Building Transformer Models with PyTorch 2.0

NLP, computer vision, and speech processing with PyTorch and Hugging Face

Prem Timsina

www.bpbonline.com

First Edition 2024

Copyright © BPB Publications, India

ISBN: 978-93-55517-494

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

Dedicated to

My beloved wife:

Sunita

and

My son ***Percival***

About the Author

Prem Timsina is the Director of Engineering at Mount Sinai Health Systems, where he oversees the development and implementation of Machine Learning Data Products. He has overseen multiple Machine Learning products that have been used as clinical decision support tool at several hospitals within New York City. With over ten years of experience in the field, Dr. Timsina is a dedicated Machine Learning enthusiast who has worked on a variety of big data challenges using tools, such as PyTorch, Deep Learning, Generative AI, Apache Spark, and various NoSQL platforms. He has contributed to the field through more than 40 publications in Machine Learning, text mining, and big data analytics. He earned his Doctor of Science degree in Information Systems from Dakota State University.

About the Reviewer

Pratik Kotian is an accomplished professional with eight years of extensive expertise in Natural Language Processing, Machine Learning, Generative AI, and Python programming. With a versatile background spanning various sectors including technology, telecommunications, finance, retail, and more, Pratik has honed his skills across different domains. His experience encompasses leadership roles in research and development, technical consultancy, and team management. Currently, Pratik serves as a Manager at Deloitte, where he leads the Generative AI Team with a focus on pioneering innovative solutions for clients. In this capacity, he leverages his expertise to drive transformative initiatives, enabling businesses to unlock value through AI-driven strategies.

Acknowledgement

I extend my deepest appreciation to my wife, Sunita Ghimire, for her constant support throughout the process of writing this book. Special love to my son Percival for being a wonderful toddler and allowing daddy to work on the book.

I am immensely thankful for dedicated collaboration of reviewers, editors, and technical expert for revising and refining this book. Additionally, I am grateful to the entire BPB Publications team who have worked tirelessly on realizing this book.

My greatest thanks to all the readers who have supported this work. Your enthusiasm on this topic has been fundamental in bringing this book to fruition.

Preface

Lately, transformer architecture has appeared as a swiss knife for Machine Learning architecture. The transformer architecture is at the heart of most recent breakthroughs in Generative Artificial Intelligence. For instance, tools like ChatGPT and BARD, perceived by many as paving stones towards artificial general intelligence, are built on Transformer foundations. Thus, it is imperative for data scientists, ML Engineers, and Technologists to understand how this architecture can solve various ML tasks.

This book provides both theoretical and practical understanding of transformer architecture. Specifically, we will cover these ML tasks: **Natural Language Processing (NLP)**, Computer Vision, Speech Processing, Tabular Data Processing, Reinforcement Learning, and Multi-Modalities. Center to the book are four major ML tasks, each explored in depth across two chapters. The initial chapter lays the groundwork by discussing the conceptual understanding. Here, we discuss the inner working of transformer architecture to solve tasks and discuss the architecture of major foundational models. Following this, the subsequent chapters focus on the practical understanding of pre-training, fine-tuning and using open source models to solve the ML tasks. This book will demonstrate practical applications through several comprehensive, end-to-end projects.

To equip with comprehensive understanding, the book has dedicated chapters for Hugging Face Ecosystem, transfer learning and deploying and serving transformer models. We will also delve deeper into best practices and debugging transfomer model developed utilizing PyTorch and Hugging Face.

The pre-requisite for this book is basic understanding of PyTorch and deep learning. This book will benefit data scientists and ML engineers who are seeking to enhance their knowledge of transformer models and learn how to develop ML engines using the transformer architecture and Hugging Face's transformer library. It will also be valuable for developers and software architects looking to integrate transformer-based models into their existing software products. Additionally, AI enthusiasts interested in the latest developments in cutting-edge ML methods will find this book useful.

In summary, you will gain a conceptual understanding of transformer architecture and practical knowledge on how to solve various ML tasks using this architecture. Happy reading!

Chapter 1: Transformer Architecture – This chapter gives the readers an overview of the evolution of NLP models over time and how each previous development has influenced the transformer architecture. The majority of this chapter discusses the conceptual understanding of the transformer architecture, illustrating details on the encoder, decoder, positional encoding, and embedding. The chapter also explains to readers about different variations of the transformer architecture and their applications in solving NLP tasks.

