An Introduction to Google AI's BERT:

Pre-training of Deep Bidirectional Transformers for Language Understanding

He Ying

Nov 23rd , 2018

Performance of BERT

BERT obtains new state-of-the-art results on 11 NLP tasks.

Thang Luong
@lmthang

正在关注

A new era of NLP has just begun a few days ago: large pretraining models (Transformer 24 layers, 1024 dim, 16 heads) + massive compute is all you need. BERT from @GoogleAI: SOTA results on everything arxiv.org/abs/1810.04805. Results on SQuAD are just mind-blowing. Fun time ahead!

翻译推文

SQuAD1.1 Leaderboard

Since the release of SQuAD1.0, the community has made rapid progress, with the best models now rivaling human performance on the task. Here are the ExactMatch (EM) and F1 scores evaluated on the test set of v1.1.

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	**87.433**	**93.160**
2 Oct 05, 2018	BERT (single model) *Google AI Language* https://arxiv.org/abs/1810.04805	85.083	91.835
2 Sep 09, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.356	91.202
2 Sep 26, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.954	91.677
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

What problem does BERT solve?

- Usually, a Transformer model needs to train a great number of parameters.(BERT Base Model has 12 * 768 * 12 = 110M parameters).
- Training parameters requires massive corpus.
- Manual labeling for massive corpus is extremely expensive.

What problem does BERT solve?

- Inspired by *A Neural Probabilistic Language Model(Yoshua Bengio et al., 2003)*, BERT uses the unsupervised methodology to train transformer models.

  (1) Can we use word vectors to represent the semantics of natural language?

  (2) How to set every word to an appropriate numeric vector?

  (3) Every article is born to be a training corpus and needs no manual labeling.

What problem does BERT solve?

- BERT aims to pre-train a generic language model , named Language Representation Model, which can be applied to other NLP tasks.

- Based on the pre-trained Language Representation Model, fine-tuning the model for a supervised downstream task, in which few parameters need to be learned from scratch.

Cores of BERT

- Pre-training
- Deep (L=12)
- Bidirectional
- Transformers
- Language Understanding

Why BERT ?

- Language model pre-training has shown to be effective for improving may natural language processing tasks.

  Sentence-level tasks such as NPI and paraphrasing

  Token-level tasks such as NER and SQuAD question answering

Why BERT ?

- Two existing strategies for applying pre-trained language representations to downstream tasks:

  (1) feature-based (ELMo, 2018)

  (2) fine-tuning (OpenAI GPT, 2018)

  Both approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations. Limiting the choice of architectures that can be used during pre-training.

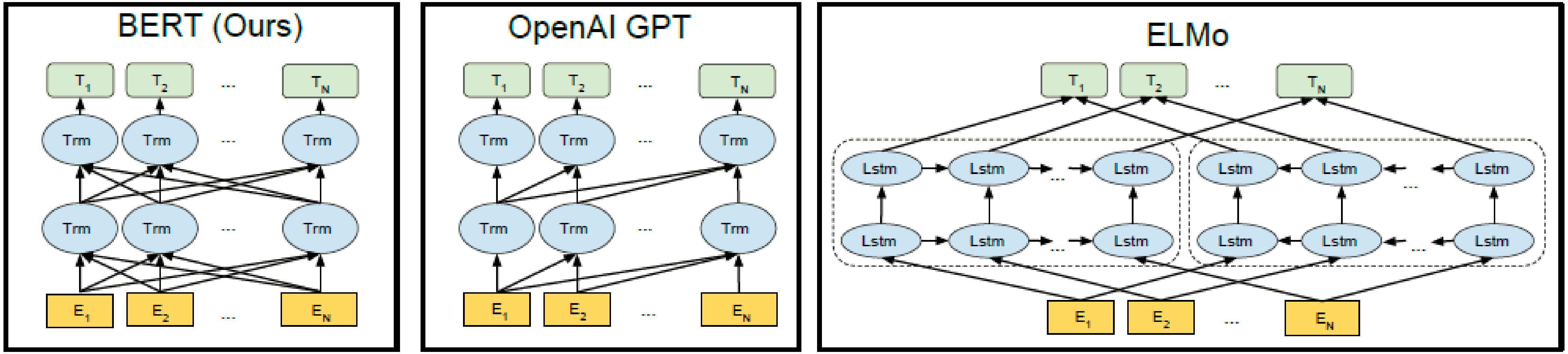
Innovation of BERT

- Proposing a new pre-training objective: the "masked language model" (MLM).

  Unlike left-to-right language model pre-training, the MLM objective allows the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer.

- Introducing a "next sentence prediction" task that jointly pre-trains text-pair representations.

Innovation of BERT



Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

Detailed Implementation of BERT

1. Model Architecture

- Denoting

  L : the number of layers(i.e., Transformer blocks)

  H : the hidden size

  A : the number of self-attention heads

Detailed Implementation of BERT

1. Model Architecture

- Two model size:

$$BERT_{BASE}: \ L = 12, H = 768, A = 12, \text{Total Parameters} = 110\text{M}$$

$$BERT_{LARGE}: \ L = 24, H = 1024, A = 16, \text{Total Parameters} = 340\text{M}$$

In all cases we set the feed-forward/filter size to be 4H,

i.e., 3072 for the H = 768 and 4096 for the H = 1024.

Detailed Implementation of BERT

2. Input Representation

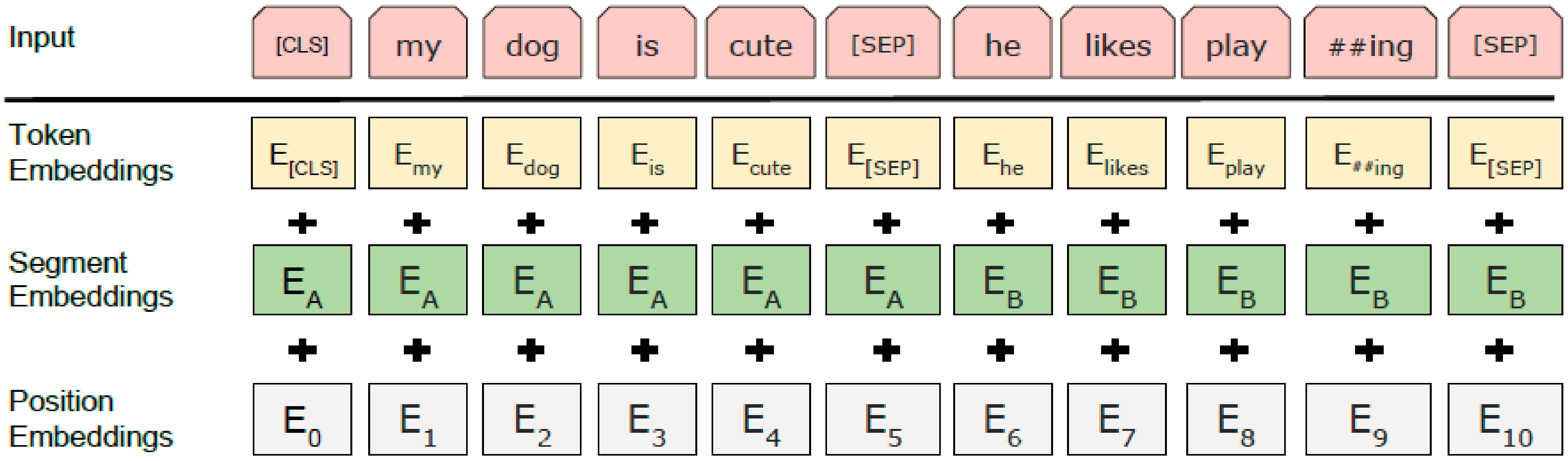
- The specifics are:

  (1) *We use WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary. We denote split word pieces with ##.*

  (2) We use learned positional embeddings with supported sequence lengths up to 512 tokens.

