Anh Nguyen

Text generation using lstm and reinforcement learning

Bachelor's Thesis

|  |  |  |
| --- | --- | --- |
| Science and Engineering  Okko Rasanen  Joni Pajarinen  April 13th 2020 |  |  |

ABSTRACT

Anh Nguyen: Text Generation using LSTM and Reinforcement Learning

Bachelor of Science Thesis

Tampere University

International Degree of Science and Engineering

April 13th 2020

Text generation tasks are more and more becoming prominent, with the help of deep learning frameworks and libraries such as Tensorflow, Keras, Pytorch, …. This paper discusses the procedures of building a text generation model using deep learning and fine-tuning it using reinforcement learning algorithms, particularly Policy Gradient Method. We discuss about building a text generation model using Tensorflow and Keras API, which will be able to generate text based on a text database from Project Gutenberg’s novels. Then we discuss briefly about reinforcement learning: policies, environments, algorithms including the policy gradient algorithm. The two processes can then be combined using the Ray framework, particularly the RLLib library of the framework, which offers scalable software primitives for reinforcement learning. This method will be able to produce a system that generates text and improve it using reward feedback from a virtual human.

Keywords: deep learning, lstm, rnn, reinforcement learning, policy gradient theorem, ray,

Rllib, keras, tensorflow, text generation

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

Preface

I would like to thank my thesis supervisors, assistant professor Okko Rasanen and Joni Pajarinen, for suggesting the topic and guiding me through my research progress and writing process. I would like to also thank Mrs. Laeticia Petit for pushing and helping me keep track of my progress and providing useful resources for tasks such as information retrieval or thesis writing guide.

Tampere, Finland, 13th April 2020

Anh Nguyen

Contents

[1. Introduction 1](#_Toc37601093)

[2. BACKGROUND 3](#_Toc37601094)

[2.1 Deep learning 3](#_Toc37601095)

[2.2 Recurrent neural network and Long short-term memory RNN 4](#_Toc37601096)

[2.3 Reinforcement Learning 6](#_Toc37601097)

[2.4 Important frameworks 8](#_Toc37601098)

[3. IMPLEMENTATION 9](#_Toc37601099)

[3.1 Defining the model 9](#_Toc37601100)

[3.1.1 Generating text 11](#_Toc37601101)

[3.2 Improving the model using Reinforcement Learning 14](#_Toc37601102)

[3.2.1 Policy Gradient Theorem 14](#_Toc37601103)

[4. RESULTS AND DISCUSSION 15](#_Toc37601104)

[5. CONCLUSION 16](#_Toc37601105)

[REFERENCES 17](#_Toc37601106)

List of Figures

[Figure 2.2.1 Recurrent Neural Network 5](#_Toc37601140)

[Figure 2.2.2 LSTM Example 5](#_Toc37601141)

[Figure 2.3.1 Machine Learning Paradigms 7](#_Toc37601142)

[Figure 2.3.2 Key concepts of Reinforcement Learning 7](#_Toc37601143)

[Figure 3.1.1 LSTM model for text generation using Keras 13](#_Toc37601144)

List of Symbols and abbreviations

RNN Recurrent Neural Networks

LSTM Long Short-Term Memory

RL Reinforcement Learning

CNN Convolutional Neural Networks

.

# Introduction

The ability for machines and computers to understand and use human’s languages has been one of the most popular topics in the field of artificial intelligence. Interactions between machines and humans have been extremely difficult, since machine code (or language) is completely incomprehensible. Hence, the field natural language processing (NLP) is derived, with the aim to help computers interact with humans and vice versa. There are useful applications of NLP, including machine translation, which has the ability to translate from a language to another, speech recognition, sentiment analysis, which predicts and analyses human’s emotions based on their input text, and text generation.

