Thesis for Master's Degree

Relation-level Regularization for Recurrent Neural Networks

Juhyeon Nam

AI Graduate School

Gwangju Institute of Science and Technology

2023

석사학위논문

순환신경망에서의 관계 단위 정규화

남주현

인공지능대학원

광주과학기술원

2023

Relation-level Regularization for Recurrent Neural Networks

Advisor: Kangil Kim

by

Juhyeon Nam

AI Graduate School

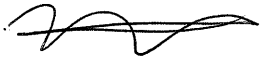
Gwangju Institute of Science and Technology

A thesis submitted to the faculty of the Gwangju Institute of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the AI Graduate School

Gwangju, Republic of Korea

June 09, 2022

Approved by



Professor Kangil Kim

Committee Chair

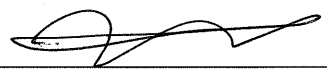
Relation-level Regularization for Recurrent Neural Networks

Juhyeon Nam

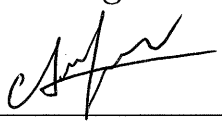
Accepted in partial fulfillment of the requirements for

the degree of Master of Science

June 09, 2022

Committee Chair 

Prof. Kangil Kim

Committee Member 

Prof. Chang Wook Ahn

Committee Member 

Prof. Ue-Hwan Kim

Dedicated to all my loved ones.

MS/AI 20204102 Juhyeon Nam. Relation-level Regularization for Recurrent Neural Networks. AI Graduate School. 2023. 31p. Advisor: Prof. Kangil Kim.

Abstract

In modern deep learning, there are two mainstream architectures dealing with sequential data: Transformers and Recurrent Neural Networks. Despite the remarkable success of transformers, it suffers from long-distance forecasting partly due to the complex, non-deterministic temporal dynamics across time steps. This limitation of the transformer is crucial, especially because of that the characteristic of sequential data consists of various relations. And here, we use the term "relation" as a set of input and output pairs in every time step having a similar relationship. And those relations repeatedly occur in sequential data.

In contrast, RNNs can handle longer sequences without increasing the model size due to their recursive nature. It is also intuitively obvious that the recurrent nature of RNNs would learn those relations much better than transformers. Thus, it is reasonable to use RNNs over transformers on sequential data, but RNNs work poorly compared to transformers.

In this work, we pointed out that the regularizations on RNNs need to make better use of these advantages of RNNs to learn recurrent relations in sequential data.

We aim to strengthen these advantages and alleviate the disadvantages of RNNs. We hypothesized that RNNs' underestimated performance is partly due to their simple architecture and regularization methods.

We proposed novel global and local regularization methods which make it possible to separately regularize the cardinality of relations and the complexity of each relation. The proposed regularization methods can be applied to any RNN-based model with a simple extension of that architecture. This extension of the model explicitly distinguishes recursively occurring relations and learns to choose relation adaptively at every inference time step.

To analyze the proposed extended model and regularization methods, we designed a toy task with Binary Counter datasets. The experimental results showed that the proposed extended architecture, proto-LSTM cells with relation loader, can solely improve the robustness over hyper-parameter sweeping in longer prediction. The performance on longer prediction of the proto-LSTMs with relation loader trained with proposed regularization methods outperformed transformer and other RNN-based baselines in terms of both accuracy and robustness.

MS/AI 남주현. 순환신경망에서의 관계 단위 정규화. 인공지능대학원. 2023. 31p.
20204102 지도교수: 김강일.

국 문 요 약

현대 딥러닝에서 시퀀스 데이터를 다루는 두 가지 주요 아키텍처에는 순환신경망(Recurrent Neural Network)과 트랜스포머(Transformer)가 있다. 그중 트랜스포머 기반의 모델들이 순환신경망 기반의 모델에 비해서 대부분의 데이터셋에 대해 놀라운 성공을 거두고 있지만 먼 타임 스텝에서의 예측에 어려움을 겪는다. 이는 트랜스포머 구조가 시간 정보를 복잡하고 비결정론적으로 학습하기 때문인데, 시퀀스 데이터는 타임 스텝에 걸쳐 다양한 관계(Relation)로 구성되어있기 때문에 이런 한계는 특히 결정적이다. 여기서 관계란, 각 타임 스텝에서 입력 및 출력 쌍의 집합을 의미하는 것으로 시퀀스 데이터에서 반복적으로 등장하는 일종의 규칙을 말한다.

반면, RNN은 재귀적 특성으로 인해 모델 크기를 늘리지 않고도 더 긴 시퀀스를 처리할 수 있다. 또한 RNN의 이러한 특성 때문에 트랜스포머보다 반복적으로 등장하는 규칙을 훨씬 더 잘 학습하리라는 것은 직관적으로 명백하다. 따라서 시퀀스 데이터에 대해서는 트랜스포머보다 순환신경망을 사용하는 것이 합리적일 것으로 보이지만, 실제로는 트랜스포머가 더 좋은 성능을 보여왔다.

본 논문에서는 시퀀스 데이터에서 반복적인 관계를 학습하기 위해 순환신경망의 이러한 장점을 더 잘 활용할 필요가 있다고 지적하고, 순환신경망의 단점을 완화하는 것을

목표로 한다. 순환신경망의 과소평가 된 성능은 트랜스포머에 비해 간단한 모델 구조와 잘못 사용되고 있는 정규화 방법 때문이라고 가정하고, 이에 대한 해결책으로 관계의 카디널리티와 각 관계의 복잡성을 별도로 정규화할 수 있는 새로운 글로벌 및 로컬 정규화 방법을 제안한다. 제안된 정규화 방법은 모델 구조의 간단한 확장으로 모든 순환신경망 기반 모델에 적용할 수 있다. 이러한 모델의 확장을 통해 반복적으로 발생하는 관계를 명시적으로 구별함으로써, 모든 추론 타임 스텝에서 모델이 직접 적절한 관계를 선택하는 법을 학습할 수 있도록 한다.

제안된 확장 모델 및 정규화 방법을 분석하기 위해 이진 카운터 데이터 세트를 사용한 실험을 설계했다. 실험 결과를 통해 본 논문에서 제안한 모델 구조 확장 방법을 사용하는 것만으로도 더 먼 타임 스텝 예측의 안정성을 확보하는 것을 확인했다. 또 제안한 정규화 방법으로 해당 확장 모델을 훈련하면 먼 타임 스텝 예측 시 정확성과 안정성 모두의 측면에서 트랜스포머나 순환신경망 기반의 베이스라인 성능을 개선함을 확인했다.

