Jiaxin Wang

A Recursive Recurrent Neural Network Decoder for Grammatical Error Correction

Computer Science Tripos – Part II

Emmanuel College

April 4, 2022

Proforma

Candidate Number:	-
Project Title:	A Recursive Recurrent Neural Network Decoder for Grammatical Error Correction
Examination:	Computer Science Tripos – Part II, 2022
Word Count:	1443 [1]
Code line count:	-
Project Originator:	-
Supervisor:	Dr Zheng Yuan, Dr Christopher Bryant

Original Aims of the Project

-

Work Completed

-

Special Difficulties

-

[1] This word count was computed by `detex diss.tex | tr -cd '0-9A-Za-z \n' | wc -w`

Declaration

I, Jiaxin Wang of Emmanuel College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed [signature]

Date [date]

Contents

List of Figures

Acknowledgements

-

Todo list

Chapter 1

Introduction

This project concerns my implementation of a *recursive recurrent neural network* model (R^2NN) proposed by Liu et al. (2014) [5] This is to be integrated in a statistical machine translation (SMT) system to attempt the task of grammatical error correction.

How well have you done?

1.1 Motivation

Grammatical Error Correction (GEC) is the task of producing a grammatically correct sentence given a potentially erroneous text while preserving its meaning. The main motivation behind this task lies in its role in helping learners of a foreign language understand the language better. In addition, native speakers can make use of such GEC tools to avoid mistakes in their writing. Being an ESL (English as a second language) learner myself, I would like to investigate models that can be used for the task of GEC in English, and I hope that it can benefit people who may find GEC tools useful.

1.2 Problem Overview

The task of Grammatical Error Correction can be seen as a machine translation process, where the input is a text which may contain grammatical errors, and the output is an error-free text. This project uses a statistical machine translation (SMT) approach to solve GEC.

A recursive recurrent neural network (R^2NN) was proposed by Liu et al. (2014) [5] for SMT. The goal of this project is to implement the proposed model to be used for GEC. The model should aim to correct all types of errors, namely grammatical, lexical, and orthographical errors. Its performance will be evaluated against a baseline SMT system, Moses[4].

1.3 Related Work

-

Chapter 2

Preparation

2.1 Starting Point

This project is based on the idea presented in the paper *A Recursive Recurrent Neural Network for Statistical Machine Translation* [5]. An R^{2}NN model was proposed, but the implementation details are not given in the paper.

The Part IB Computer Science Tripos course Artificial Intelligence[1] gives an introduction to neural networks and explains how forwarding and backpropagation works. Prior to this project I did not have any coding experience with neural networks. I have found it useful to follow the coding examples on PyTorch[2] website.

An example SMT system is available on Moses[3] website. It provides a detailed tutorial on how to install Moses, how to prepare corpus and how to train the SMT.

2.2 Theory

2.2.1 SMT system

A typical SMT system consists of four main components: The language model (LM), the translation model (TM), the reordering model and the decoder[6]. The LM computes the probability of a given sequence being valid. The TM builds a translation table which contains mappings of words/phrases between source and target corpora. The reordering model learns about phrase reordering of translation. The decoder finds a translation candidate who is most likely to be the translation of the source sentence. In the task of GEC, this would be the most probable correction to the original erroneous sentence.

Level of detail?

[1] https://www.cl.cam.ac.uk/teaching/2021/ArtInt/
[2] https://pytorch.org/tutorials/
[3] https://www.statmt.org/moses/?n=Moses.Baseline

2.2.2 Moses SMT

The Moses baseline system uses KenLM as the language model. For the translation model, it uses GIZA++ to obtain a word alignment model. The model should be trained to produce a phrase table and associated scores. Reordering tables are also created during this process. Eventually, the Moses decoder will find the best translation candidate given input based on the scores it calculated in previous stages.

2.2.3 Feedforward Neural Network

A feedforward neural network consists of three parts: an input layer, one or more hidden layers, and an output layer. In a feedforward neural network, data only flows in one direction (forward) from input to output. Figure 2.1 shows an example of a feedforward neural network with one hidden layer.