Detailed Implementation of BERT

2. Input Representation



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

Detailed Implementation of BERT

3. Pre-training Task I: Masked Language Model

- The difficulty to train a deep bidirectional model:

  Standard conditional language models can only be trained left-to-right or right-to-left, since bidirectional conditioning would allow each word to indirectly "see itself" in a multi-layered context.

- A straightforward approach:

  Mask some percentage of the input tokens at random, and then only predict only those masked tokens. (Taylor, 1953)

Detailed Implementation of BERT

3. Pre-training Task I: Masked Language Model

- Mask 15% of all WordPiece tokens in each sequence at random.
- Only predict the masked words rather than reconstructing the entire input

Detailed Implementation of BERT

3. Pre-training Task I: Masked Language Model

- Two downsides:

  (1) Creating a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.

  To mitigate this, we do not always replace "masked" words with the actual [MASK] token.

Detailed Implementation of BERT

3. Pre-training Task I: Masked Language Model

- Two downsides:

  80% of the time: Replace the word with the [MASK] token,

  e.g., my dog is hairy → my dog is [MASK]

  10% of the time: Replace the word with a random word,

  e.g., my dog is hairy → my dog is apple

  10% of the time: Keep the word unchanged,

  e.g., my dog is hairy → my dog is hairy.

Detailed Implementation of BERT

3. Pre-training Task I: Masked Language Model

- Two downsides:

  (2) Only 15% of tokens are predicted in each batch, which suggests that more pre-training steps may be required for the model to converge.

  Experiments demonstrate that MLM does converge marginally slower than a left-to-right model (which predicts every token), but the empirical improvements of the MLM model far outweigh the increased training cost.

Detailed Implementation of BERT

3. Pre-training Task II: Next Sentence Prediction

- Many important downstream tasks such as QA and NPI are based on understanding the relationship between two text sentences, which is not directly captured by language modeling.
- Pre-train a binarized next sentence prediction task that can be trivially generated from any monolingual corpus.

Detailed Implementation of BERT

3. Pre-training Task II: Next Sentence Prediction

- For example,

  Input = [CLS] the man went to [MASK] store [SEP]

  he bought a gallon [MASK] milk [SEP]

  Label = IsNext

  Input = [CLS] the man [MASK] to the store [SEP]

  penguin [MASK] are flight ##less birds [SEP]

  Label = NotNext

- The final pre-trained model achieves 97%-98% accuracy at this task.

Detailed Implementation of BERT

4. Pre-training Procedure

- Pre-training corpus:

  (1) BooksCorpus (800M words) (Zhu et al., 2015)

  (2) English Wikipedia (2500M words, only the text passages)

- Experiment specifics:

  (1) Batch size: 256

  (2) Length of tokens: 512

  (3) Number of iterations: 1,000,000 steps

  (4) Number of epoch: about 40 epochs over the 3.3 billion word corpus.

  (5) Learning rate: 1e-4; L2 weight decay: 0.01

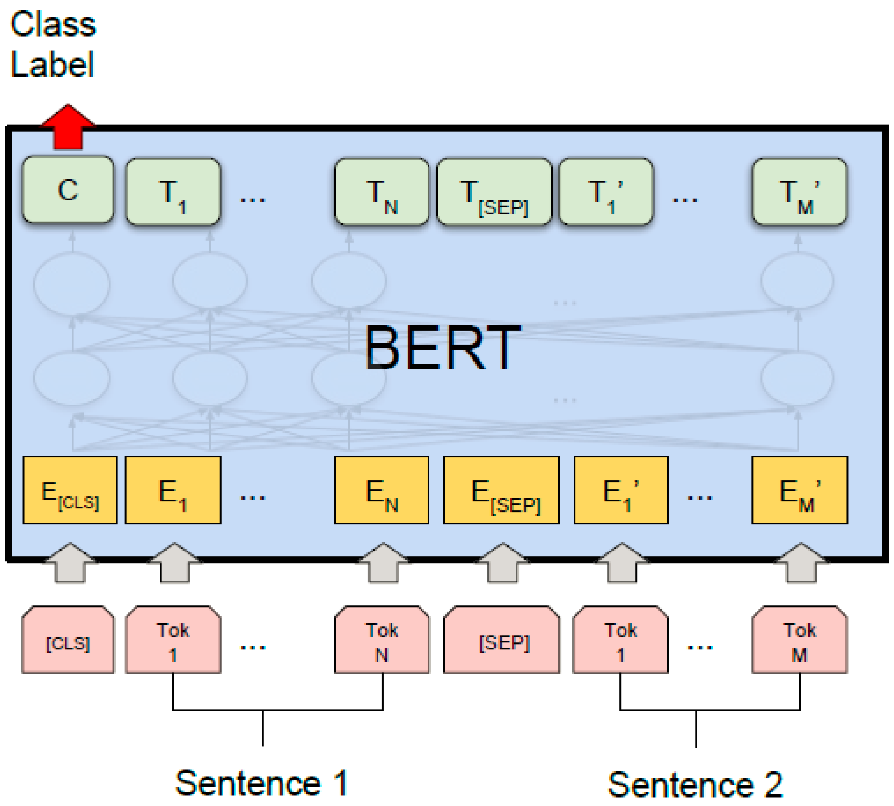
  (6) Dropout probability: 0.1 on every layer.

Detailed Implementation of BERT

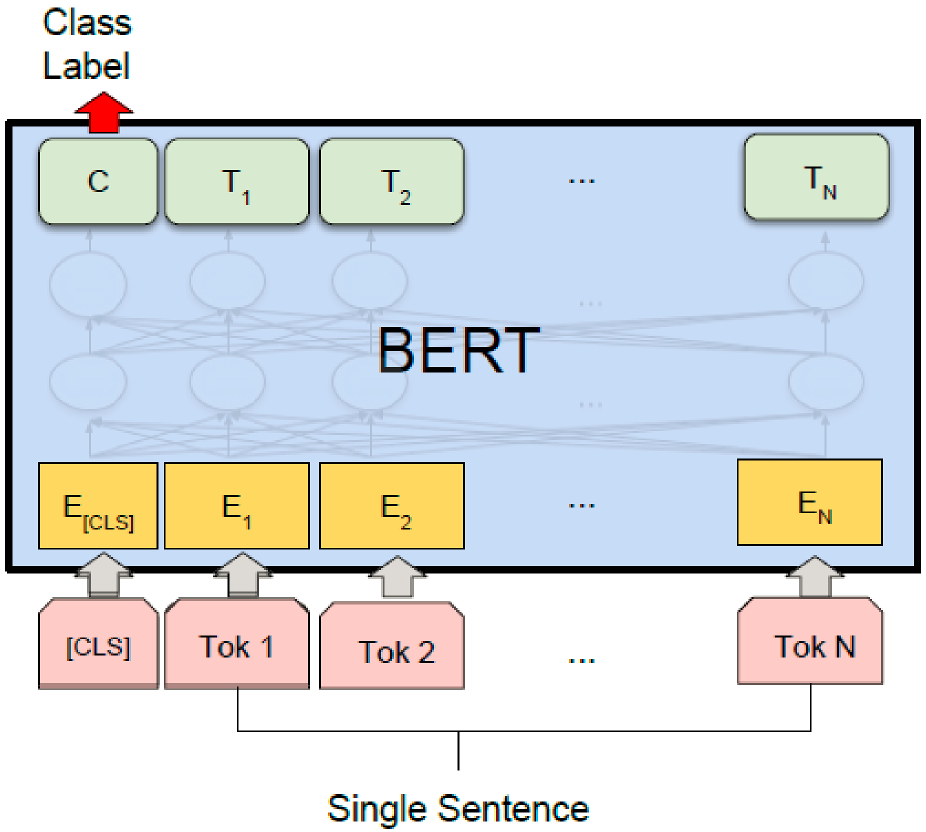
5. Fine-tuning Procedure

- Sequence-level classification tasks (Using the special [CLS] word embedding)
- Token-level prediction tasks (slightly modified in a task-specific manner)
- The optimal hyperparameter values are task-specific, but we found the following range of possible values to work well across all tasks:
    (1) Batch size: 16, 32
    (2) Learning rate: 5e-5, 3e-5, 2e-5
    (3) Number of epoch: 3, 4
- Fine-tuning is typically very fast.