Contents

Abstract (English) i

Abstract (Korean) iii

List of Contents v

List of Tables vii

List of Figures viii

List of Algorithms ix

1 Introduction 1

2 Related Works and Notations 4

2.1 Relation . . . 4

2.2 Recurrent Neural Networks(RNNs) . . . 4

2.2.1 Recurrent Neural Network(RNN) and Basic Notations . . . 4

2.2.2 Long-Short Term Memory(LSTM) . . . 5

2.3 Regularization Methods for RNNs and Its Limitations . . . 7

2.4 Relational Models . . . 8

2.4.1 Ensemble Method . . . 8

2.4.2 Mixture of Experts . . . 9

2.4.3 Multiple-weight Recurrent Neural Network . . . 9

2.4.4 Relational Memory Core . . . 10

3 Approach 12

3.1 Multiple Proto-LSTM Cells . . . 12

3.2 Relation Loader . . . 13

3.3 Local and Global regularization . . . 14

3.3.1 Cell Noise Injection . . . 14

3.3.2 Cell and Non-Cell L2 regularization . . . 14

4 Experiments 18

4.1 Experimental Settings . . . 18

4.2 Quantitative Results . . . 19

4.3 Qualitative Analysis 21
4.3.1 Analysis on Regularization Methods 21
4.3.2 Proto-LSTM Cell loading distribution analysis 23

5 Conclusion **25**

Summary **26**

References **28**

Acknowledgements **32**

List of Tables

4.1 Hyper-parameters of the grid search on Binary counter datasets with LSTM. A set of values achieving the best performance is bolded. 19

4.2 Hyper-parameters of the grid search on Binary counter datasets with RMC. A set of values achieving the best performance is bolded. 19

4.3 Hyper-parameters of the grid search on Binary counter datasets with Transformer. A set of values achieving the best performance is bolded. . 20

4.4 Hyper-parameters of the grid search on Binary counter datasets with Proto-LSTMs with relation loader(Ours). A set of values achieving the best performance is bolded. 20

4.5 Evaluation result of each model with regularization. Evaluation has been conducted with the increasing number of bits in Binary Counter Datasets. The best sentence accuracy from each dataset is bolded. . . . 22

List of Figures

3.1 Architecture of 4 proto-LSTM Cells with relation loader. 15

3.2 Regularization methods for 4 proto-LSTM Cells with relation loader explained. 17

4.1 Performance of LSTM(baseline) and 4 Proto-LSTMs with Loading layer. Both use L2 regularizer. 20

4.2 Performance of 4 Proto-LSTMs with Loading layer with different regularization methods 23

4.3 Proto-LSTM cells loading probability distributions at each time step in inference 24

4.4 Proto-LSTM cells loading probability distributions at every epoch of training 24

List of Algorithms

1 Algorithmic Procedure for Binary Counter 19

Chapter 1

Introduction

In modern deep learning, there are two mainstream architectures dealing with sequential data: Transformers and Recurrent Neural Networks. In general, transformers outperform RNNs in a multitude of tasks dealing with sequential data. It is partly due to the ability to learn the complex context in data by integrating information of all time steps in depth by employing multiple layers of self-attention mechanism.

Despite the remarkable success of transformers, it suffers from long-distance forecasting partly due to the complex, non-deterministic temporal dynamics across time steps [1]. In addition, transformer-based models are unable to process long sequences due to their self-attentive mechanism, which scales quadratically with the sequence length [2, 3, 4, 5]. Consequentially, transformers lack the inductive ability to handle sequences longer than any sequences seen in the training time[6]. This limitation of the transformer is crucial, especially because of that the characteristic of sequential data consists of various relations. And here, we use the term "relation" as a set of input and output pairs in every time-step having a similar relationship. And those relations repeatedly occur in sequential data.

If those recurrent relations are learned properly, the model will not be dependent on the length of the sequence. And the performance of the model will be preserved regardless of the sequence length since the sequence can be simply interpreted by some

repeated relations at every time step. However, the limitation of transformers which is the lack of inductive ability on the length of a sequence suggests that the transformer cannot catch that recurrent relation properly.

In contrast, RNNs[7] can handle longer sequences without increasing the size of the model due to its recursive nature. It learns to represent observations from previous time steps as a compressed hidden vector, which is the bottleneck of RNNs. The long-short term memory(LSTM)[8] alleviates this bottleneck by utilizing gates and allowing the model to determine whether the information is to be stored or to be forgotten. Those variants of RNN-based architectures naturally contain the inductive bias in sequential information by loading input sequences step-by-step. It is also intuitively obvious that the recurrent nature of RNNs would learn those relations much better than transformers. Thus, it is reasonable to use RNNs over transformers on sequential data, but RNNs work poorly compared to transformers.

In this work, we point out that the conventional use of RNNs and regularization are not making good use of these advantages of RNNs to learn recurrent relations in sequential data. We aim to strengthen the advantages and alleviate the disadvantages of RNNs. We hypothesize that RNNs' underestimated performance is partly due to their simple architecture and regularization methods. Those are hindering RNNs from learning and regularizing repeatedly occurring relations separately by regularizing those various, yet simple, recurrent relations to be a single complex relation. In other words, existing regularization methods on RNN suppress the cardinality of relations and the complexity of each relation at the same time. Therefore, we propose novel global and

local regularization methods which make it possible to separately regularize the cardinality of relations and the complexity of each relation. The proposed regularization methods can be applied to any RNN-based model with a simple extension of that architecture. This extension of the model explicitly distinguishes recursively occurring relations and learns to choose relation adaptively at every inference time step.

Chapter 2

Related Works and Notations

2.1 Relation

First, we define the concept of "relation". In this work, a relation is defined by a set of input and output pairs having similar relationships. We use the term recurrent relation as the meaning of a relation occurring across time steps in sequential data. It is the opposite of recursive relation, which means a repeated structural relation such as a parse tree.

2.2 Recurrent Neural Networks(RNNs)

2.2.1 Recurrent Neural Network(RNN) and Basic Notations

One of the most prevailing model architectures to deal with sequential data is the recurrent neural network, which was first introduced in 1982 by Hopfield.[9] Recurrent neural network (RNN) takes the inputs explicitly sequentially and passes the internal state through time to utilize the information from arbitrarily long previous time steps. At every time step, RNN takes the input of the current time step and internal state from the previous time step. An input sequence can be denoted $(x^{(1)}, x^{(2)}, ..., x^{(T)})$ where the length of the sequence is T. We use superscripts with parentheses for time and each input $x^{(t)}$ at time step t is a real-valued vector. In the same manner, a target sequence

can be denoted as $(y^{(1)}, y^{(2)}, ..., y^{(T)})$. However, the length of the input sequence and the target sequence do not have to be the same, which means that the target does not have to be sequential data. The most basic RNN architecture can be described as the equation below.

$$\mathbf{h^{(t)}} = \sigma(W_{xh}\mathbf{x^{(t)}} + W_{hh}\mathbf{h^{(t-1)}} + \mathbf{b_h} \tag{2.1}$$

At time step t, the internal state, which is commonly known as the hidden state, $\mathbf{h^{(t)}}$ is computed by a fully-connected layer taking the concatenated result of the input $x^{(t)}$ and the previous hidden state $h^{(t-1)}$. W_{xh} and W_{hh} are weights of a fully-connected layer between an input and a hidden layer, and $\mathbf{b_h}$ is a bias.

$$\hat{y}^{(t)} = softmax(W_{ho}\mathbf{h^{(t)}} + \mathbf{b_o}) \tag{2.2}$$

The above equation computes a prediction $\mathbf{\hat{y}^t}$ from a hidden state $\mathbf{h^{(t)}}$. W_{ho} is a weight between the hidden layer and output of time step t, and b_o is a bias.