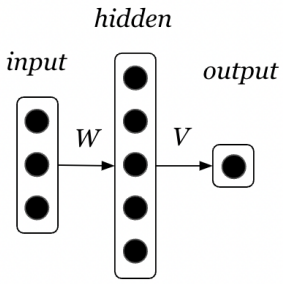

Figure 2.1: Example of a feedforward neural network

2.2.4 Recurrent Neural Network (RNN)

Recurrent neural networks are usually used to deal with sequences. They allow access to the input data as well as past data to compute the next state. As shown in Figure 2.2, the hidden layer h_t is computed using both input x_t at time t and the hidden state h_{t-1} which contains the history information from time 0 to $t-1$. For each timestep t, the hidden state can be expressed as:

$$h_t = Wx_t + Uh_{t-1}$$

2.2.5 Recursive Neural Network (RvNN)

A recursive neural network has a tree-like structure. Figure 2.3 illustrates an example of a basic RvNN architecture. The parent node representation is computed from its child nodes' representation as follows:

$$p_{n,n+1} = f(W[x_n; x_{n+1}])$$

where f is the activation function. The same weight matrix W will be applied recursively over the input.

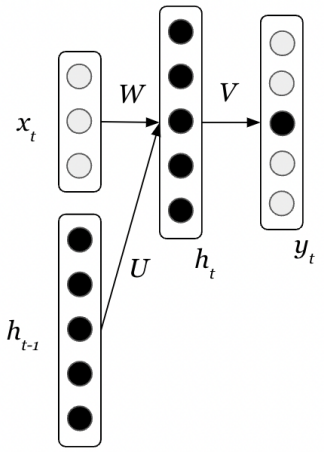

Figure 2.2: Example of a recurrent neural network (RNN)

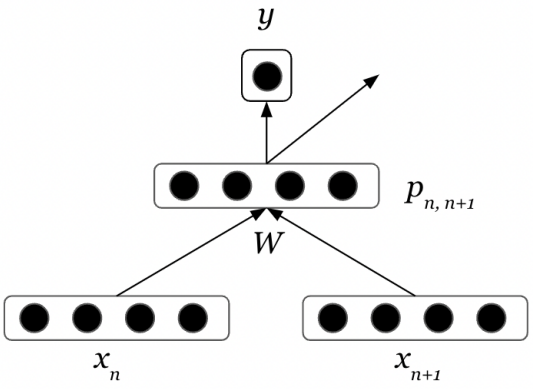

Figure 2.3: Example of a recursive neural network (RvNN)

2.2.6 Recursive Recurrent Neural Network (R^2NN)

The R^2NN proposed by Liu et al.[5] combines the features of RvNN and RNN. It has a tree-like structure similar to RvNN, with recurrent vectors added to integrate global information. As shown in Figure 2.4, $s^{[l,m]}$ and $s^{[m,n]}$ is the representation of child nodes $[l, m]$ and $[m, n]$. The recurrent input vectors, $x^{[l,m]}$ and $x^{[m,n]}$ are added to the two child nodes respectively. They encode the global information, such as language model scores and distortion model scores. A third recurrent input vector $x^{[l,n]}$ is added to the parent node $[l, n]$. The parent node representation is computed as

$$s_j^{[l,n]} = f(\sum_i \hat{x}_i^{[l,n]} w_{ji})$$

where $\hat{x}$ is the concatenation of vectors $[x^{[l,m]}; s^{[l,m]}; x^{[m,n]}; s^{[m,n]}]$, and f is the $HTanh$ function. The output, $y^{[l,n]}$, is computed as

$$y^{[l,n]} = \sum_j ([s^{[l,n]}; x^{[l,n]}])_j v_j$$

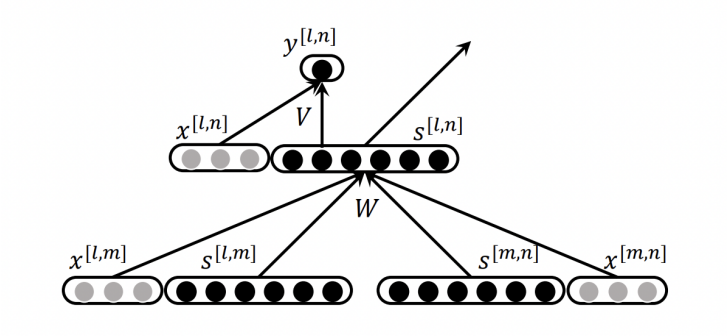

Figure 2.4: Recursive recurrent neural network (Liu et al., 2014, p.1494)

