NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

NTU Academy for Professional and Continuing Education

(SCTP) Advanced Professional Certificate

Data Science and AI

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

3.10 Natural Language Processing - Advanced

Module Overview

3.1 Probability and Statistics for Machine Learning

3.2 Introduction to Machine Learning

3.3 Supervised Learning

3.4 Supervised Learning - Advanced

3.5 Unsupervised Learning

3.6 Time Series Data & Forecasting

3.7 Neural Network & Deep Learning

3.8 Computer Vision (CV)

3.9 Natural Language Processing (NLP)

3.10 NLP - Advanced

Copyright © 2024 Nanyang Technological University. All rights reserved.

Lesson Objectives

- Deep Learning for NLP
- Recurrent Neural Networks (RNN)
- Attention Mechanism
- Transformers

Copyright © 2024 Nanyang Technological University. All rights reserved.

Common NLP Tasks

- Text Classification
- Information Extraction (e.g. NER)
- Question Answering
- Summarisation
- Text Generation
- Style Transfer
- Translation

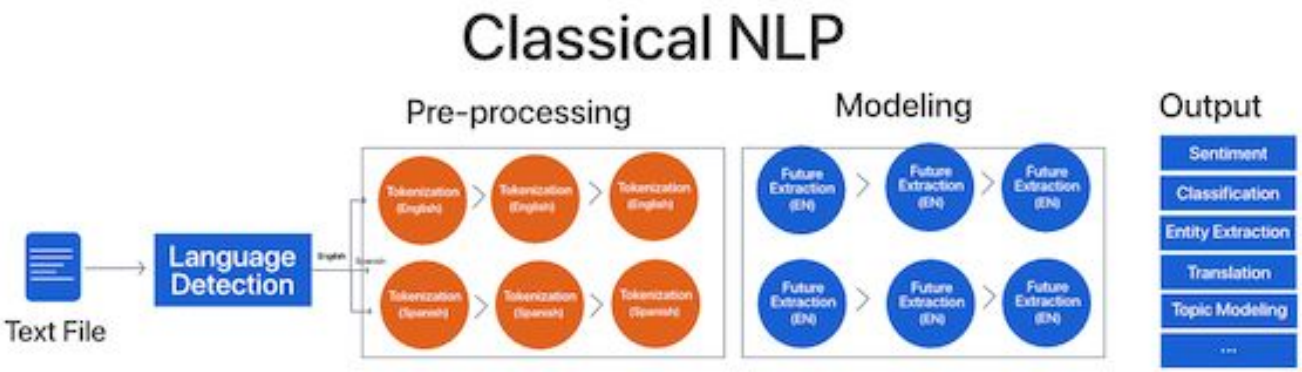
…

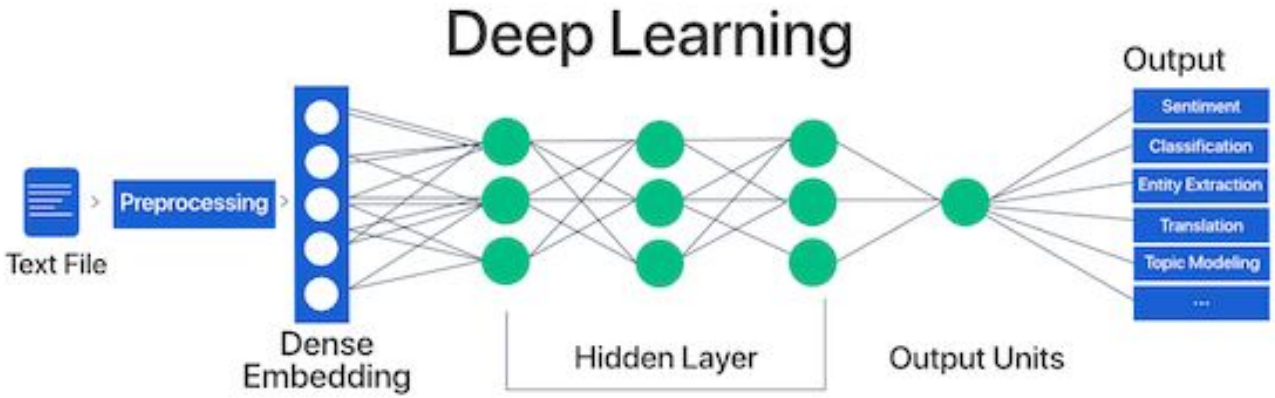
Style Attribute	*Source Attribute / Sentence*	*Target Attribute / Sentence*
Politeness	Polite: *"**Could you please** send me the data**?**"*	Impolite: *"send me the data**!!**"*
Toxicity	Offensive: *"I hope they pay out the ***."*	Non-offensive: *"I hope they pay **what they deserve**."*
Simplicity	Expert: "*Many cause **dyspnea**, pleuritic chest pain, or both."*	Layman: *"The most common symptoms, regardless of the type of fluid in the pleural space or its cause, are **shortness of breath** and chest pain."*
Biasedness	Biased: *"A new downtown is being developed **which will bring back**..."*	Neutral: *"A new downtown is being developed **which its promoters hope** will bring back..."*
Authorship	Shakespearean: *"My lord, the queen **would** speak with you**, and presently.**"*	Contemporary: *"My lord, the queen **wants to** speak with you **right away.**"*

https://blog.fastforwardlabs.com/2022/03/22/an-introduction-to-text-style-transfer.html

Copyright © 2024 Nanyang Technological University. All rights reserved.

Deep Learning for NLP





Copyright © 2024 Nanyang Technological University. All rights reserved.

Deep Learning for NLP

Limitations of Traditional Models:

Sparsity: Traditional text representations like bag-of-words result in sparse vectors that are not efficient for capturing the semantic meaning of words.

Feature Engineering: Crafting features requires domain expertise and is often labor-intensive and time-consuming.

Contextual Information: Capturing context and word order is challenging with traditional models.

Copyright © 2024 Nanyang Technological University. All rights reserved.

Deep Learning for NLP

Emergence of Deep Learning Models:

Deep learning models introduced end-to-end learning, where raw data can be fed into the model, and the model itself learns the features that are most predictive for the task.

Word Embeddings: Deep learning introduced word embeddings (e.g., Word2Vec, GloVe), which are dense vector representations of words capturing semantic meaning.

Sequence Models: RNNs and their variants like LSTMs and GRUs were developed to handle sequences and capture temporal dependencies in text.

Attention Mechanisms: Attention allows models to focus on different parts of the input sequence when predicting an output, improving performance on tasks like machine translation.

Transformer Models: The Transformer architecture, which relies solely on attention without recurrence, has achieved state-of-the-art results in many NLP tasks.

Copyright © 2024 Nanyang Technological University. All rights reserved.

Text Generation

Mary → had

Mary had → a

Mary had a → little

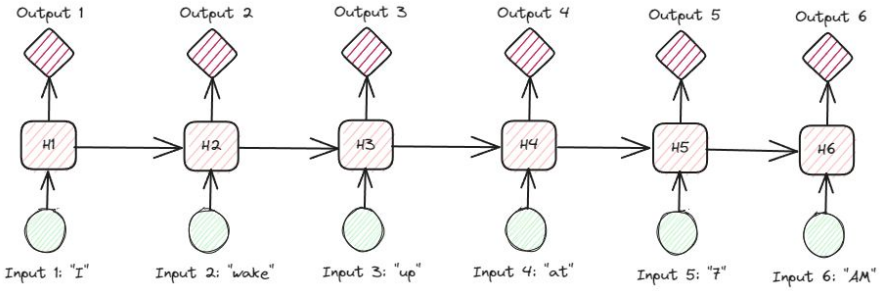
Mary had a little → lamb

Copyright © 2024 Nanyang Technological University. All rights reserved.

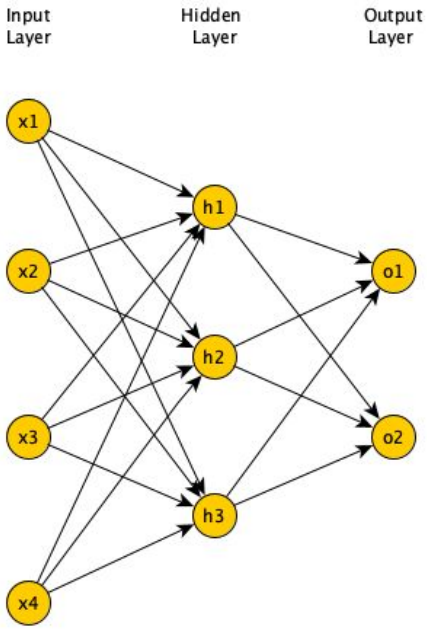
Why do we need RNNs?

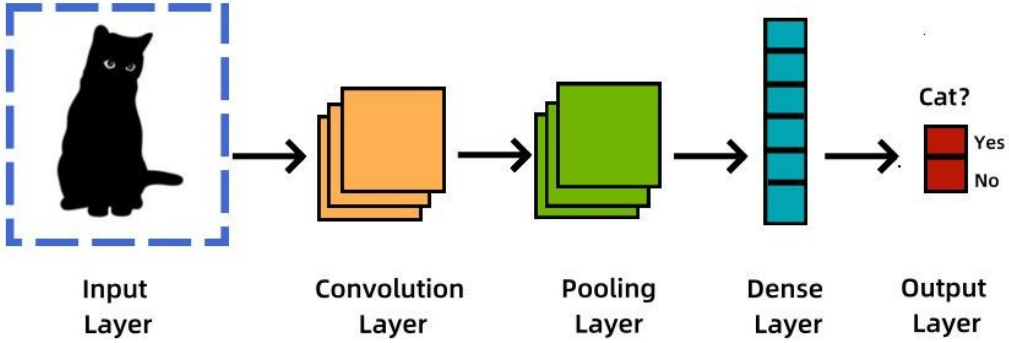
Feedforward Neural Networks (FNNs) takes inputs and process each input independently through a number of hidden layers **without considering the order and context of other inputs**. Due to which it is unable to handle sequential data effectively and capture the dependencies between inputs.

As a result, FNNs are not well-suited for sequential processing tasks such as, language modeling, machine translation, speech recognition, time series analysis, and many other applications that requires sequential processing. **To address the limitations posed by traditional neural networks, RNN comes into the picture.**



Copyright © 2024 Nanyang Technological University. All rights reserved.