Experiment Results

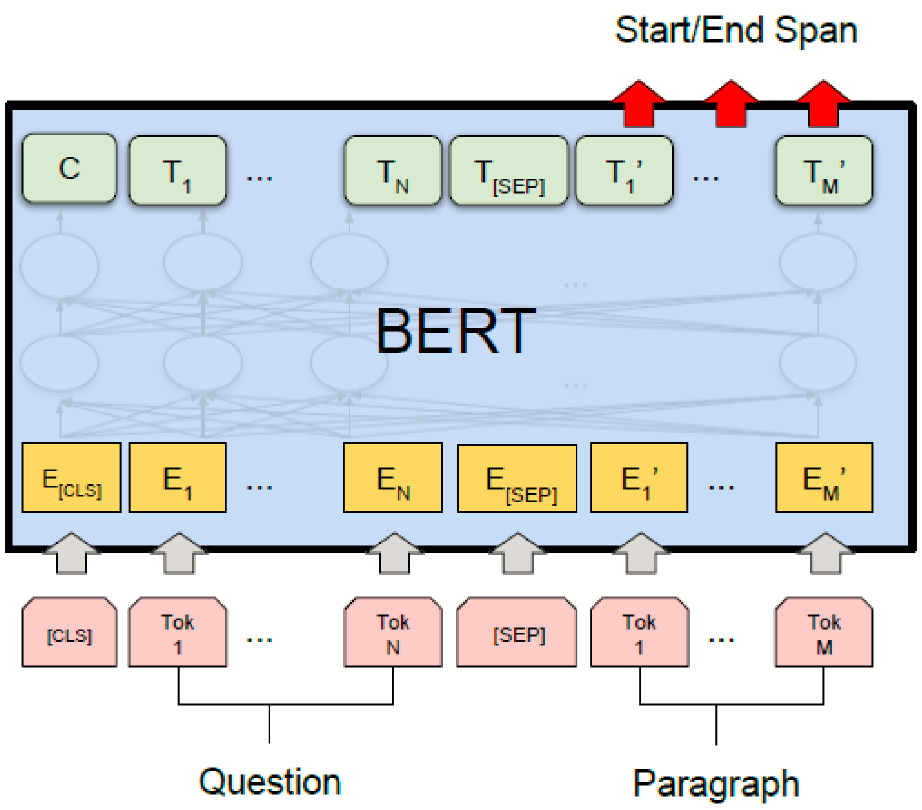


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

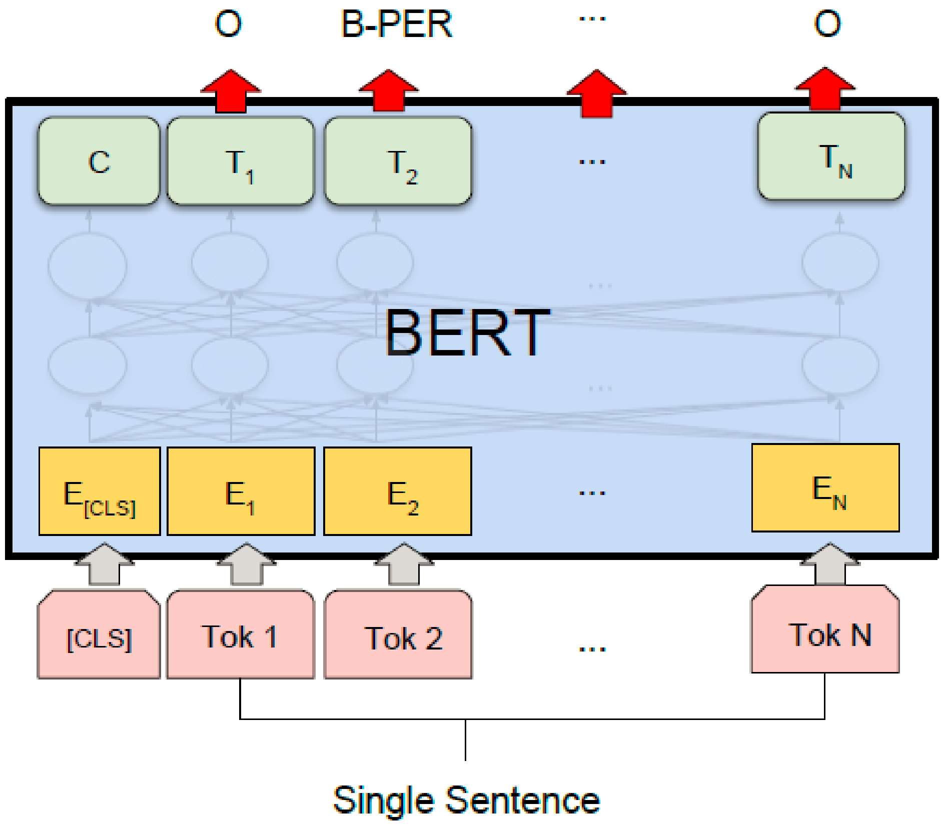


(b) Single Sentence Classification Tasks: SST-2, CoLA

Experiment Results



(c) Question Answering Tasks:
SQuAD v1.1



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

Experiment Results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); $BERT_{BASE}$ = (L=12, H=768, A=12); $BERT_{LARGE}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.

Experiment Results

System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

Table 3: CoNLL-2003 Named Entity Recognition results. The hyperparameters were selected using the Dev set, and the reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table 4: SWAG Dev and Test accuracies. Test results were scored against the hidden labels by the SWAG authors. †Human performance is measure with 100 samples, as reported in the SWAG paper.

Contributions of BERT

- Demonstrating the importance of bidirectional pre-training for language representations.
- Pre-trained representations eliminate the needs of many heavily engineered task-specific architectures.
- BERT advances the state-of-the-art for eleven NLP tasks.

Other Messages of BERT

- Deep learning is representation learning.
- Scale matters. (Data + Model + Computing Capability)
- Pre-training is important.

Other Messages of BERT

- Price of pre-training a BERT Model:

  For TPU pods:

  4 TPUs * ~$2/h (preemptible) * 24 h/day * 4 days = $768 (base model)

  16 TPUs = ~$3k (large model)

  For TPU:

  16 tpus * $8/hr * 24 h/day * 4 days = 12k

  64 tpus * $8/hr * 24 h/day * 4 days = 50k

Thank you !