This research will be focusing on text generation application of NLP. In particular, the system designed and implemented will be able to generate useful text based on an existing text database. Since most NLP applications rely heavily on the use of machine learning algorithms, the aforementioned system will also be utilizing machine learning algorithms, particularly deep learning models. Deep learning is a branch of machine learning which uses neural networks to solve problems such as feature extraction or classification. Moreover, besides building a deep learning model, the system also implements reinforcement learning algorithms in order to improve the quality of the text generated. Reinforcement learning is also a subfield of machine learning whose principle is to have a software-defined agent taking a set of actions, known as policy, in an environment so as to maximize a specific reward schema. Finally, the whole system is evaluated using a custom evaluation function which measures the accuracy of the text generated.

Chapter 2 presents the key concepts or theoretical background of deep learning, reinforcement learning and its algorithms. Moreover, a popular deep learning architecture is also discussed, followed by a brief explanation of the frameworks and libraries used in this research. Chapter 3 explains the implementation of the whole system, from data gathering, data preprocessing, to building the deep learning model and the reinforcement learning architecture needed to train the text database. Chapter 4 discusses the result obtained from chapter 3 and evaluates the system’s quality, along with data visualization for better insights. Last but not least, chapter 5 concludes the research with generalized claims and findings from chapter 3 and 4.

# BACKGROUND

This chapter explains key concepts of the research: deep learning as well as reinforcement learning concepts. In terms of deep learning, the structure of the Recurrent Neural Network (RNN), particularly Long Short-term Memory RNN, will be elaborated. Deep learning frameworks used in the research will also be mentioned and explained briefly. As regards reinforcement learning, key definitions such as policy, environment and reinforcement learning algorithms are prerequisites to understanding: i) how reinforcement learning works and ii) the mechanics of the solution model presented in later chapters.

## Deep learning

Deep learning is precisely a subset of machine learning and one of its algorithms. It replicates the human brain to perform tasks such as data processing, pattern recognition and information interpretation. The most fundamental level of deep learning models are neural networks with multiple layers [1]. However, the number of layers in a traditional network is around 2-3 layers while that of deep learning models can be up to 150 layers. Applications of deep learning include machine translation, self-driving cars, digital marketing, and so on. However, the purpose of this research is towards the natural language processing application of deep learning, specifically generating text from an existing database and thus improving the model.

Implementation of deep learning models can be done using various frameworks and libraries such as Tensorflow, Keras API, Pytorch, and so on. The libraries which are used in this research are Tensorflow and Keras. Tensorflow is an open-source end-to-end framework which supports building machine learning or state of the art machine learning models [3]. On the other hand, Keras is a high-level API for building neural networks, which runs on top of Tensorflow [4].

## Recurrent neural network and Long short-term memory RNN

A recurrent neural network (RNN) is a type of artificial neural networks that has a more complex structure. It is often used for natural language processing problems such as speech recognition, text classification, and so on. In an RNN architecture, neurons that are connected to each other form a loop, which means that the RNN utilizes both feedforward and feedback structure of a neural network. Hence, unlike traditional feed-forward neural networks, recurrent neural networks store their states after processing a sequence of input and thus use their internal states to process future inputs [5].

Understanding how RNNs work is the prerequisite to understanding the structure of Long short-term memory RNNs (LSTM).

A picture containing text

Description automatically generated

Figure 2.2.1 Recurrent Neural Network

A drawing of a cartoon character

Description automatically generated

Figure 2.2.2 LSTM Example

## Reinforcement Learning

Reinforcement learning is a subset of machine learning, one of three machine learning problems, including supervised and unsupervised learning. The objective of reinforcement learning is to utilize an agent in a specific environment and use it to optimize a certain cumulative reward by learning a good strategy to perform actions.

Understanding key concepts of reinforcement learning is the prerequisite to understanding how the research model can be improved using virtual feedback. In a reinforcement learning problem, there is an environment, which can be defined by an arbitrary model. An agent which acts on an environment has information about its state, an array of action it can take, then receives a scalar reward from the environment known as feedback. However, the agent may fully understand the environment, or does not have any information about the environment model at all. Hence, it is essential that the agent balances between exploration and exploitation.