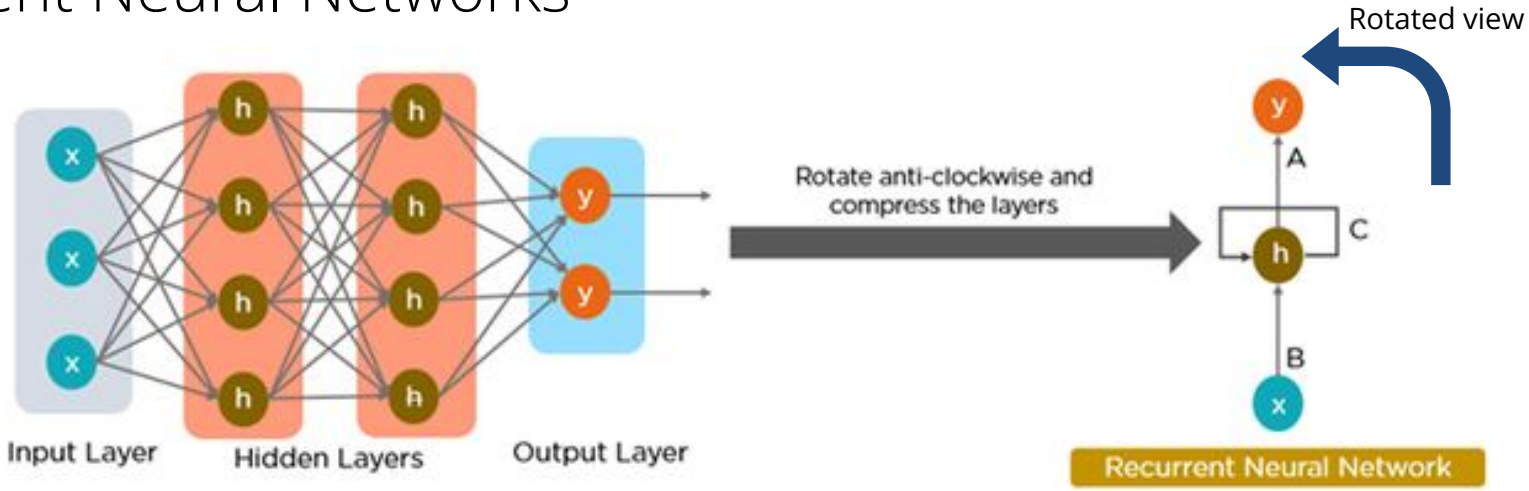
Feedforward networks lack Spatial awareness → **CNN**

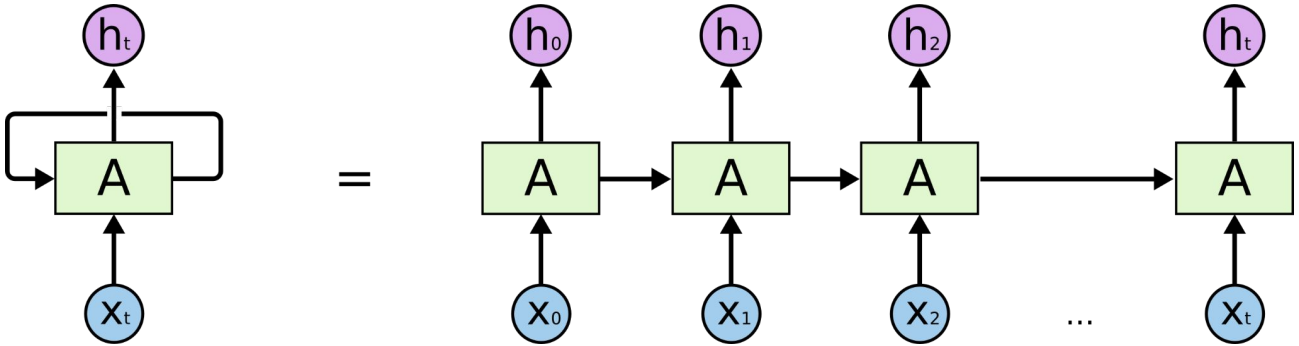
Recurrent Neural Networks

Feedforward networks lack memory → **RNN**

Copyright © 2024 Nanyang Technological University. All rights reserved.

Recurrent Neural Networks





https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Copyright © 2024 Nanyang Technological University. All rights reserved.

Recurrent Neural Network

Step 1: Prepare the Data

You have a big corpus of text (like Penn Treebank).

- Tokenize the text → turn words into integers (using a tokenizer + vocab)
- Create sequences of tokens → input and target pairs

Example:

Suppose the tokenized data is:

["The", "quick", "brown", "fox", "jumps", "over"]

Inputs

```
["The", "quick", "brown", "fox", "jumps"]
```

Targets

```
["quick", "brown", "fox", "jumps", "over"]
```

Copyright © 2024 Nanyang Technological University. All rights reserved.

Recurrent Neural Network

Step 2: Initialize the Model

Create an RNN model:

- **Embedding layer** → to map tokens into dense vectors.
- **RNN layer** → to process sequences.
- Fully connected layer → to map hidden states to vocab predictions.

Also set up:

- Loss function → typically CrossEntropyLoss (for classification).
- Optimizer → like Adam (to update weights).

Define vocab size and embedding size

Memory: RNNs maintain a hidden state (or memory) at each time step, which is passed to the next time step.

This hidden state allows the network to retain information about past inputs, which is crucial for learning sequential data (e.g., time series, language).

Typically - tanh activation function

Copyright © 2024 Nanyang Technological University. All rights reserved.

Recurrent Neural Network

Step 3: Set the Initial Hidden State

Before feeding each batch into the RNN, initialize the hidden state:

```
hidden = model.init_hidden(batch_size)
```

- This is usually a tensor of zeros at the start.

Step 4: Feed a Batch of Inputs

Step 5: Calculate the Loss

Step 6: Backpropagate the Loss

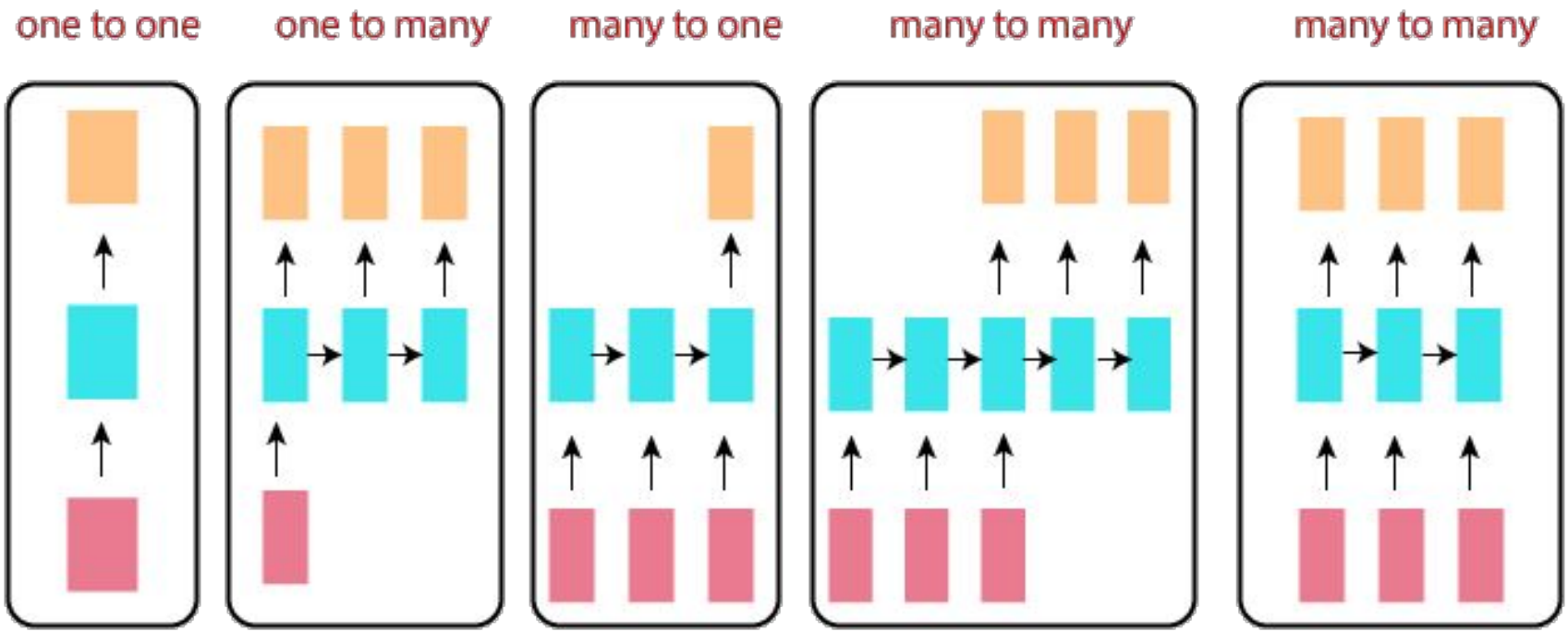
Step 7: Update the Model Weights

Step 8: Repeat for Many Epochs

RNNs maintain a hidden state (or memory) that gets updated at each time step based on the input sequence, and it needs to be initialized before processing the first batch.

Copyright © 2024 Nanyang Technological University. All rights reserved.

Types of RNNs



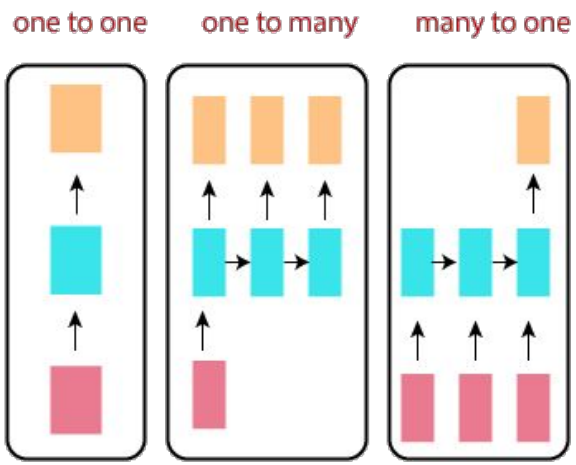
Copyright © 2024 Nanyang Technological University. All rights reserved.

