

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

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

To View Complete
BPB Publications Catalogue
Scan the QR Code:



www.bpbonline.com

Kup ksi k

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 transformer 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 explained through three projects implemented in PyTorch: 1. Classifier—IMDB sentiment 2. Text Generation—Shakespeare 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 querying 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.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

business@bpbonline.com for more details.

At **www.bpbonline.com**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

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

1. Transformer Architecture	1
Introduction	1
Structure	2
Objectives	2
Chronology of NLP model development	2
<i>Recurrent neural network</i>	3
<i>Limitation of RNN</i>	4
<i>LSTM</i>	4
<i>Limitation of LSTM</i>	5
<i>Cho's (2014) RNN encoder decoder</i>	5
<i>Bahdanau's (2014) attention mechanism</i>	6
Transformer architecture	7
<i>Embedding</i>	8
<i>Positional encoding</i>	9
<i>Model input</i>	11
<i>Encoding layer</i>	12
<i>Attention mechanism</i>	14
<i>Self-attention</i>	14
<i>Multiheaded attention</i>	15
<i>Decoder layer</i>	15
Training process of transformer	17
Inference process of transformer	17
Types of transformers and their applications	17
<i>Encoder only model</i>	17
<i>Decoder-only model</i>	18
<i>Encoder-decoder model</i>	19
Conclusion	19
Quiz	19
Answers	22

2. Hugging Face Ecosystem.....	23
Introduction	23
Structure	23
Objectives	24
<i>System resources</i>	24
Overview of Hugging Face.....	24
<i>Key component of Hugging Face</i>	25
Tokenizers	26
Create your custom Tokenizer	26
Training.....	27
Inference	27
Use pre-trained tokenizer from Hugging Face	28
Datasets.....	29
Using Hugging Face dataset.....	29
Using the Hugging Face dataset on PyTorch.....	30
Models	31
Environmental setup	32
Training	32
Inference	33
Sharing your model on Hugging Face.....	35
Model	35
Spaces.....	35
Conclusion	36
Quiz	37
Answers.....	38
3. Transformer Model in PyTorch.....	39
Introduction	39
Structure	39
Objectives	40
System resources	40
Transformer components in PyTorch.....	40

Embedding.....	42
<i>Example</i>	43
Positional encoding.....	43
Masking	45
Encoder component of a transformer	46
Decoder component of a transformer	48
Transformer layer in PyTorch.....	50
Conclusion	51
Quiz	52
Answers.....	54
4. Transfer Learning with PyTorch and Hugging Face.....	55
Introduction	55
Structure	56
Objectives	56
System requirements	56
Need of transfer learning	57
Using transfer learning.....	58
Where can you get pre-trained model	60
Popular pre-trained model	60
<i>NLP</i>	60
<i>Computer vision</i>	61
<i>Speech processing</i>	61
Project: Develop classifier by fine tuning BERT-base-uncased.....	62
<i>Custom dataset class</i>	65
<i>DataLoader</i>	67
<i>Inference</i>	70
Conclusion	72
Quiz	72
Answers.....	75
5. Large Language Models: BERT, GPT-3, and BART.....	77
Introduction	77
Structure	78

Objectives	78
Large language model.....	78
Key determinants of performance.....	80
<i>Size of network: Number of encoder and decoder layers</i>	80
<i>Number of model parameters</i>	80
<i>Max-sequence length</i>	80
<i>Size of embedding dimension</i>	80
<i>Pre-training dataset size and types</i>	80
Pioneering LLMs and their impact.....	81
<i>BERT and its variants</i>	81
<i>BERT pre-training</i>	81
<i>BERT fine-tuning</i>	82
<i>BERT Variations</i>	83
<i>Applications</i>	83
<i>Generative pre-trained Transformer</i>	84
<i>Pre-training of GPT</i>	84
<i>Applications</i>	86
<i>Bidirectional and Auto-Regressive Transformers</i>	86
<i>Pre-training</i>	86
<i>Application</i>	87
Creating your own LLM	87
<i>Clinical-Bert</i>	88
Conclusion	89
Quiz	90
Answers.....	92
6. NLP Tasks with Transformers.....	93
Introduction	93
Structure	93
Objectives	94
System requirements	94