2.2.2 Long-Short Term Memory(LSTM)

The next prominent improvement of RNN architecture, Long-Short Term Memory (LSTM)[8], resolves the short dependency problem and vanishing gradient problem in the RNN by introducing the gating mechanism. The gating mechanism consists of three different gates; input gate, forget gate, and output gate. In addition to the hidden vector, LSTM utilizes a state vector that allows the model to determine how long a

particular signal will convey through time. The detailed model architecture is explained by the equations below.

$$\begin{aligned}
\mathbf{i}^{(\mathbf{t})} &= \sigma(W_{xi}\mathbf{x}^{(\mathbf{t})} + W_{hi}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b_i}) \\
\mathbf{f}^{(\mathbf{t})} &= \sigma(W_{xf}\mathbf{x}^{(\mathbf{t})} + W_{hf}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b_f}) \\
\mathbf{o}^{(\mathbf{t})} &= \sigma(W_{xo}\mathbf{x}^{(\mathbf{t})} + W_{ho}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b_o})
\end{aligned} \tag{2.3}$$

$\mathbf{i}^{(\mathbf{t})}$, $\mathbf{f}^{(\mathbf{t})}$, and $\mathbf{o}^{(\mathbf{t})}$ are input gate, forget gate and output gate, respectively. Role of those gates can be explained by the next equation.

$$\begin{aligned}
\tilde{\mathbf{c}}^{(\mathbf{t})} &= tanh(W_{xg}\mathbf{x}^{(\mathbf{t})} + W_{hg}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b_g}) \\
\mathbf{c}^{(\mathbf{t})} &= \mathbf{f}^{(\mathbf{t})} * \mathbf{c}^{(\mathbf{t-1})} + \mathbf{i}^{(\mathbf{t})} * \tilde{\mathbf{c}}^{(\mathbf{t})} \\
\mathbf{h}^{(\mathbf{t})} &= \mathbf{o}^{(\mathbf{t})} * tanh(\mathbf{c}^{(\mathbf{t})})
\end{aligned} \tag{2.4}$$

First, $\tilde{\mathbf{c}}^{(\mathbf{t})}$ is computed by fully-connected layer followed by a $tanh$ activation function. Then, the forget gate $\mathbf{f}^{(\mathbf{t})}$ determines the remaining information from the previous cell state $\mathbf{c}^{(\mathbf{t-1})}$ and it allows the model to "forget" some information from the previous time steps. However, the input gate $\mathbf{i}^{(\mathbf{t})}$ decides the amount of preserving information from the computed input $\tilde{\mathbf{c}}^{(\mathbf{t})}$. Combining those two gates, the updated cell state $\mathbf{c}^{(\mathbf{t})}$ is calculated. The simpler version of LSTM is the Gated Recurrent Unit(GRU) which combines the input gate and the forget gate by assuming the resulting sum of those gates be one [10]. LSTM has a superior ability to learn long-range dependencies compared to simple RNNs.

2.3 Regularization Methods for RNNs and Its Limitations

Variants of recurrent neural networks suffer from exploding and vanishing gradients [11]. Regularization strategies such as dropout[12] and batch normalization[13] are proven to be useful in a feed-forward and convolutional neural network. However, naively applying those techniques to recurrent neural networks has not been successful[14]. Since naively applying existing regularization methods harms the long-term dependencies in RNNs and LSTMs, we need a different strategy for regularization, especially for RNNs and LSTMs.

L1 and L2 regularizations are the most standard regularization methods and are widely used to prevent overfitting. Both are implemented by the penalty term in the loss function. The first, L1 regularization, often called Lasso[15], encourages the sum of the absolute value of parameters in the model to decrease. It makes small weights even close to zero, therefore works as the feature selection [16]. As the name implies, the mathematical expression is the same as the L1 norm as follows. L is the loss function of the model.

$$w^*_{L_1} = arg\min_w L(y, \hat{y}|w) + \lambda \sum_{i=1}^{N} |w_i| \tag{2.5}$$

On the other hand, the objective of the L2 regularization is to minimize the sum of squared weights in the model [17]. As can be seen in the mathematical expression below, L2 regularization generally decreases the weights, unlike L1 regularization.

$$w^*_{L_2} = arg\min_w L(y, \hat{y}|w) + \lambda \sum_{i=1}^{N} w_i^2 \tag{2.6}$$

Due to the differences in behavioral characteristics of those regularizations, usage of L1 regularization makes the model notably insensitive to the presence of extraneous features, compared to L2 regularization [18]. This means that if several relations composite the training dataset, existing standard regularization will fail to learn those relations.

2.4 Relational Models

2.4.1 Ensemble Method

The ensemble method is combining the outputs from a multitude of models to get a more robust and accurate prediction. The strategy of combining outputs is the key to the ensemble method. The most typical ways of combining outputs are by weighted summation of the final logit vector and unweighted voting of the final output label. The reason why the ensemble method enhances the performance from the individual model's output is that the error in the individual model is uncorrelated to others. In other words, every model used for an ensemble makes different errors, and the most frequent prediction among those is highly likely to be the correct prediction.[19] Ensemble method can be viewed as the architecture combining multiple relations which are trained on multiple models. However, it does not guarantee each model to learn mutually exclusive relations.

2.4.2 Mixture of Experts

Mixture of experts[20] is similar to the ensemble method, but it trains individual models simultaneously. It divides a problem space into homogeneous regions.[21] Several works applying the mixture of experts to sequential architecture such as LSTM[22] and transformer[23] showed the improvement of the model performance. However, the regularization method for a mixture of experts based on RNN architecture has not been actively explored.