Types of RNNs

- **One-to-One**: One input → One output (simple classification).
  This is the simplest scenario where a single input is processed to produce a single output, similar to a standard feedforward neural network. [this is not RNN]

- **One-to-Many**: One input → Many outputs (e.g., image captioning).
  A single input is used to generate a sequence of outputs. Essentially, the RNN processes the single input and then "decodes" it into multiple outputs over time.

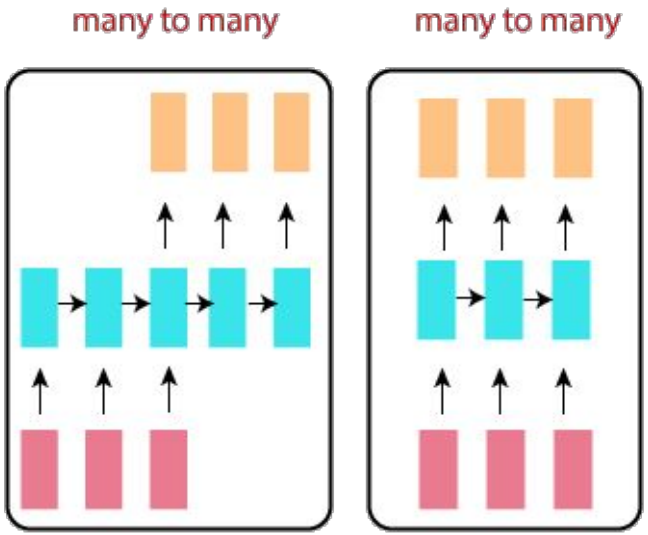
- **Many-to-One**: Many inputs → One output (e.g., sentiment analysis).
  a sequence of inputs is processed to produce a single output. The RNN takes an entire sequence as input (e.g., a time series or a sentence) and generates a single output after processing the entire sequence.



Copyright © 2024 Nanyang Technological University. All rights reserved.

Types of RNNs

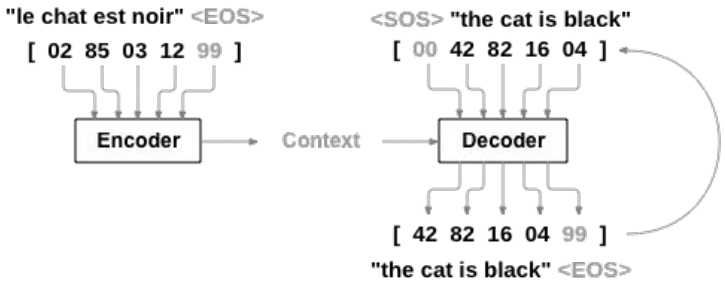
- **Encoder-Decoder (Many-to-Many)**: Useful for sequence transformation tasks where the input and output sequences may have different lengths (e.g., machine translation, summarization).
- **Direct (Many-to-Many)**: Useful when the input and output sequences are of the same length, and each element of the output corresponds to an element of the input (e.g., NER, POS tagging).



Copyright © 2024 Nanyang Technological University. All rights reserved.

Types of RNNs

- **Encoder-Decoder (Many-to-Many)**: Useful for sequence transformation tasks where the input and output sequences may have different lengths (e.g., machine translation, summarization).
- **Direct (Many-to-Many)**: Useful when the input and output sequences are of the same length, and each element of the output corresponds to an element of the input (e.g., NER, POS tagging).



Copyright © 2024 Nanyang Technological University. All rights reserved.

Code-along

Jupyter Notebook

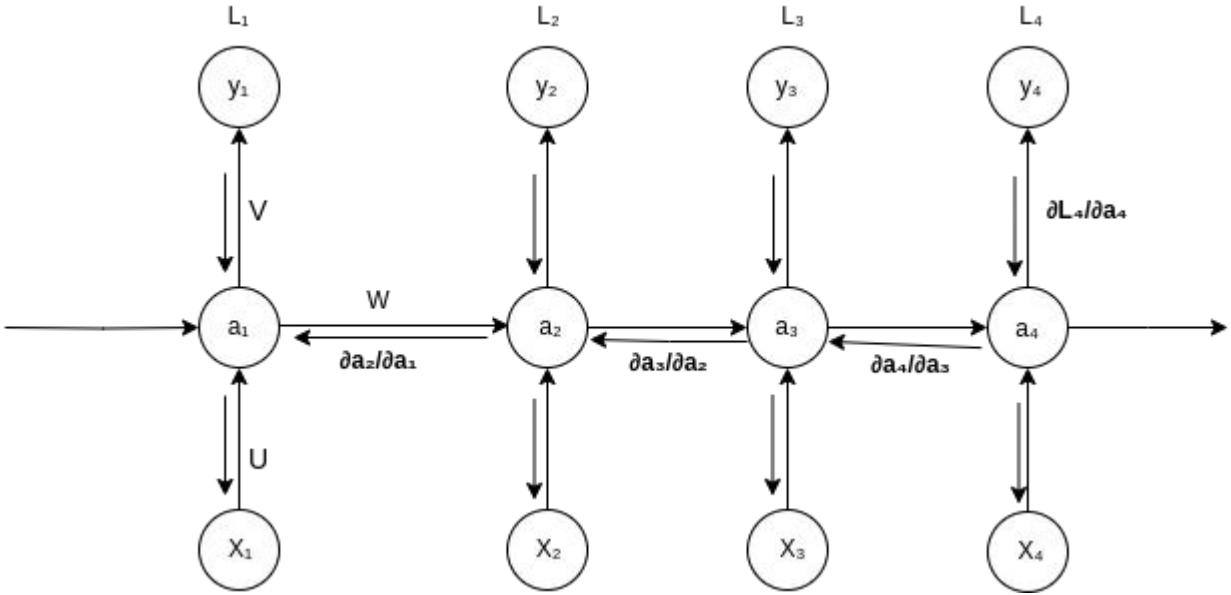
Part 1: Recurrent Neural Networks for Text Generation

Copyright © 2024 Nanyang Technological University. All rights reserved.

Backpropagation Through Time

The key difference is that the gradients are "propagated back in time" across multiple time steps, not just one layer like in standard networks.

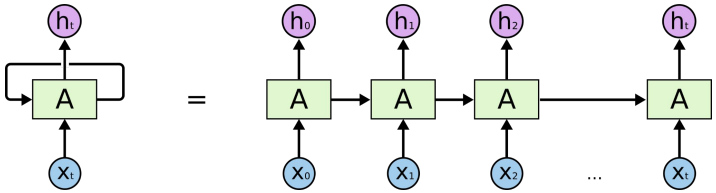
This allows the RNN to learn from long sequences, but it also means that it's harder to train on long sequences due to issues like vanishing or exploding gradients.



[Source link: Medium article](#) (RNN Architecture Explained)

Copyright © 2024 Nanyang Technological University. All rights reserved.

Vanishing Gradient intuition



$$h_t = \sigma(wh_{t-1}).$$

$$\frac{\partial h_{t'}}{\partial h_t} = \prod_{k=1}^{t'-t} w\sigma'(wh_{t'-k})$$

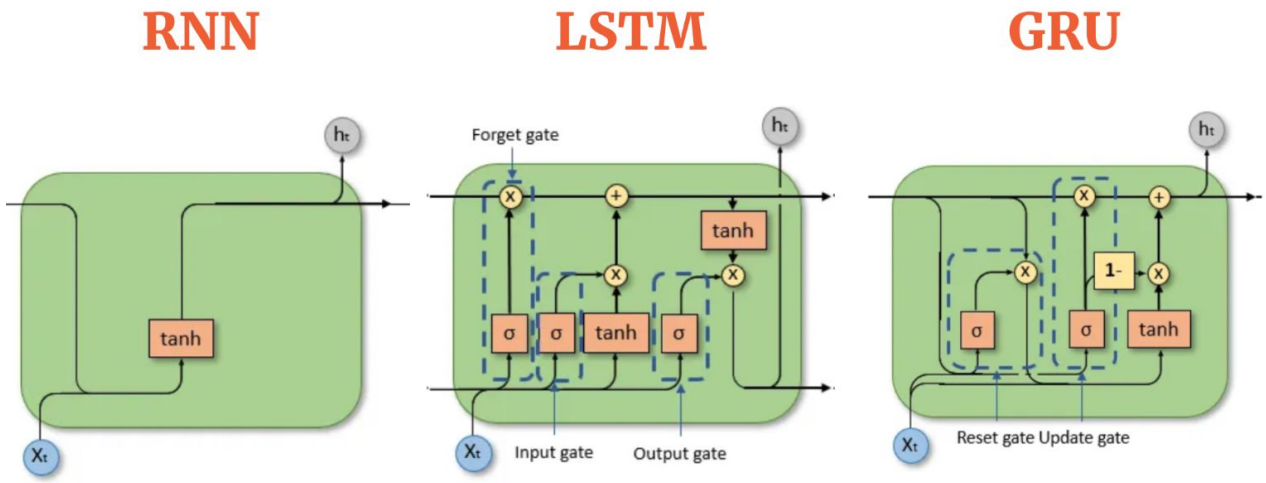
$$= \underbrace{w^{t'-t}}_{!!!} \prod_{k=1}^{t'-t} \sigma'(wh_{t'-k})$$

If W < 1: vanishing gradient
If W > 1: exploding gradient

Copyright © 2024 Nanyang Technological University. All rights reserved.

