Deep Learning Sessions Lisboa

Introduction to Transformers

Lisboa, 10 December, 2019

Pedro Lobato Ferreira
Applied AI, Unbabel
pedro.lobato@unbabel.com

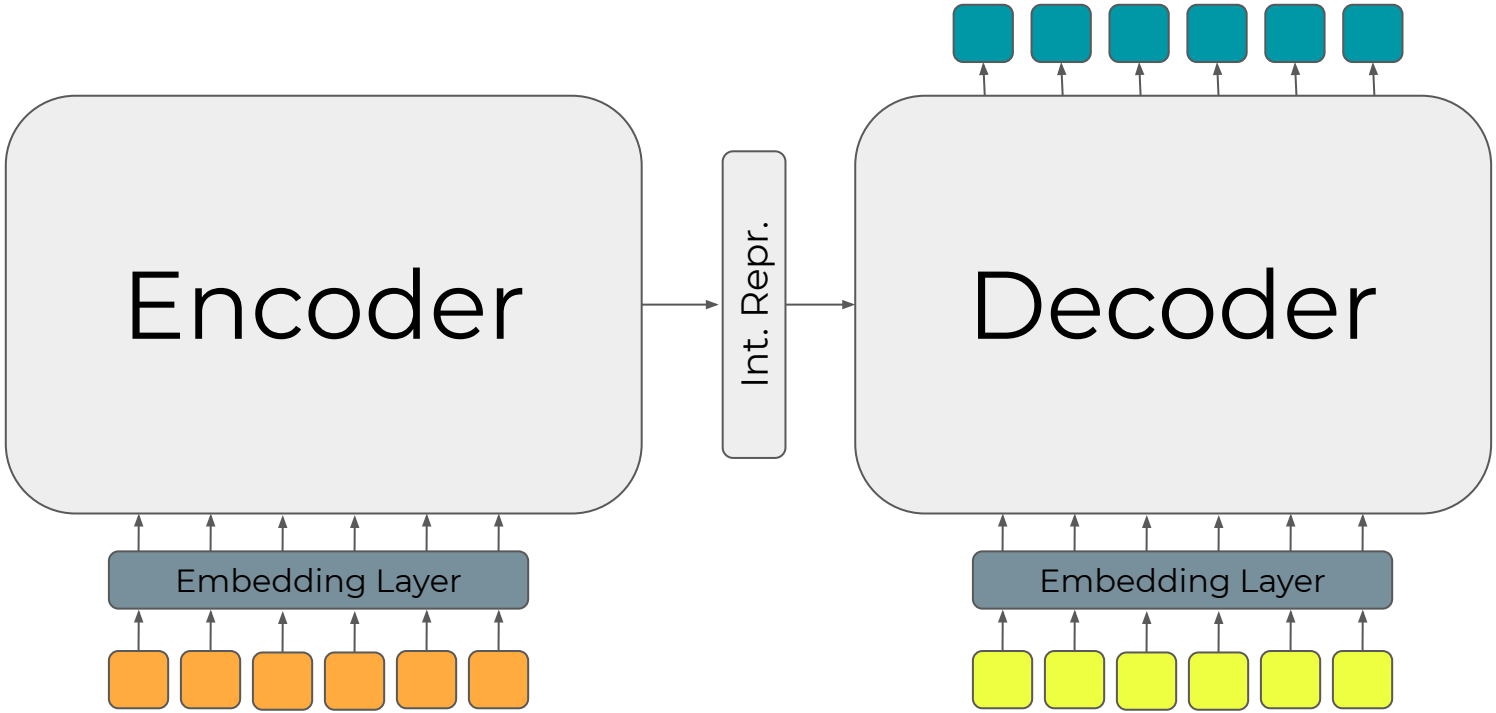
Agenda

- Sequence-to-Sequence with Attention
- Transformer
- Sequence-to-Sequence Transformer
- Some Transformer Applications (in NLP)
- Guide for a Practical Approach
- Resources