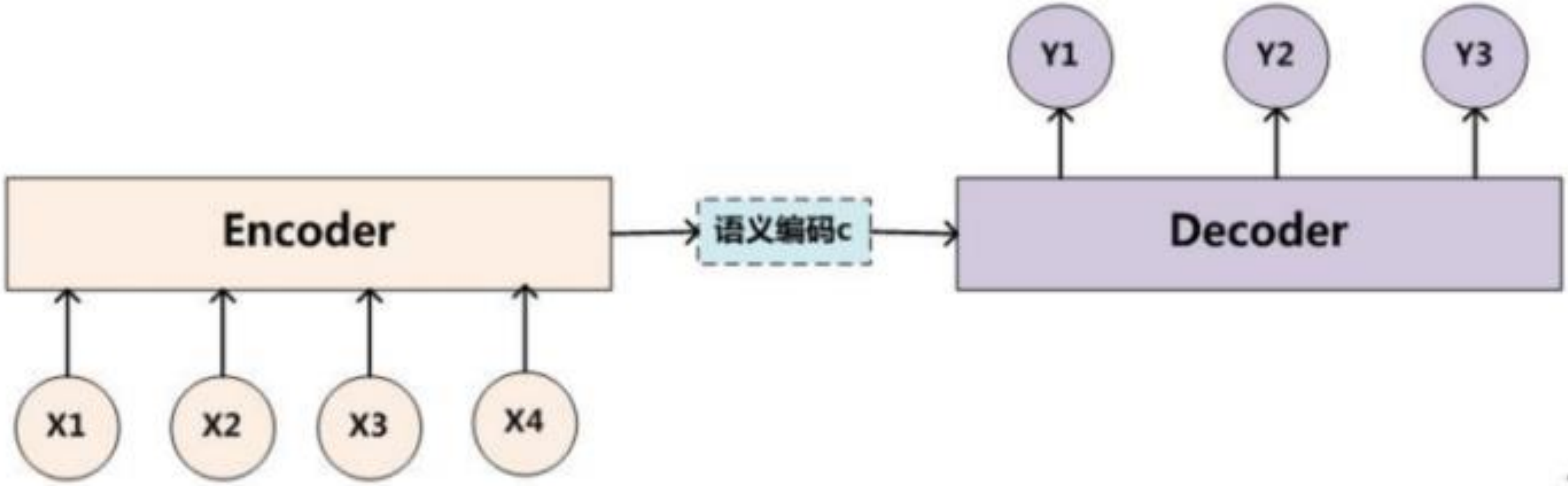
Attention is All You Need

Google AI, 2017

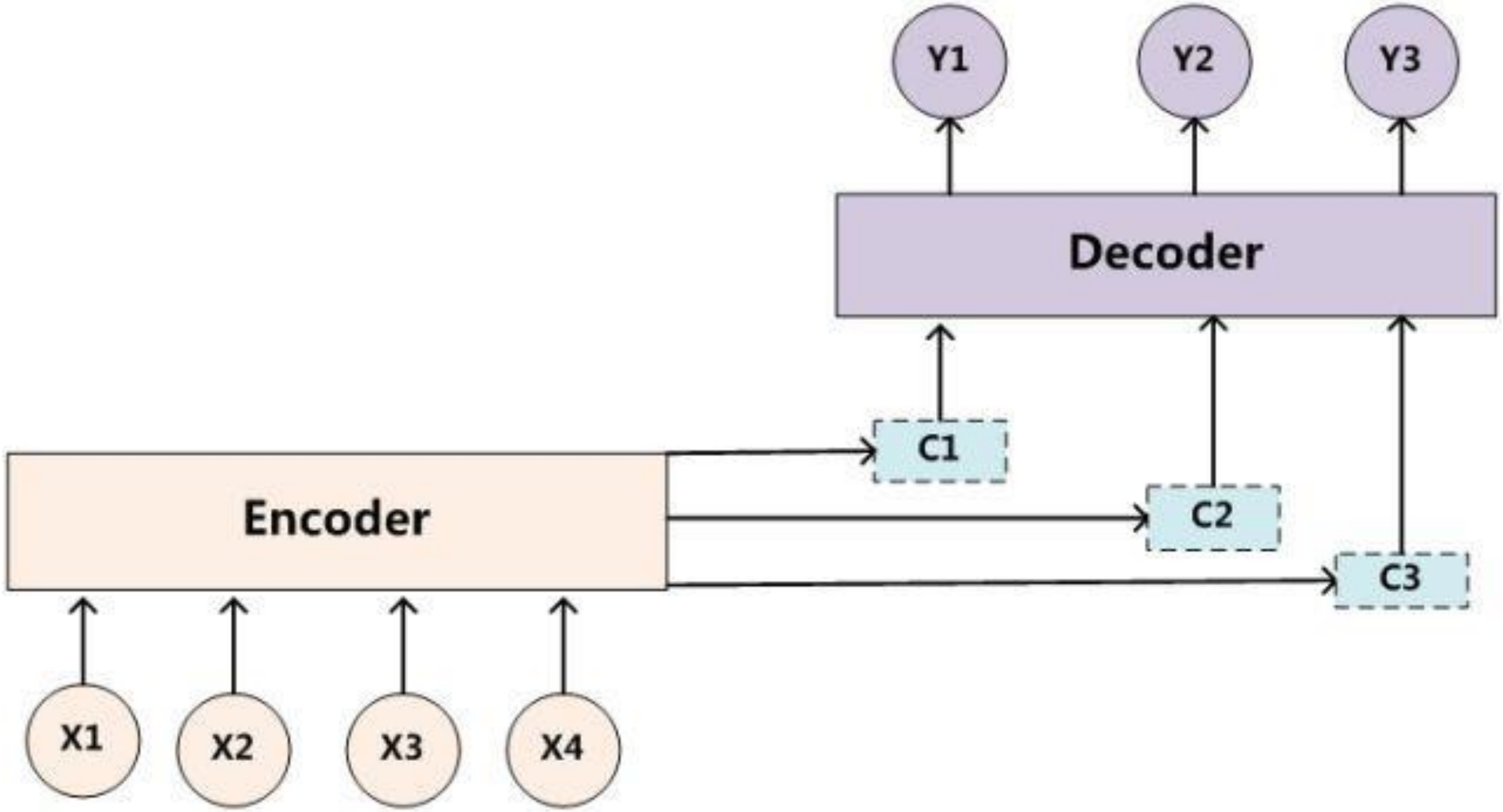
The Illustrated Transformer:

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

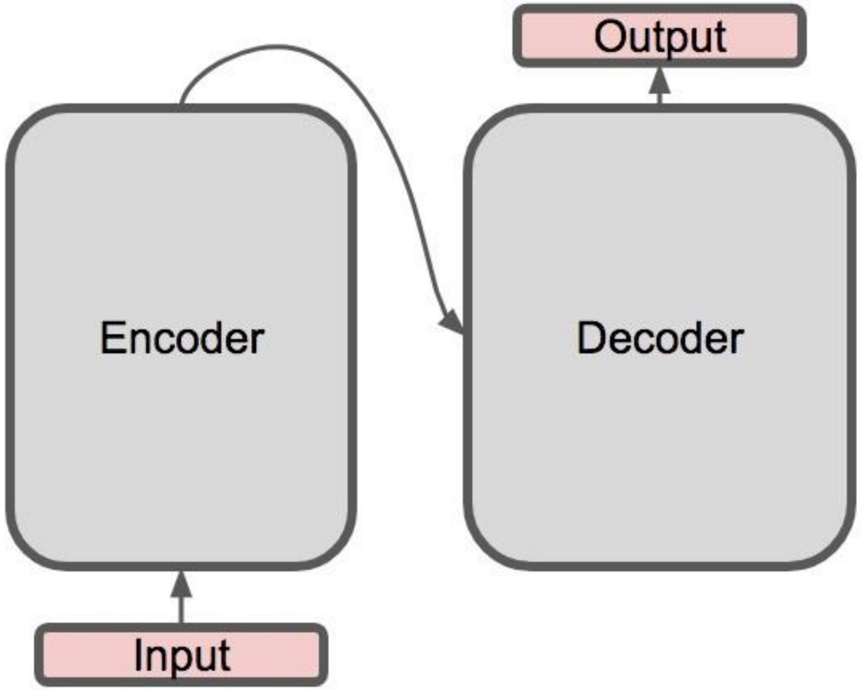
Attention Mechanism

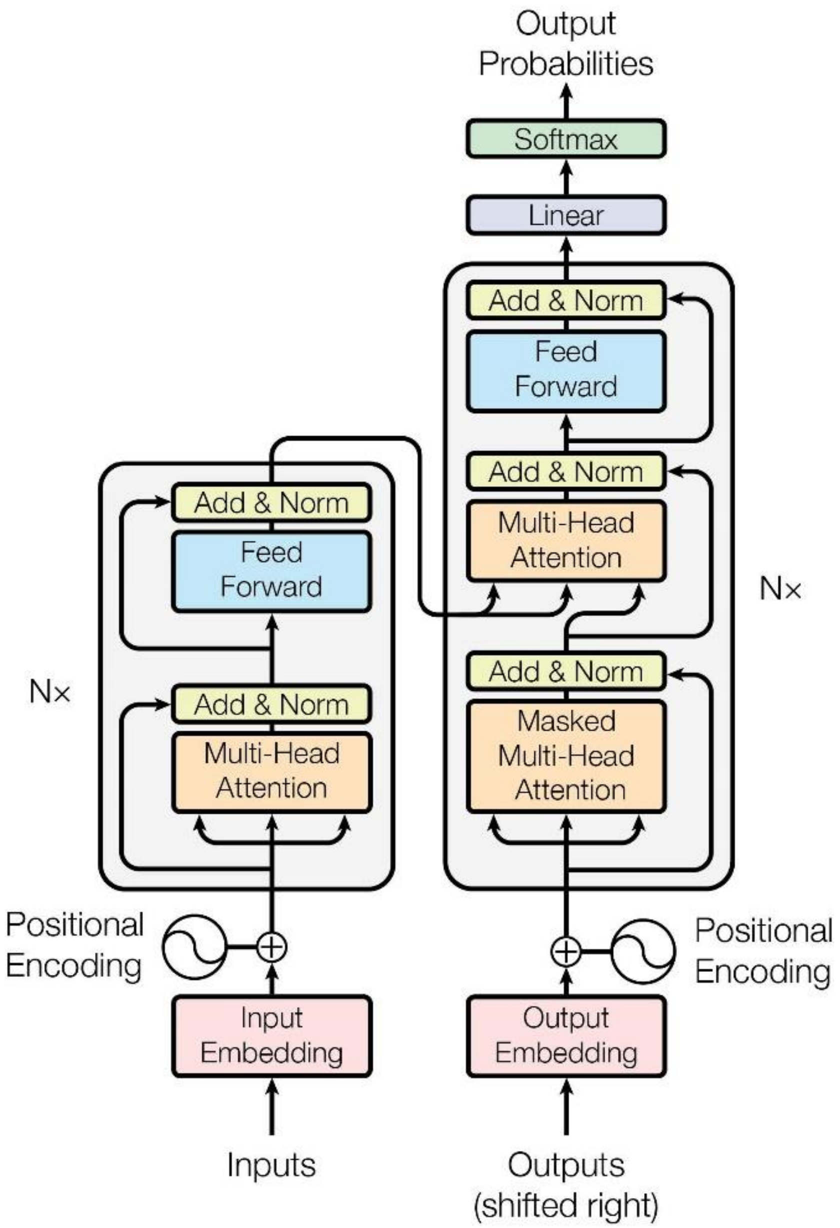


Attention Mechanism

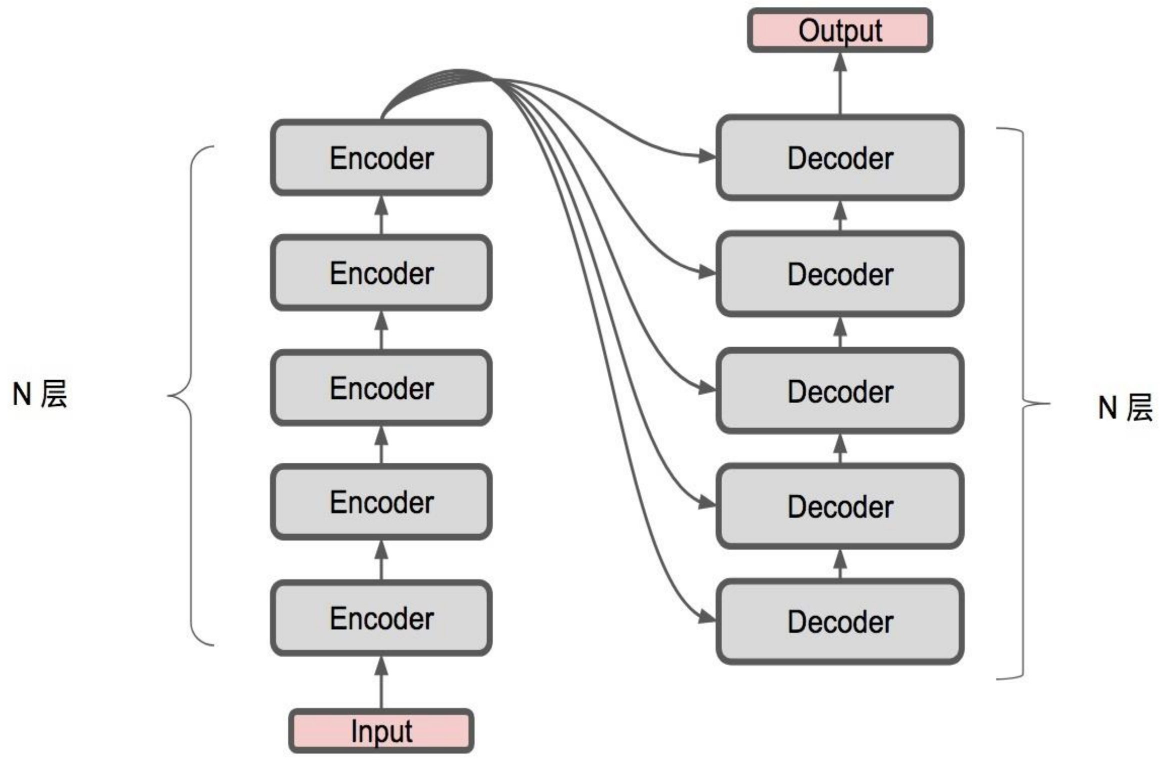