RNN Variations

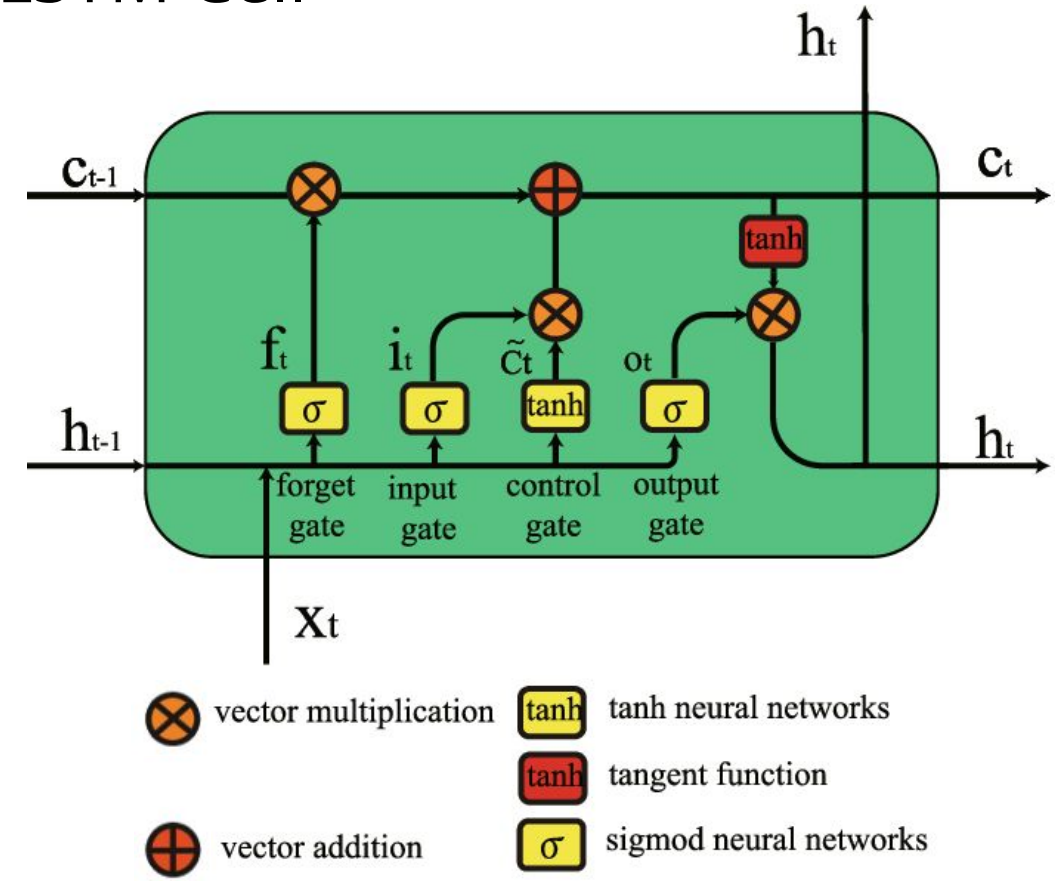
LSTM is better than RNN because it can remember long-term dependencies with its memory cells, reducing the vanishing gradient problem, while GRU is better than RNN because it uses fewer gates and is computationally more efficient while still handling long-term dependencies well.



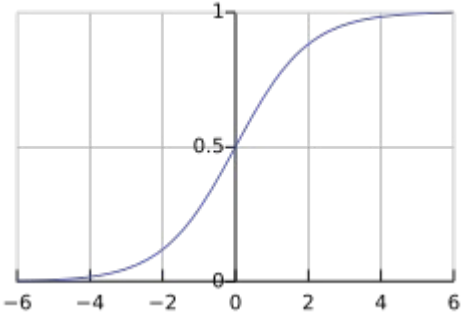
[Source Article: Sequence Models]

Copyright © 2024 Nanyang Technological University. All rights reserved.

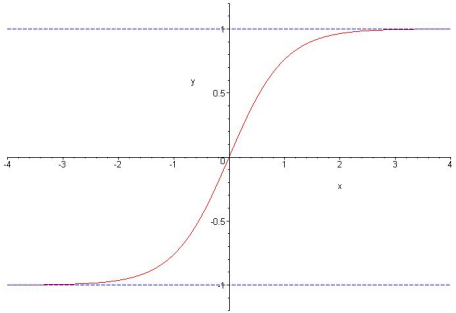
LSTM Cell



Sigmoid



tanh



Copyright © 2024 Nanyang Technological University. All rights reserved.

LSTM Cell

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

i_t, f_t, g_t, o_t are the input, forget, cell, and output gates,

Copyright © 2024 Nanyang Technological University. All rights reserved.

LSTM Cell

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

i_t, f_t, g_t, o_t are the input, forget, cell, and output gates,

Copyright © 2024 Nanyang Technological University. All rights reserved.

LSTM Cell

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

i_t, f_t, g_t, o_t are the input, forget, cell, and output gates,

Copyright © 2024 Nanyang Technological University. All rights reserved.

LSTM Cell

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

i_t, f_t, g_t, o_t are the input, forget, cell, and output gates,

Copyright © 2024 Nanyang Technological University. All rights reserved.

LSTM Cell

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= \boxed{o_t} \odot \tanh(c_t)
\end{aligned}
$$

i_t, f_t, g_t, o_t are the input, forget, cell, and output gates,

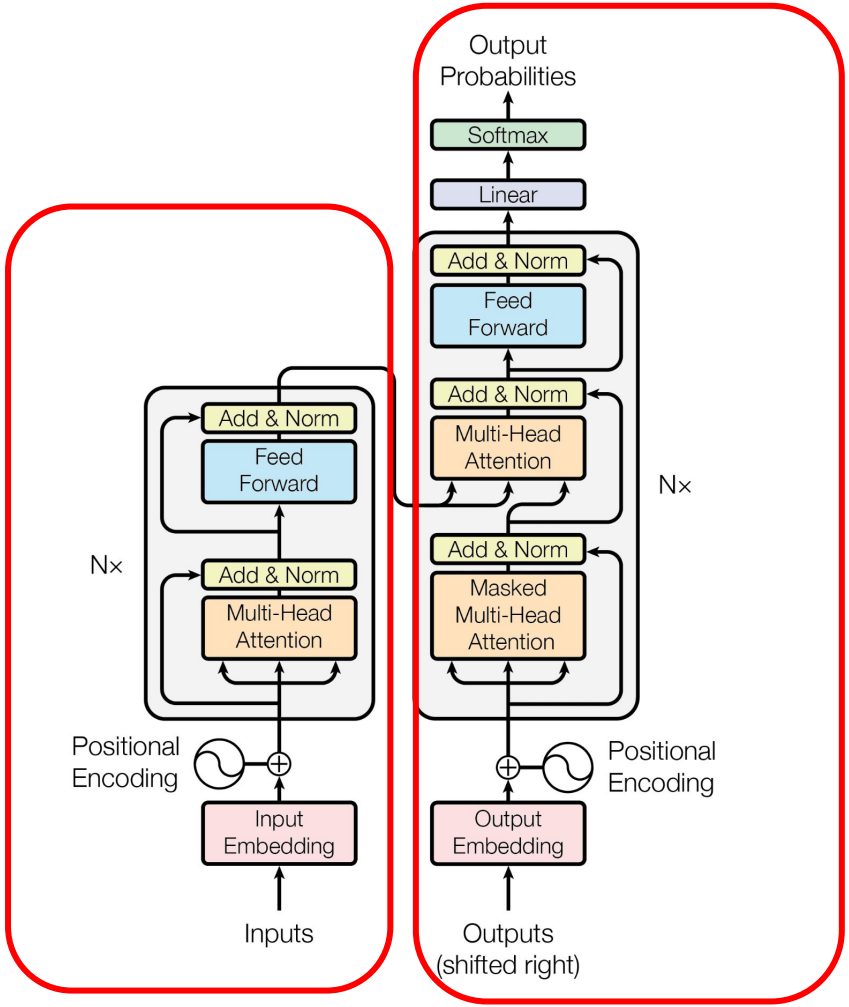
Copyright © 2024 Nanyang Technological University. All rights reserved.

Transformers

Copyright © 2024 Nanyang Technological University. All rights reserved.