2.4.3 Multiple-weight Recurrent Neural Network

Multiple-weight recurrent neural network introduced novel pradigm of multi-weight recurrent neural network, mimicking human ability[24]. Authors pointed out that existing variants of recurrent neural networks are not built to deal with multi-faceted data, and suggest novel paradigm which is readily applicable to numerous RNN-family network architectures. The novel model allows multiple matrices to update state vector. The following equations describe a multiple-weight recurrent neural network:

$$
\begin{aligned}
\mathbf{i}^{(\mathbf{t})} &= \sigma(W_{xi}\mathbf{x}^{(\mathbf{t})} + W_{hi}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b_i}) \\
\mathbf{f}^{(\mathbf{t})} &= \sigma(W_{xf}\mathbf{x}^{(\mathbf{t})} + W_{hf}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b_f}) \\
\mathbf{o}^{(\mathbf{t})} &= \sigma(W_{xo}\mathbf{x}^{(\mathbf{t})} + W_{ho}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b_o}) \\
\tilde{\mathbf{c}}^{(\mathbf{t})_\mathbf{i}} &= tanh(W^i_{xg}\mathbf{x}^{(\mathbf{t})} + W^i_{hg}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b^i_g}) \\
\mathbf{c}^{(\mathbf{t})} &= \mathbf{f}^{(\mathbf{t})} * \mathbf{c}^{(\mathbf{t-1})} + \mathbf{i}^{(\mathbf{t})} * \sum_{i=1}^{K} p^i * \tilde{\mathbf{c}}^{(\mathbf{t})_\mathbf{i}} \\
\mathbf{h}^{(\mathbf{t})} &= \mathbf{o}^{(\mathbf{t})} * tanh(\mathbf{c}^{(\mathbf{t})})
\end{aligned}
\tag{2.7}
$$

The only difference from LSTM is the way to compute $c^{(t)}$, where K is the number of weight matrices. Weighted sum of K gated inputs, $\tilde{\mathbf{c}}^{(\mathbf{t})_\mathbf{i}}$, are used to update state vector. The weights of each gated input, p^i, are also learnt using input and the state vector.

$$
\begin{aligned}
\tilde{p}^i &= W^i_{xp}x + W^i_{cp}c + b^i_p \\
p^i &= \frac{exp(\tilde{p}^i)}{\sum_j exp(\tilde{p}^j)}
\end{aligned}
\tag{2.8}
$$

The weights go through a softmax normalization as the equation above, converting into probability-like value corresponding to each weight matrix. This allows the model to make decision for internal state vector based on multiple matrices at every time step.

2.4.4 Relational Memory Core

While multiple-weight recurrent neural network utilizes multiple weight matrices, Relational Memory Core(RMC) exploits external memory to make the state vector

self-attentive[25]. Memory-based neural networks model temporal data by leveraging an ability to remember information for long periods. However, The authors of this paper argue that It is unclear, whether they also have an ability to perform complex relational reasoning with the information they remember. They first confirm their intuitions that standard memory architectures may struggle at tasks involving relational reasoning. They then improve upon these shortfalls by using a new memory module – so called, a Relational Memory Core (RMC) - which employs multi-head dot product attention to allow memories to interact. Since they also call out the relational learning problem in recurrent structure, we compared performance with our model in experiments

Chapter 3

Approach

3.1 Multiple Proto-LSTM Cells

A basic building block of our model is LSTM. Our model consists of multiple LSTM cells which tend to learn recursively occurring relations in sequential data separately. We named those LSTM cells "Proto-LSTM Cells" since each of those is a prototype of LSTM representing different relations. Each proto-LSTM cell can be viewed as an expert in the Mixture of experts architecture[20]. The following equations describe how the proto-LSTM cells work:

$$
\begin{aligned}
\mathbf{i^{(t)}} &= \sigma((\sum_{k=1}^{K} p^k * W_{xi}^k)\mathbf{x^{(t)}} + (\sum_{k=1}^{K} p^k * W_{hi}^k)\mathbf{h^{(t-1)}} + \sum_{k=1}^{K} p^k * \mathbf{b_i^k}) \\
\mathbf{f^{(t)}} &= \sigma((\sum_{k=1}^{K} p^k * W_{xf}^k)\mathbf{x^{(t)}} + (\sum_{k=1}^{K} p^k * W_{hf}^k)\mathbf{h^{(t-1)}} + \sum_{k=1}^{K} p^k * \mathbf{b_f^k}) \\
\mathbf{o^{(t)}} &= \sigma((\sum_{k=1}^{K} p^k * W_{xo}^k)\mathbf{x^{(t)}} + (\sum_{k=1}^{K} p^k * W_{ho}^k)\mathbf{h^{(t-1)}} + \sum_{k=1}^{K} p^k * \mathbf{b_o^k}) \\
\mathbf{\tilde{c}^{(t)}} &= tanh((\sum_{k=1}^{K} p^k * W_{xg}^k)\mathbf{x^{(t)}} + (\sum_{k=1}^{K} p^k * W_{hg}^k)\mathbf{h^{(t-1)}} + \sum_{k=1}^{K} p^k * \mathbf{b_g^k})
\end{aligned}
\tag{3.1}
$$

$$
\mathbf{c^{(t)}} = \mathbf{f^{(t)}} * \mathbf{c^{(t-1)}} + \mathbf{i^{(t)}} * \mathbf{\tilde{c}^{(t)}}
$$

$$
\mathbf{h^{(t)}} = \mathbf{o^{(t)}} * tanh(\mathbf{c^{(t)}})
$$

The above equations are different from the equations of an LSTM only in a way by replacing all weight and bias terms as the weighted summation of each proto-LSTM

cell's weights and biases, where K represents the number of proto-LSTM cells. The weighted summation is done as the equation below.

$$\begin{aligned} W &= \sum_{k=1}^{K} p^k * W^k \\ \mathbf{b} &= \sum_{k=1}^{K} p^k * \mathbf{b^k} \end{aligned} \tag{3.2}$$

This architecture might look similar to the multiple-weights recurrent neural network[24]. The proto-LSTM cell utilizes whole parameters in each cell, such as W^k and $\mathbf{b^k}$, and uses the weighted summation of those as a single loaded LSTM cell at each time step. However, multiple-weights recurrent neural network only employs a weighted sum of state vectors, not the model's parameters.

3.2 Relation Loader

The weight p^k is a loading probability of kth proto-LSTM cell. The model is also trained to compute the loading probability from input $x^{(t)}$ and the hidden vector $h^{(t)}$. The loading probability is the same as an attention score for each proto-LSTM cell.

$$\tilde{p}^k = W_x^k x^{(t)} + W_h^k h^{(t-1)} + b \tag{3.3}$$

As the above equation, the Relation Loader is a fully-connected layer taking concatenated vector of the input $x^{(t)}$ and the hidden vector $h^{(t)}$.

$$p^k = \frac{exp(\tilde{p}^k)}{\Sigma_i exp(\tilde{p}^i)} \tag{3.4}$$

The weight of each proto-LSTM cell, $\tilde{p}^k$, is go through a softmax normalization and is converted into the loading probability of each proto-LSTM cell, p^k.