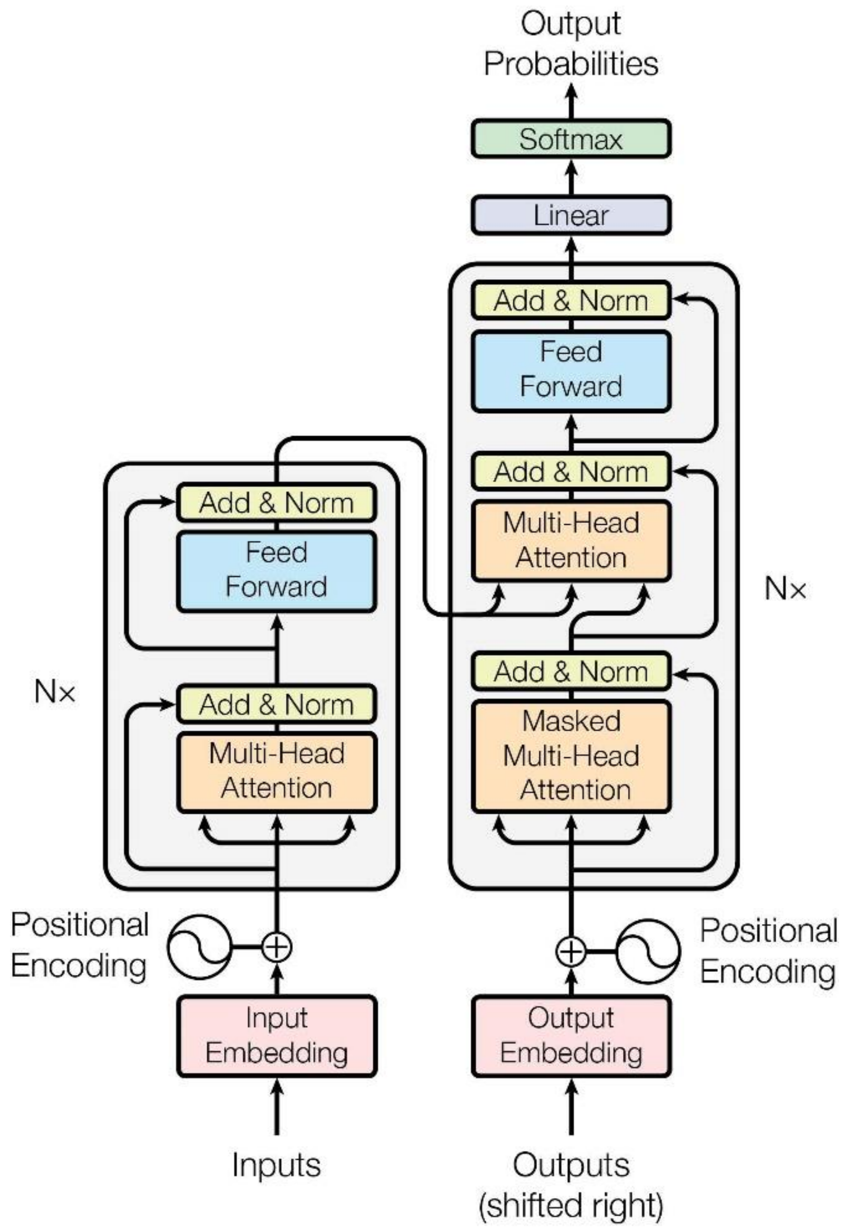
A Transformer Model



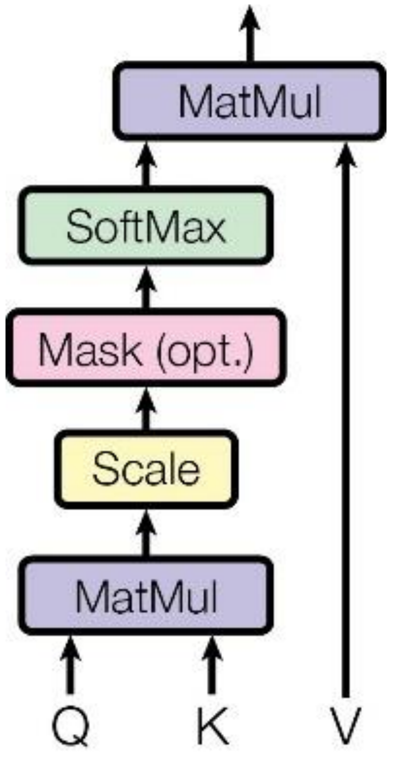


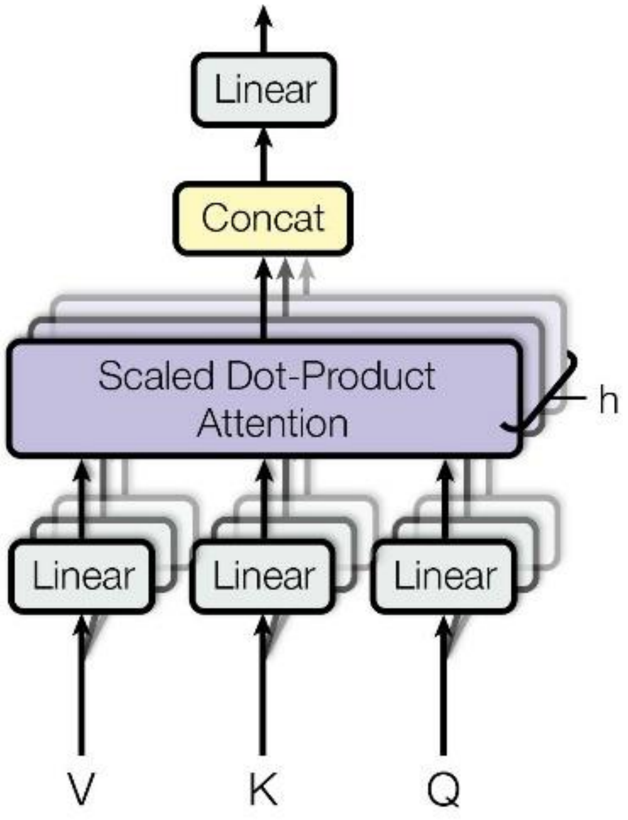
A Transformer Model