Transformers

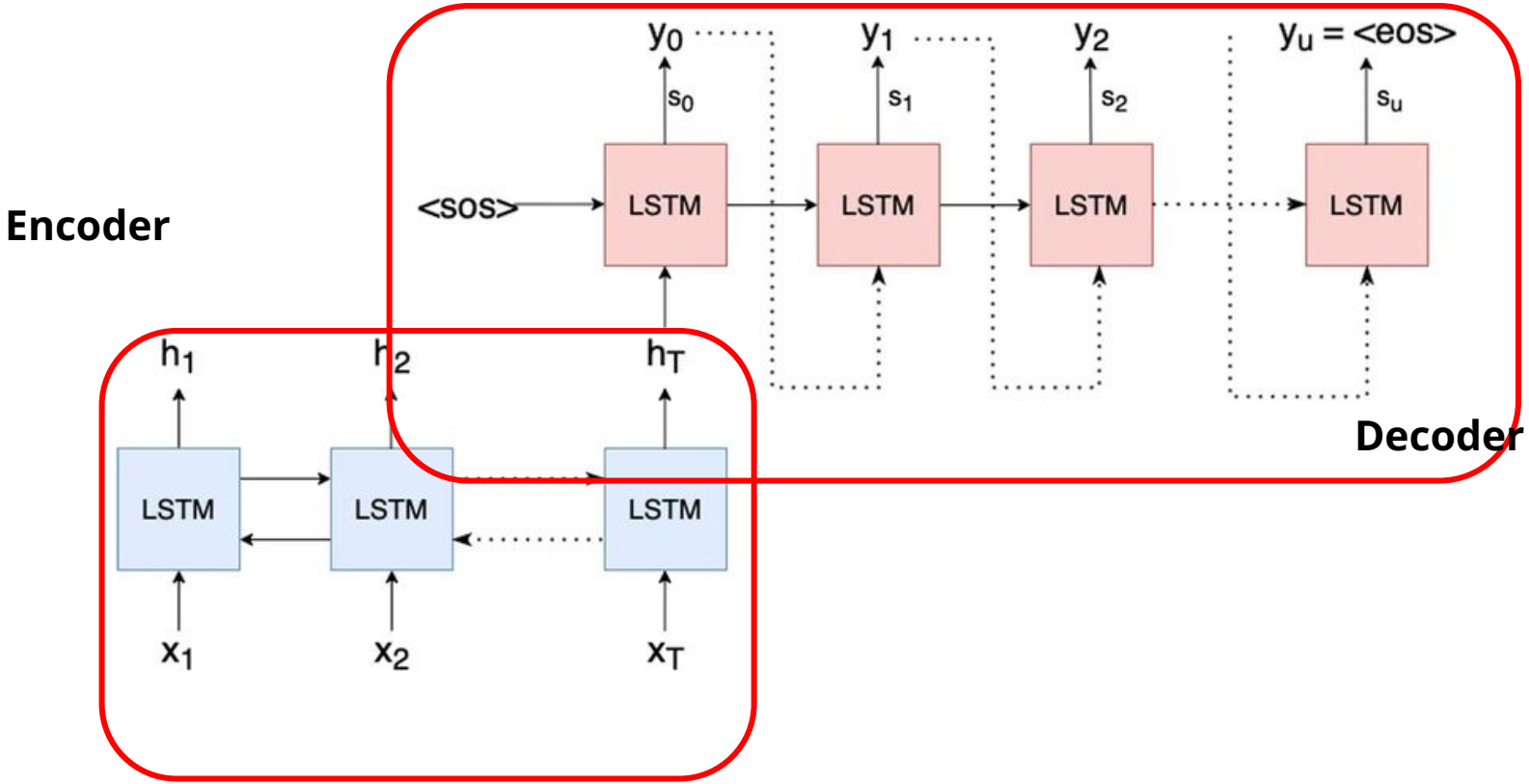
Encoder



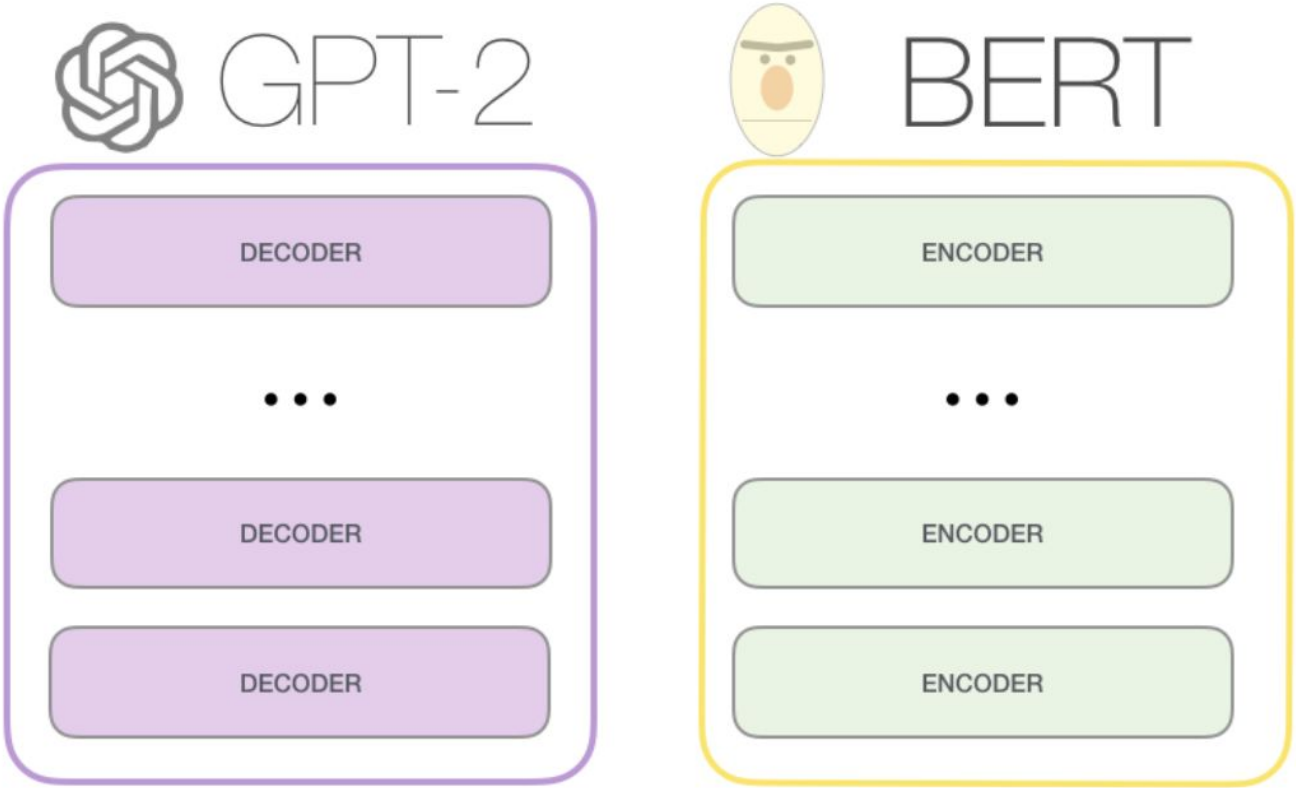
Decoder

Copyright © 2024 Nanyang Technological University. All rights reserved.

Transformers vs seq2seq

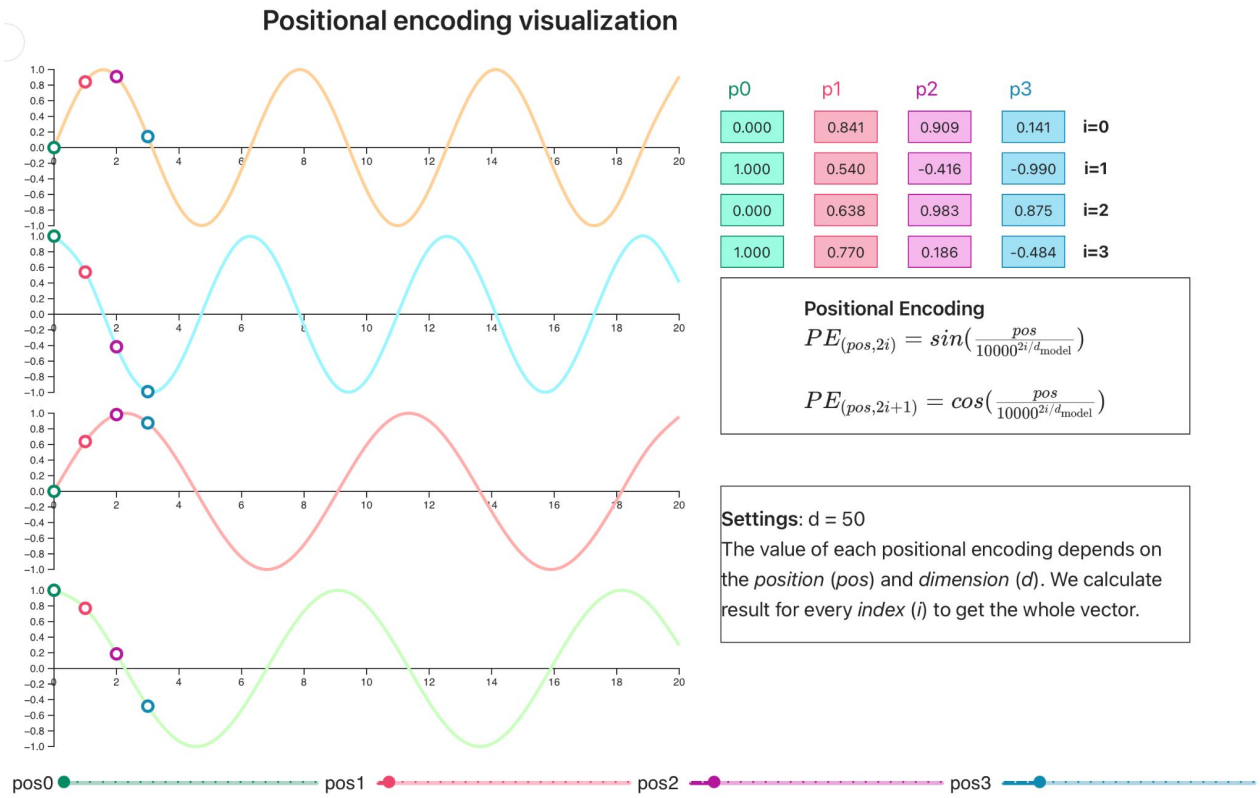


Copyright © 2024 Nanyang Technological University. All rights reserved.



Copyright © 2024 Nanyang Technological University. All rights reserved.

Positional encoding - sinusoidal



Encode from token -> (N, tokens, D)

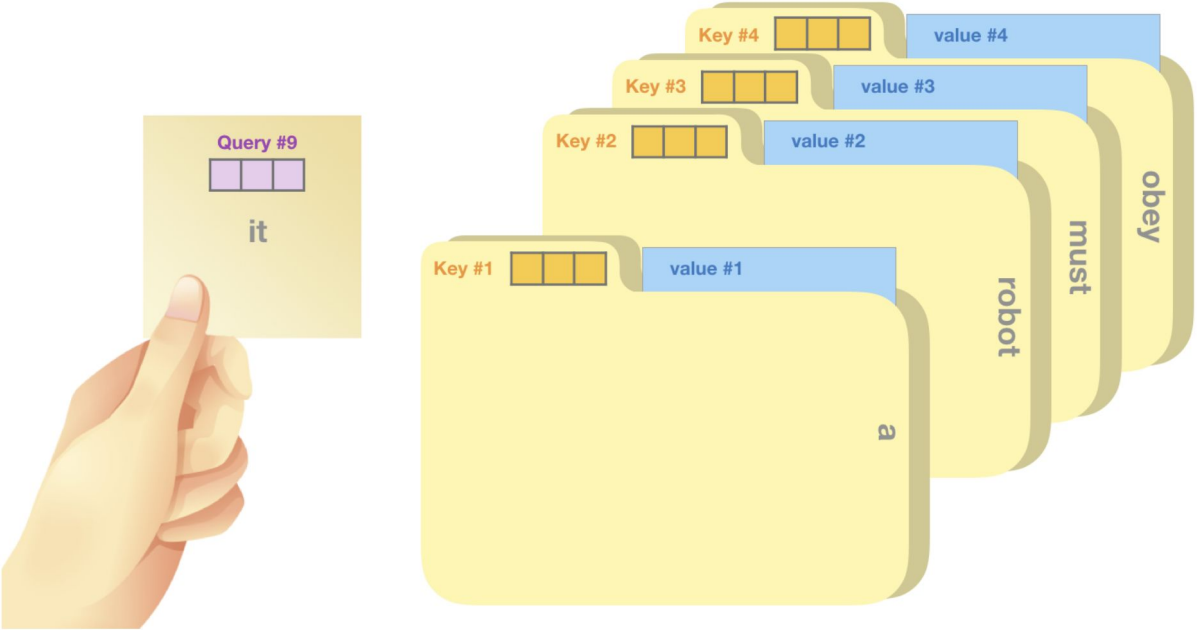
We generate for 1 sample: (tokens, D)

https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers

Copyright © 2024 Nanyang Technological University. All rights reserved.