Chapter 2: Hugging Face Ecosystem – This chapter provides a thorough understanding of the core functionalities and features of the Hugging Face ecosystem, specifically focusing on the transformers, datasets, and tokenizers libraries. The chapter explains how to use the Hugging Face ecosystem for using pre-trained models, fine-tuning existing models, and sharing your models. We will walk you through each step of this process, using practical examples, specifically fine-tuning the Dreambooth model (personalizing text to image generation).

Chapter 3: Transformer Model in PyTorch – This chapter will give the readers a detailed understanding of the PyTorch implementation of the transformer architecture, thoroughly examining its various components. This includes learning how to build models in different configurations, such as encoder only, decoder only, and the combined encoder-decoder setup in PyTorch. All of these concepts are explaned through three projects implemented in Pytorch: 1. Classifier—IMDB sentiment 2. Text Generation—Shakespear poet 3. Machine Translation—English to German. This chapter is all about getting you comfortable and confident with how Transformer models work in PyTorch.

Chapter 4: Transfer Learning with PyTorch and Hugging Face – This chapter provides a complete picture of what transfer learning is, why it is useful, and where it can be used. We will showcase the transfer learning by building the real news vs. fake news project.

Chapter 5: Large Language Models: BERT, GPT-3, and BART – This chapter discusses the key concepts of **Large Language Models** (**LLMs**). It will also discuss the key determinants of LLM performance. Additionally, we will look at the architecture of pioneering LLMs. We will conclude this chapter by showcasing how you can create your own LLM with your data.

Chapter 6: NLP Tasks with Transformers – This chapter will provide a detailed understanding of key NLP tasks and the corresponding transformer models used to solve these tasks. We will also discuss handling long sequences in transformers. We will explore these concepts through three projects: 1. Handling long sequences by chunking, 2. Handling long sequences with hierarchical attention, and 3. Generating Shakespeare-like text using GPT-2 and Tiny Shakespeare.

Chapter 7: CV Model Anatomy: ViT, DETR, and DeiT – This chapter will provide a foundational understanding of image pre-processing techniques and their significance in computer vision tasks. The chapter delve into the architecture and workings of the **Vision**

Transformer (**ViT**), **Distilled Vision Transformer** (**DeiT**) and **Detection Transformer** (**DETR**). We will illustrate all these concepts by three projects.

Chapter 8: Computer Vision Tasks with Transformers – This chapter serves as a comprehensive understanding of various computer vision tasks and their applications. We will look at three main tasks here, namely, Image Segmentation, Classification, and Image Generation. We will explain these concepts through training and developing three machine learning models: 1. Food Image Segmentation, 2. Comparison of DEIT and RESNET, and 3. Dog Image Generation.

Chapter 9: Speech Processing Model Anatomy: Whisper, SpeechT5, and Wav2Vec – This chapter provides a foundational understanding of speech pre-processing, and a detailed analysis of Whisper, SpeechT5, and Wav2Vec Architecture.

Chapter 10: Speech Tasks with Transformers – This chapter will provide a comprehensive understanding of various speech processing tasks and their applications in real-world scenarios. We will look at into three major tasks here: 1. Text-to-Speech 2. Automatic Speech Recognition, and 3. Audio-to-Audio. We will explain these concepts through real world projects

Chapter 11: Transformer Architecture for Tabular Data Processing – This chapter will look at the following architecture: 1. Google's TAPAS for quering the tabular data, 2. TabTransformer for Structured Data, and 3. FT Transformer for structured data. We will explain these architecture through real world examples.

Chapter 12: Transformers for Tabular Data Regression and Classification – This chapter explores the application of transformers in tabular data processing. We will also delve into the implementation of transformers such as TabTransformer, FT Transformer, and TabNet for solving classification and regression problems. The chapter illustrates these models by solving both classification and regression problems and comparing the results with XGBoost. In summary, the goal of this chapter is to provide a detailed explanation of how we can use Transformer Architecture for machine learning with structured data.

Chapter 13: Multimodal Transformers, Architectures and Applications – This chapter is an explanation of how transformers can handle multiple data types in a single model. We will discuss two major architectures: ImageBind Architecture (Meta's architecture that combines text, audio, IMU, thermal, depth, and image) and CLIP Architecture (text and image). This chapter also explains different multi-modal tasks.

Chapter 14: Explore Reinforcement Learning for Transformer – This chapter discusses the fundamentals of **Reinforcement Learning (RL)** and the most common tools in PyTorch, as well as the process of building an RL model. This chapter will walk you through developing a Trading Model using tools like Gym, Stable Baselines3, and Yfinance. Additionally, this chapter illustrates two major RL architectures: Decision transformer and trajectory transformer, which are significant transformer architectures used in RL.