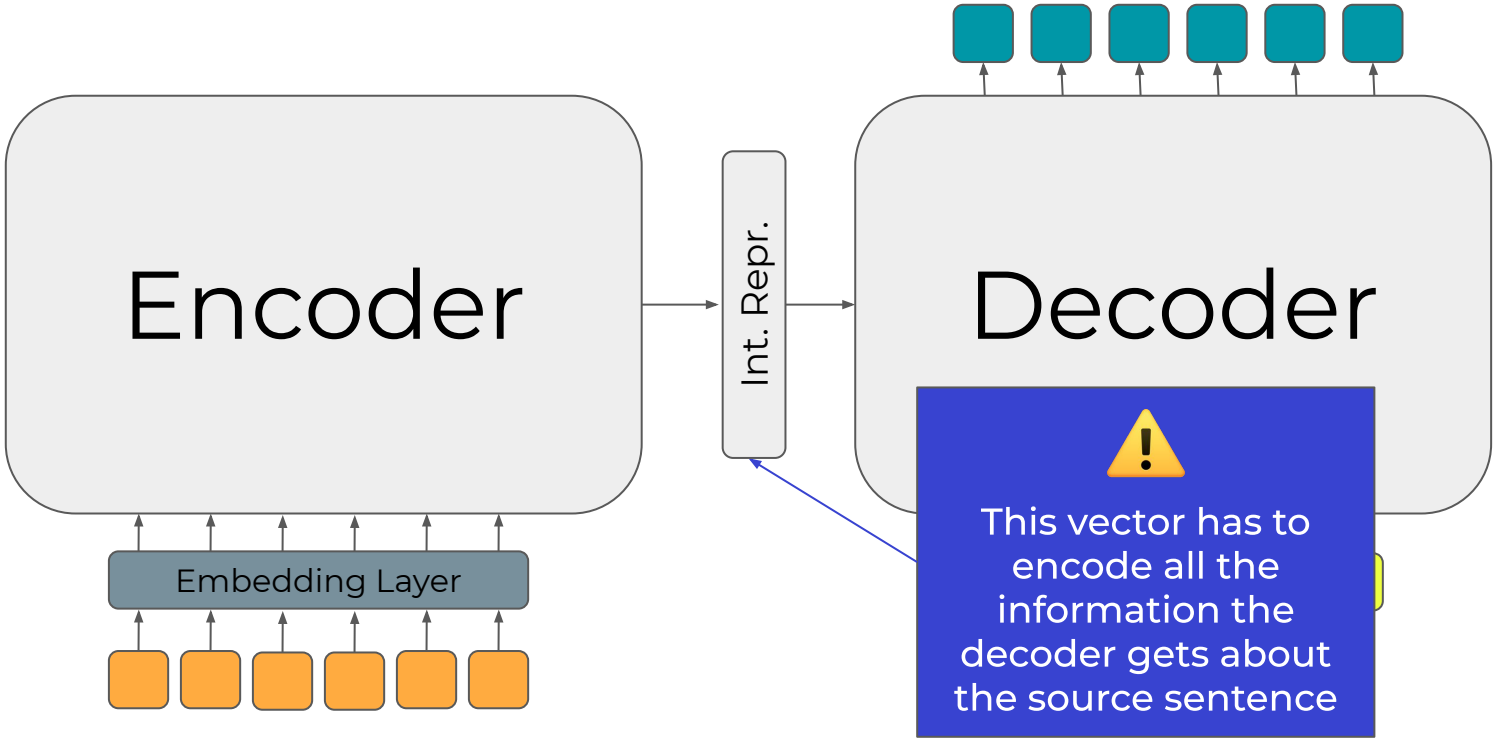
01

Sequence-to-Sequence with Attention

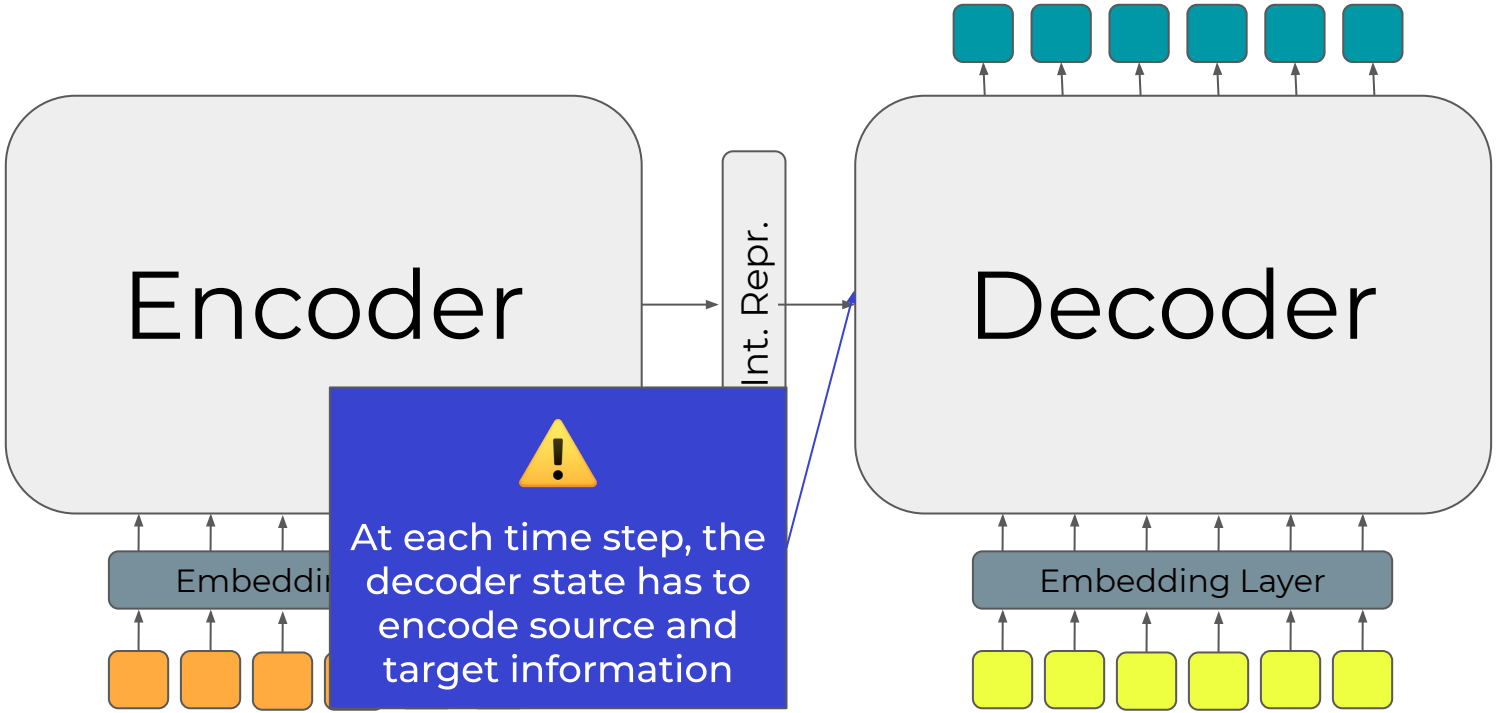
Sequence-to-Sequence



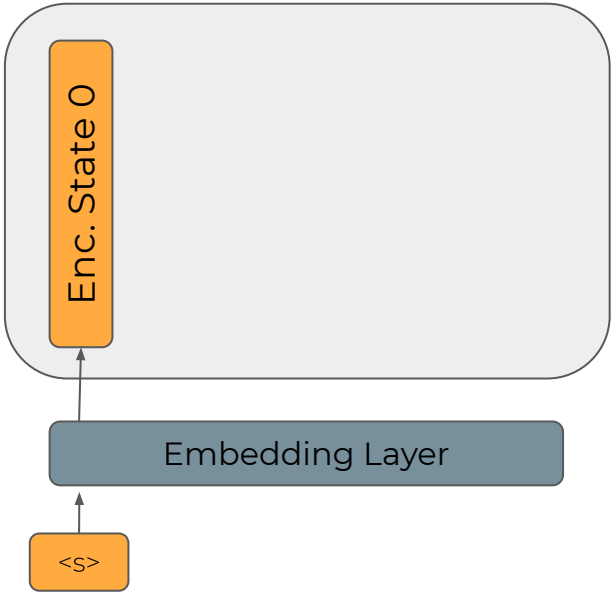
Sequence-to-Sequence



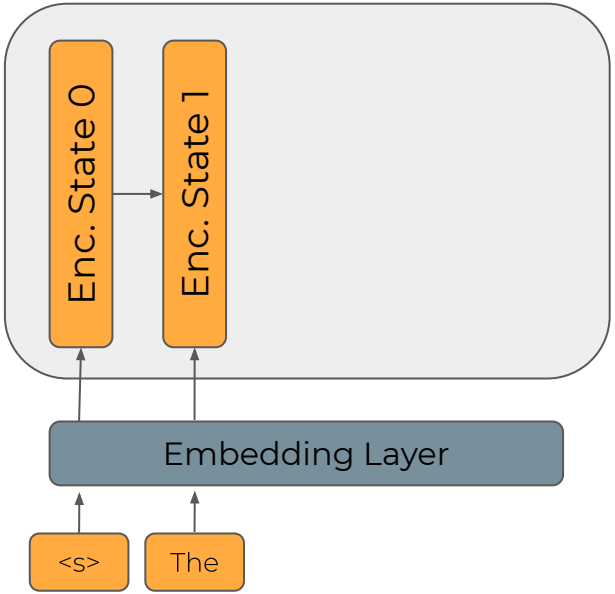
Sequence-to-Sequence



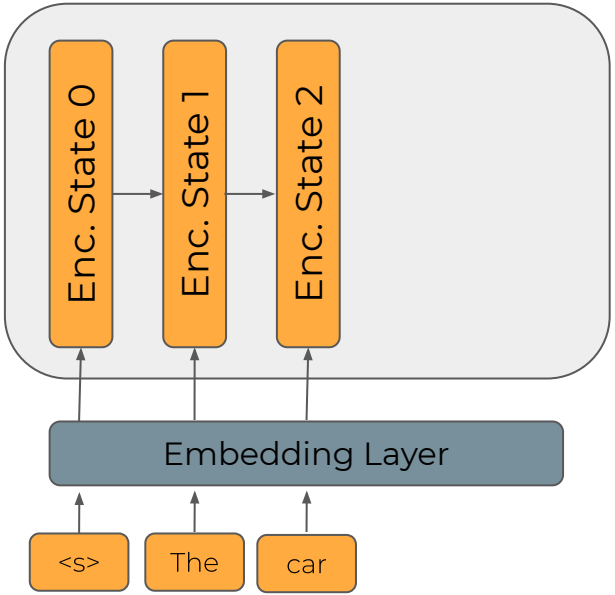
Sequence-to-Sequence with Attention



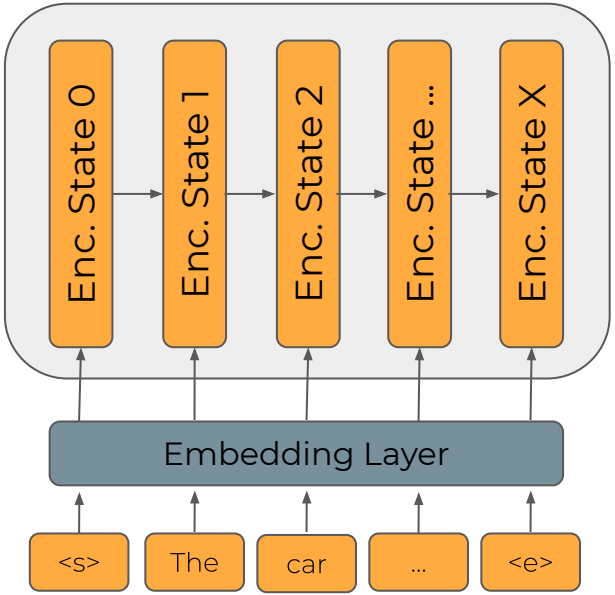
Sequence-to-Sequence with Attention



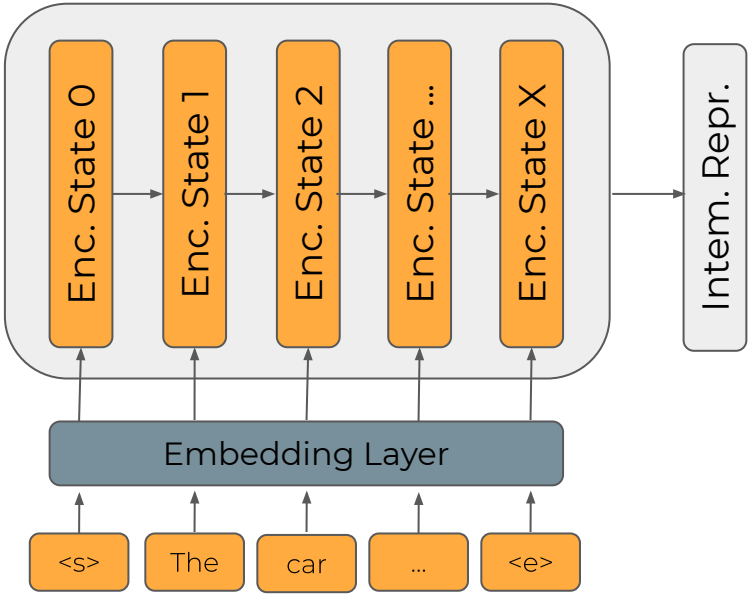
Sequence-to-Sequence with Attention



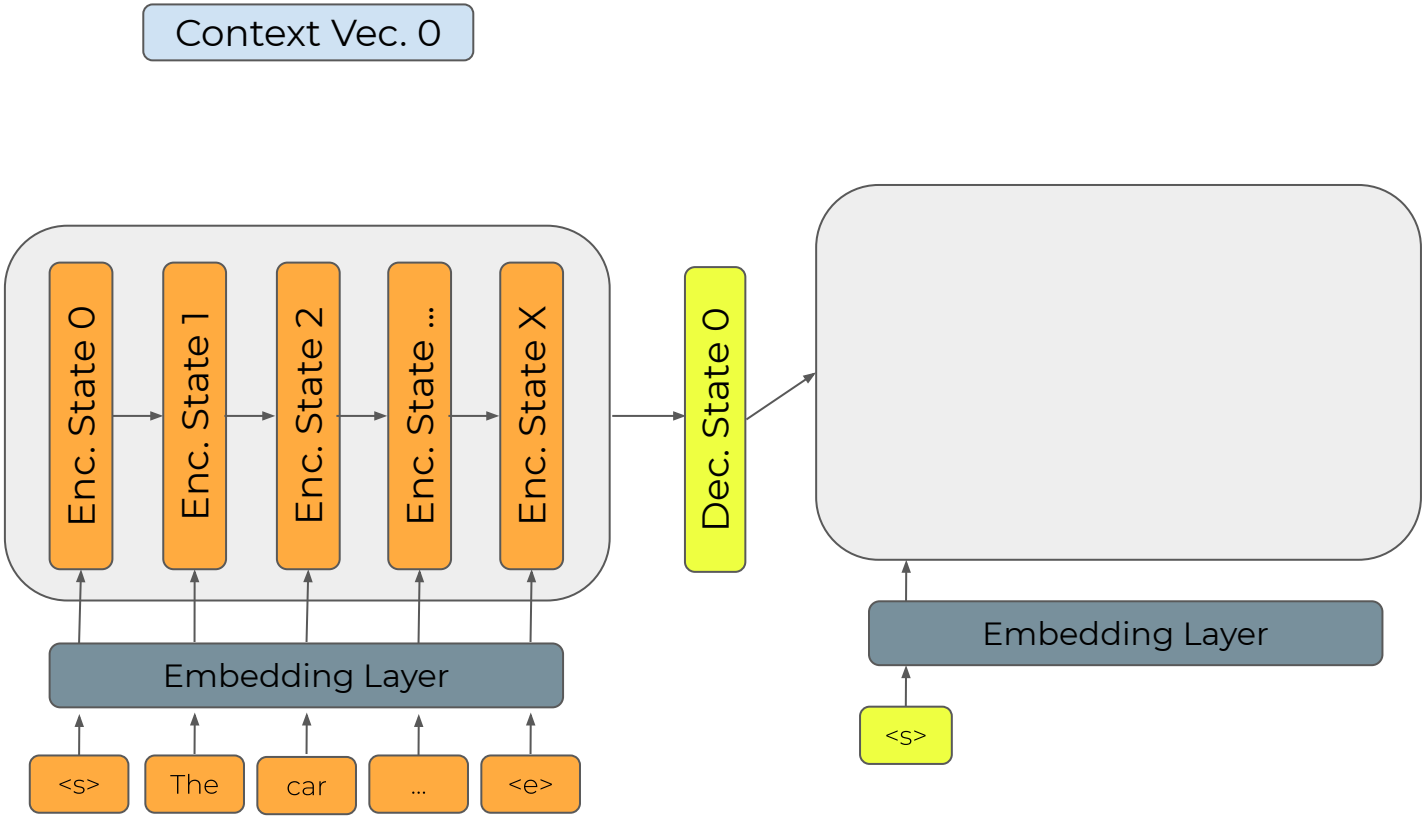
Sequence-to-Sequence with Attention



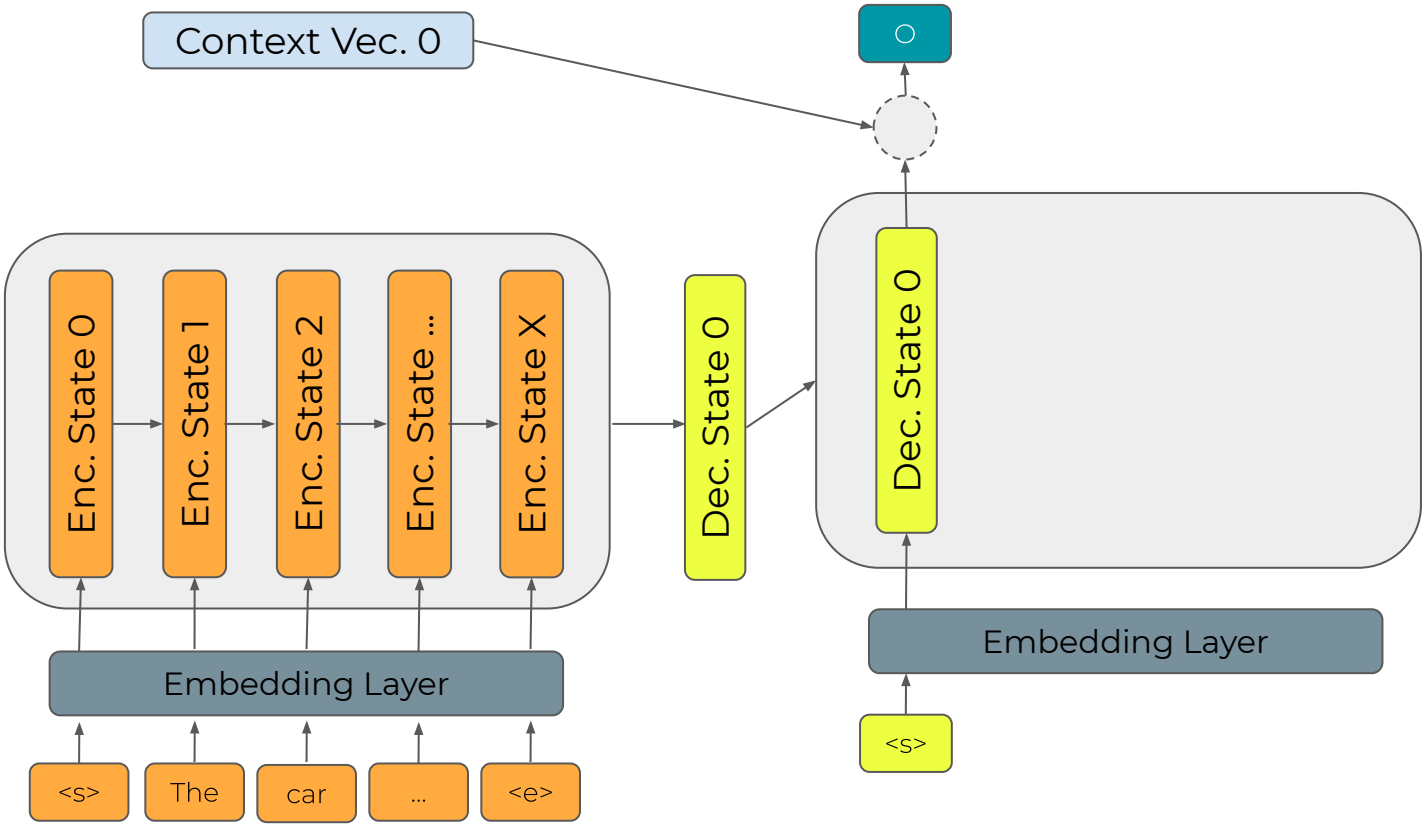
Sequence-to-Sequence with Attention



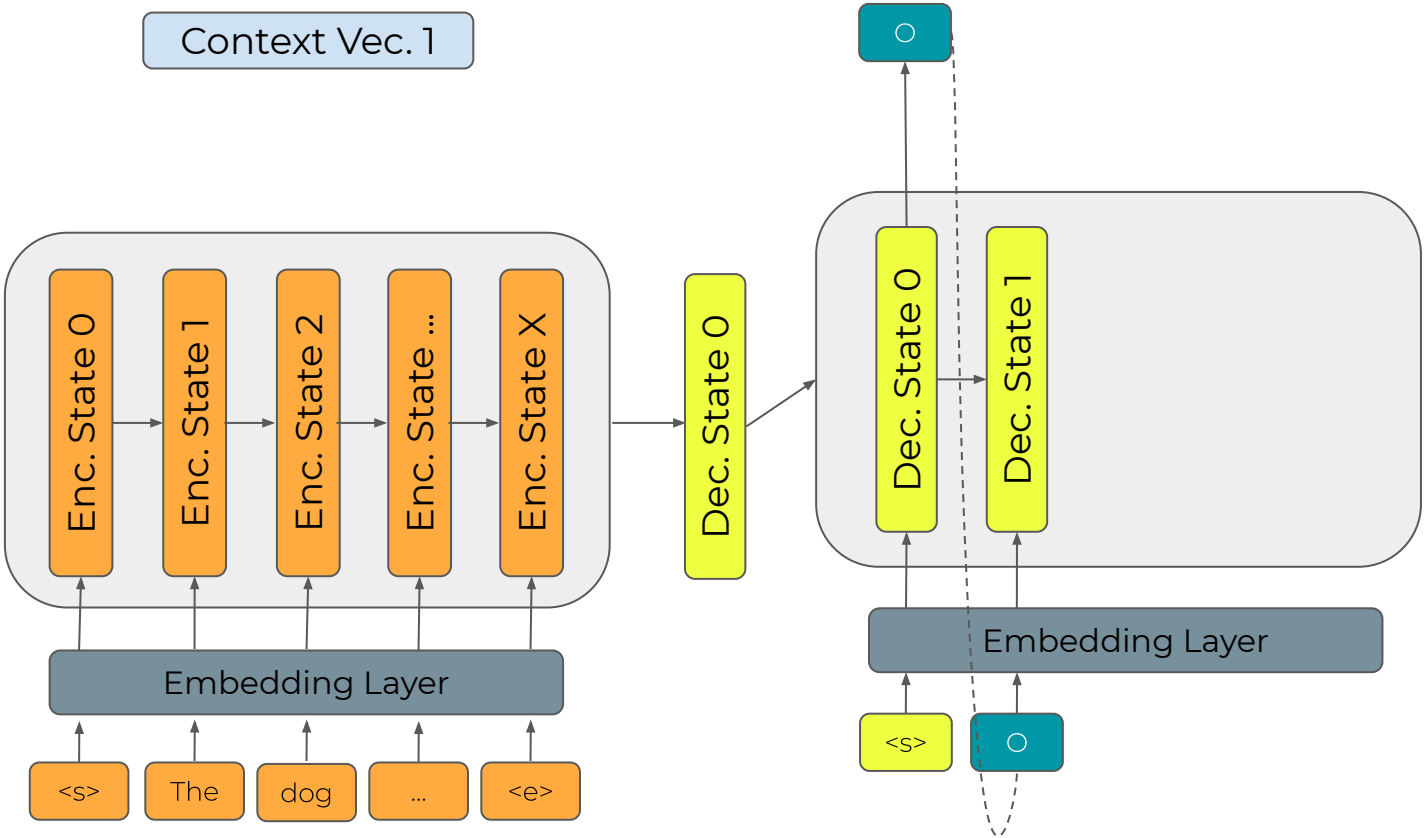
Sequence-to-Sequence with Attention



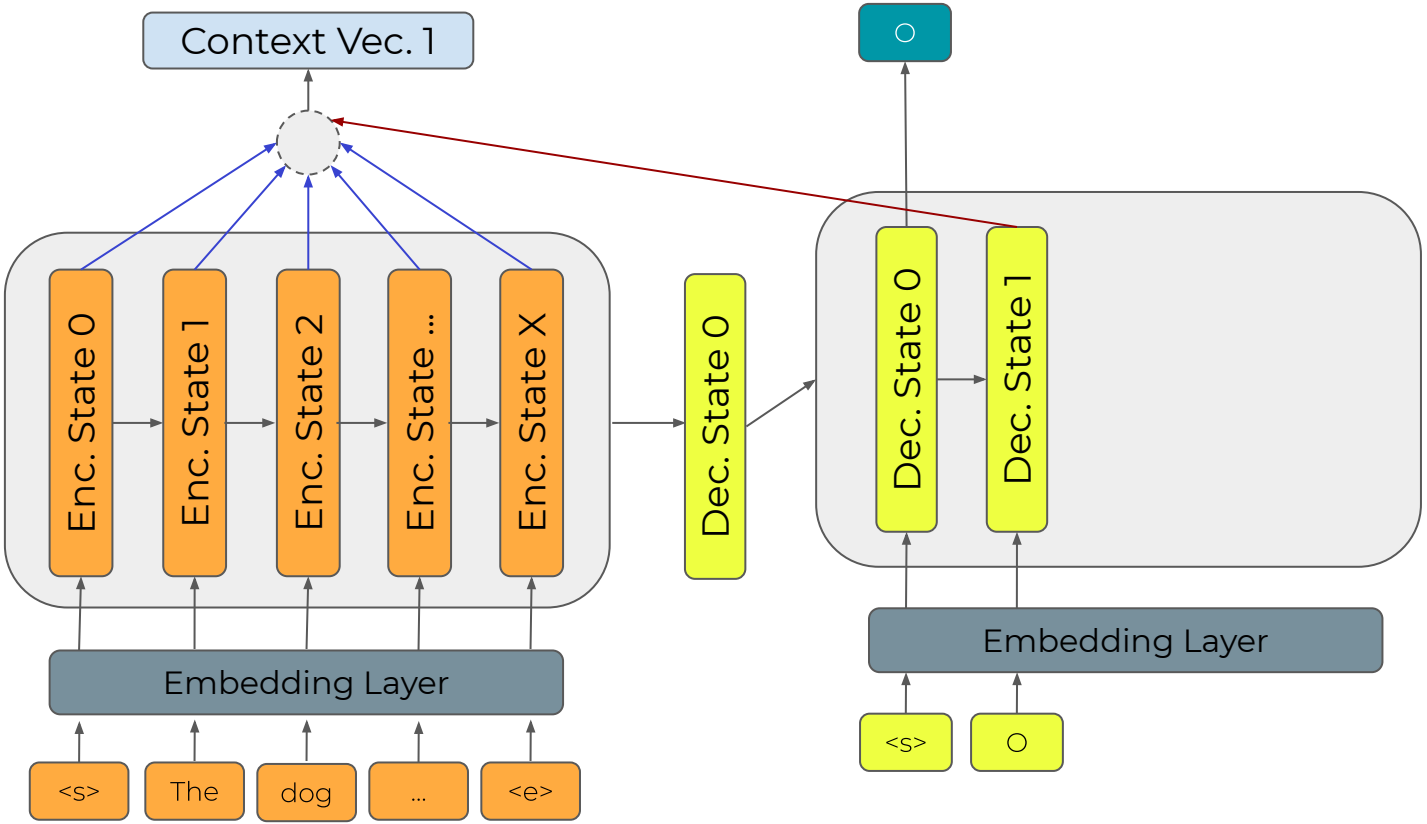
Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