Self attention

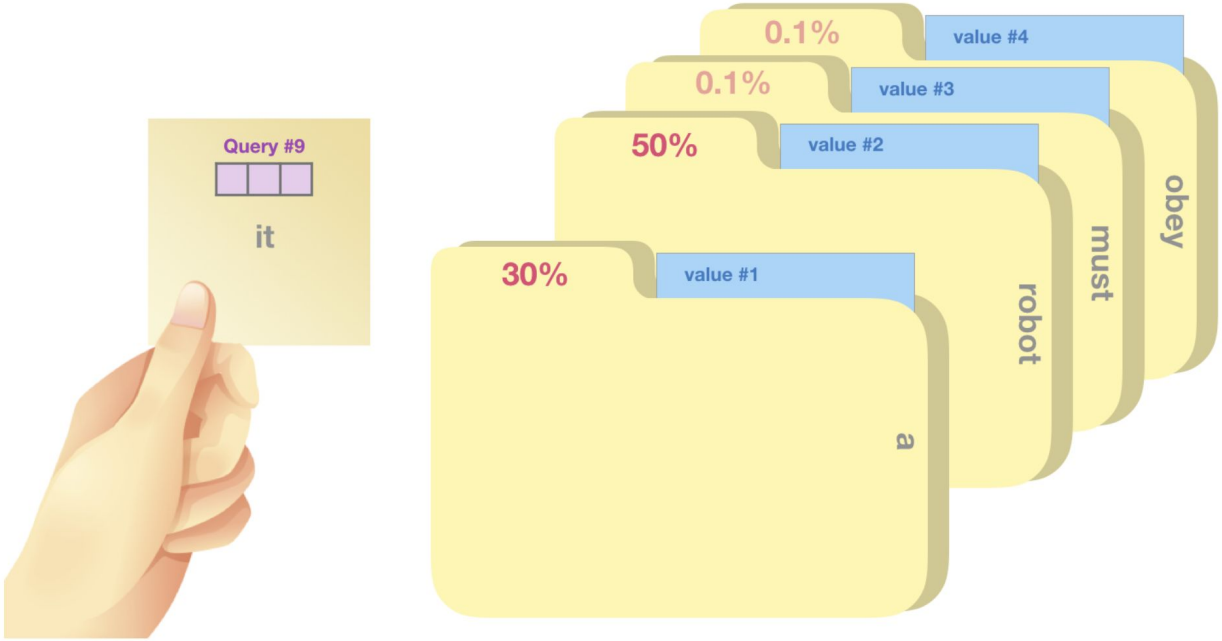
> ***Second Law of Robotics***
> *A robot must obey the orders given **it** by human beings except where **such orders** would conflict with the **First Law**.*



Copyright © 2024 Nanyang Technological University. All rights reserved.

Self attention

> ***Second Law of Robotics***
> *A robot must obey the orders given **it** by human beings except where **such orders** would conflict with the **First Law**.*



Copyright © 2024 Nanyang Technological University. All rights reserved.

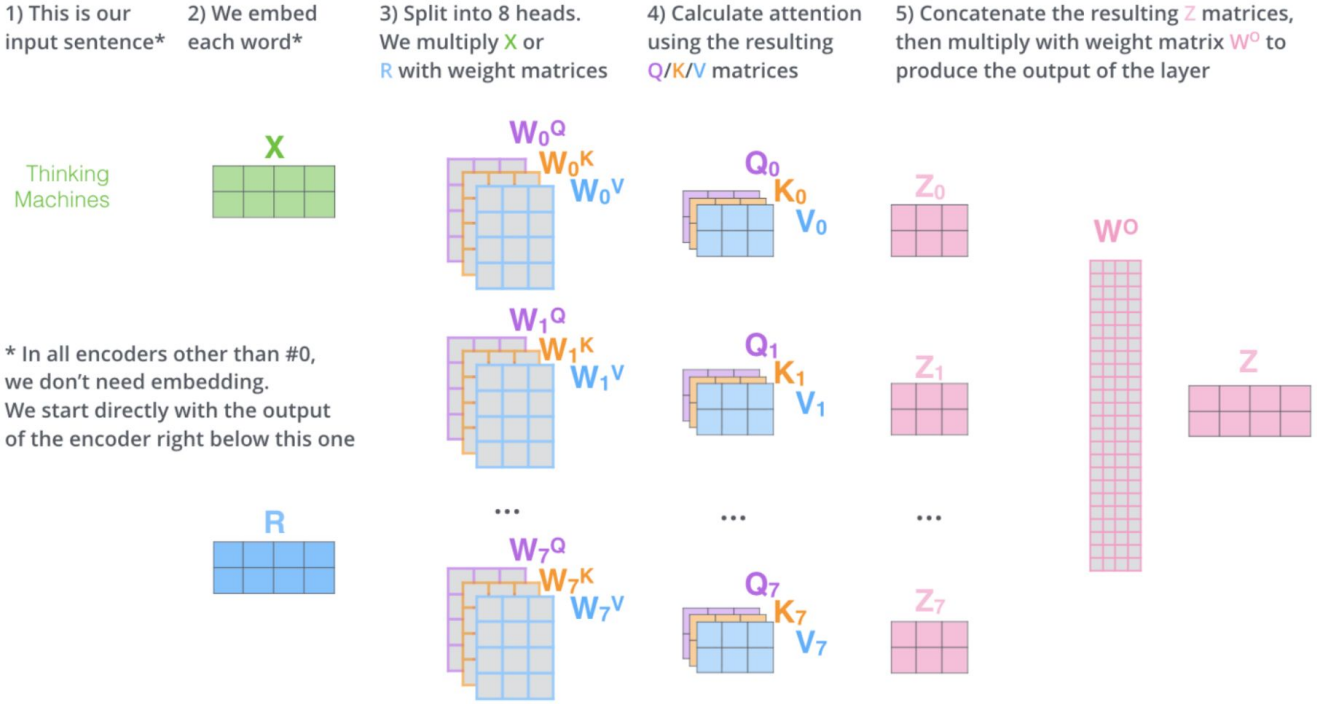
Self attention

> ***Second Law of Robotics***
> *A robot must obey the orders given **it** by human beings except where **such orders** would conflict with the **First Law**.*

Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	

Copyright © 2024 Nanyang Technological University. All rights reserved.

Multi headed self-attention



Copyright © 2024 Nanyang Technological University. All rights reserved.

Self attention self read

https://jalammar.github.io/illustrated-gpt2/

https://jalammar.github.io/illustrated-transformer/

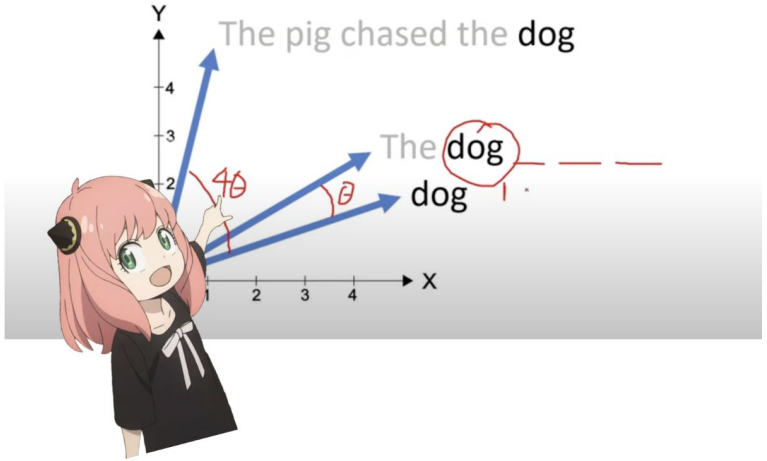
Copyright © 2024 Nanyang Technological University. All rights reserved.

Modern: RoPE

For token at position p, we apply rotation matrix R(p). We rotate the vector by an angle proportion to p

Apply rotation to Query and Key vector and attention is computed

Rotary Positional Embeddings



1. Good for long sequences
2. Can extrapolate/ generalize to longer sequences -> good for large context window
3. Complements with attention: attention captures relative position while RoPE captures absolute position
4. Modifies query and key vector before attention is computed

Copyright © 2024 Nanyang Technological University. All rights reserved.

Language Processing with BERT

The 3 minute intro
(with example applications)

Copyright © 2024 Nanyang Technological University. All rights reserved.

BERT

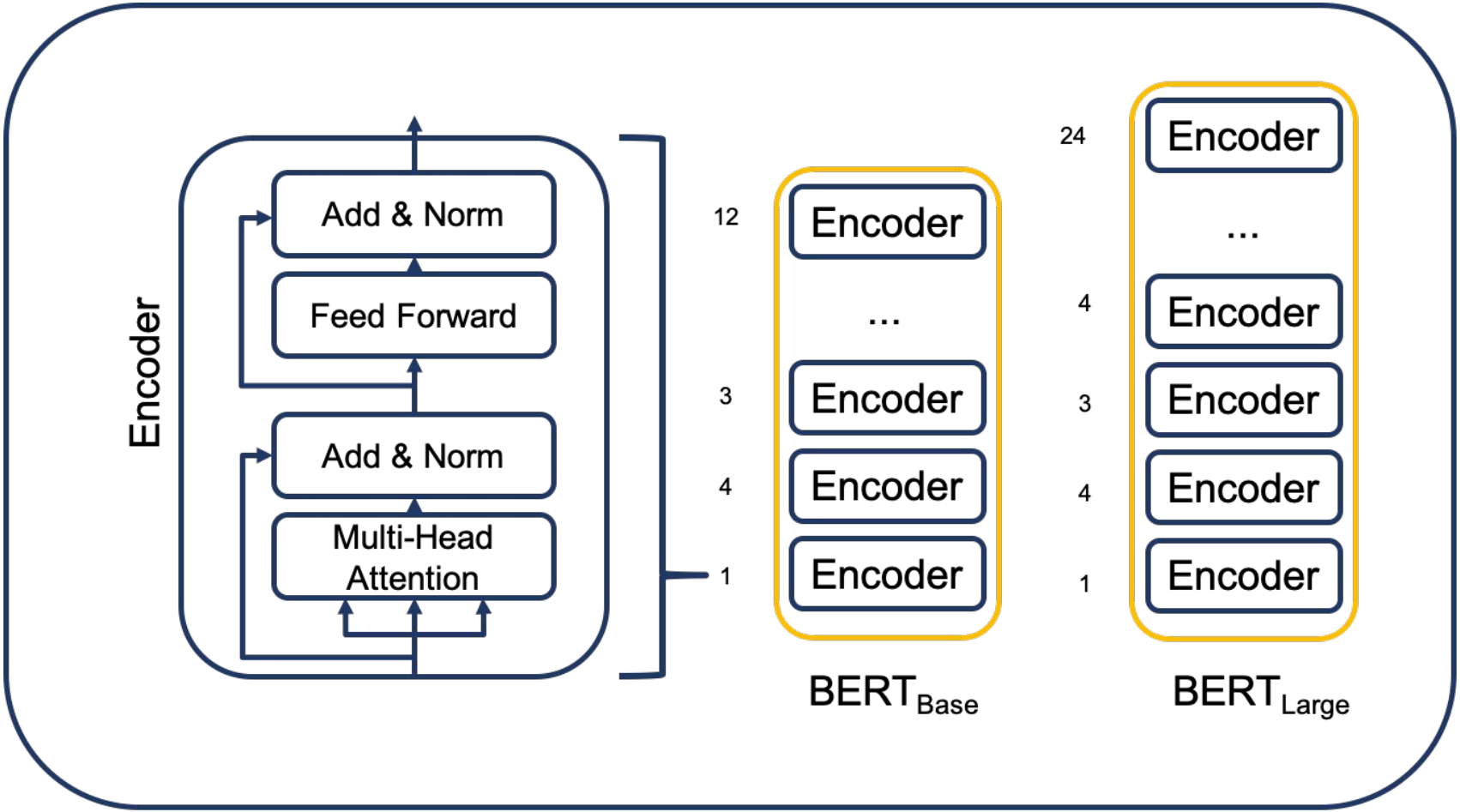
BERT (standing for Bidirectional Encoder Representations from Transformers) is an open-source model developed by Google in 2018

