 2.3: The bidirectional RNN.

Source: http://web.stanford.edu/class/cs224n/lectures/lecture9.pdf

attention

2.1.2 Relevant Literature

2.2 Branch Prediction

2.2.1 Basic Knowledge

2.2.2 Relevant Literature

2.3 Dataset

Introduce the branch prediction championship competition and the related dataset.

2.4 Training Procedure

This part introduce the metrics used during training, like dropout, adam optimization, learning rate ...

Chapter 3

Inplementation

In this chapter, we will provide the experimental inlementation in details, including the data processing, the default architecture of the model, and the detailed experimental design.

3.1 Data Preprocess

3.2 Model Architecture

3.3 Experiment Design

We designed four group of experiments. In each group, we fixed the

3.3.1 Varying History Length

3.3.2 Varying Training Set Size

3.3.3 Varying Network Types

3.3.4 Hyperparameters

Chapter 4

Results

In this chapter, we will show the experiment results of different models. We evaluate the performance of the models based on the prediction accuraccy on the test set. We fixed the parameters and change the key variables in each group of experiments.

4.1 Results Based on Different History Size

4.2 Results Based on Different Training Set Size

4.3 Results Based on Different Network Type

4.4 Results after Parameter Optimization

References

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with lstm.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on* (pp. 6645–6649).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Lee, B., Malishevsky, A., Beck, D., Schmid, A., & Landry, E. (n.d.). Dynamic branch prediction. *Oregon State University*.

Michaud, P., Seznec, A., & Uhlig, R. (1996). *Skewed branch predictors* (Unpublished doctoral dissertation). INRIA.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association.*

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

Seznec, A. (2006). A case for (partially)-tagged geometric history length predictors. *Journal of InstructionLevel Parallelism*.