3.3 Local and Global regularization

The main reason for separating proto-LSTM cells by employing the Relation Loader is to separately handle the local and global regularity. The local regularity represents the complexity of individual relations which are trained separately over each proto-LSTM cell. And the global regularity represents the cardinality of the relations.

3.3.1 Cell Noise Injection

To regularize the complexity of each proto-LSTM cell, which is the local regularity, we used cell noise injection. Following equations describe how noise is injected into the ith proto-LSTM cell:

$$\begin{aligned} \tilde{W}^k &= W^k + \epsilon * \eta_w^k; \quad \eta_w^k \sim \mathcal{N}(0, {\sigma_w^k}^2) \\ \tilde{\mathbf{b}}^k &= \mathbf{b^k} + \epsilon * \eta_b^k; \quad \eta_b^k \sim \mathcal{N}(0, \sigma_b^{k^2}) \end{aligned} \tag{3.5}$$

Gaussian noise is added to the weights W^k and the bias $\mathbf{b^k}$ in each proto-LSTM cell. The noise is generated at every time-step for each proto-LSTM cell, with the same standard deviation as the weights, ${\sigma_w^k}^2$, or bias, $\sigma_b^{k^2}$, and zero means.

3.3.2 Cell and Non-Cell L2 regularization

L2 regularization term in general looks like the below:

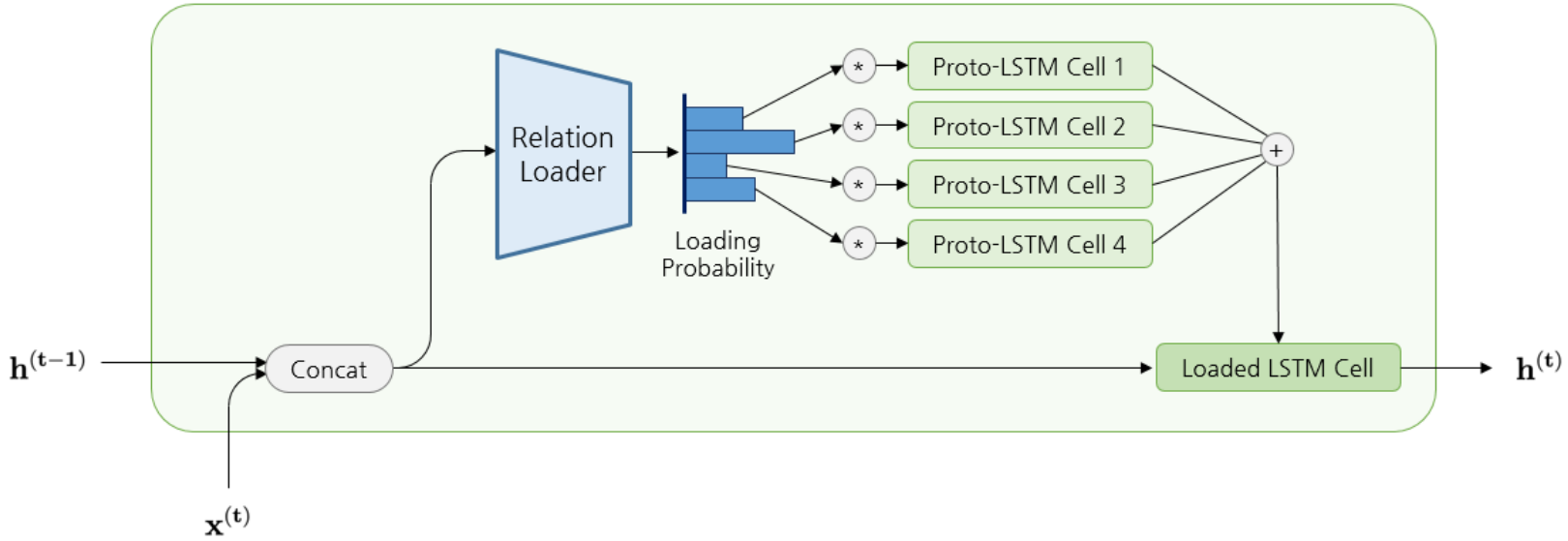


Figure 3.1: Architecture of 4 proto-LSTM Cells with relation loader.

$$\lambda \sum_{i=1}^{N} w_i^2 \tag{3.6}$$

There is N number of parameters in the model. This term regularizes all parameters in the model. However, the separated parameters over multiple proto-LSTM cells allow the model to regularize the complexity of learned relation in proto-LSTM cells apart from the whole model complexity as the following equation:

$$\begin{aligned} &\lambda_{cell} \sum_{i=1}^{N} \nVdash_{cell} * w_i^2 + \lambda_{non-cell} \sum_{i=1}^{N} \nVdash_{cell}^{-} * w_i^2; \\ \text{where,} \quad &\nVdash_{cell} = \begin{cases} 1, & \text{if x} \in \text{proto-LSTM cell parameters;} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{3.7}$$