Google's release of BERT, an open-source natural language processing framework, revolutionized NLP with its unique bidirectional training, which enables the model to have more context-informed predictions about what the next word should be.

By understanding context from all sides of a word, BERT outperformed previous models in tasks like question-answering and understanding ambiguous language. Its core uses Transformers, connecting each output and input element dynamically.

BERT, pre-trained on Wikipedia, excelled in various NLP tasks, prompting Google to integrate it into its search engine for more natural queries. This innovation sparked a race to develop advanced language models and significantly advanced the field's ability to handle complex language understanding.

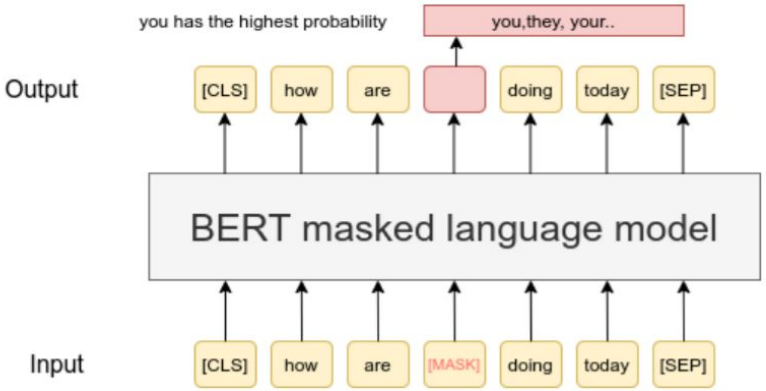
Copyright © 2024 Nanyang Technological University. All rights reserved.



https://pub.towardsai.net/demystifying-bert-efe9ac3c1c74

Copyright © 2024 Nanyang Technological University. All rights reserved.

BERT Pretraining



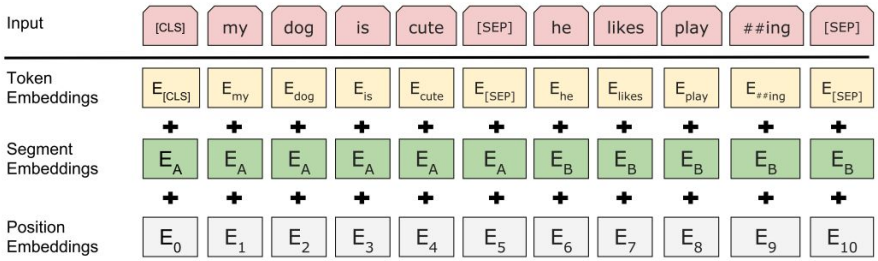


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Masked Language Modeling

Next Sentence Prediction

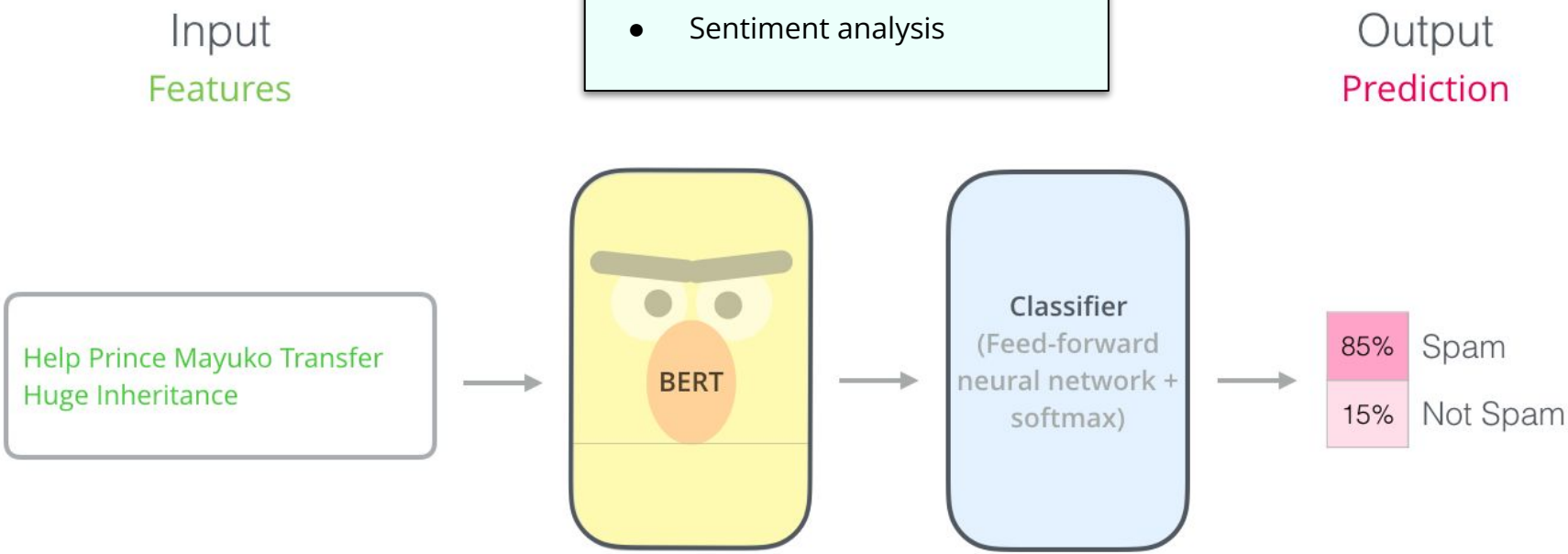
Model		Parameters	Layers	Hidden	Embedding
BERT	base	108M	12	768	768
	large	334M	24	1024	1024
	xlarge	1270M	24	2048	2048

Copyright © 2024 Nanyang Technological University. All rights reserved.

BERT in action - Finetuning

BERT has been successfully applied to many NLP tasks, including:

- Text classification
- Question answering
- Named entity recognition
- Sentiment analysis



https://jalammar.github.io/illustrated-bert/

Copyright © 2024 Nanyang Technological University. All rights reserved.

BERT Model Variants: Comparison & Use Cases

BERT

BASE MODEL
Bidirectional Encoder Representations from Transformers (2018)

PARAMETERS
BERT-base: 110M | BERT-large: 340M

KEY FEATURES
- Pre-trained on MLM and NSP
- Bidirectional context
- Original transformer architecture

STRENGTHS
- Strong baseline performance
- Well-documented
- Extensive support

LIMITATIONS
- Static masking
- NSP adds complexity
- Slower training

WHEN TO USE
- General-purpose NLP

RoBERTa

FULL NAME
Robustly Optimized BERT Approach (2019)

PARAMETERS
RoBERTa-base: 125M | RoBERTa-large: 355M

KEY FEATURES
- Dynamic masking patterns
- Removes NSP task
- Trained on 10× more data

STRENGTHS
- Best overall accuracy
- Improved training
- Better generalization

LIMITATIONS
- Higher compute needs
- Longer training
- Larger model size

WHEN TO USE
- Max accuracy priority

DistilBERT

FULL NAME
Distilled BERT (2019)

PARAMETERS
66M (40% smaller than BERT-base)

KEY FEATURES
- Knowledge distillation
- 6 layers vs 12
- 97% of BERT performance
- 60% faster inference

STRENGTHS
- Great speed/accuracy trade-off
- Lower memory footprint
- Fast inference
- Production-ready

LIMITATIONS
- 3% accuracy reduction
- Not for complex tasks
- Less context depth

WHEN TO USE
- Real-time apps
- Speed critical

ALBERT

FULL NAME
A Lite BERT (2019)

PARAMETERS
ALBERT-base: 12M | xxlarge: 235M

KEY FEATURES
- Factorized embeddings
- Cross-layer sharing
- SOP instead of NSP
- Reduced parameters

STRENGTHS
- Smallest param count
- Memory efficient
- Scales well
- Good despite size

LIMITATIONS
- Slower than DistilBERT
- Sharing adds compute
- May underperform

WHEN TO USE
- Very limited memory

Copyright © 2024 Nanyang Technological University. All rights reserved.