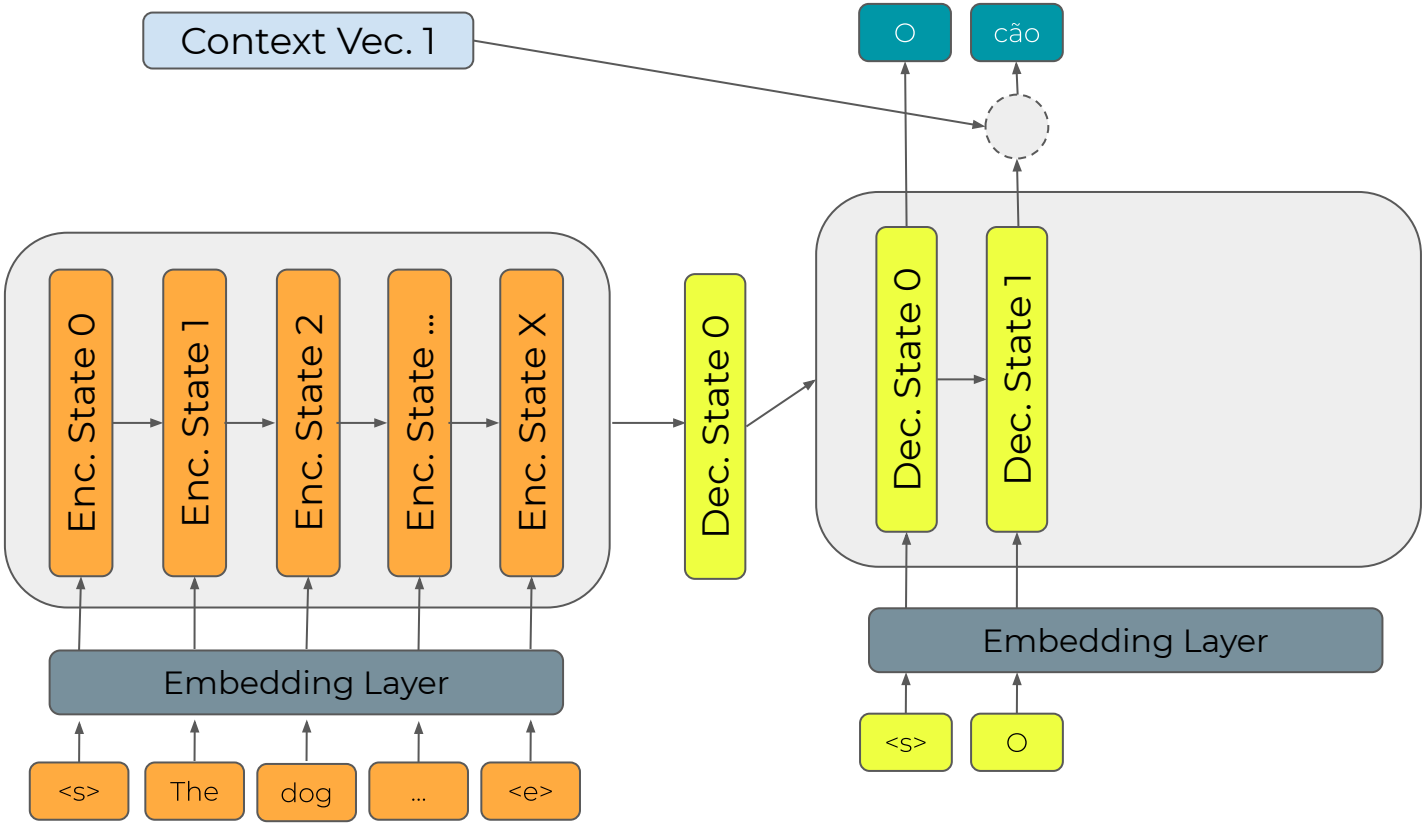
Attention

$$e_{ts} = a(h_t, hs)$$

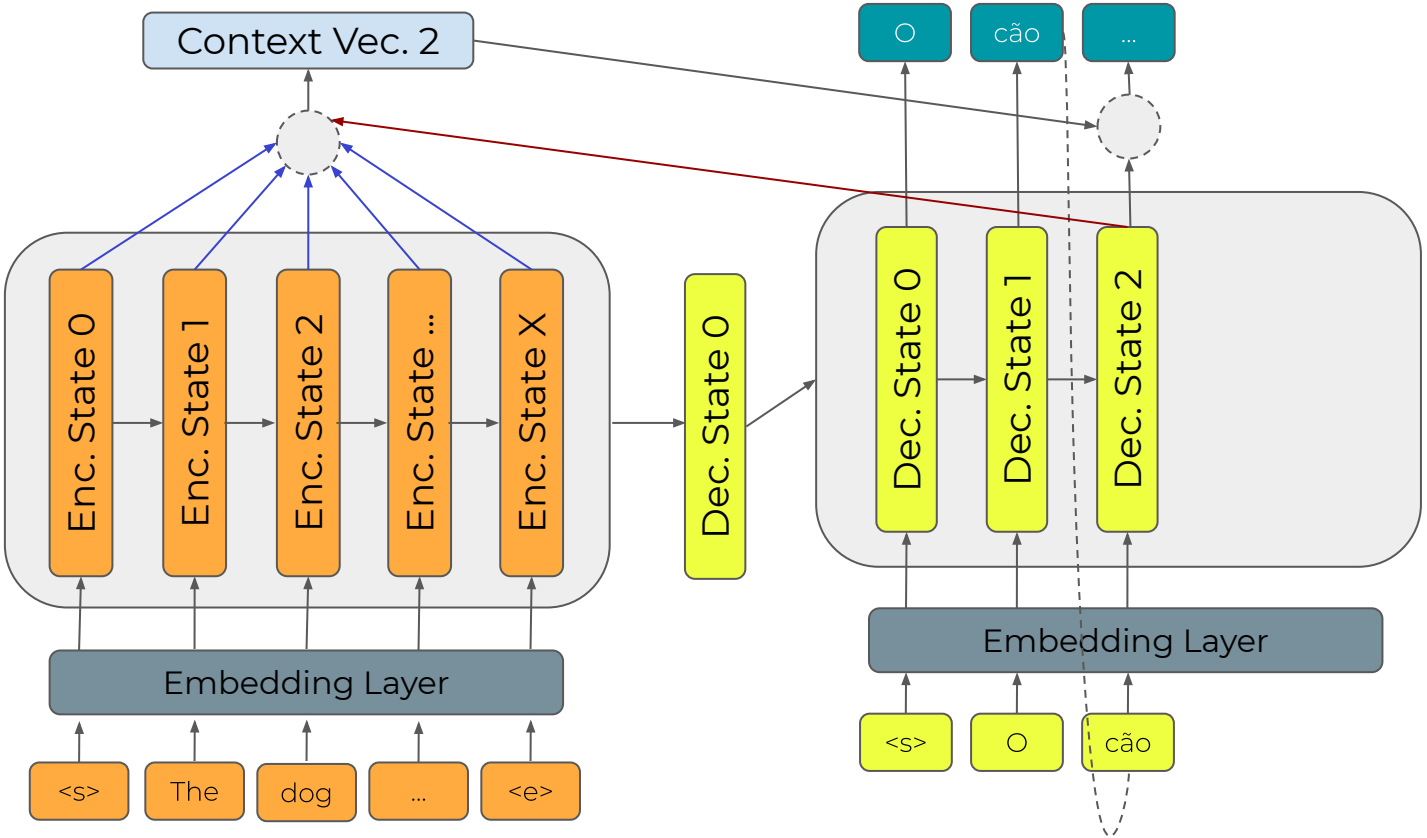
$$\alpha_{ts} = \frac{e_{ts}}{\sum_{k=1}^{X} e_{tk}}$$

$$c_t = \sum_{s=1}^{X} \alpha_{ts} h_s$$

Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



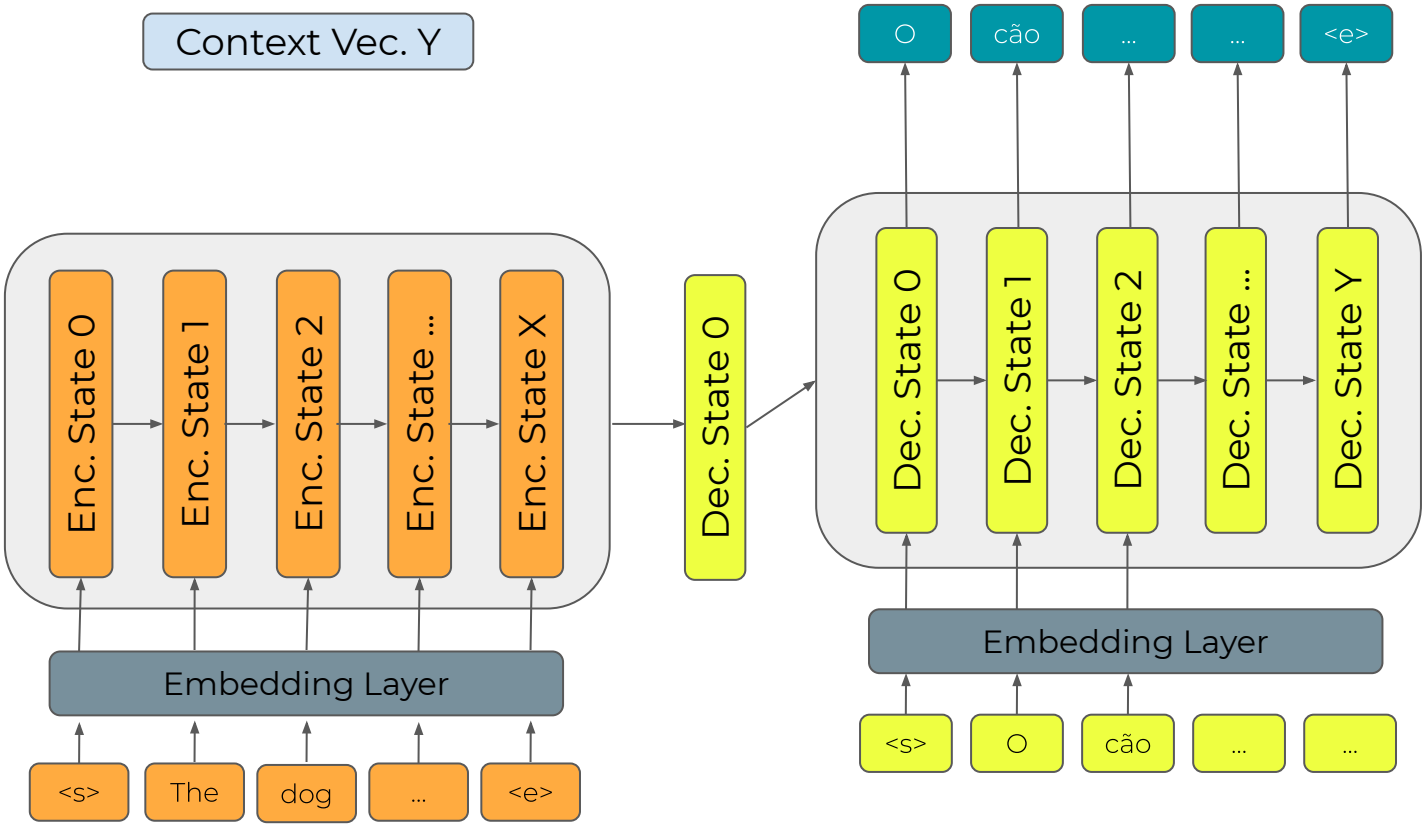
Attention

$$e_{ts} = a(h_t, hs)$$

$$\alpha_{ts} = \frac{e_{ts}}{\sum_{k=1}^{X} e_{tk}}$$

$$c_t = \sum_{s=1}^{X} \alpha_{ts} h_s$$

Sequence-to-Sequence with Attention



02

Transformer

Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

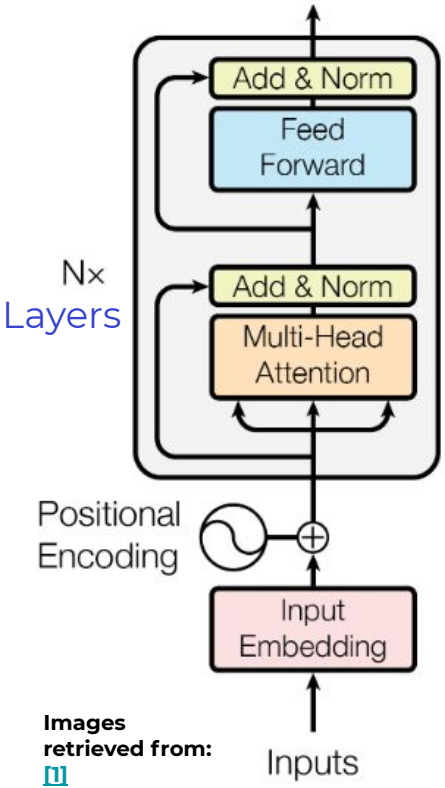
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer,

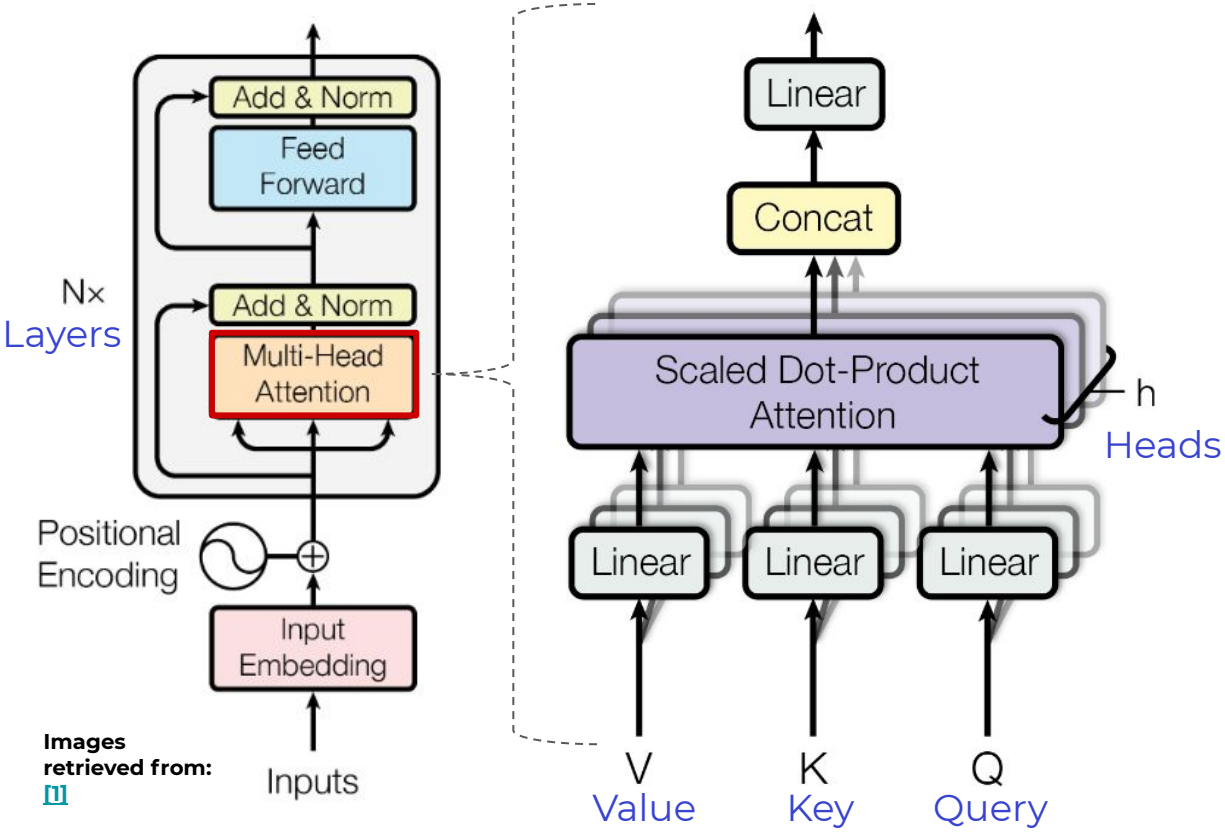
Retrieved from: [1]

Transformer Encoder



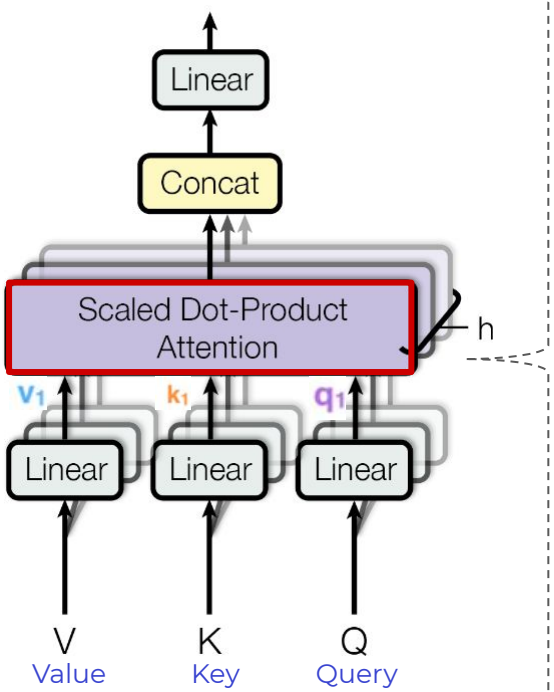
Images retrieved from:
[1]

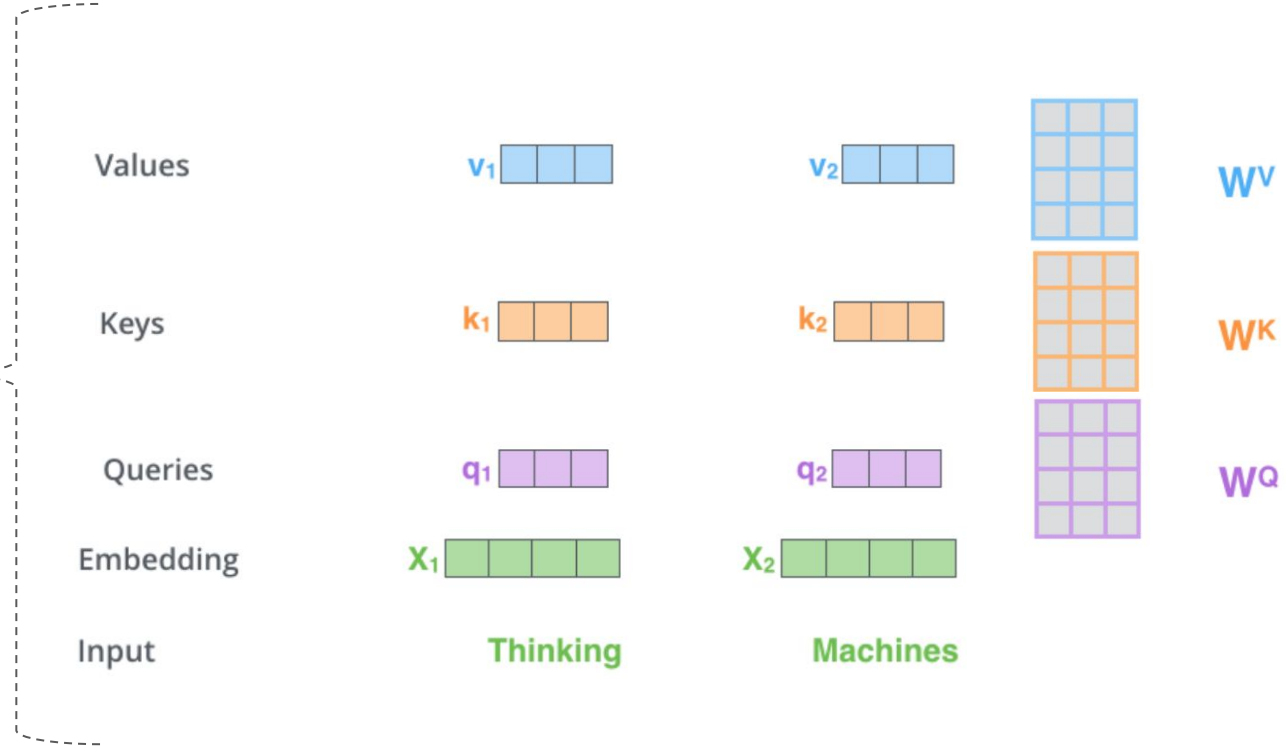
Transformer Encoder



Images retrieved from:
[1]