NLP tasks.....	94
Text classification	96
<i>Most appropriate architecture for text classification</i>	96
<i>Text classification via fine-tuning transformer</i>	96
<i>Handling long sequence</i>	97
<i>Project 1: Document chunking</i>	97
<i>Project 2: Hierarchical attention</i>	98
Text generation	99
<i>Project 3: Shakespeare like text generation</i>	99
<i>Data preparation</i>	99
<i>Training</i>	101
ChatBot with transformer	103
<i>Project 4: Clinical question answering transformer</i>	104
<i>Data preparation</i>	104
<i>Model declaration</i>	104
<i>Creating prompt and tokenization</i>	104
Training with PEFT and LORA.....	105
Conclusion	107
Quiz	107
Answers.....	109
7. CV Model Anatomy: ViT, DETR, and DeiT	111
Introduction	111
Structure	112
Objectives	112
System requirements	112
Image pre-processing	113
<i>Example of image pre-processing</i>	114
Vision transformer architecture	116
<i>Project 1: AI eye doctor</i>	118
Distillation transformer.....	119
<i>Advantages of DeiT</i>	120
<i>Exercise</i>	120

Detection transformer.....	120
<i>Project 2: Object detection model</i>	122
Conclusion	124
Quiz	125
Answers.....	127
8. Computer Vision Tasks with Transformers.....	129
Introduction	129
Structure	129
Objectives	130
System requirements	130
Computer vision task	130
<i>Image classification</i>	131
<i>Exercise</i>	132
<i>Image segmentation</i>	132
<i>Project 1: Image segmentation for our diet calculator</i>	133
Diffusion model: Unconditional image generation	134
<i>Forward diffusion</i>	135
<i>Backward diffusion</i>	136
<i>Inference process</i>	136
<i>Learnable parameters</i>	137
<i>Project 2: DogGenDiffusion</i>	137
Conclusion	137
Quiz	138
Answers.....	140
9. Speech Processing Model Anatomy: Whisper, SpeechT5, and Wav2Vec	141
Introduction	141
Structure	141
Objectives	142
System requirements	142
Speech processing	142
<i>Example of speech pre-processing</i>	143

Whisper	145
<i>Project 1: Whisper_Nep</i>	147
<i>Task</i>	147
<i>Approach</i>	147
Wav2Vec.....	147
<i>Applications of Wav2Vec</i>	150
SpeechT5.....	150
<i>Input/Output representation</i>	150
<i>Cross-modal representation</i>	152
<i>Encoder-decoder architecture</i>	152
<i>Pre-training</i>	152
<i>Fine-tuning and applications</i>	152
Comparing Whisper, Wav2Vec 2.0 and Speech T5.....	152
Conclusion	153
Quiz	153
Answers.....	156
10. Speech Tasks with Transformers.....	157
Introduction	157
Structure	158
Objectives	158
System requirements	158
Speech processing tasks	159
<i>Speech to text</i>	159
<i>Project 1: Custom audio transcription with ASR using Whisper</i>	160
Text-to-speech.....	162
<i>Project 2: Implementing text-to-Speech</i>	162
Audio to audio.....	165
<i>Project 3: Audio quality improvement through noise reduction</i>	165
Conclusion	167
Quiz	168
Answers.....	170

11. Transformer Architecture for Tabular Data Processing	171
Introduction	171
Structure	171
Objectives	172
System requirements	172
Tabular data representation using transformer	172
<i>TAPAS architecture</i>	173
<i>Pretraining objective</i>	174
<i>Fine-tuning</i>	175
<i>Applications</i>	175
<i>Example</i>	175
TabTransformer architecture	177
FT transformer architecture	179
<i>Feature tokenizer</i>	179
<i>Concatenation of numerical and categorical feature</i>	180
<i>Transformer</i>	180
Conclusion	181
Quiz	182
Answers	184
12. Transformers for Tabular Data Regression and Classification	185
Introduction	185
Structure	185
Objectives	185
System requirements	186
Transformer for classification	186
<i>Dataset</i>	187
<i>Target</i>	188
<i>Pre-process the data</i>	188
<i>Declare the configuration</i>	188
<i>Train and evaluate the model with three models</i>	190
<i>Evaluation</i>	191
<i>Analysis</i>	192