2.3 Requirement Analysis

Based on the Project Structure section from my project proposal, the following requirements have been identified:

Data preprocessing

- Data should be prepared in a form that is accepted by Moses SMT and R^{2}NN SMT
- Preprocessing should be done carefully to avoid accidentally correcting some of the grammatical errors
  - E.g. capitalisation errors may go undetected if all sentences are lowercased during data preprocessing

Moses SMT for GEC

- A language model should be trained to model the probability of a given sentence being valid
- A translation model should be trained to construct a phrase translation table
- A reordering model should be trained to learn the reordering of phrases
- With the above three models and Moses decoder, a complete Moses SMT system should be built

R^{2}NN SMT for GEC

Following the R^{2}NN paper by Liu et al.[5],

- Phrase pair embeddings (PPE) should be learned by building a one-hidden-layer neural network and a recurrent neural network
- A recursive recurrent neural network (R^{2}NN) should be built and used as a decoder

Evaluation

- The performance of both SMT systems should be evaluated using F0.5 scores

2.4 Choice of Tools

2.4.1 Programming Language

Python is chosen to be the main programming language as it provides many libraries that are commonly used for natural language processing. For this project I will be using `python 3.8` and PyCharm as my IDE.

2.4.2 Libraries

PyTorch
The `PyTorch`[4] library is one of the most popular machine learning frameworks. There are other similar libraries (such as `TensorFlow`) but I find `PyTorch` tutorials are easier to follow.

NumPy
My project is likely to involve statistical processing. I would be using the `NumPy`[5] library for this purpose.

pandas
`pandas`[6] is a powerful library for processing tabular data. This would be used to manipulate phrase tables in my project.

2.4.3 Dataset

The dataset introduced in BEA 2019 Shared Task[1] will be used in this project. I chose the two corpora (FCE v2.1 and W&I+LOCNESS v2.1) which are immediately downloadable from the website[7] to start with. The corpora have been standardised to be easily evaluated by ERRANT[2, 3]. ERRANT is a toolkit used to annotate parallel data and compare hypothesis against reference to produce various evaluation metrics including F0.5 score. I may request other corpora as an extension of my project.

2.4.4 Version Control and Backup

Git will be used for version control. The entire project, including all the written code and my dissertation, will be pushed to GitHub regularly.

[4]https://pytorch.org/
[5]https://numpy.org/
[6]https://pandas.pydata.org/
[7]https://www.cl.cam.ac.uk/research/nl/bea2019st/#data

Chapter 3

Implementation

3.1 Repository Overview

-

3.2 Moses SMT

-

3.2.1 Pre-processing

-

3.2.2 Language Model

-

3.3 R^2NN SMT

-

3.3.1 Pre-processing

-

3.3.2 Phrase Pair Embedding

Chapter 4

Evaluation

Chapter 5

Conclusion

Bibliography

[1] Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy, August 2019. Association for Computational Linguistics.

[2] Christopher Bryant, Mariano Felice, and Ted Briscoe. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[3] Mariano Felice, Christopher Bryant, and Ted Briscoe. Automatic extraction of learner errors in ESL sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[4] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, page 177–180, USA, 2007. Association for Computational Linguistics.

[5] Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. A recursive recurrent neural network for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1491–1500, 2014.

[6] Zheng Yuan. *Grammatical error correction in non-native English.* PhD thesis, University of Cambridge, Cambridge, United Kingdom, 3 2017.

Appendix A

Project Proposal