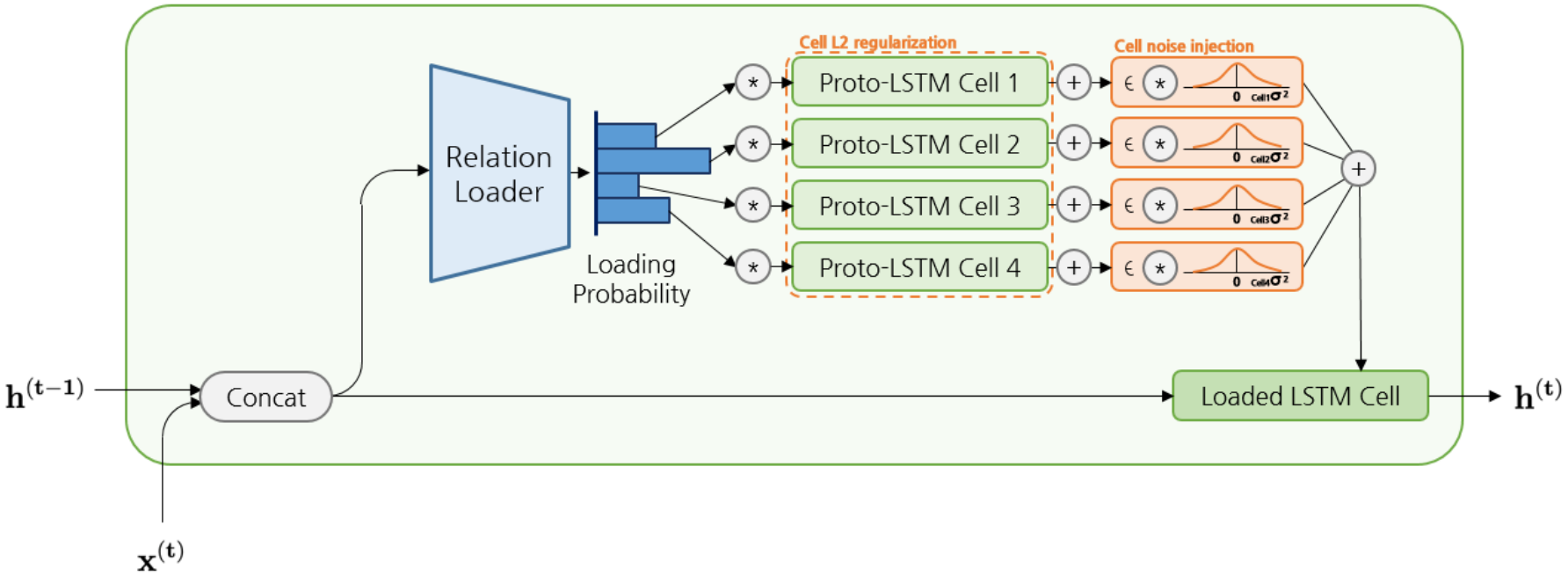


Figure 3.2: Regularization methods for 4 proto-LSTM Cells with relation loader explained.

Chapter 4

Experiments

4.1 Experimental Settings

We test our model on toy datasets which consist of sequences composed by recursively occurring simple relations. To confirm that the model is sufficiently generalized, the test set should tolerably variant from the training set but can be explained by the simple rules learned from the training set.

Therefore, we propose a toy dataset called Binary Counter Dataset, in which the input is a binary number and the output is also a binary number but it is increased by one from the input. Due to the simplicity of rules in binary incrementor, which is denoted in Algorithm 1, humans can easily learn the rules and can adopt those rules to different digits of binary numbers. Once the rules are learned, it is easy to extend them to longer digits. A human can surely extend and adapt those rules but the deep learning model often fails to learn a multitude of simple recurring rules due to the regularization.

In this paper, we have used 3 digits Binary Counter Dataset as the train set and 6, 8, 10, 12, 14, and 16 digits Binary Counter Datasets as the test sets. Because the increase in numbers begins with the least significant bits in the binary number, the data was generated by reversing the order of each digit.

Algorithm 1: Algorithmic Procedure for Binary Counter

Data: X = Input binary number with k bits
Result: $Y = i$ bits binary number increased by 1 from the input

```
Initialization: updating ← True;
for i=1:k do
    if updating then
        Y[i] ← ∼ X[i];
        if X[i] is 0 then
            updating ← False;
    else
        Y[i] ← X[i]
```

4.2 Quantitative Results

We have set the accuracies of single LSTM, RMC, and Transformer on the Binary Counter test set as baselines. We use L2 regularization with five different regularization parameters as shown in table 4.1, table 4.2, table 4.3, and table 4.4.

Hyper-parameter	Values
Hidden size	{6, **8**, 10}
Learning rate	{0.01, **0.05**}
L2 regularization (λ_{whole})	{0.0, 0.0001, **0.001**, 0.01, 0.1}

Table 4.1: Hyper-parameters of the grid search on Binary counter datasets with LSTM. A set of values achieving the best performance is bolded.

Hyper-parameter	Values
(Hidden size, Key size, # Attention heads)	{(6, 6, 1), (6, 3, 2), **(8, 8, 1)**, (8, 4, 2), (8, 2, 4), (10, 10, 1), (10, 5, 2), (10, 2, 5)}
Learning rate	{0.01, **0.05**}
L2 regularization (λ_{whole})	{0.0, 0.0001, **0.001**, 0.01, 0.1}

Table 4.2: Hyper-parameters of the grid search on Binary counter datasets with RMC. A set of values achieving the best performance is bolded.

Hyper-parameter	Values
(Hidden size, Key size, # Attention heads)	{(48, 6, 2), (64, 8, 2), (80, 10, 2), (72, 6, 3), (96, 8, 3), (120, 10, 3), (96, 6, 4), (128, 8, 4), **(160, 10, 4)**}
# Layers	{1, 2, **3**}
Learning rate	{**0.01**, 0.05}
L2 regularization (λ_{whole})	{0.0, 0.0001, 0.001, **0.01**, 0.1}

Table 4.3: Hyper-parameters of the grid search on Binary counter datasets with Transformer. A set of values achieving the best performance is bolded.

Hyper-parameter	Values
Hidden size	{6, **8**, 10}
# Proto-LSTMs	{2, **3**, 4, 6, 8, 10}
Learning rate	{0.01, **0.05**}
L2 regularization (λ_{cell})	{0.0, 0.0001, **0.001**, 0.01, 0.1}
L2 regularization ($\lambda_{non-cell}$)	{0.0, 0.0001, **0.001**, 0.01, 0.1}
Noise injection (ϵ)	{0.0, 0.001, 0.01, 0.1, **1.0**}

Table 4.4: Hyper-parameters of the grid search on Binary counter datasets with Proto-LSTMs with relation loader(Ours). A set of values achieving the best performance is bolded.

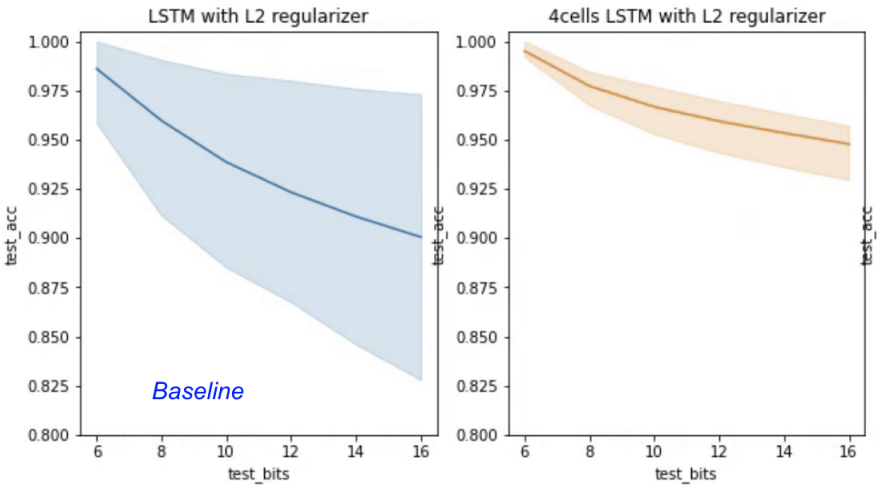


Figure 4.1: Performance of LSTM(baseline) and 4 Proto-LSTMs with Loading layer. Both use L2 regularizer.