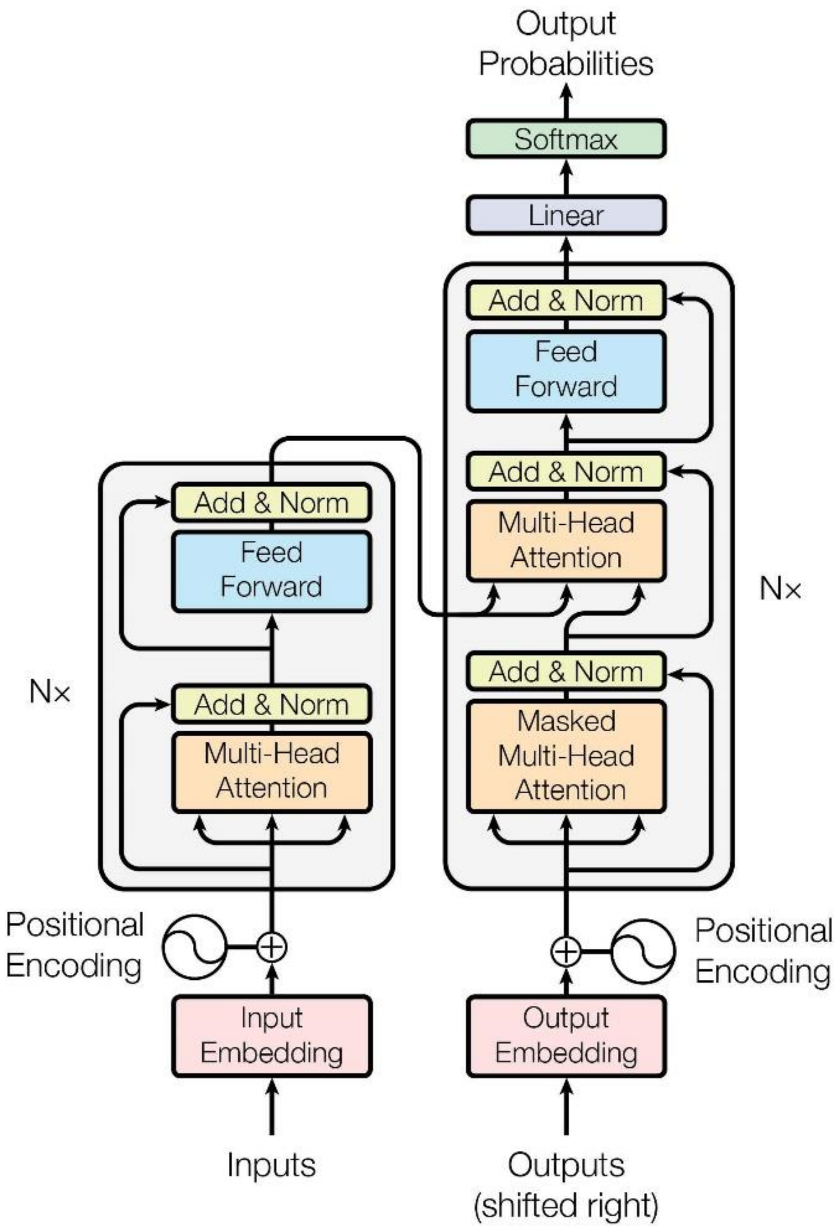




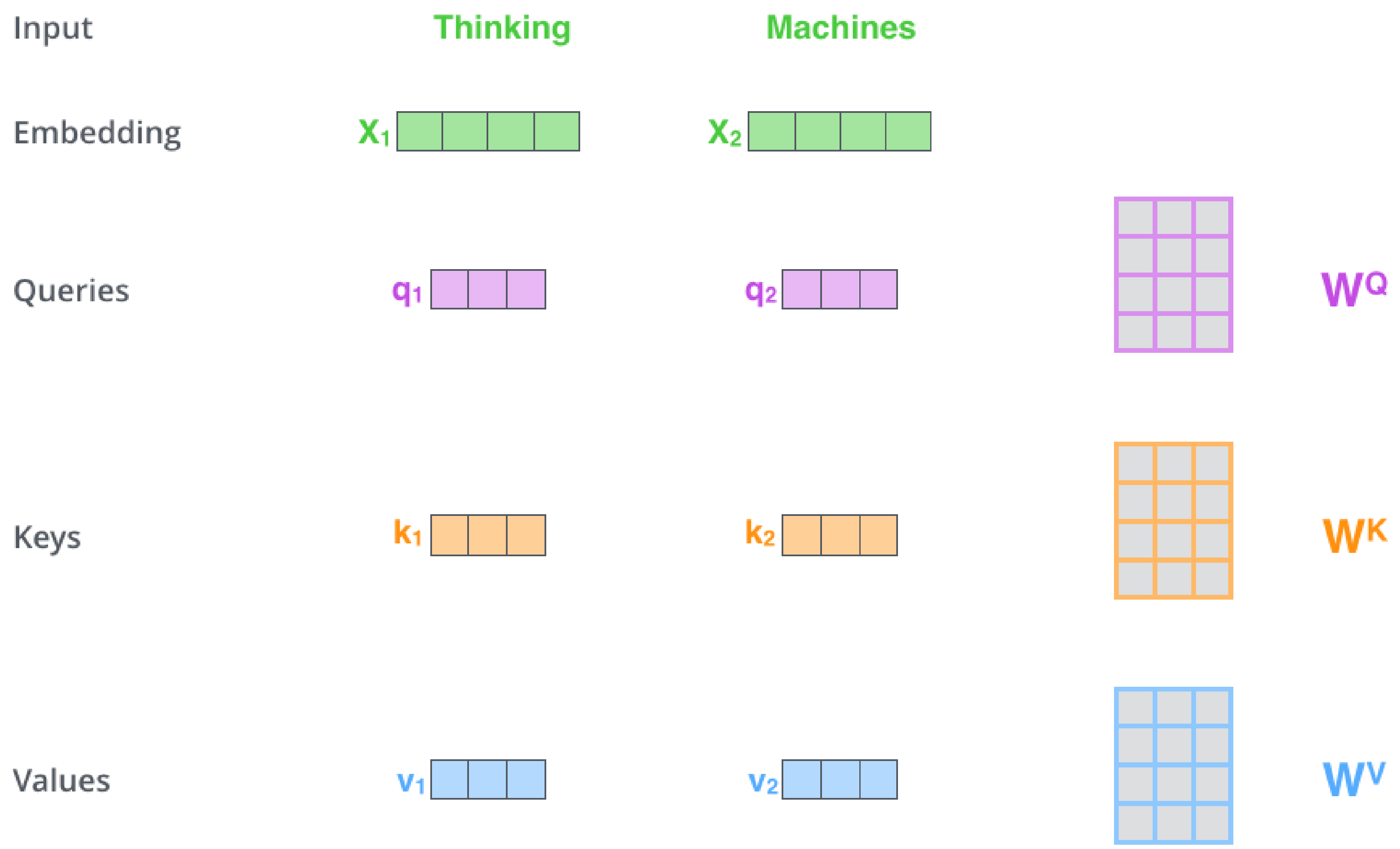
A Transformer Model



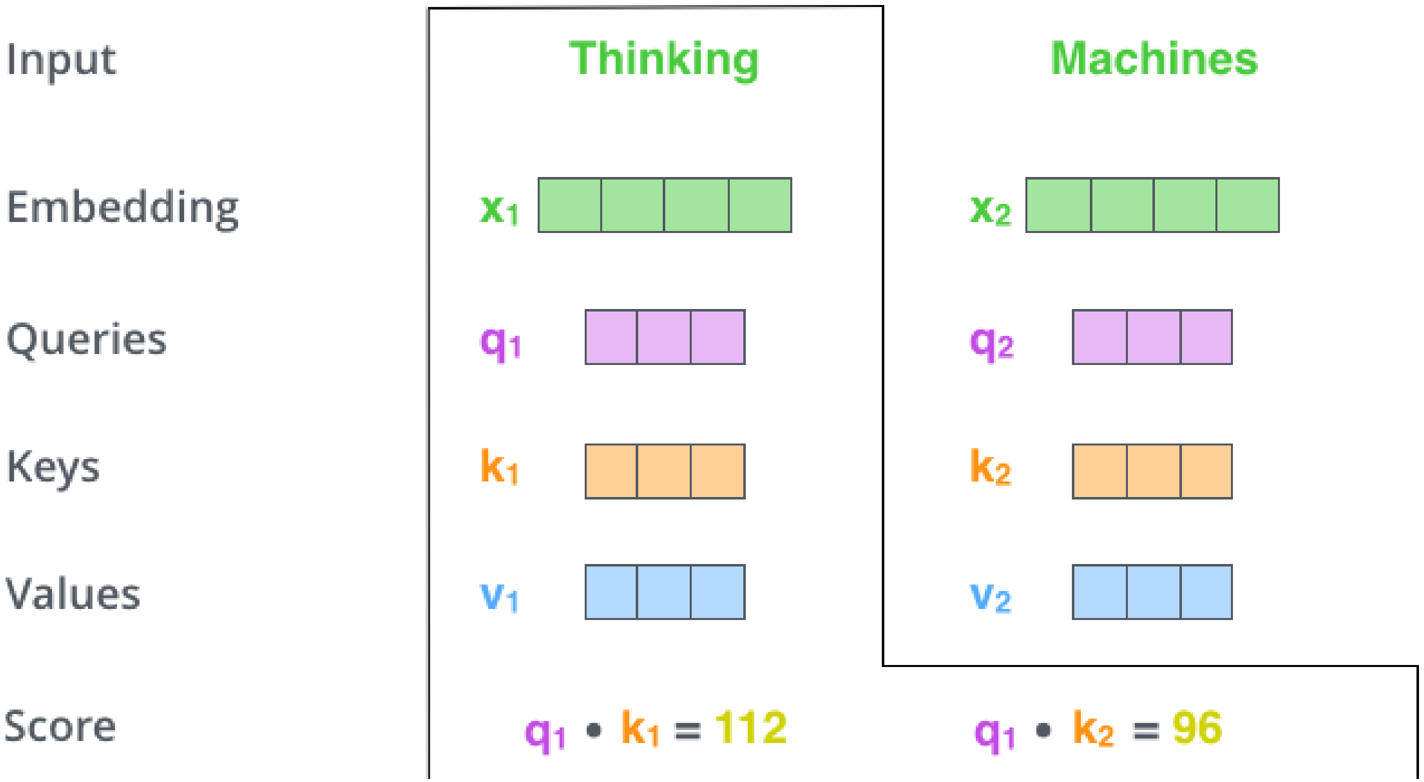




A Transformer Model



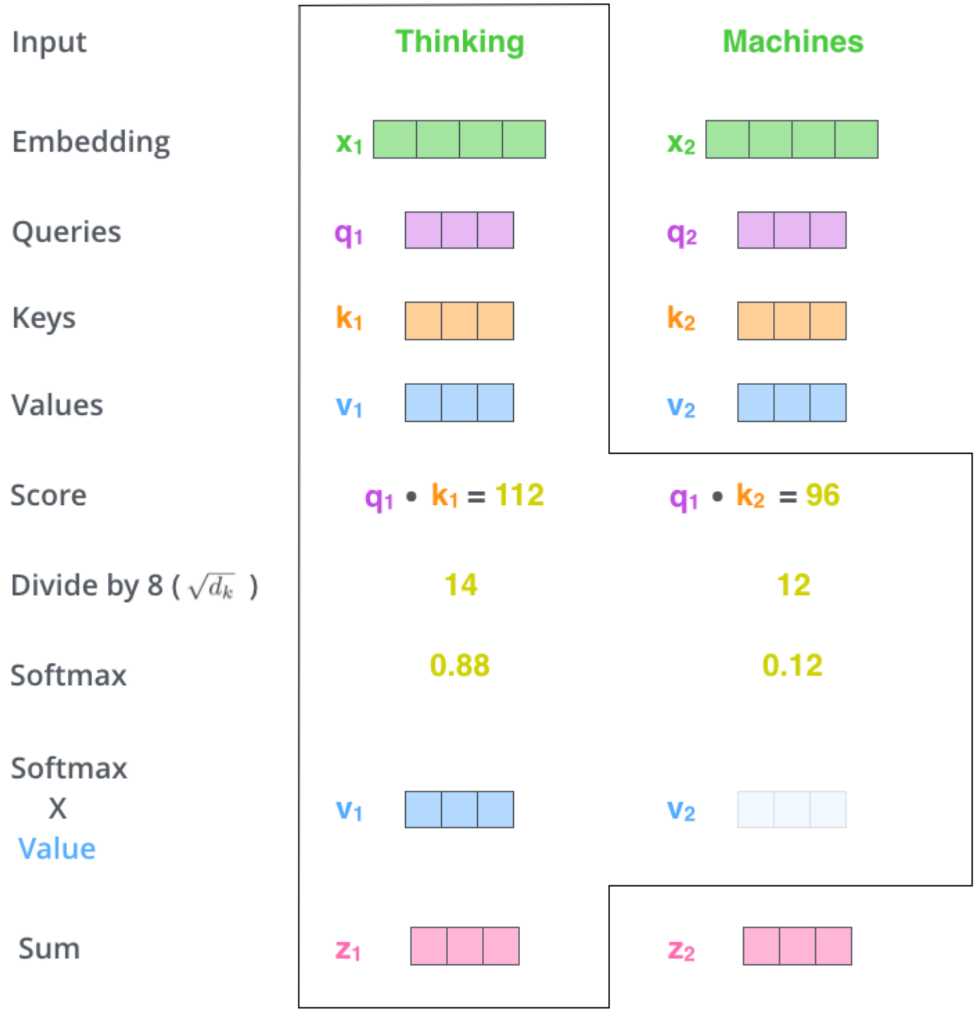