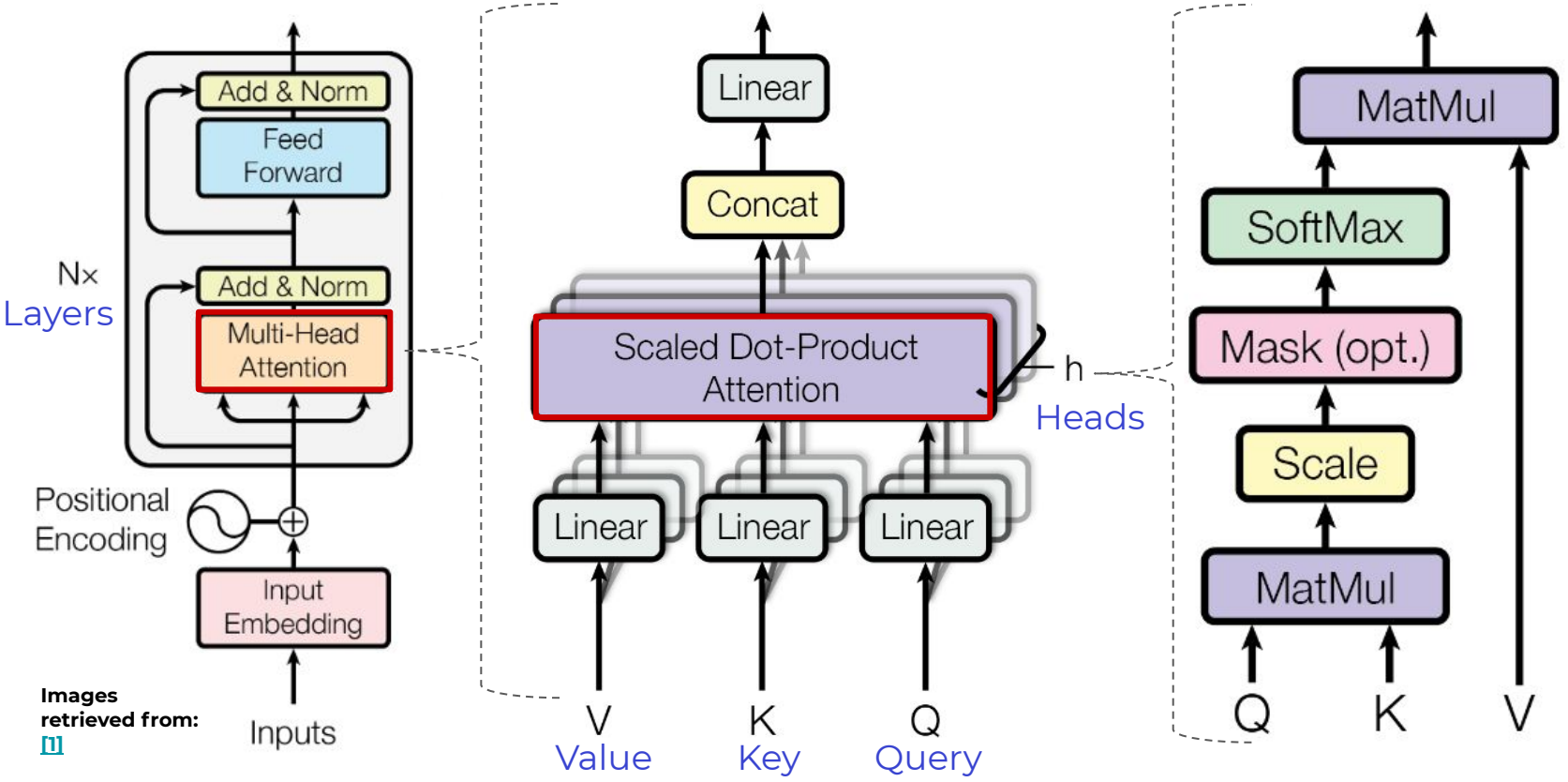
Multi-Head (Self) Attention





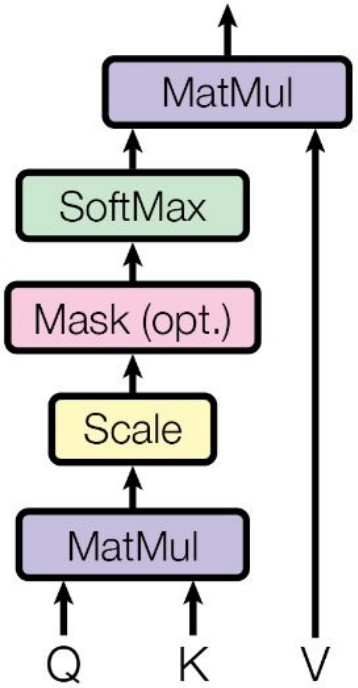
Images retrieved from:
[1] and [2]

Transformer Encoder



Images retrieved from:
[1]

Where have we seen this before?



Images retrieved from: [1]

Attention Now

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

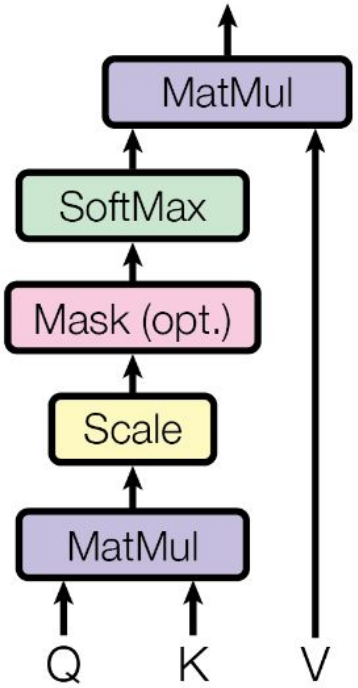
Attention a few slides ago

$$e_{ts} = a(h_t, hs)$$

$$\alpha_{ts} = \frac{e_{ts}}{\sum_{k=1}^{X} e_{tk}}$$

$$c_t = \sum_{s=1}^{X} \alpha_{ts} h_s$$

Where have we seen this before?



Images retrieved from: [1]

Attention Now

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

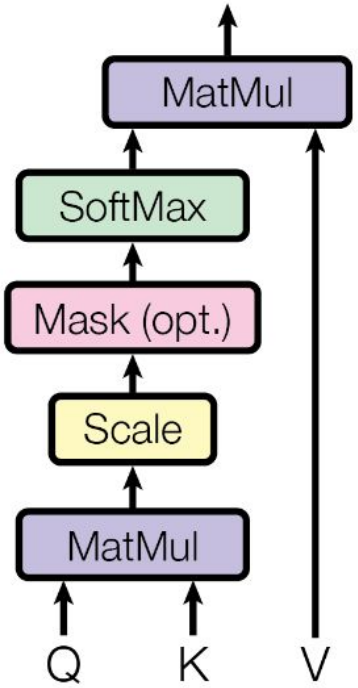
Attention a few slides ago

$$e_{ts} = a(h_t, hs)$$

$$\alpha_{ts} = \frac{e_{ts}}{\sum_{k=1}^{X} e_{tk}}$$

$$c_t = \sum_{s=1}^{X} \alpha_{ts} h_s$$

Where have we seen this before?



Images retrieved from: [1]

Attention Now

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

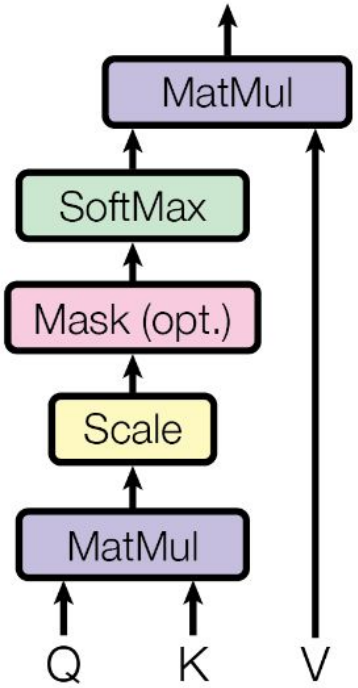
Attention a few slides ago

$$e_{ts} = a(h_t, hs)$$

$$\alpha_{ts} = \frac{e_{ts}}{\sum_{k=1}^{X} e_{tk}}$$

$$c_t = \sum_{s=1}^{X} \alpha_{ts} h_s$$

Where have we seen this before?



Images retrieved from: [1]

Attention Now

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

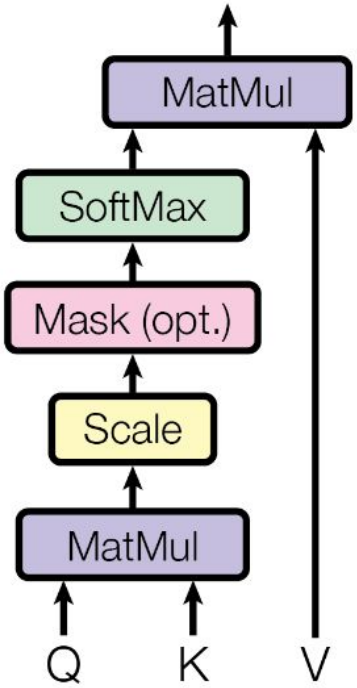
Attention a few slides ago

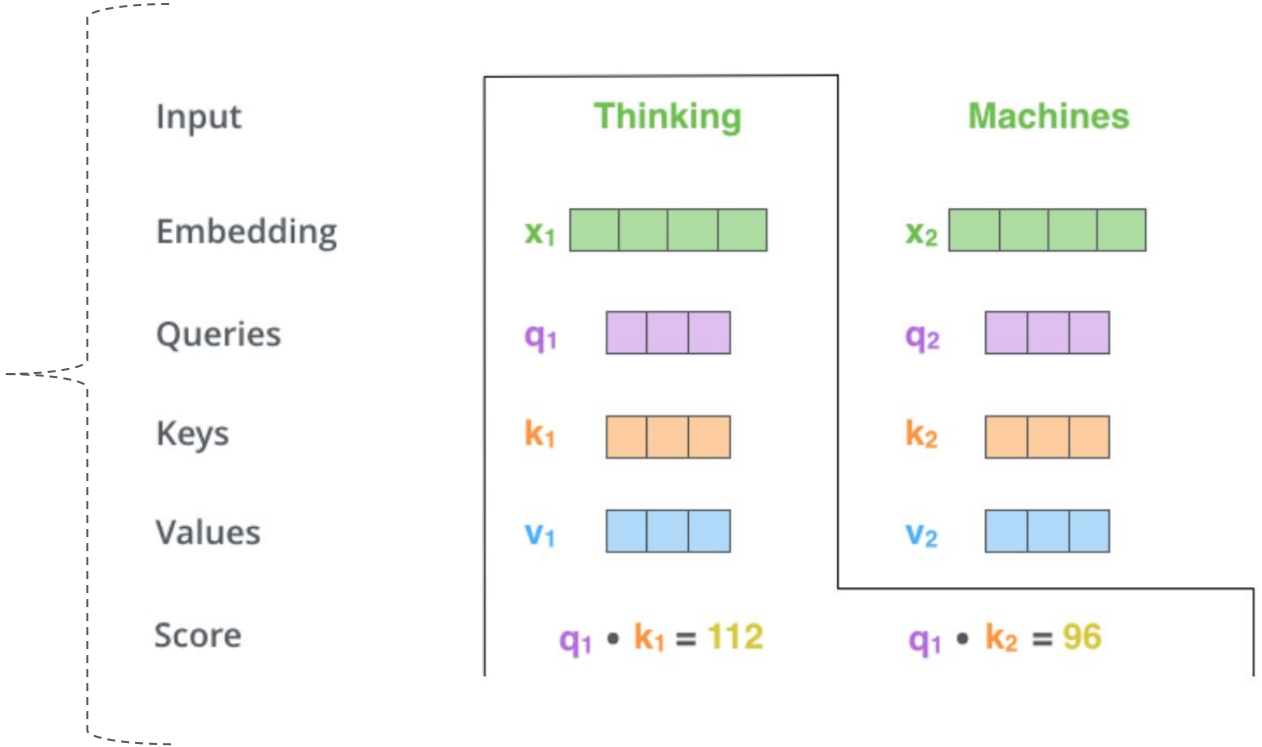
$$e_{ts} = a(h_t, hs)$$

$$\alpha_{ts} = \frac{e_{ts}}{\sum_{k=1}^{X} e_{tk}}$$

$$c_t = \sum_{s=1}^{X} \alpha_{ts} h_s$$

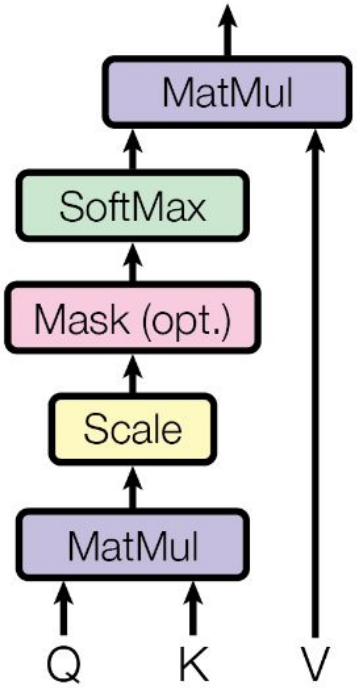
Multi-Head (Self) Attention

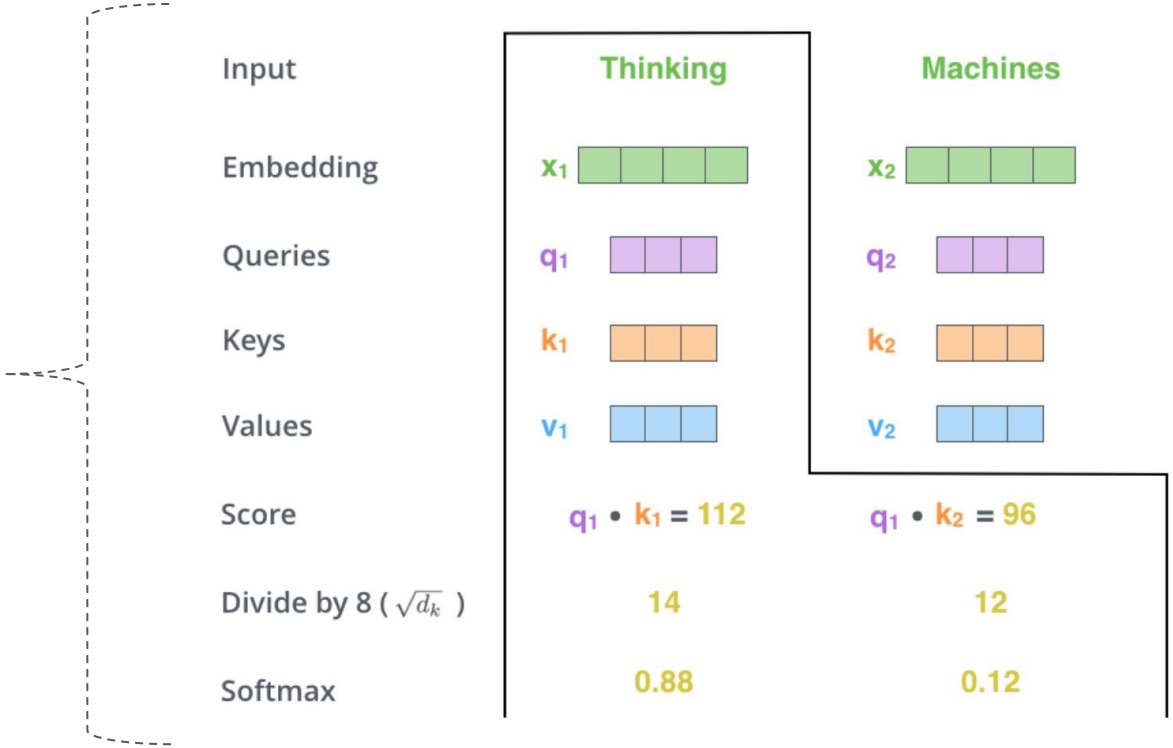




Images retrieved from:
[1] and [2]

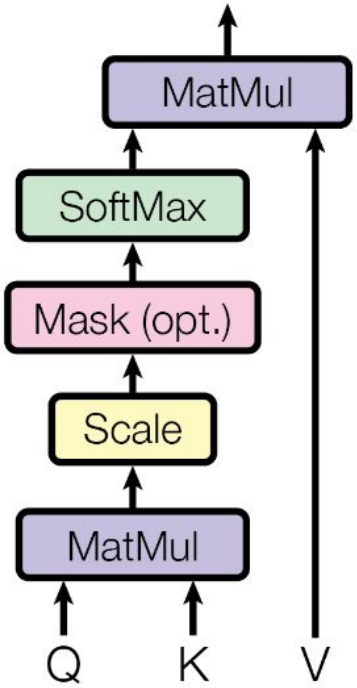
Multi-Head (Self) Attention

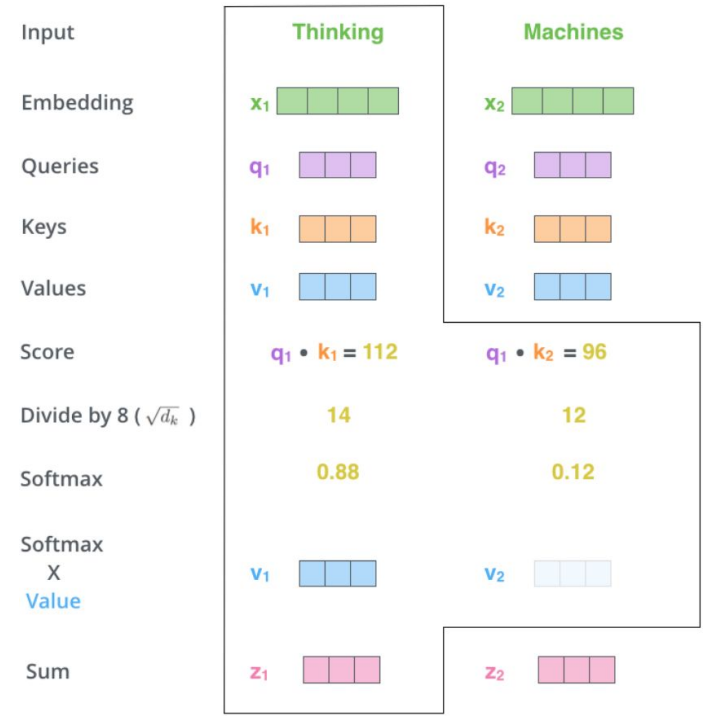




Images retrieved from:
[1] and [2]