The graphs in 4.1 compare performances on longer-range predictions of the LSTM(baseline) model and the model with 4 proto-LSTM cells. It shows that even only the extended architecture of the recurrent model we suggested without the regularization methods can improve the performance of longer-range predictions and its robustness on hyper-parameter sweeping.

Compared test accuracies with all baseline models are shown in table 4.5. The sequence-level accuracies of all models on Binary Counter Datasets with the gradually increasing number of bits are in this table 4.5. The last three rows are the results of our models with different combinations of regularization methods. The results show that all RNN-based models, LSTM, RMC, and our model with different regularization methods have monotonically decreasing performance for longer predictions. It is obvious since the prediction for the longer sequence is redundant with the shorter prediction. However, our model with the proposed regularization method consistently achieves better performances compared to all baselines. And the transformer shows notable behavior; its performance is inconsistent and even looks random for longer prediction. This can be interpreted as the limitation of the transformer, which is the lack of inductive ability to handle longer sequences.

4.3 Qualitative Analysis

4.3.1 Analysis on Regularization Methods

The graph in 4.2 shows the performances on longer-range predictions with different regularization methods. Graphs in the first row are using L2 regularization in a con-

Models	# bits in each Binary Counter Datasets					
	6 bits	8 bits	10 bits	12 bits	14 bits	16 bits
LSTM(with L2)	**1.0**	0.9972	0.9957	0.9951	0.9949	0.9948
RMC(with L2)	0.9788	0.9557	0.9365	0.9195	0.9048	0.8927
Transformer(with L2)	0.7667	0.2	0.5667	0.4	0.4667	0.7
Ours(with L2)	**1.0**	0.9931	0.9889	0.9872	0.9859	0.9848
Ours(with cell&non-cell L2)	**1.0**	0.9984	0.9963	0.9944	0.9925	0.9906
Ours(with both regularization)	**1.0**	**0.9986**	**0.9973**	**0.9964**	**0.9960**	**0.9959**

Table 4.5: Evaluation result of each model with regularization. Evaluation has been conducted with the increasing number of bits in Binary Counter Datasets. The best sentence accuracy from each dataset is bolded.

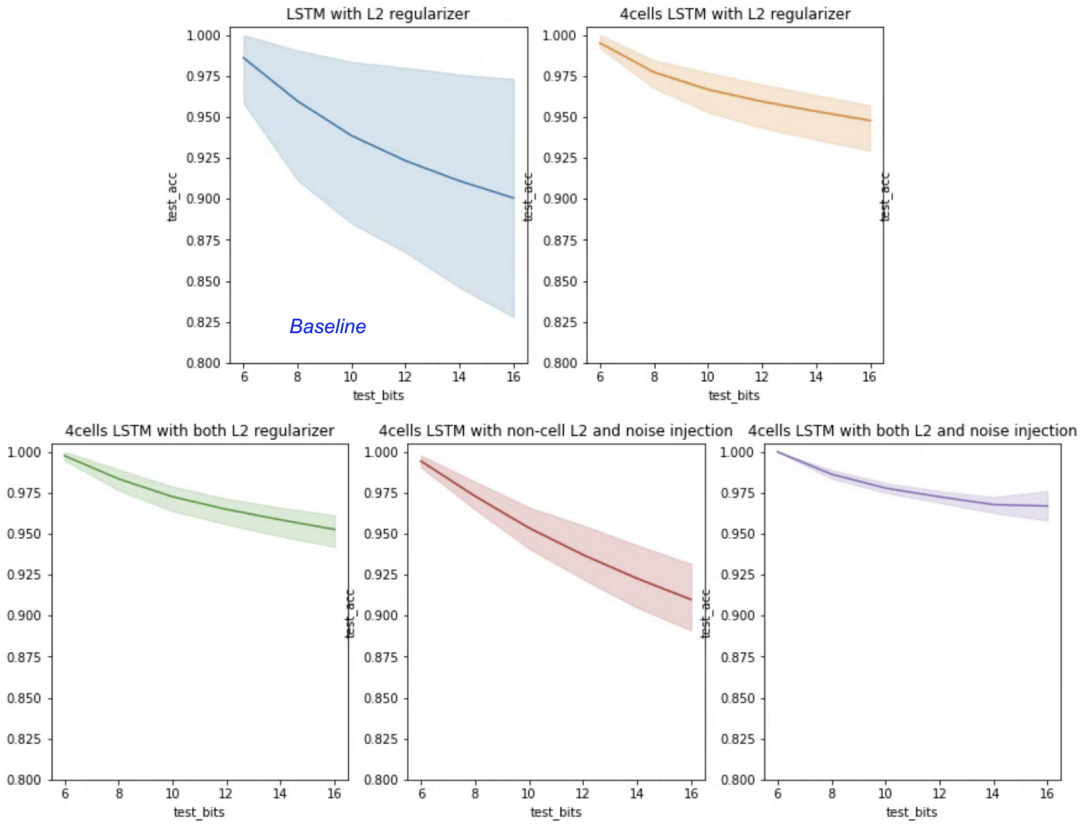


Figure 4.2: Performance of 4 Proto-LSTMs with Loading layer with different regularization methods

ventional way, and the below three graphs are the combinations of the regularization methods we suggest in this paper. The results show that the combination of both regularization methods, cell and non-cell L2 regularization, and noise injection, has the most robust performance and also achieved the best accuracy. It also shows that the proposed regularization methods not only improve the accuracy for longer prediction, but also robustness on hyper-parameter sweeping.

4.3.2 Proto-LSTM Cell loading distribution analysis

The graph in 4.3 shows that the model is trained to separate the relations.