Chapter 15: Model Export, Serving, and Deployment – This chapter provides a comprehensive exploration of the machine learning lifecycle, focusing on model serialization, export, and deployment. Specifically, the chapter illustrates exporting PyTorch models to interoperable formats like ONNX, as well as the usage of PyTorch Script and Pickle. The chapter also provides a practical illustration of serving a PyTorch model using FastAPI and sharing model through Hugging Face. The goal is to equip readers with the knowledge and tools needed to efficiently export, serve, and deploy machine learning models.

Chapter 16: Transformer Model Interpretability, and Experimental Visualization – This chapter discusses the concepts of model interpretability and explainability. The chapter explores various tools that can be used for model interpretability and explainability. It provides a practical example of using CAPTUM for interpreting transformer models. Additionally, the chapter showcases how to use TensorBoard for model visualization, logging, and interpretation

Chapter 17: PyTorch Models: Best Practices and Debugging– The chapter discusses practical guidelines and best practices for building transformer models using both the general PyTorch Library and the Hugging Face Library. It then discusses a structured approach to debugging PyTorch models. The chapter illustrates all these concepts through real-world examples.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/hydtz8g

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Building-Transformer-Models-with-PyTorch-2.0.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

CHAPTER 1
Transformer Architecture

Introduction

Imagine you are a software engineer working on an exciting project and searching for a programming language to help create software quickly and efficiently. You hear about a revolutionary new type of language that is the Swiss knife of programming language: this language is most efficient in creating **Machine Learning (ML)** models—plus, this programming language creates stunning websites faster than other web development frameworks and supports hardware programming. Furthermore, its performance in network programming and other related tasks is also outstanding. Would it not be interesting to learn about this powerful programming language?

Similar developments can be observed in the world of ML frameworks. The transformer architecture is an incredibly versatile ML architecture. Transformers were initially developed for **Natural Language Processing** (**NLP**). Due to their superior results, this architecture has rendered other NLP architectures like RNN and **long short-term memory networks (LSTM**) obsolete. More recently, transformers have begun impacting other ML fields as well. According to SUPERB (**https://superbbenchmark.org/leaderboard**) the best foundational model for speech processing is also based on the transformer. Furthermore, transformers have shown excellent results in computer vision and other machine learning fields as well. Therefore, transformers have the potential to converge all AI frameworks into a solitary, highly adaptable architecture.

In this chapter, we will look into the **base architecture** of this versatile machine learning in depth. The chapter specifically focuses on understanding the original transformer architecture proposed by *Vaswani et al. (2017)*. Since the transformer was originally proposed for NLP—we will understand important NLP models and how the transformer was influenced by those models.

Structure

This chapter covers the following topics:

- Chronology of NLP model development.
- Transformer architecture
- Training process of transformer
- Inference process of transformer
- Types of transformers and their applications

Objectives

This book chapter intends to provide readers with a broad understanding of the evolution and significant milestones in the development of NLP models, with a special emphasis on the transformer architecture. It seeks to offer an in-depth examination of various NLP models, drawing comparisons and highlighting the distinctive ways in which the transformer model addresses the limitations of its predecessors. A key focus will be placed on investigating the essential components that make up the transformer architecture. Additionally, the chapter aims to educate readers about the different variations of the transformer model, showcasing their broad spectrum of applications in the field of NLP. The overarching theme of this chapter is to trace the journey of NLP models' development, culminating in the rise of the transformer as a ground-breaking innovation in the landscape of language processing technologies.

Chronology of NLP model development

The transformer was originally proposed for NLP, specifically machine translation, by *Vaswani et al*. in 2017[1]. It is currently the most popular and effective model in NLP, as well as other wide-ranging tasks (speech processing, computer vision, and others). However, the development of the transformer was not a sudden occurrence. In fact, it was the culmination of years of research and development in NLP models, with each model building upon the previous ones. Let us examine the chronological history of different NLP models. This is important because as we understand the transformer architecture,

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need. Advances in neural information processing systems, 30.*

we will be able to contextualize it within the historical development of NLP models, their shortcomings, and how transformer is unique and versatile.