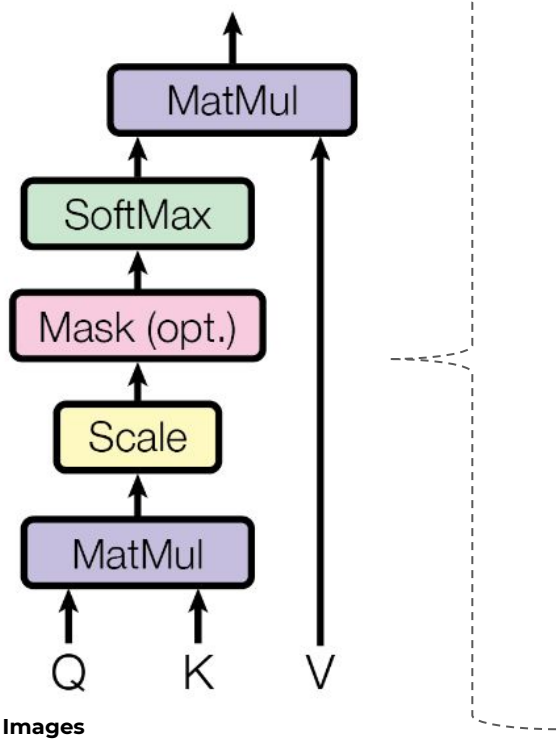
Multi-Head (Self) Attention

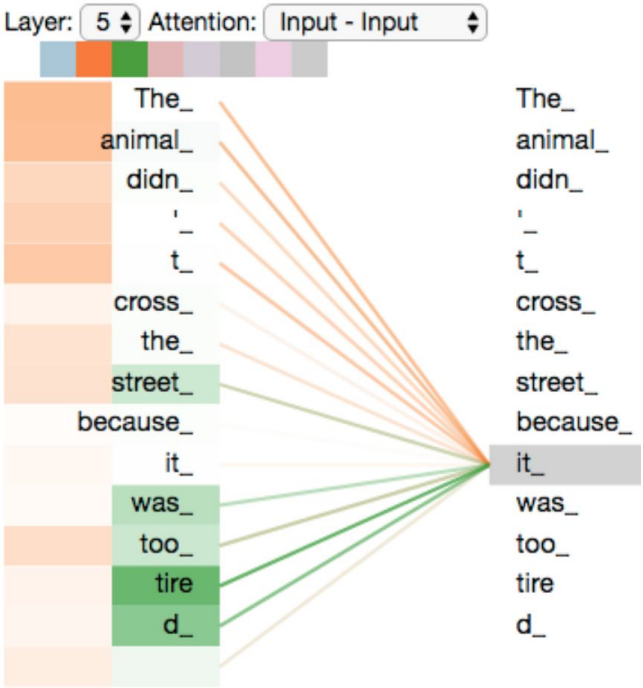




Images retrieved from:
[1] and [2]

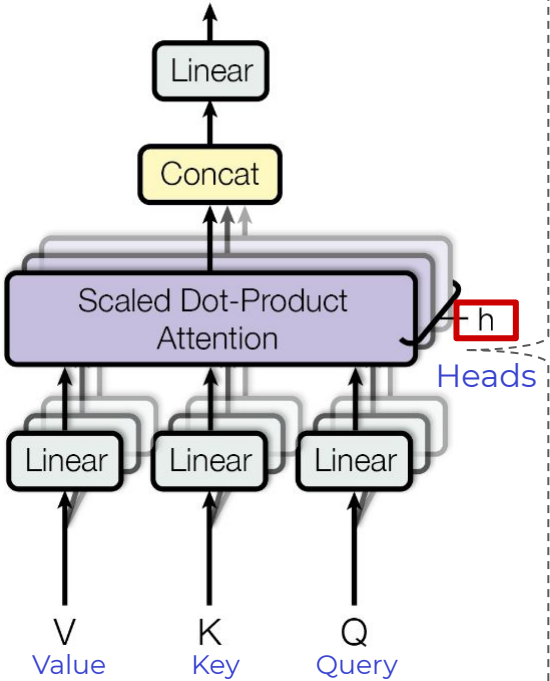
Multi-Head (Self) Attention





Images retrieved from:
[1] and [2]

Multi-Head (Self) Attention

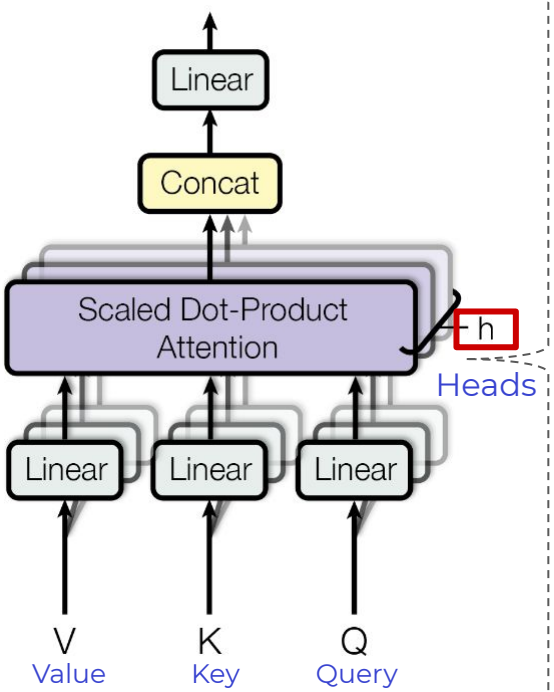


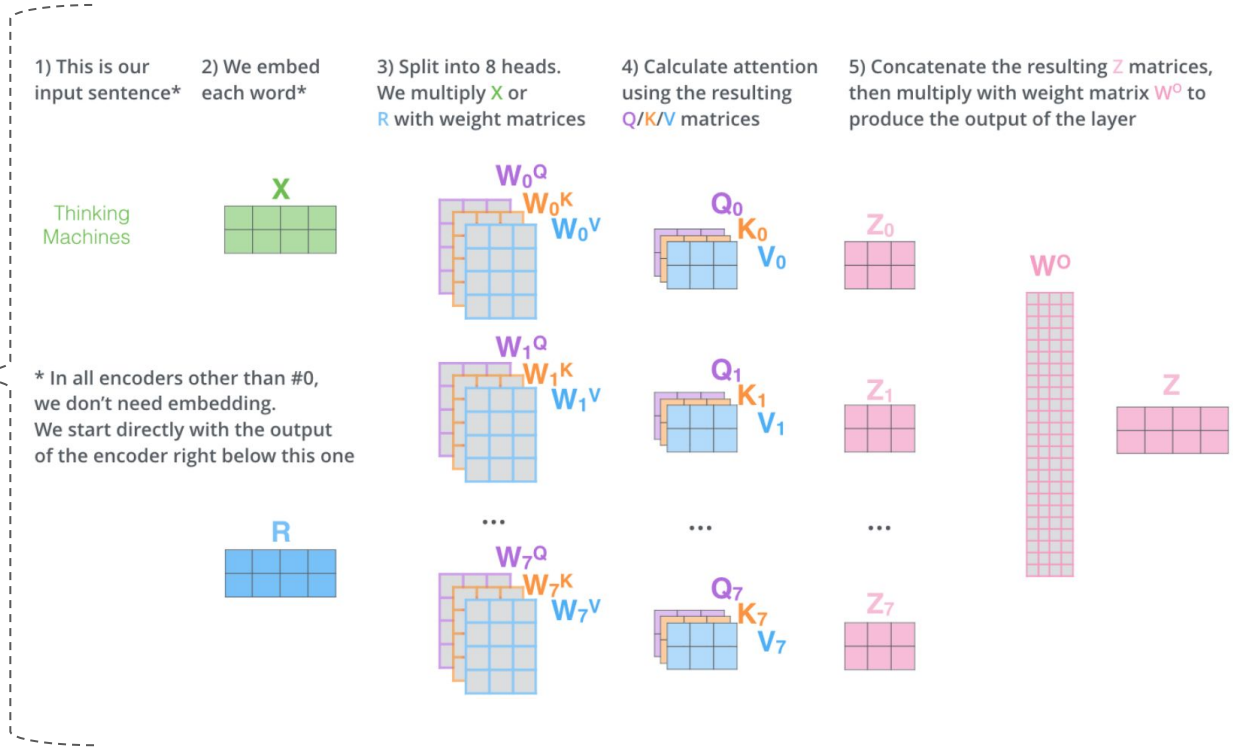
Images retrieved from:
[1]

Multi-Head Attention:

- Makes it possible for the output vector of the same word in different heads to focus on different parts of the sentence (during self-attention).
- By having different sets of weights for the matrices that create the Query, Key and Value vectors, and by being initialized differently, each head will have its own representation subspace.

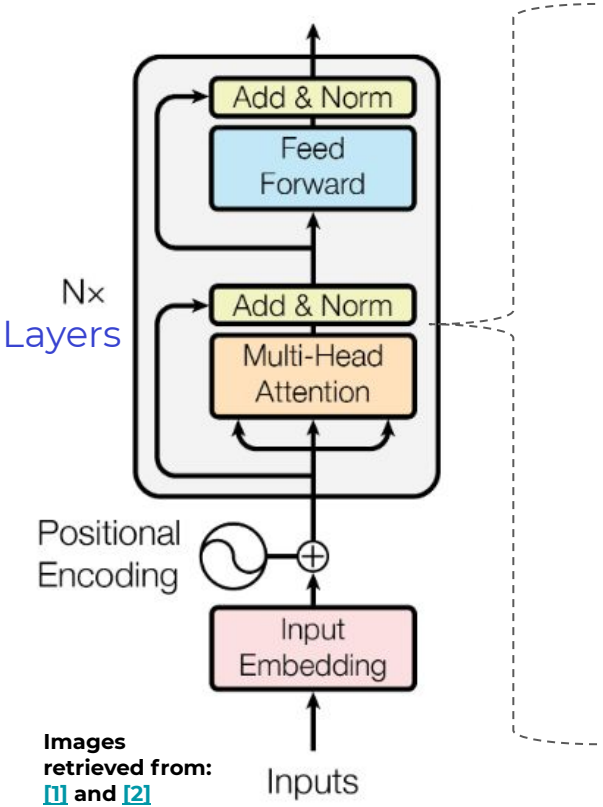
Multi-Head (Self) Attention

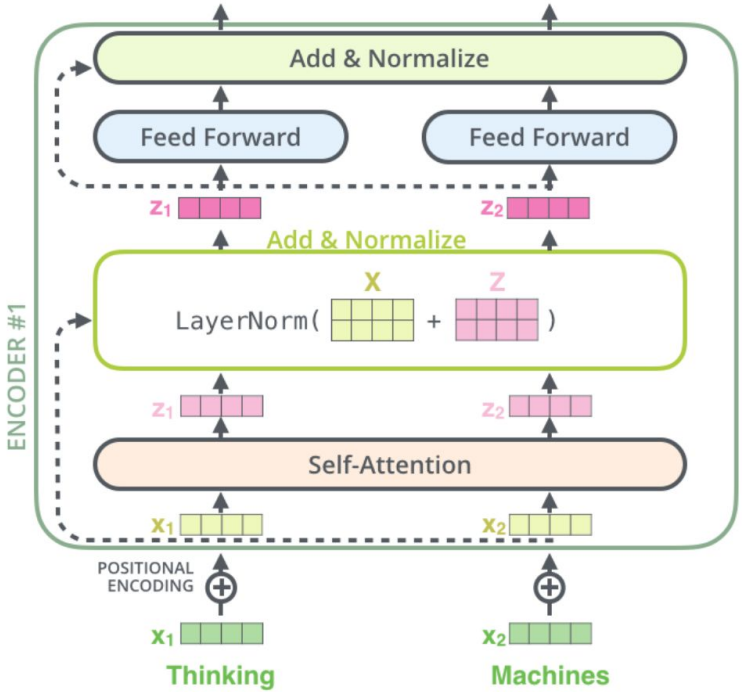




Images retrieved from:
[1] and [2]

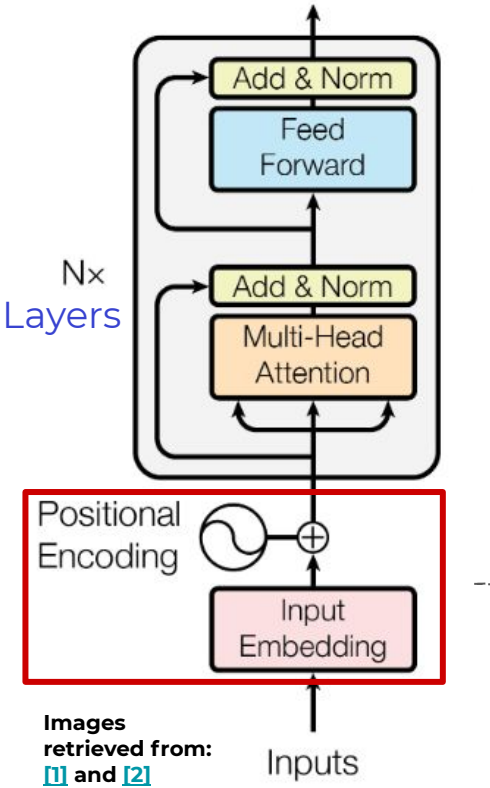
Transformer Encoder





Images retrieved from: [1] and [2]

One detail missing...

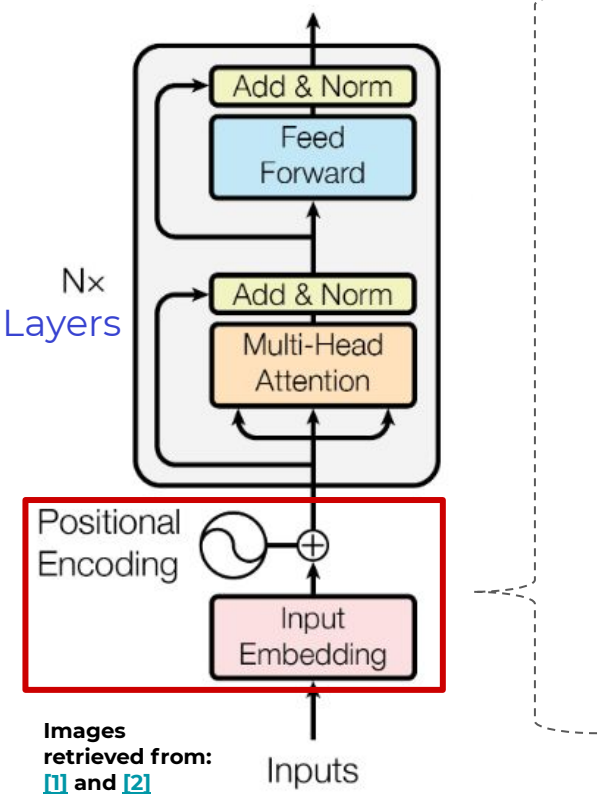


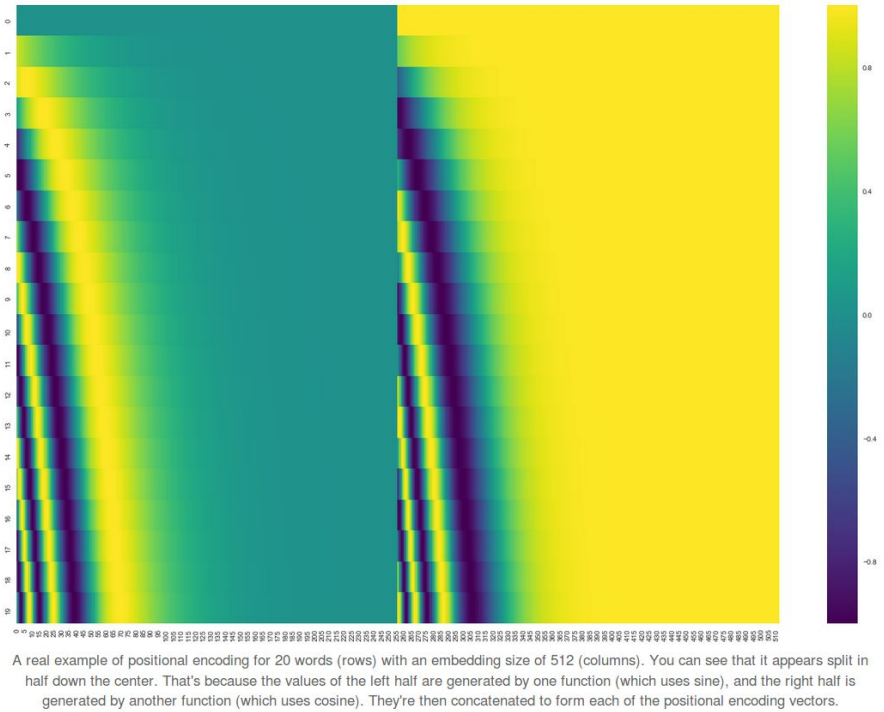
Images retrieved from: [1] and [2]

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

One detail missing...