Code-along

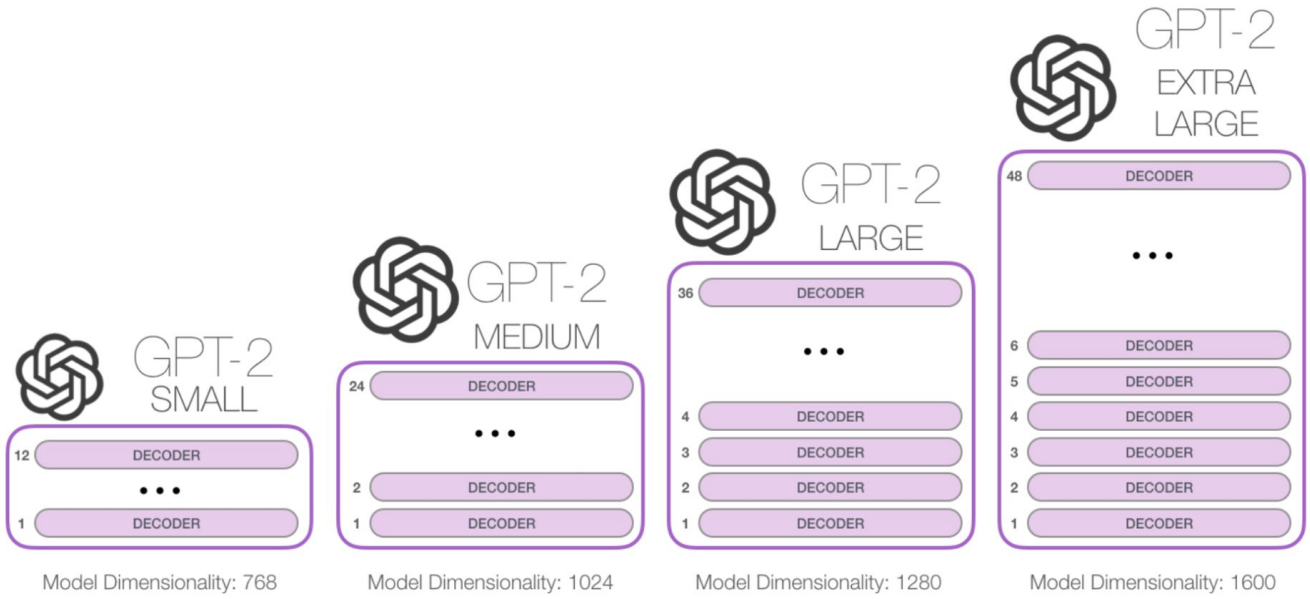
Jupyter Notebook

Part 2: Sentiment Analysis and NER with BERT

Copyright © 2024 Nanyang Technological University. All rights reserved.

GPT - Generative Pretrained Transformer

The GPT (Generative Pretrained Transformer) models are a series of deep learning models developed by OpenAI. These models are designed to generate human-like text by predicting the next word in a sequence of words. The architecture of GPT is similar to the decoder part of the Transformer model. It uses masked self-attention, where each token can only attend to previous tokens in the self-attention layers of the transformer.



Copyright © 2024 Nanyang Technological University. All rights reserved.

Masked attention

	Features				Labels
position:	1	2	3	4	
Example: 1	robot	must	obey	orders	must
2	robot	must	obey	orders	obey
3	robot	must	obey	orders	orders
4	robot	must	obey	orders	<eos>

Copyright © 2024 Nanyang Technological University. All rights reserved.

Masked attention

Features

Example:	position: 1	2	3	4
1	robot	must	obey	orders
2	robot	must	obey	orders
3	robot	must	obey	orders
4	robot	must	obey	orders

Scores
(before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Apply Attention Mask →

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

Then, applying softmax on each row produces the actual scores we use for self-attention:

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

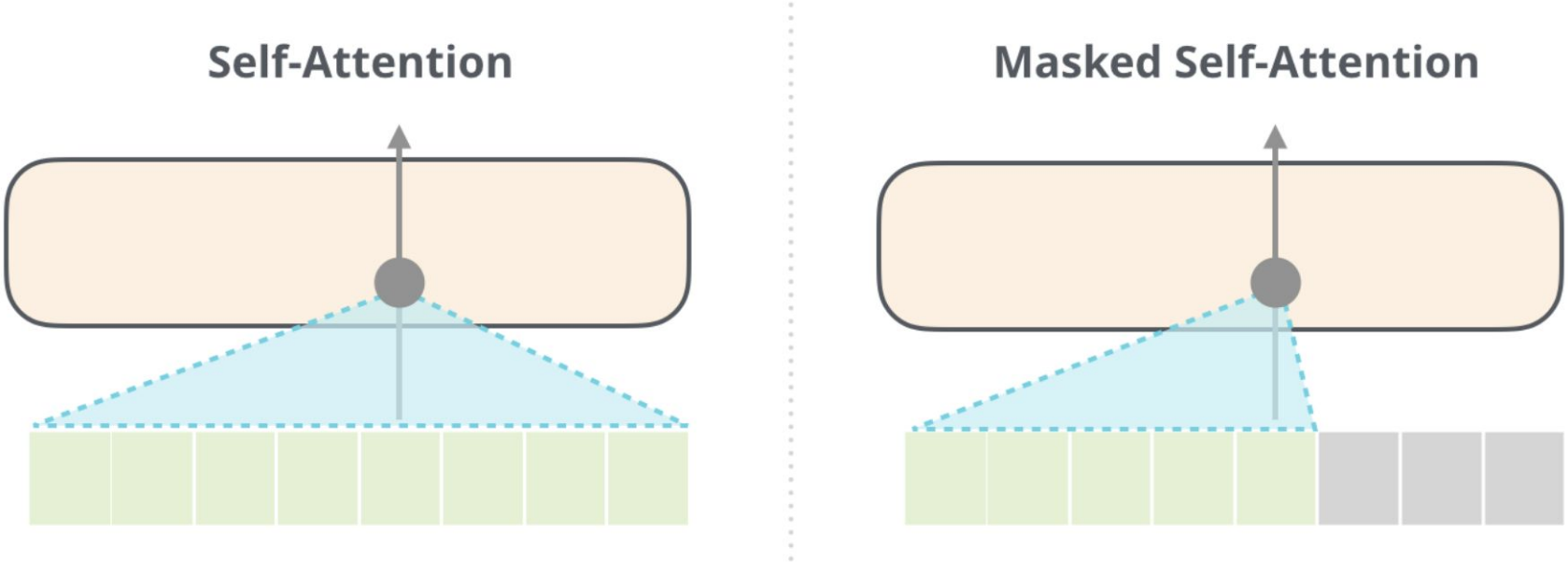
Softmax
(along rows) →

Scores

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

Copyright © 2024 Nanyang Technological University. All rights reserved.

Masked vs bidirectional



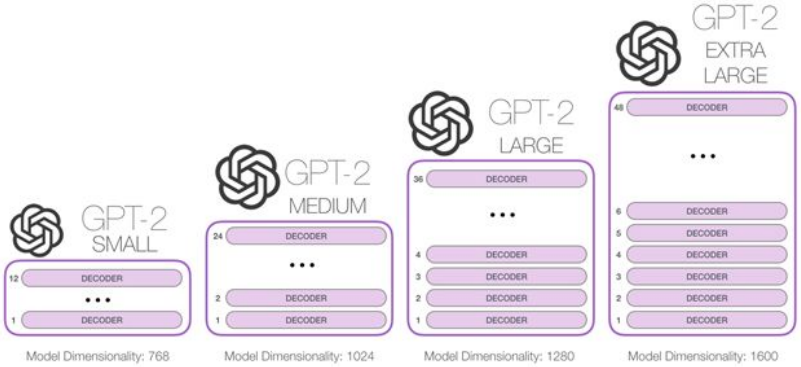
Copyright © 2024 Nanyang Technological University. All rights reserved.

GPT-2

We've trained a large-scale unsupervised language model which generates coherent paragraphs of text, achieves state-of-the-art performance on many language modeling benchmarks, and performs rudimentary reading comprehension, machine translation, question answering, and summarization—all without task-specific training.

Our model, called GPT-2 (a successor to GPT), was trained simply to predict the next word in 40GB of Internet text. Due to our concerns about malicious applications of the technology, we are not releasing the trained model. As an experiment in responsible disclosure, we are instead releasing a much smaller model for researchers to experiment with, as well as a technical paper.

GPT-2 is a large transformer-based language model with 1.5 billion parameters, trained on a dataset[A] of 8 million web pages. GPT-2 is trained with a simple objective: predict the next word, given all of the previous words within some text. The diversity of the dataset causes this simple goal to contain naturally occurring demonstrations of many tasks across diverse domains. GPT-2 is a direct scale-up of GPT, with more than 10X the parameters and trained on more than 10X the amount of data.



https://openai.com/index/better-language-models/#sample1

Copyright © 2024 Nanyang Technological University. All rights reserved.

GPT Evolution

Key Features Across All GPT Models:

- **Transformer Architecture:** All GPT models use the Transformer architecture, which allows for better handling of sequential data and long-range dependencies in text.
- **Pretraining and Fine-tuning:** These models are pre-trained on large datasets and can then be fine-tuned for specific tasks, making them highly adaptable.
- **Language Generation:** The models are generative, meaning they create text based on patterns and context learned during training.

https://platform.openai.com/tokenizer

https://medium.com/@vipul.koti333/evolution-of-gpt-models-gpt-1-to-gpt-4-0238ee07a29b

Feature	GPT-1	GPT-2	GPT-3	GPT-4
Year	2018	2019	2020	2023
Paper Title	"Improving Language Understanding by Generative Pre-training"	"Language Models are Unsupervised Multitask Learners"	"Language Models are Few-Shot Learners"	"GPT-4 Technical Report"
Parameters	117M	1.5B	175B	Undisclosed but larger than GPT-3
Architecture	12-layer transformer decoder	48-layer transformer decoder	96-layer transformer	Optimized transformer architecture, multimodal capabilities
Context Window	512 tokens	1024 tokens	2048 tokens	Up to 128k tokens depending on model
Key Objective	Generative pre-training for language modeling	Task conditioning for multitask learning	In-context learning and few-shot learning	Multimodal capabilities and advanced reasoning
Training Dataset	BooksCorpus	WebText (40GB from 8M documents)	Common Crawl, WebText2, Books1, Books2, Wikipedia	Multimodal datasets (text + images), higher quality and scale
Zero-Shot Learning	Limited	Improved over GPT-1	Major advancement, effective in many tasks	Further enhanced with stronger reasoning and visual input

Copyright © 2024 Nanyang Technological University. All rights reserved.

Other resources?

https://github.com/naklecha/llama3-from-scratch

Copyright © 2024 Nanyang Technological University. All rights reserved.

Code-along

Jupyter Notebook

Part 3: Text Generation with GPT-2

Copyright © 2024 Nanyang Technological University. All rights reserved.

End of Lesson - Exit Ticket

Survey Link

https://www.menti.com/al2yt8p1zw8i

Copyright © 2024 Nanyang Technological University. All rights reserved.