In the upcoming section, we will explore the timeline of NLP model evolution and contrast various NLP models. *Figure 1.1* shows the chronology of NLP research:

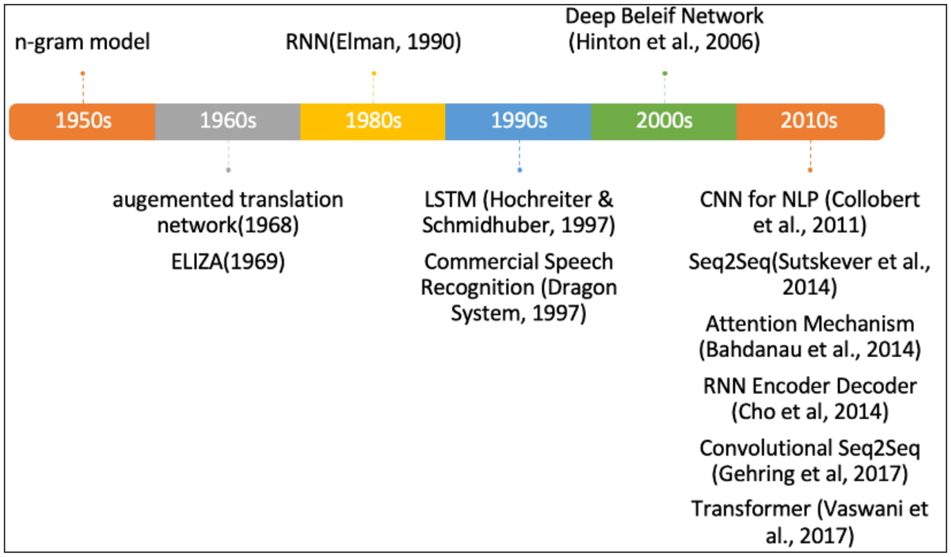
n-gram model
RNN(Elman, 1990)
Deep Beleif Network (Hinton et al., 2006)
1950s
1960s
1980s
1990s
2000s
2010s
augemented translation network(1968)
ELIZA(1969)
LSTM (Hochreiter & Schmidhuber, 1997)
Commercial Speech Recognition (Dragon System, 1997)
CNN for NLP (Collobert et al., 2011)
Seq2Seq(Sutskever et al., 2014)
Attention Mechanism (Bahdanau et al., 2014)
RNN Encoder Decoder (Cho et al, 2014)
Convolutional Seq2Seq (Gehring et al, 2017)
Transformer (Vaswani et al., 2017)

***Figure 1.1**: Chronology of NLP models development*

The transformer model was the culmination of all the previous research developments. *Vaswani et al*., cited a few of that original research. Specifically, *Vaswani et al*. cited the following research, and the transformer model seems to have been highly influenced by them.

In the following sections, we will discuss a few of the most important NLP models, their benefits, and their shortcomings.

Recurrent neural network

First, let us discuss the concept of next-word prediction. For instance, let us say we have a sentence, *The color of the sky is ….* Based on the information already processed by our brain, we can predict that the next word in this sentence would be *blue*. However, this prediction is not solely based on previous words, but rather on multiple preceding words.

Traditional machine learning algorithms, such as linear regression and multilayer perceptron, are not equipped to store previous information and utilize it for predictions. These algorithms do not have the capability to retain information from prior inputs. Here, recurrent neural networks come into play, which is capable of retaining prior information and utilizing it for making accurate predictions.

Figure 1.2 shows the structure of RNN. Here, each cell takes the output of its previous cell as its input. This allows the network to retain information from previous time steps and incorporate it into the computation at each subsequent iteration:

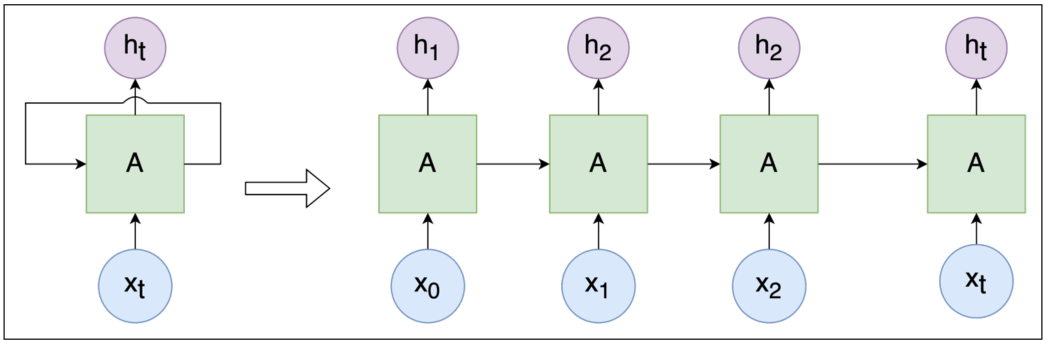

***Figure 1.2**: RNN structure*

Limitation of RNN

Let us consider the following example: *England is my hometown. I spent my whole life there. I just moved to Spain two days ago. I can speak only one language, which is ....* In this example, the next word is *English*. The most important contextual word, in this case, is *England*, which appears at the beginning of the sentence. However, in some cases, the relevant information may be located far away from where it is needed in an RNN. For example, in this case, the gap between the relevant information and the predicted word is about 26-time steps, that is, *England* is at time step 1, and the predicted word is at time step 27. This large gap can pose a problem for RNNs, as they may not be able to retain contextual information over such long sequences, or the weights associated with that information may become very small. This is due to the structure of RNNs, where the gradients can become very small or even zero as they are repeatedly multiplied by the weight matrices in the network. This can make it difficult for the network to learn and can cause training to be slow or even fail altogether.

LSTM

To overcome the issue of the vanishing gradient problem, LSTM was introduced.

In contrast to RNNs, LSTMs have a memory gate that allows them to store information about long-term dependencies in data. Furthermore, they possess a forget gate which helps filter out unnecessary information from previous states.

Another advantage of LSTMs is their low likelihood of encountering the problem of vanishing gradients. This occurs when gradients become very small or even zero during backpropagation, making it difficult for the network to learn. LSTMs address this issue by employing gates that regulate information flow through the network, allowing it to retain relevant details and discard irrelevant ones. *Figure 1.3* shows the comparison of RNN and LSTM structures. As compared to RNN, LSTM structure is complex:

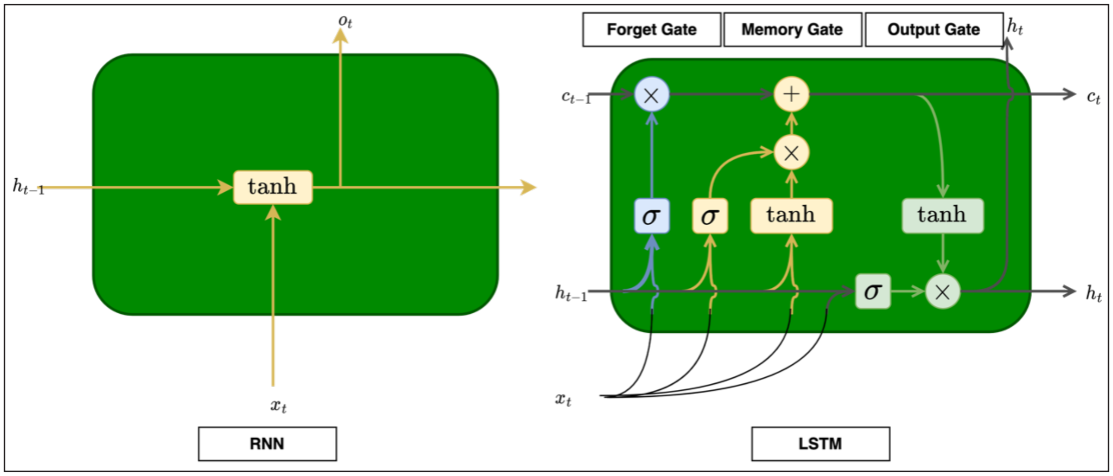

Figure 1.3*: Comparison of RNN and LSTM*

Limitation of LSTM

Limited ability to handle long sequences: Even though LSTM has a memory gate, they still struggle to handle long sequences. This is because they use a fixed length hidden state, which may be a problem if the input sequence is very long.

LSTMs process sequences sequentially, this can be slow and limit the ability to parallelize computations across multiple processors.

Cho's (2014) RNN encoder decoder[2]

The RNN encoder-decoder model is a sequence-to-sequence algorithm. It has three major components. Let us explore the components of an RNN encoder-decoder model with an example of English-to-French translation:

- **Encoder**: This is an RNN that encodes a variable-length input sequence (in this case, an English sentence) into a fixed-length vector.
- **Encoded vector**: The fixed-length vector output by the encoder.
- **Decoder**: This is also an RNN that takes the encoded vector as input and produces a variable-length output sequence (in this case, the French translation of the English input sequence).

[2] 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*.