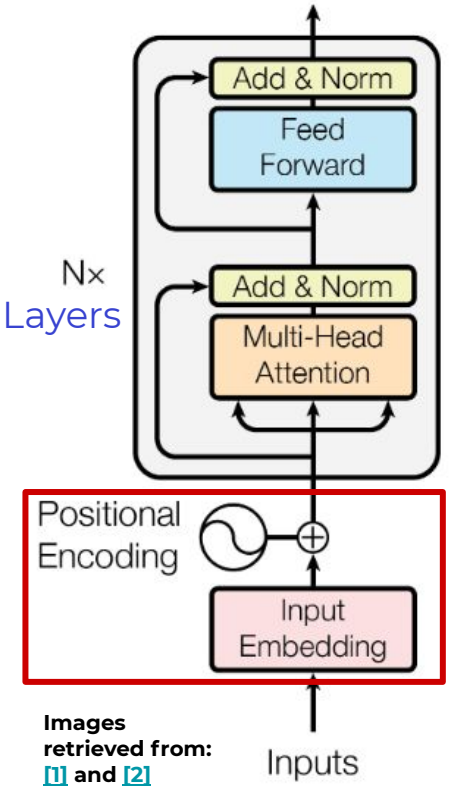


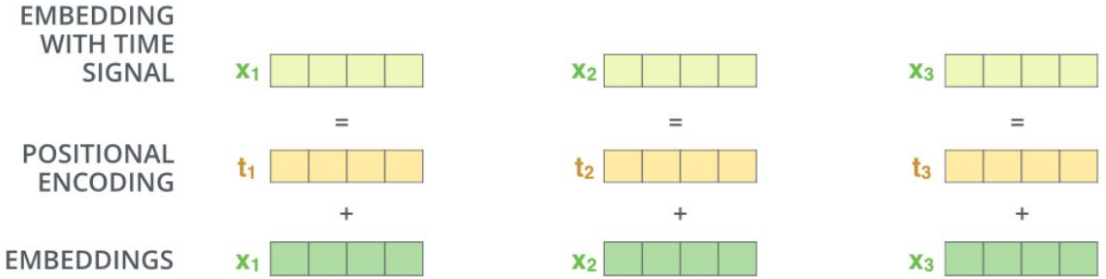


A real example of positional encoding for 20 words (rows) with an embedding size of 512 (columns). You can see that it appears split in half down the center. That's because the values of the left half are generated by one function (which uses sine), and the right half is generated by another function (which uses cosine). They're then concatenated to form each of the positional encoding vectors.

Images retrieved from:
[1] and [2]

One detail missing...



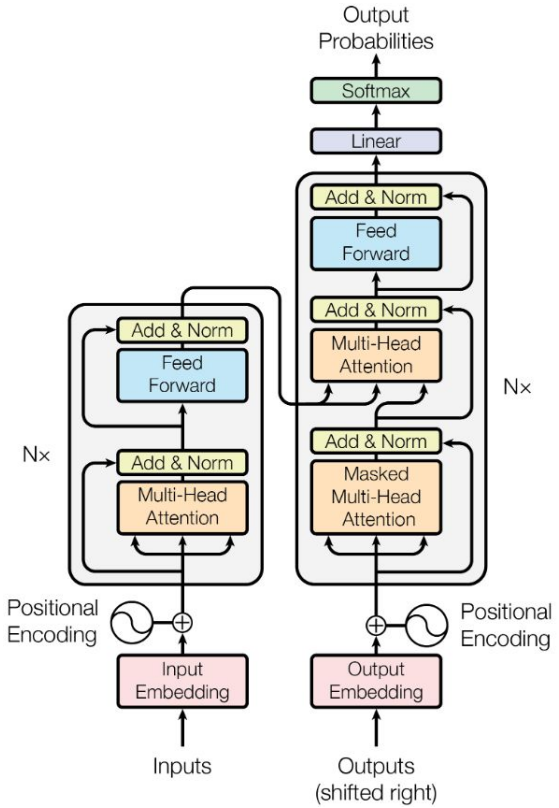


Images retrieved from: [1] and [2]

03

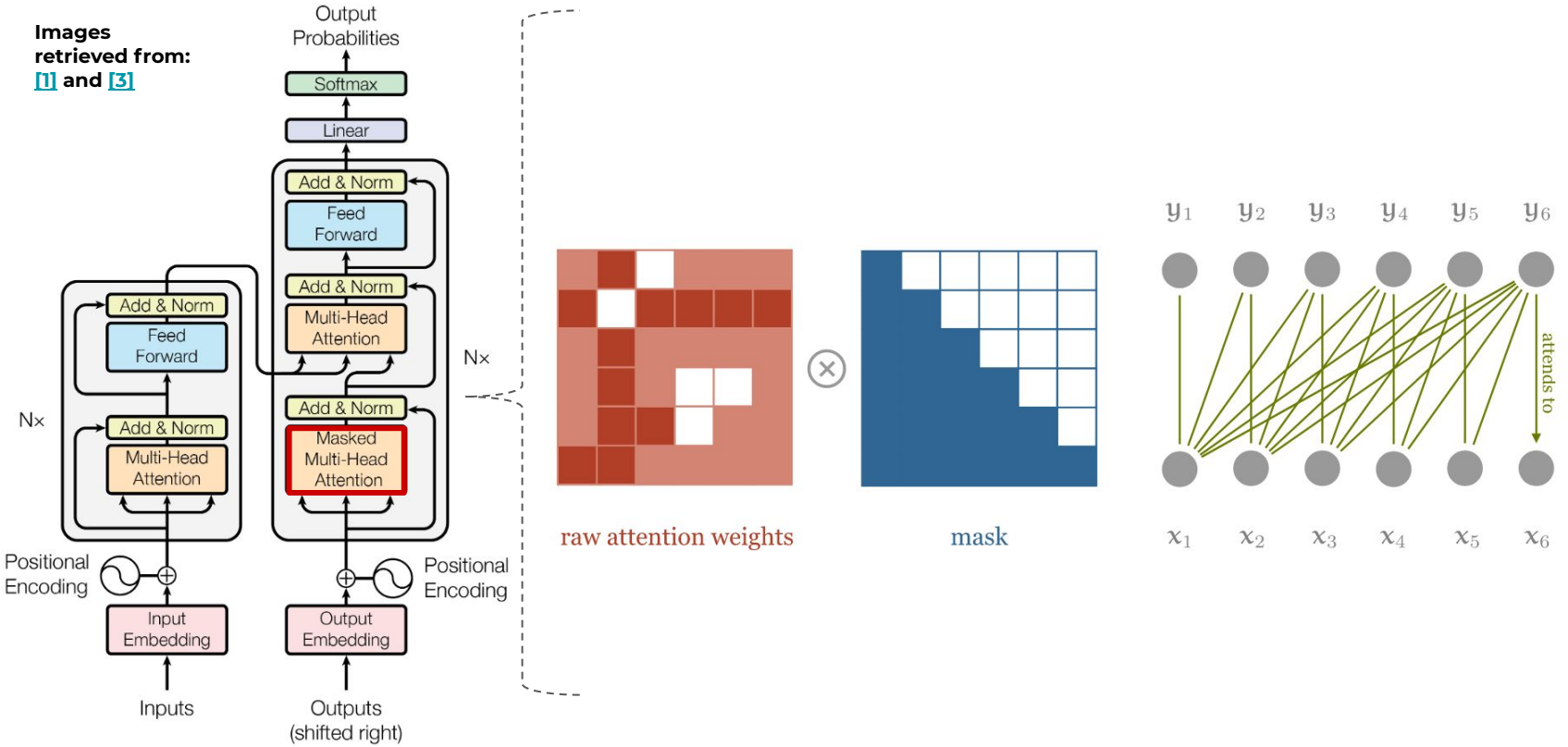
Sequence-to-Sequence Transformer

Transformer Encoder + Decoder



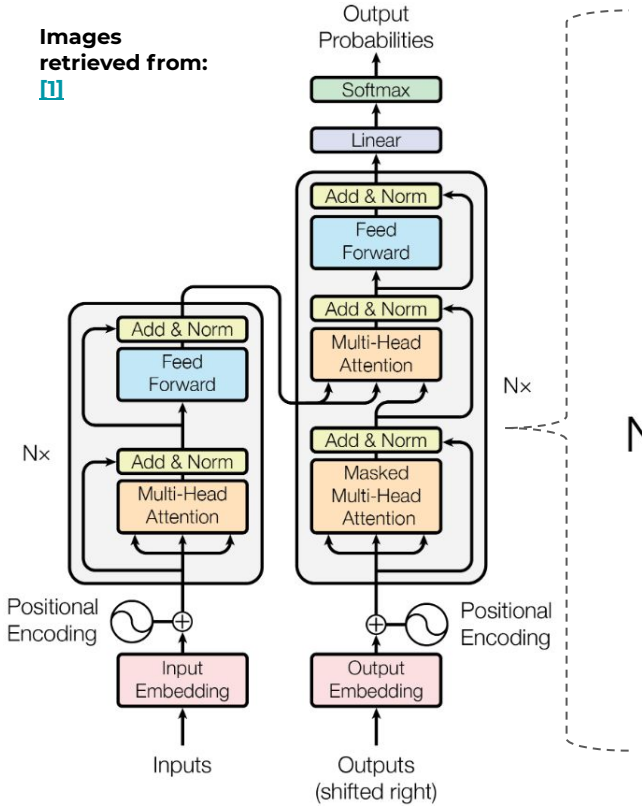
Images retrieved from: [1]

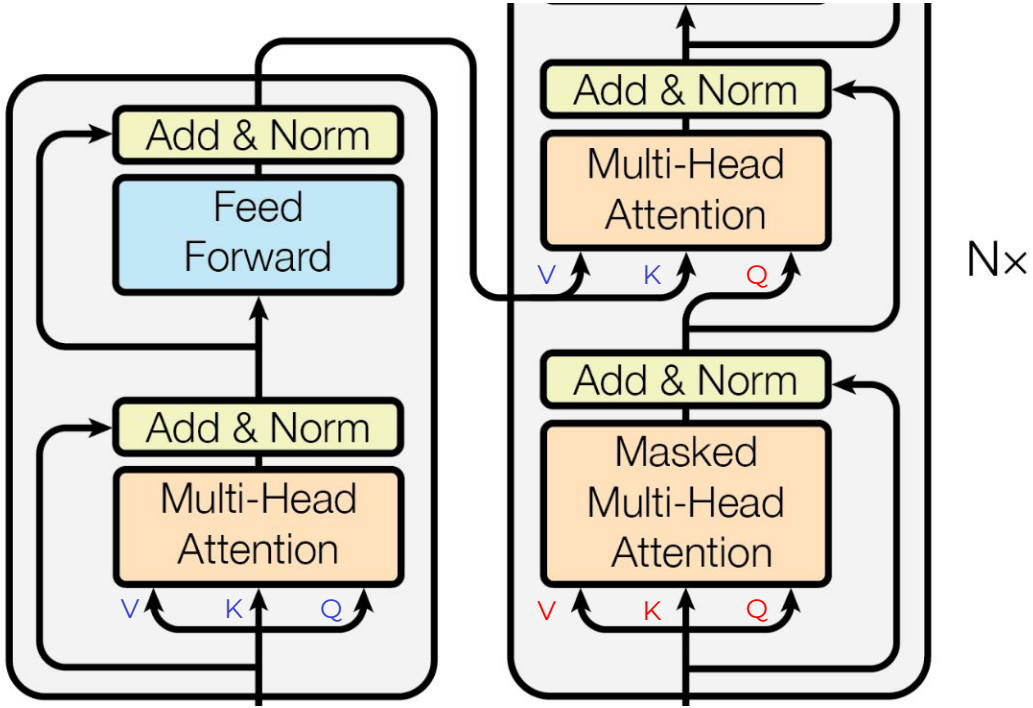
Masked Multi-Head (Self) Attention



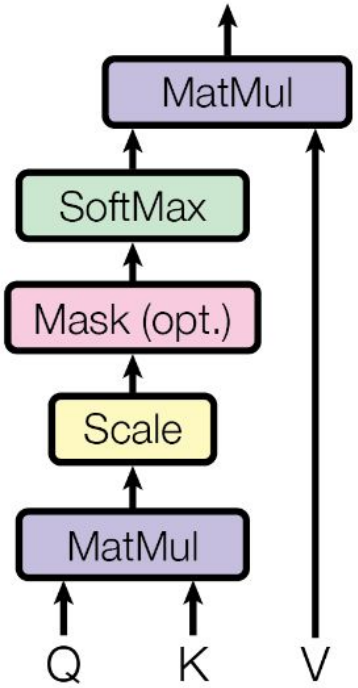
Decoder Multi-Head Attention

Images retrieved from:
[1]





Where have we seen this before?



Images retrieved from: [1]

Attention Now

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Attention a few slides ago

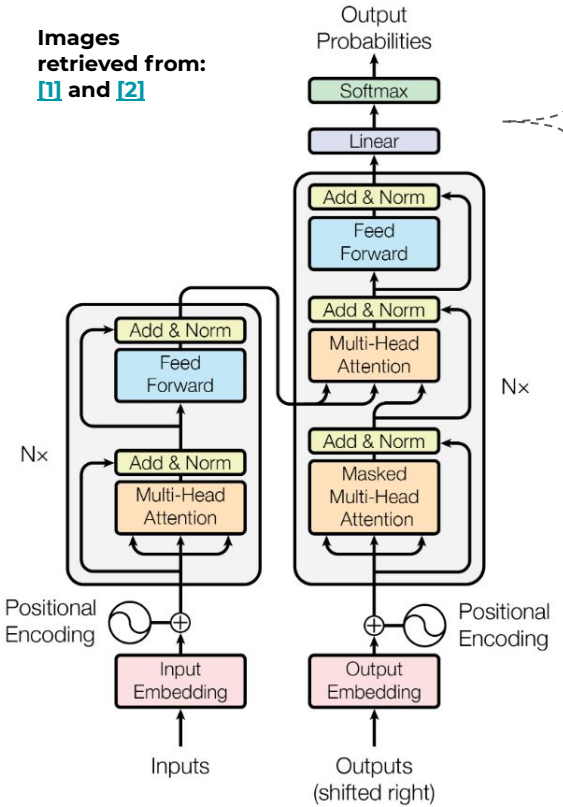
$$e_{ts} = a(h_t, hs)$$

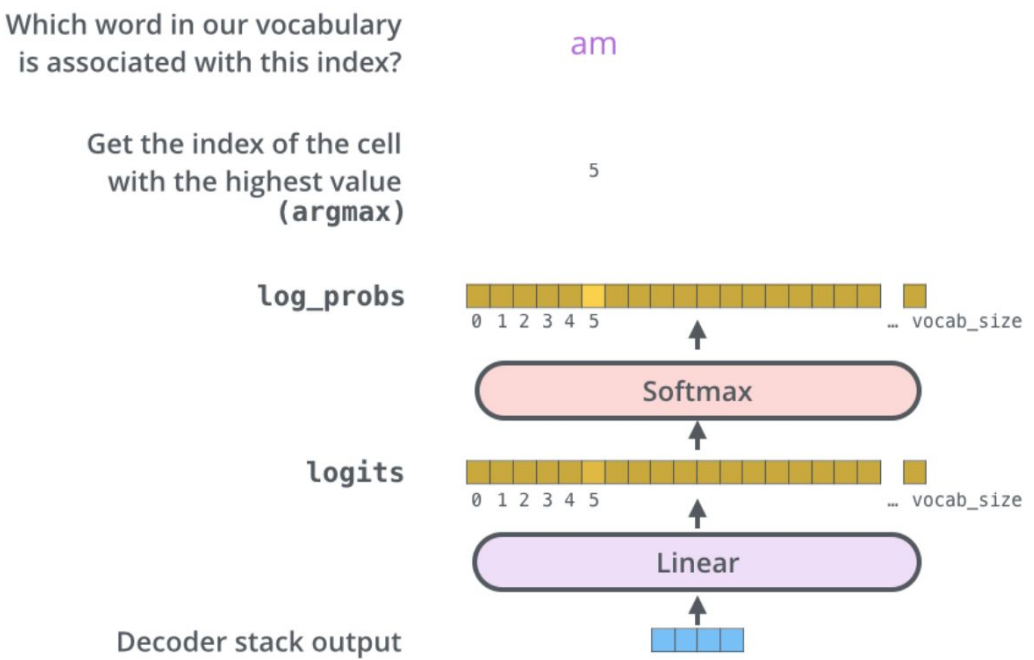
$$\alpha_{ts} = \frac{e_{ts}}{\sum_{k=1}^{X} e_{tk}}$$

$$c_t = \sum_{s=1}^{X} \alpha_{ts} h_s$$

Decoder Multi-Head Attention

Images retrieved from: [1] and [2]





04

Some Transformer Applications (in NLP)