Transformer for regression	193
<i>The dataset</i>	193
<i>Pre-process the data</i>	193
<i>Define model configuration</i>	195
<i>Train and evaluate</i>	197
Conclusion	197
Quiz	198
Answers.....	200
13. Multimodal Transformers, Architectures and Applications.....	201
Introduction	201
Structure	201
Objectives	202
System requirements	202
Multimodal architecture	202
<i>ImageBind</i>	203
<i>Demonstration</i>	204
CLIP	204
<i>Pre-training objective</i>	205
<i>Applications and usage</i>	205
Multimodal tasks	205
<i>Feature extraction</i>	206
<i>Text-to-image</i>	208
<i>Image to-text</i>	210
<i>Visual question answering</i>	211
Conclusion	212
Quiz	213
Answers.....	215
14. Explore Reinforcement Learning for Transformer.....	217
Introduction	217
Structure	218
Objectives	218

System requirements	218
Reinforcement learning	218
Important techniques in PyTorch for RL	219
<i>Stable Baseline3</i>	219
<i>Gymnasium</i>	220
Project 1: Stock Market Trading with RL	221
Transformer for reinforcement learning	221
<i>Decision transformer</i>	221
<i>Trajectory transformer</i>	223
<i>Input</i>	223
Conclusion	224
Quiz	224
Answers	227
15. Model Export, Serving, and Deployment	229
Introduction	229
Structure	229
Objectives	230
System resources	230
Model export and serialization	230
<i>PyTorch model export and import</i>	231
<i>torch.save</i>	231
<i>torch.load</i>	231
<i>torch.nn.Module.load_state_dict</i>	231
<i>Saving multiple models</i>	234
Exporting model on ONNX Format	235
Serving model with FastAPI	236
<i>Benefits of FastAPI</i>	236
<i>Application of FastAPI for model serving</i>	236
<i>Project: FastAPI for semantic segmentation model serving</i>	237
Serving Pytorch model in mobile devices	238
Deploying HuggingFace's Transformers model on AWS	239

<i>Deployment using Amazon SageMaker</i>	239
<i>Deployment using AWS Lambda and Amazon API gateway</i>	240
Conclusion	241
Quiz	242
Answers.....	244
16. Transformer Model Interpretability, and Experimental Visualization	245
Introduction	245
Structure	246
Objectives	246
Explainability vs. interpretability	246
<i>Interpretability</i>	247
<i>Explainability</i>	247
Tools for explainability and interpretability	247
CAPTUM for interpreting Transformer prediction.....	248
<i>Model loading</i>	248
<i>Input preparation</i>	249
<i>Why Baseline Tensor</i>	249
<i>Layer Integrated Gradients</i>	250
<i>Visualization</i>	252
TensorBoard for PyTorch models.....	253
Conclusion	257
Quiz	258
Answers.....	260
17. PyTorch Models: Best Practices and Debugging	261
Introduction	261
Structure	262
Objectives	262
Best practices for building transformer models	262
<i>Working with Hugging Face</i>	262
<i>General consideration with Pytorch model</i>	267
The art of debugging in PyTorch	269

<i>Syntax errors</i>	269
<i>Runtime errors</i>	269
<i>Shape mismatch</i>	269
<i>CUDA errors</i>	271
<i>Loss computation issues</i>	271
<i>Mismatched configuration</i>	271
<i>Memory error</i>	271
<i>Library/Dependency errors</i>	272
<i>Logical errors</i>	273
<i>General guidelines for debugging PyTorch ML models</i>	274
<i>Conclusion</i>	275
<i>Quiz</i>	275
<i>Answers</i>	278
Index	279-285

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 2017¹. 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:

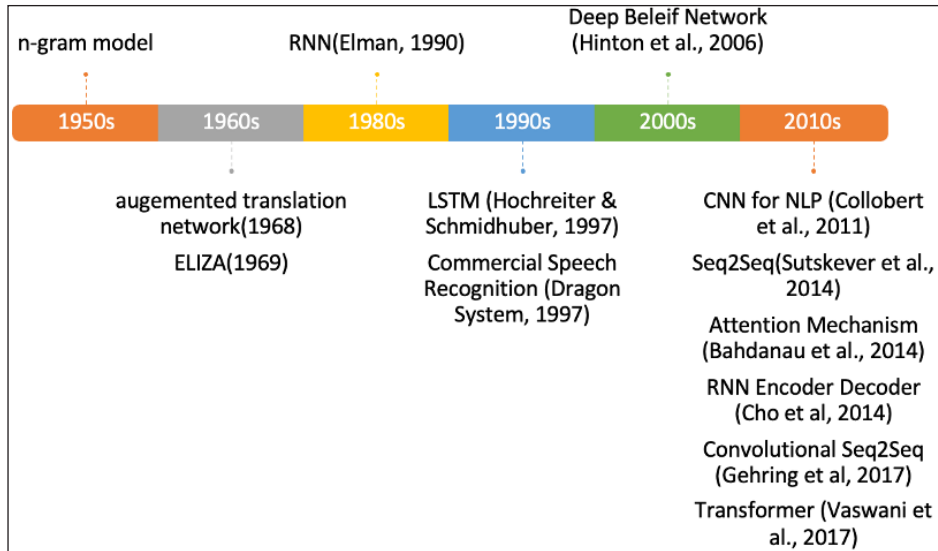


Figure 1.1: Chronology of NLP models development

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:

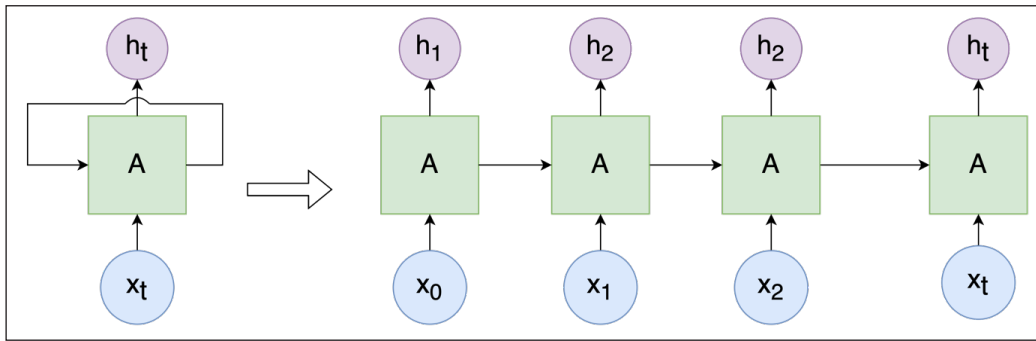


Figure 1.2: RNN structure

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:

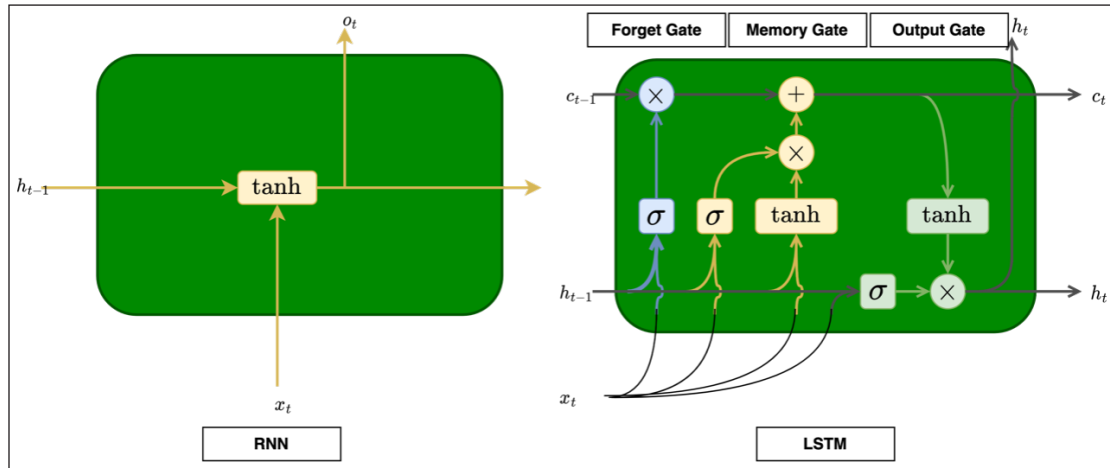


Figure 1.3: Comparison of RNN and LSTM

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 decoder²

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*.