Policies can be developed with respect to an agent and the environment it acts on. A policy π(s) can be defined as a set of actions for an agent to take in order to maximize the feedback reward [6]. Mathematically, a deterministic policy can be expressed as a function of state s which outputs action a:

In the case of stochastic policy:

A close up of a logo

Description automatically generated

Figure 2.3.1 Machine Learning Paradigms

A close up of a logo

Description automatically generated

Figure 2.3.2 Key concepts of Reinforcement Learning

## Important frameworks

The programming language used in the research is Python. Besides essential libraries for data processing and helper functions, there are two frameworks that are considered vital for creating a text generation model: Tensorflow/Keras and Ray, specifically RLLib. Tensorflow is a framework providing tools for machine learning applications, while Keras is a high-level API for building neural networks. Keras will be used for the creation of the LSTM model for generating text. Additionally, keras also provides helper built-in functions for data preprocessing such as to\_categorical function to one-hot encode the data labels, or functions to configure the text generation model.

On the other hand, on building a reinforcement learning model for improving output text, Ray will be used, including the RLLib library. Ray is a framework that provides a system for handling AI applications, which includes an interface and other schedulers [7]. RLLib, on the other hand, is a package which comes along with the Ray framework, providing scalable algorithms to support the creation of reinforcement learning models. This includes custom environments, policies and other reinforcement learning algorithms such as Deep Q-learning (DQN) or Policy Gradient Algorithm (PPO).

# IMPLEMENTATION

This chapter will explain the procedure of building a text generation model from a database and apply reinforcement learning algorithms onto it. Generally, the process can be outlined as the following: i) Finding a text database, ii) Building a deep learning model to study the database, iii) Applying reinforcement learning algorithms using RLLib.

## Defining the model

Before constructing a LSTM model using Keras, a text database must be prepared and processed. For this research, novels’ text from Project Gutenberg will be used. The text can be found on the respective website, collected into a text file for Python to easily read and process.

In terms of data preprocessing, after being read by Python, the text string will be filtered by omitting all punctuations and symbols so that only alphanumerical values are chosen. Furthermore, the text will be transformed into lowercase. Two python dictionaries are created for the purpose of simplifying character referencing, one as a mapping from character to integer or index and vice versa. In order to fit the model onto the processed text database, data must be split into a training set and a test set (validation set). Keras’ LSTM model takes a sequence of text as an input. Hence, for each training and test set, the input data (known as x) will be a sequence of 40 characters, whereas their labels (known as y) will be the next character of the following sequence. However, since Keras’ neural networks are unable to process characters or strings as inputs, characters must be converted into numbers (or indexes), which can be done by utilizing the previous aforementioned python dictionary mapping character to integer. Last but not least, the training data can be one-hot encoded using Keras’ to\_categorical built-in function. One-hot encoding is a process where categorical data is transformed into a binary label matrix which will make machine learning algorithms, especially neural networks, perform better.

(Write more on one hot encode)

In terms of creating the text generation model, Keras’ Sequential model is used as a baseline, followed by two LSTM layers of 128 hidden units each. The output of the LSTM layers will be put into a fully connected layer with 50 neurons, with the ReLU activation function. An activation function (or transfer function) is defined as a weighted function of an input neuron and the bias term which has the purpose of eliminating unnecessary neurons in a neural network layer [9]. Activation functions can be either linear or non-linear, however, in this fully connected layer, the ReLU function is non-linear and can be represented by the following equation:

Finally, the activated neurons from the fully connected layer will be put into a classification layer which has the number of neurons equal to the number of distinct characters found in the text database. As regards activation function, the softmax function is another non-linear function which calculates the probability distribution of each class; in this case, probability of each alphanumeric character. Softmax can be expressed as following:

This means that the probability of each character ranges from 0 to 1, thus, the sum of those probabilities will always be 1. Hence, the output character can be chosen by picking the class with the highest probability. That is why softmax is considered perfect for classification as well as other models [8].