Transformer-Based Transfer Learning

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

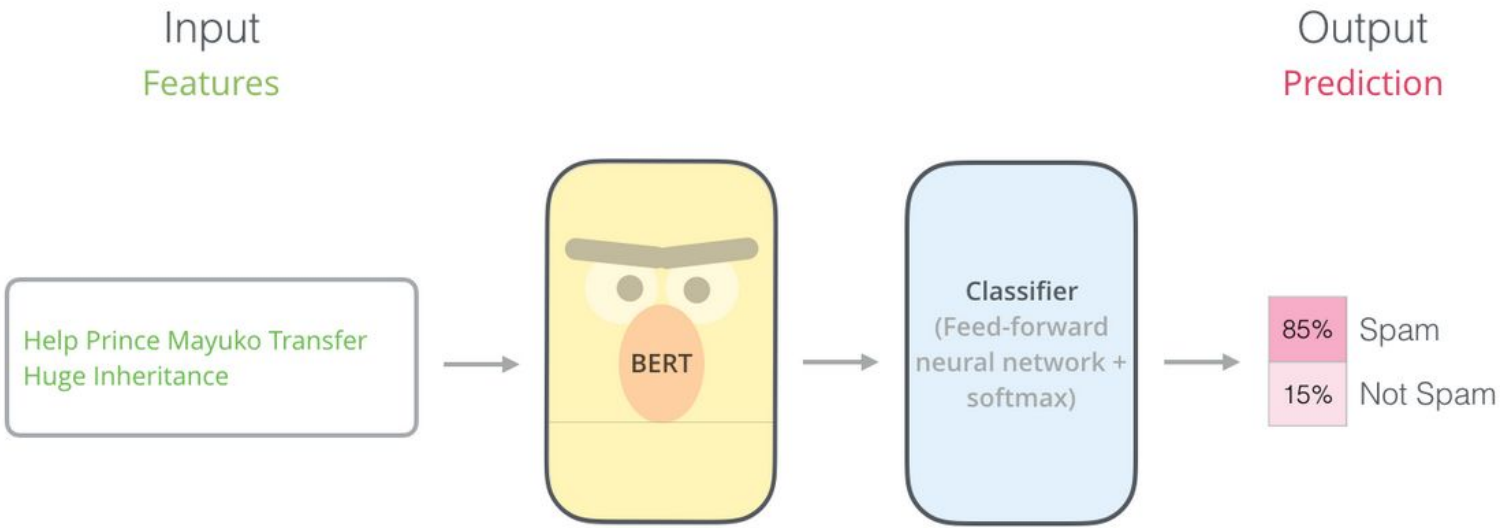
Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-

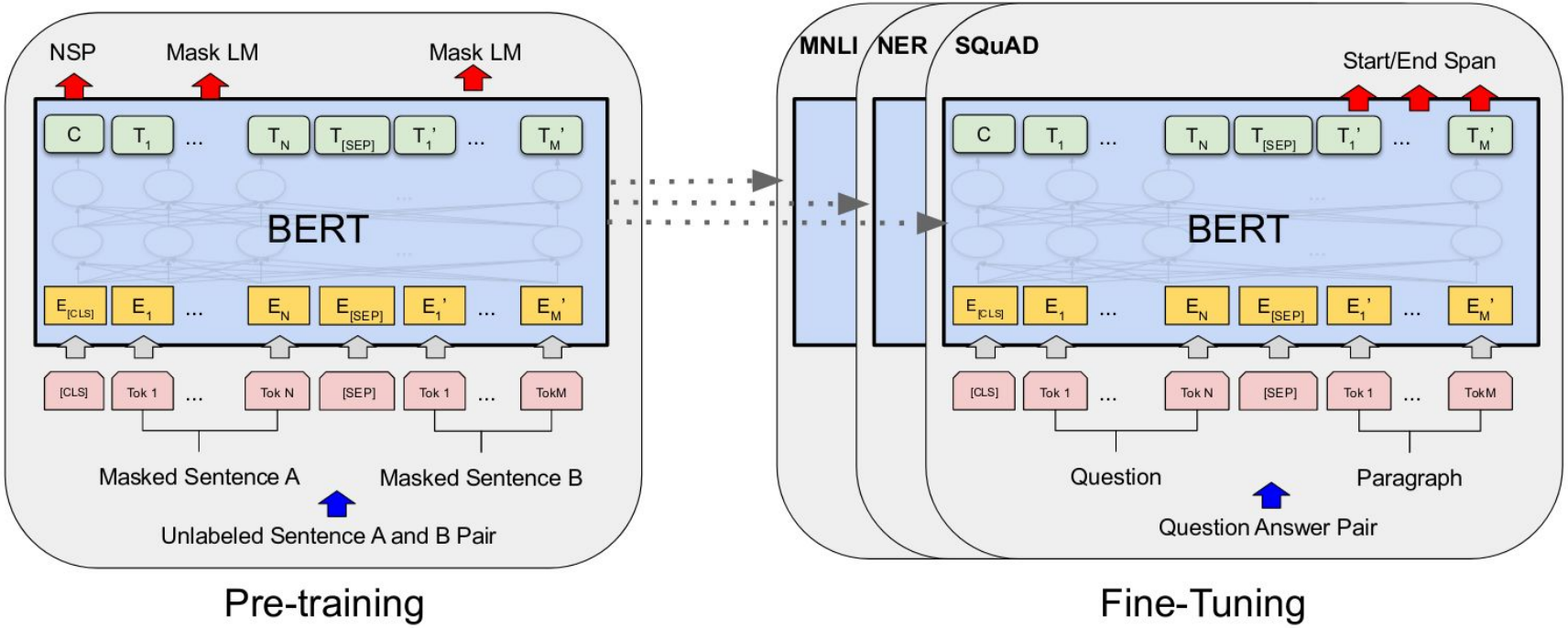
Retrieved from: [4]

Transformer-Based Transfer Learning



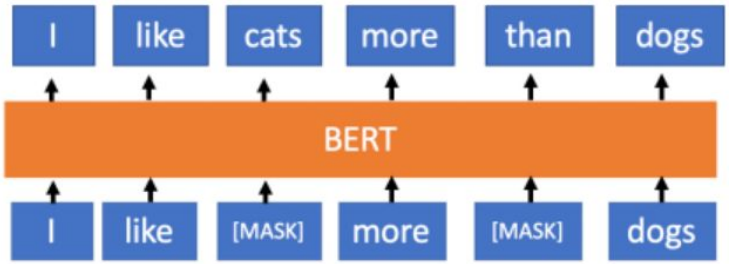
Images Retrieved from: [5]

Transformer-Based Transfer Learning



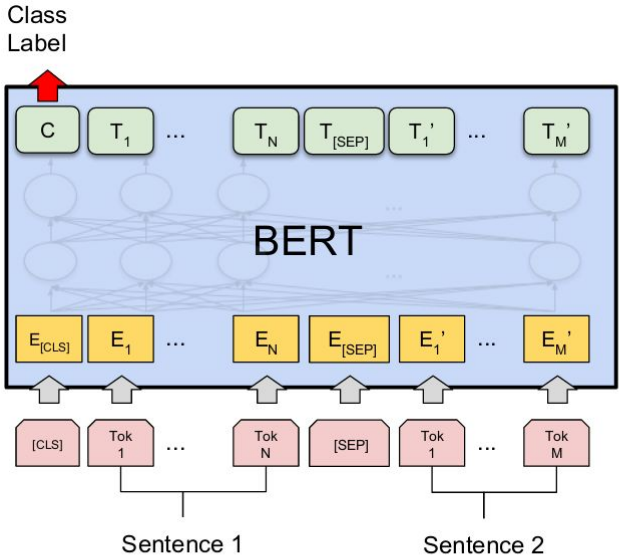
Images Retrieved from: [4]

Transformer-Based Transfer Learning

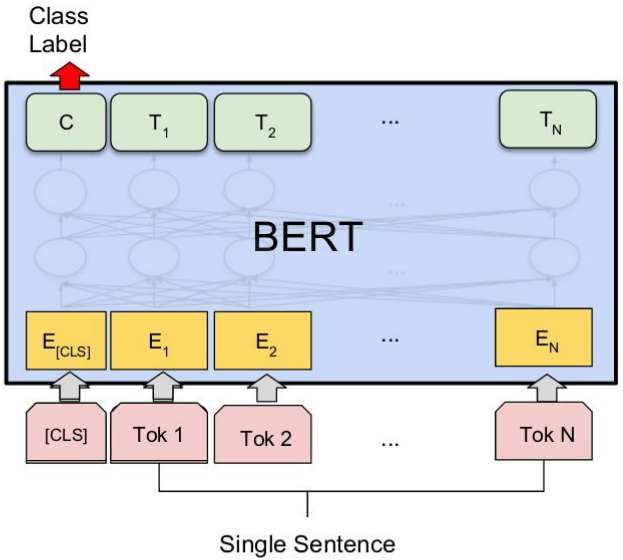


Images Retrieved from: Link

Transformer-Based Transfer Learning



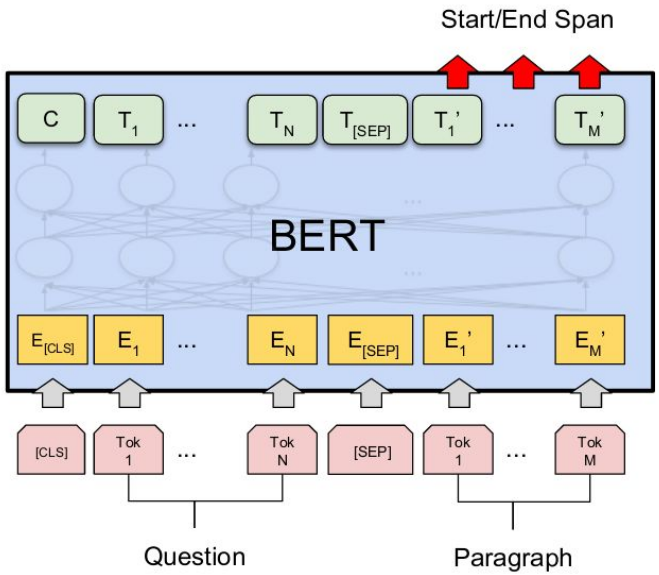
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



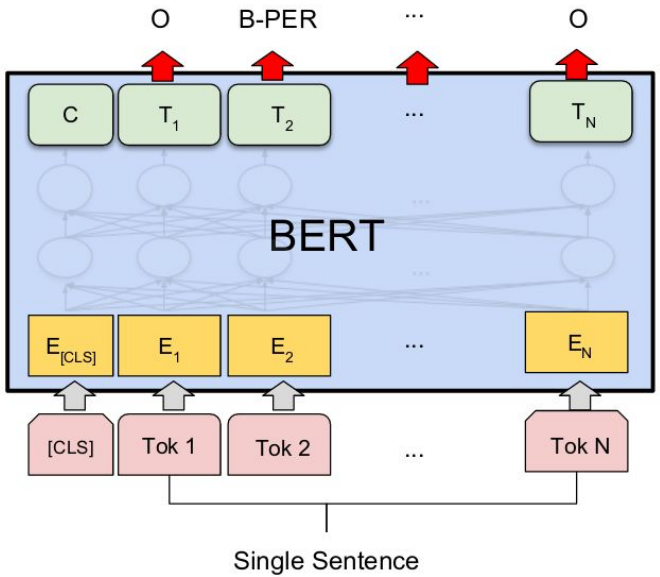
(b) Single Sentence Classification Tasks: SST-2, CoLA

Images Retrieved from: [4]

Transformer-Based Transfer Learning



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Images Retrieved from: [4]

Transformer-Based Transfer Learning

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	
1	Microsoft D365 AI & MSR AI	MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	5
2	T5 Team - Google	T5		89.7	70.8	97.1	91.9/89.2	92.5/92.1	74.6/90.4	92.0	91.7	96.7	92.5	93.2	5
3	ERNIE Team - Baidu	ERNIE		89.6	72.2	97.5	93.0/90.7	92.9/92.5	75.0/90.6	91.2	90.6	98.0	90.3	90.4	4
4	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.5	71.3	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.7	90.4	99.2	87.4	91.8	4
5	XLNet Team	XLNet (ensemble)		89.5	70.2	97.1	92.9/90.5	93.0/92.6	74.7/90.4	90.9	90.9	99.0	88.5	92.5	4
6	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5	91.3	91.0	99.2	89.2	91.8	5
7	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3	91.1	90.7	98.8	88.7	89.0	5
8	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	98.9	88.2	89.0	4
9	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0	92.8	91.2	93.6	95.9	
10	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9	87.6	87.2	93.9	80.9	65.1	3

Retrieved from: [GLUE Benchmark Leaderboard]

Transformer-Based Transfer Learning

Rank	Name	Model	URL	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AX-g	AX-b
1	SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.8	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	99.3/99.7	76.6
2	T5 Team - Google	T5		88.9	91.0	93.0/96.4	94.8	88.2/62.3	93.3/92.5	92.5	76.1	93.8	92.7/91.9	65.6
3	Facebook AI	RoBERTa		84.6	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	91.0/78.1	57.9
4	IBM Research AI	BERT-mtl		73.5	84.8	89.6/94.0	73.8	73.2/30.5	74.6/74.0	84.1	66.2	61.0	97.8/57.3	29.6
5	SuperGLUE Baselines	BERT++		71.5	79.0	84.8/90.4	73.8	70.0/24.1	72.0/71.3	79.0	69.6	64.4	99.4/51.4	38.0
		BERT		69.0	77.4	75.7/83.6	70.6	70.0/24.1	72.0/71.3	71.7	69.6	64.4	97.8/51.7	23.0
		Most Frequent Class		47.1	62.3	21.7/48.4	50.0	61.1/0.3	33.4/32.5	50.3	50.0	65.1	100.0/50.0	0.0
		CBoW		44.5	62.2	49.0/71.2	51.6	0.0/0.5	14.0/13.6	49.7	53.1	65.1	100.0/50.0	-0.4
		Outside Best		-	80.4	-	84.4	70.4/24.5	74.8/73.0	82.7	-	-	-	-
-	Stanford Hazy Research	Snorkel [SuperGLUE v1.9]		-	-	88.6/93.2	76.2	76.4/36.3	-	78.9	72.1	72.6	-	47.6

Retrieved from: [SuperGLUE Benchmark Leaderboard]

Transformer-Based Transfer Learning

SEARCH

Understanding searches better than ever before

Pandu Nayak
Google Fellow and Vice President, Search

Published Oct 25, 2019

If there's one thing I've learned over the 15 years working on Google Search, it's that people's curiosity is endless. We see billions of searches every day, and 15 percent of those queries are ones we haven't seen before--so we've built ways to return results for queries we can't anticipate.

When people like you or I come to Search, we aren't always quite sure about the best way to formulate a query. We might not know the right words to use, or how to spell something, because often times, we come to Search looking to learn--we don't necessarily have the knowledge to begin with.

At its core, Search is about understanding language. It's our job to figure out what you're searching for and surface helpful information from the web, no matter how you spell or combine the words in your query. While we've continued to improve our language understanding capabilities over the years, we sometimes still don't quite get it right, particularly with complex or conversational queries. In fact, that's one of the reasons why people often use "keyword-ese," typing strings of words that they think we'll understand, but aren't actually how they'd naturally ask a question.

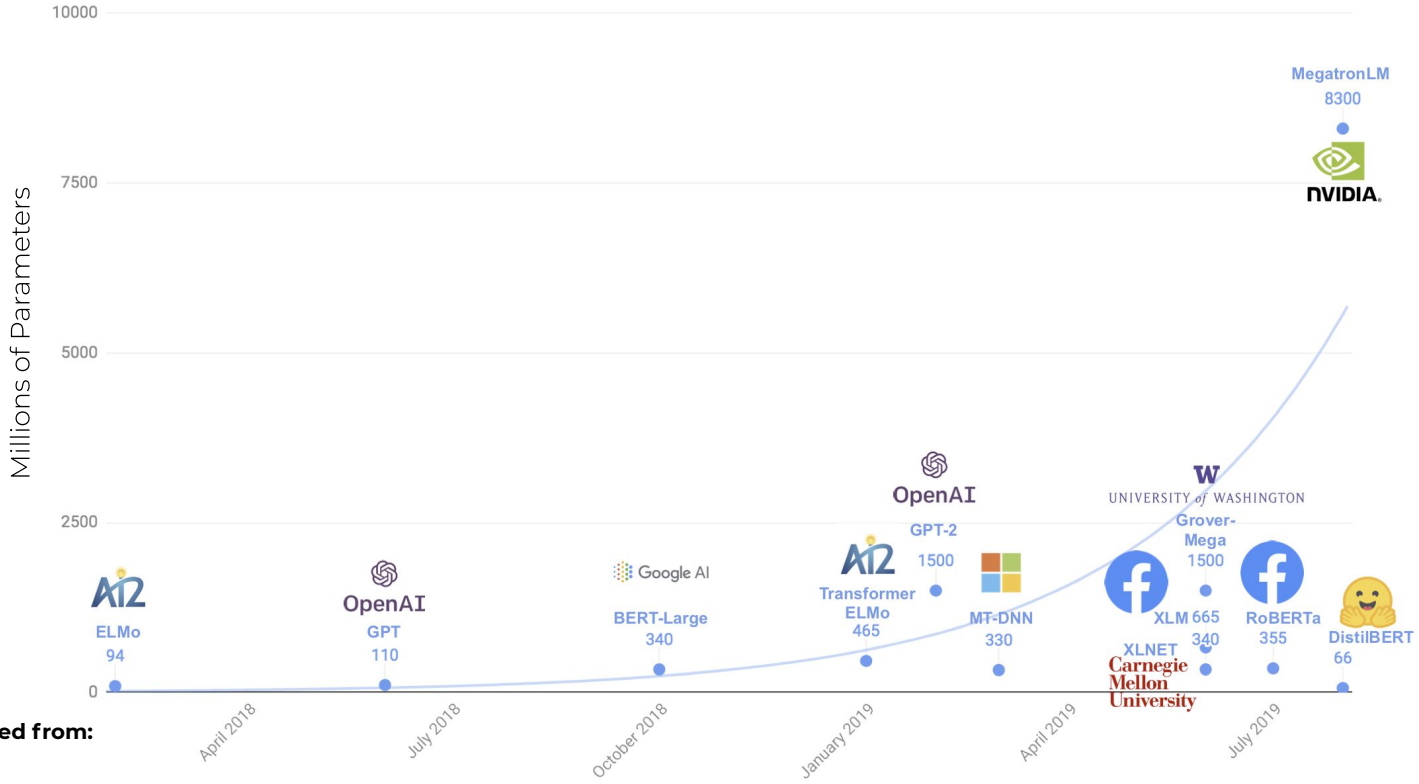
With the latest advancements from our research team in the science of language understanding--made possible by machine learning--we're making a significant improvement to how we understand queries, representing the biggest leap forward in the past five years, and one of the biggest leaps forward in the history of Search.