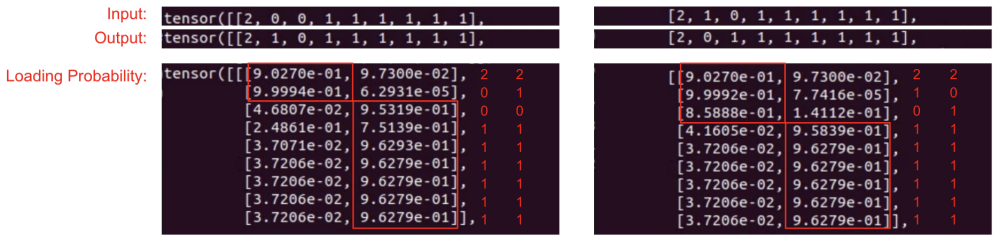


Figure 4.3: Proto-LSTM cells loading probability distributions at each time step in inference

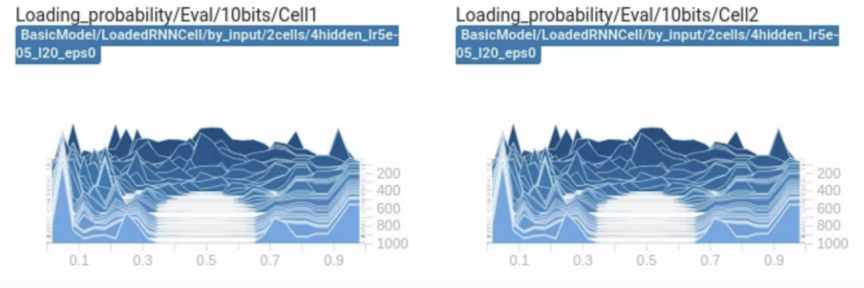


Figure 4.4: Proto-LSTM cells loading probability distributions at every epoch of training

The graph in 4.4 shows how the loading probability distribution changes as the training epoch increases. It is a stack of histograms of loading probability, where the training starts from the farthest histogram and ends on the nearest histogram. The distribution seems to be random at the beginning of the training, but it converges to 0 or 1 as the epoch increases. This means that the model tends to select only one proto-LSTM, and successfully separate the trained relation as in 4.3.

Chapter 5

Conclusion

In this work, we pointed out the disadvantages of the existing regularization method on RNN-based models and transformers, and show that those are not reliable for sequential data consisting of repeated relations. We proposed the extended architecture based on LSTM, proto-LSTMs with relation loader, which can capture those repeated relations distinctly. Furthermore, we proposed novel regularization methods which can regularize the complexity and the cardinality of those repeated relations separately.

To analyze the proposed model and regularization method, we designed a toy task with Binary Counter datasets. The training dataset consists of sequences composed of repeated relations, and the test dataset consists of longer sequences composed of the same relations from the training dataset. The experimental results show that the proposed extended architecture, proto-LSTM cells with relation loader, can solely improve the robustness over hyper-parameter sweeping in longer prediction. Also, we empirically analyzed the model's behavior and showed that explicitly separated model architecture has captured repeated relations distinctly and the relation loader tends to choose only one of them as the training epochs increases. The performance on longer prediction of the proto-LSTMs with relation loader trained with proposed regularization methods outperformed transformer and other RNN-based baselines in terms of both accuracy and robustness.

Summary

Relation-level Regularization for

Recurrent Neural Networks

In this work, we pointed out that the regularizations on RNN-based architectures need to make better use of the advantages of the recurrent nature of the models to learn recurrent relations in sequential data. We aim to strengthen these advantages and alleviate the disadvantages of RNNs. We hypothesized that RNNs' underestimated performance is partly due to their simple architecture and regularization methods.

We proposed novel global and local regularization methods which make it possible to separately regularize the cardinality of relations and the complexity of each relation. The proposed regularization methods can be applied to any RNN-based model with a simple extension of that architecture. This extension of the model explicitly distinguishes recursively occurring relations and learns to choose relation adaptively at every inference time step.

To analyze the proposed extended model and regularization methods, we designed a toy task with Binary Counter datasets. The experimental results showed that the proposed extended architecture, proto-LSTM cells with relation loader, can solely improve the robustness over hyper-parameter sweeping in longer prediction. The performance on longer prediction of the proto-LSTMs with relation loader trained with proposed regularization methods outperformed transformer and other RNN-based baselines in

terms of both accuracy and robustness.

References

1. B. Tang and D. S. Matteson, "Probabilistic transformer for time series analysis," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 23592–23608, Curran Associates, Inc., 2021.

2. I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *CoRR*, vol. abs/2004.05150, 2020.

3. J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling," *CoRR*, vol. abs/1911.05507, 2019.

4. S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *CoRR*, vol. abs/2006.04768, 2020.

5. K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. J. Colwell, and A. Weller, "Rethinking attention with performers," *CoRR*, vol. abs/2009.14794, 2020.

6. X. Liu, H. Yu, I. S. Dhillon, and C. Hsieh, "Learning to encode position for transformer with continuous dynamical model," *CoRR*, vol. abs/2003.09229, 2020.

7. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

8. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

9. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, pp. 2554–2558, Apr. 1982.

10. K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

11. Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

12. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

13. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 448–456, PMLR, 07–09 Jul 2015.

14. S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," in *International Conference on Learning Representations*, 2018.

15. R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

16. A. Y. Ng, "On feature selection: Learning with exponentially many irrelevant features as training examples," in *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 404–412, Morgan Kaufmann, 1998.

17. A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems 4* (J. E. Moody, S. J. Hanson, and R. P. Lippmann, eds.), pp. 950–957, Morgan-Kaufmann, 1992.

18. A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, (New York, NY, USA), p. 78, Association for Computing Machinery, 2004.

19. T. G. Dietterich, "Ensemble methods in machine learning," *Lecture Notes in Computer Science*, vol. 1857, pp. 1–??, 2000. It is a good classic article reviewing ensemble methods. He shows intutively why ensembles is a good idea: Statistical, computational, representational.

20. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

21. T. Baldacchino, E. J. Cross, K. Worden, and J. Rowson, "Variational bayesian

mixture of experts models and sensitivity analysis for nonlinear dynamical systems," *Mechanical Systems and Signal Processing*, vol. 66-67, pp. 178–200, 2016.

22. N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *CoRR*, vol. abs/1701.06538, 2017.

23. W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *CoRR*, vol. abs/2101.03961, 2021.

24. Z. Cao, L. Wang, and G. de Melo, "Multiple-weight recurrent neural networks," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 1483–1489, 2017.

25. A. Santoro, R. Faulkner, D. Raposo, J. W. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. P. Lillicrap, "Relational recurrent neural networks," *CoRR*, vol. abs/1806.01822, 2018.

Acknowledgements

First of all, I would like to express my deepest gratitude to my advisor, professor Kangil Kim, for his invaluable patience and feedback. His guidance carried me through all project stages, from my bachelor's degree to my master's degree. Also, this endeavor would not have been possible without my defense committee, professor Chang Wook Ahn and professor Ue-Hwan Kim, who generously provided knowledge and expertise.

I am also grateful to my lab mates for late-night feedback sessions and countless help while proceeding with my degree, including moral support. Appreciation should also go to my dearest friends for always being there for me. I would like to extend my honest gratitude to professor Soo Jeong Lee for always welcoming me and giving me the most sincere advice.

Lastly and most importantly, words cannot express my gratitude to my mother. I know she will always be there for me, and this fact alone gives me the strength to get up again. My warmest thanks will always belong to my mother.