Overall, the structure of our text generation model is relatively small and simple, since only a computationally efficient yet straightforward model is needed for this task. However, to be able to prepare the model for training, it needs to be compiled. In Keras, compiling the model means setting up configurations for the model so as to optimize its training process, which in this case, defining the loss function and optimizer. In terms of loss function, since the network is a multi-class model, categorical crossentropy is chosen. Mathematically, categorical crossentropy loss is the difference between the distribution of the predicted output and that of the actual output (label) [10] . It can be expressed as the following equation:

where is a one-hot encoded prediction vector and is the actual ground truth distribution. Hence, this loss is suitable for model evaluation when the model output is one-hot encoded.

On the other hand, another important configuration to consider is the model’s optimizer, in which case Adam optimizer is used. Optimizers in a Keras model are algorithms or methods used to update the weights or learning rate of the model so as to minimize the loss function [11]. Particularly, Adaptive Moment Estimation (Adam) is an optimization algorithm which is considered an upgrade version of the Stochastic Gradient Descent (SGD) algorithm. It is effective and often outperforms other algorithms in deep learning applications [12]. Keras sets default parameters for its optimizers, however, in this case, the learning rate was specifically set to 0.0001.

Training the dataset can take a relatively large amount of time. Hence, it is extremely unnecessary to train the model from scratch more than once. However, Keras has callbacks which helps to save and load the weights of a model after training so that they can be reused at any time. Callbacks are a set of built-in functions that can be used to monitor the training process. For instance, ModalCheckpoint saves the weight into a file that can later be loaded into an existing model, or ReduceLROnPlateau readjusts the learning rate if a target property of the model does not improve within a specified number of epochs. In this research, only ModalCheckpoint is used. Finally, the model is fit onto the training dataset and trained for 200 epochs (iterations) with batch size of 64. Batch size is defined as a hyperparameter specifying the interval after which the model’s parameters are updated. For example, a batch size of 64 means that after training each 64 data samples, the model will update its parameters.

### Generating text

After the training process, an evaluation method needs to be derived in order to assess the efficiency of the model. However, diversity is needed so as to keep the output creative and not monotone – generating the same output for the same input text. Hence, a sample function is implemented with the purpose of sample an index from the one-hot encoded predicted vector, based on the “temperature” or level of diversity specified when called. This temperature is a float parameter ranging from 0 to 1, with 0 having no diversity and 1 meaning complete randomized output. The principle of the function can be seen below

A screenshot of a cell phone

Description automatically generated

Figure 3.1.1 LSTM model for text generation using Keras

## Improving the model using Reinforcement Learning

With a relatively small dataset, it is understandable that the model itself is unable to generate perfect English text. Therefore, to improve the model in order to generate more useful and creative, yet meaningful outputs, one way is to use reinforcement learning algorithms. This section will explain the used reinforcement learning algorithm, along with its implementation using the RLLib framework.

### Policy Gradient Theorem

Policy Gradient Theorem is

# RESULTS AND DISCUSSION

# CONCLUSION

# REFERENCES

1. C. Heinz, B. Moses, J. Hoffmann, Listings - Typeset source code listings using LaTeX, Comprehensive TeX Archive Network (CTAN), 2006. Available: http://www.ctan.org/pkg/listings
2. T. Oetiker, H. Partl, I. Hyna, E. Schlegl, The Not So Short Introduction to LATEX2ε – Or LATEX2ε in 157 minutes, Version 5.03, 2014, 171 p. Available: <http://www.ctan.org/tex-archive/info/lshort/english/>
3. K. Ruohonen, Matemaattisen tekstin kirjoittaminen, Tampereen teknillinen yli-opisto, 2009, 7 s. Available: http://math.tut.fi/~ruohonen/D-tyo-ohje.pdf
4. E. Salminen, Practical advice for writing publications, course material, TKT-9617 Scientific Publishing, Tampere University of technology, Nov 2009 (updated Apr 2014), 101 p. Available http://www.cs.tut.fi/~ege/Misc/salminen\_figures\_styles\_v15.pdf
5. Guide to Writing a Thesis in Technical Fields: Instructions for Master of Science and Bachelor of Science Theses, Tampere University, Tampere, 2019. Available: POP > Study info > Master's thesis > MSc % thesis guidelines