Applying BERT models to Search

Last year, we introduced and open-sourced a neural network-based technique for natural language processing (NLP) pre-training called Bidirectional Encoder Representations from Transformers, or as we call it--BERT, for short. This technology enables anyone to train their own state-of-the-art question answering system.

Retrieved from:
[Link]

Transformer-Based Transfer Learning



Retrieved from:
[Link]

GPT-2

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

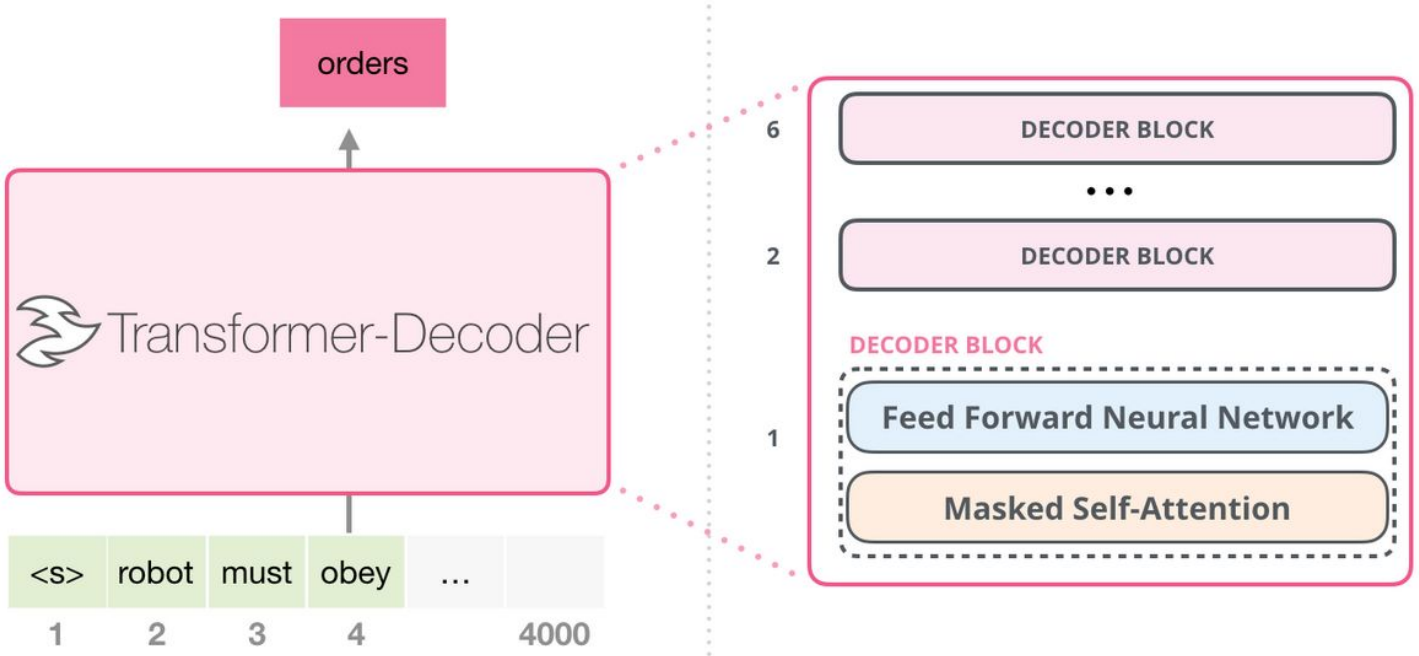
Making use of the Transformer Decoder

Retrieved from: Link

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

GPT-2



Images Retrieved from: [6]

GPT-2

Write With Transformer

Get a modern neural network to auto-complete your thoughts.

This web app, built by the Hugging Face team, is the official demo of the 🤗 `/transformers` repository's text generation capabilities.

Star 18,345

Retrieved from:
Link

Uber AI

See all models and checkpoints

🚕 Uber AI Plug and Play Language Model (PPLM)

Star 18,345

Retrieved from:
Link

GPT-2

Talk to Transformer

See how a modern neural network completes your text. Type a custom snippet or try one of the examples. Learn more below.

Follow @AdamDanielKing for more neat neural networks.

Sponsor the site. Show your product to 100,000s of monthly visitors.

Custom prompt

It's a cold December night, and here we are, listening to a talk about transformers. It's almost ending, but there's still time..

Retrieved from:
Link

GPT-2

It's a cold December night, and here we are, listening to a talk about transformers. It's almost ending, but there's still time...the talk is about how something that sounds kind of basic, like switching the circuit board on and off. Can really dramatically improve a car, or a house.

BUT, something far more complex, or expensive, is available. And what that does is it changes how people create life. By design.

Alone. Into a system.

We all know the story of Edison creating the lightbulb.

Retrieved from:
Link

05

Guide for a Practical Approach

First, play around with existing libraries

State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch

Transformers (formerly known as `pytorch-transformers` and `pytorch-pretrained-bert`) provides state-of-the-art general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch.

Features

- As easy to use as pytorch-transformers
- As powerful and concise as Keras
- High performance on NLU and NLG tasks
- Low barrier to entry for educators and practitioners

Retrieved from: Link

Or if you are used to spacy..

spacy-transformers

This package (previously `spacy-pytorch-transformers`) provides spaCy model pipelines that wrap Hugging Face's `transformers` package, so you can use them in spaCy. The result is convenient access to state-of-the-art transformer architectures, such as BERT, GPT-2, XLNet, etc. For more details and background, check out our blog post.

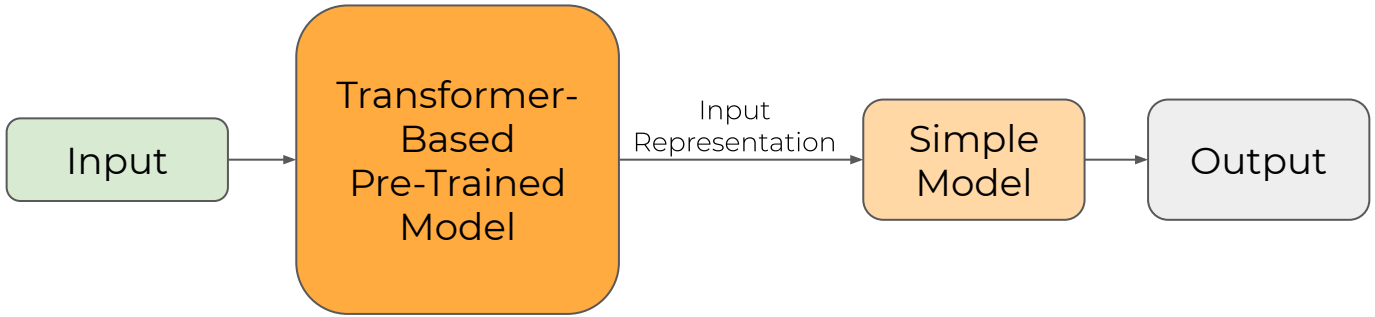
build never built | pypi v0.5.1 | release v0.5.1 | code style black | Open in Colab

Features

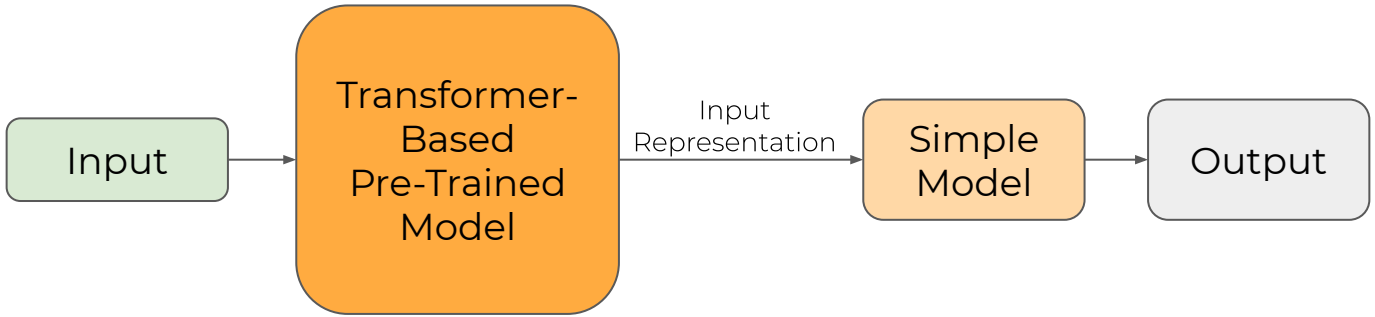
- Use **BERT**, **RoBERTa**, **XLNet** and **GPT-2** directly in your spaCy pipeline.
- **Fine-tune** pretrained transformer models on your task using spaCy's API.
- Custom component for **text classification** using transformer features.
- Automatic **alignment** of wordpieces and outputs to linguistic tokens.
- Process multi-sentence documents with intelligent **per-sentence prediction**.
- Built-in hooks for **context-sensitive vectors** and similarity.
- Out-of-the-box serialization and model packaging.

Retrieved from: Link

Try fine-tuning a model

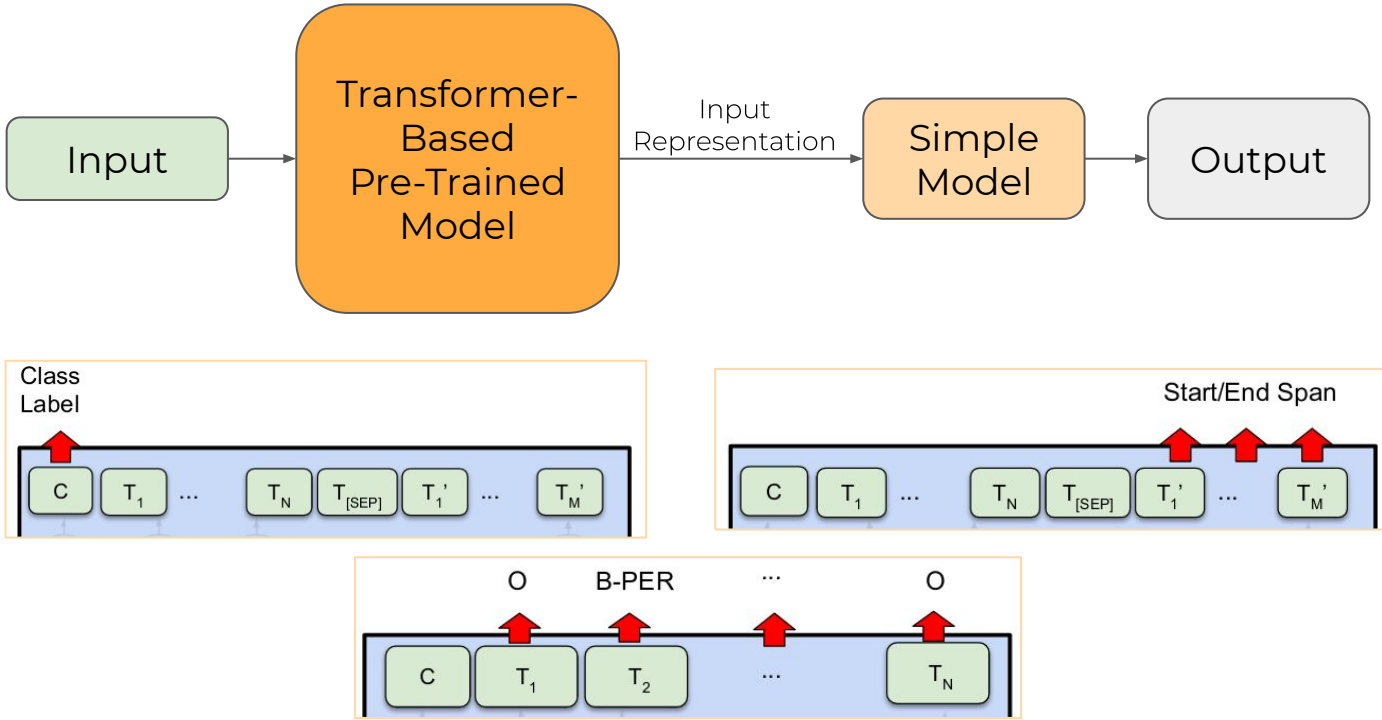


Try fine-tuning a model

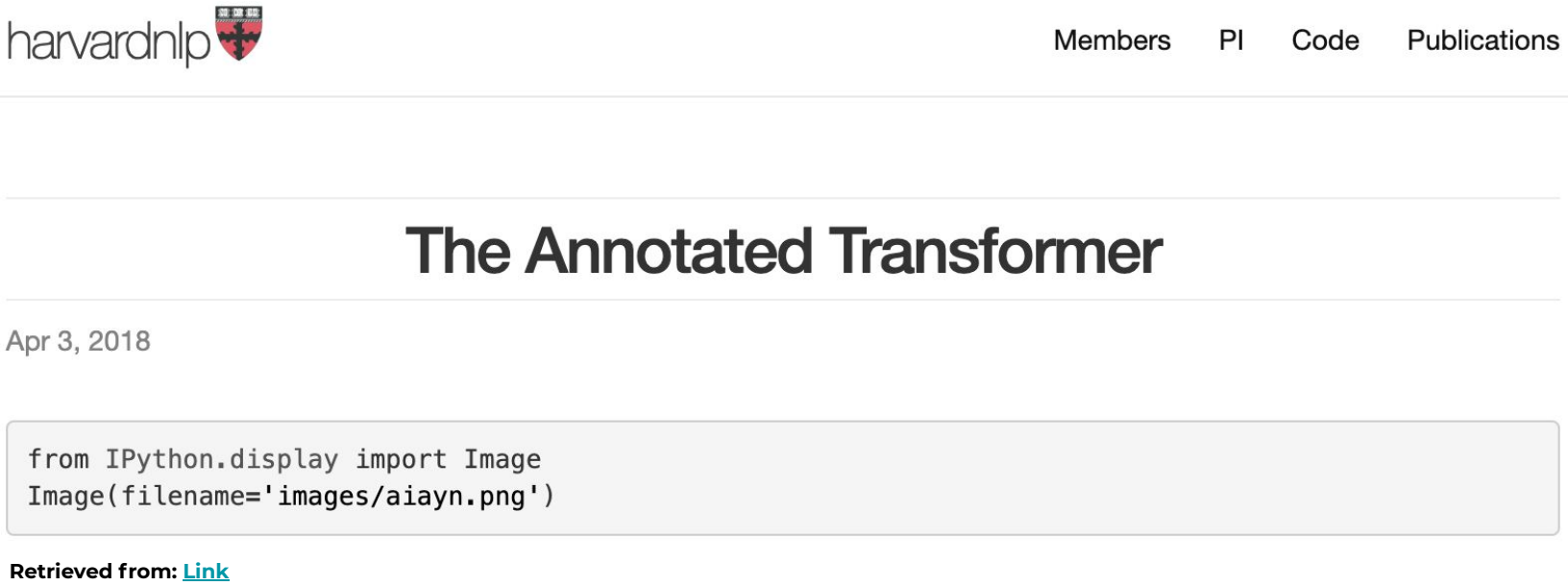



```
# Transformers has a unified API
# for 8 transformer architectures and 30 pretrained weights.
#          Model          | Tokenizer          | Pretrained weights shortcut
MODELS = [(BertModel,       BertTokenizer,       'bert-base-uncased'),
          (OpenAIGPTModel,  OpenAIGPTTokenizer,  'openai-gpt'),
          (GPT2Model,       GPT2Tokenizer,       'gpt2'),
          (CTRLModel,       CTRLTokenizer,       'ctrl'),
          (TransfoXLModel,  TransfoXLTokenizer,  'transfo-xl-wt103'),
          (XLNetModel,      XLNetTokenizer,      'xlnet-base-cased'),
          (XLMModel,        XLMTokenizer,        'xlm-mlm-enfr-1024'),
          (DistilBertModel, DistilBertTokenizer, 'distilbert-base-uncased'),
          (RobertaModel,    RobertaTokenizer,    'roberta-base')]
```

Try fine-tuning a model



Finally, learn to implement the details



Retrieved from: Link

06

Resources

Resources

- [1] - *Attention is All you Need*
- [2] - The Illustrated Transformer
- [3] - Transformers from scratch
- [4] - *BERT: Pre-training of Deep Bidirectional Transformer for Language Understanding*
- [5] - The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)
- [6] - The Illustrated GPT-2 (Visualizing Transformer Language Models)
- [EXTRA] - The Annotated Transformer
- [EXTRA] - *Image Transformer*
- [EXTRA] - *Training Tips for the Transformer Model*

Questions?