A Transformer Model

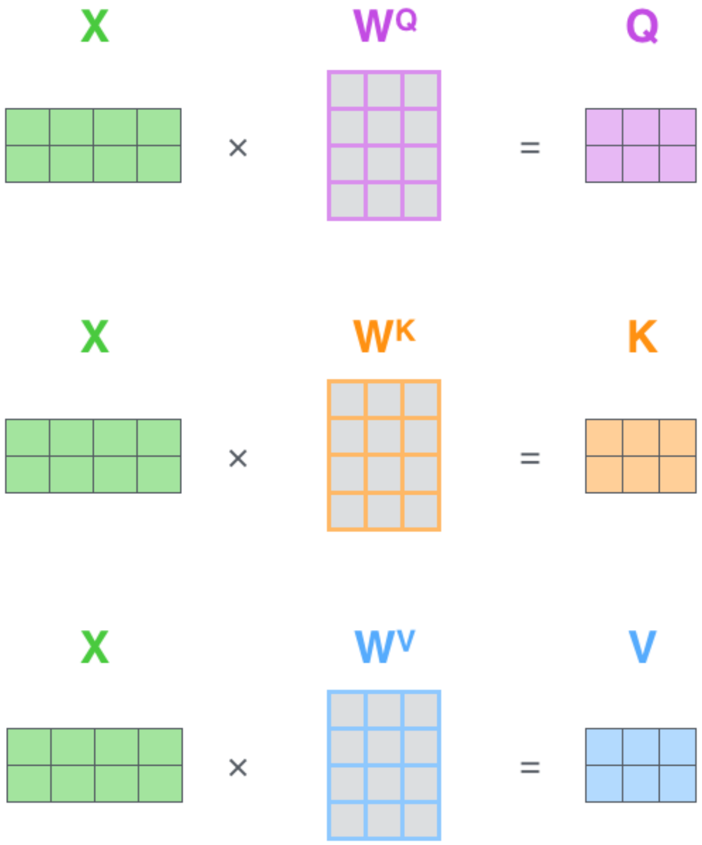


A Transformer Model

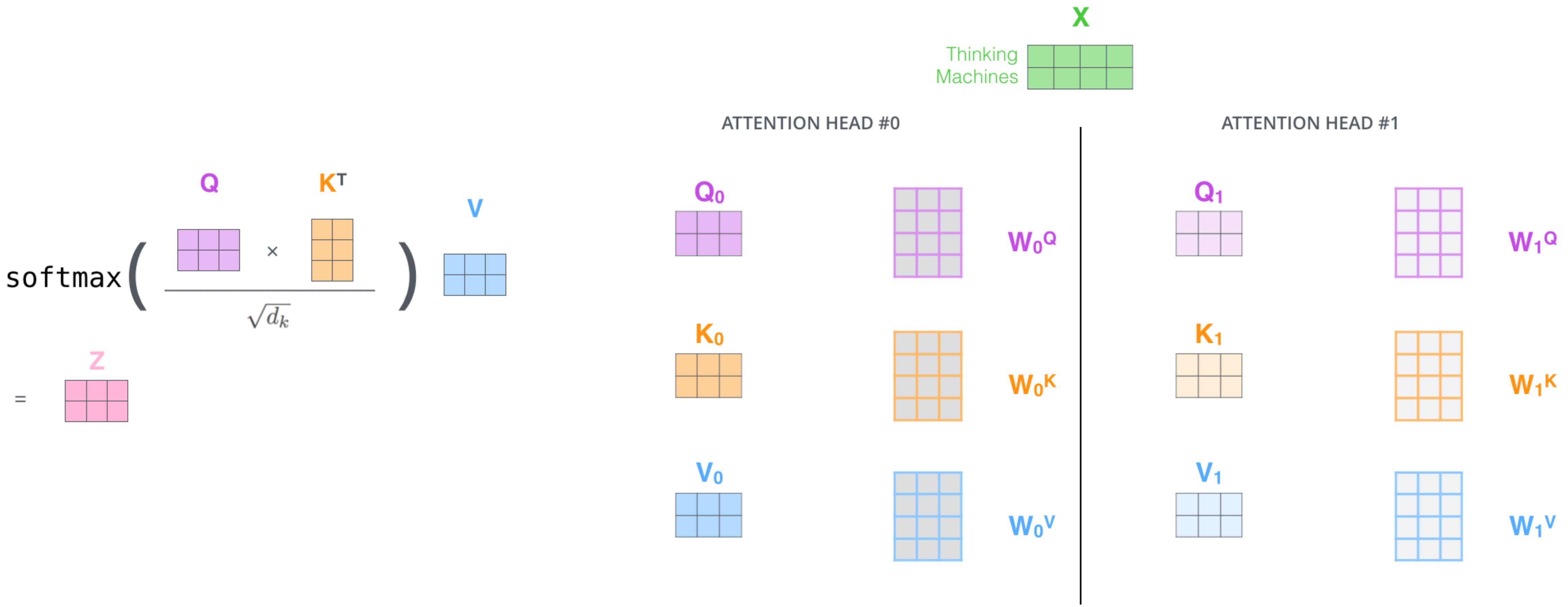
Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12

A Transformer Model

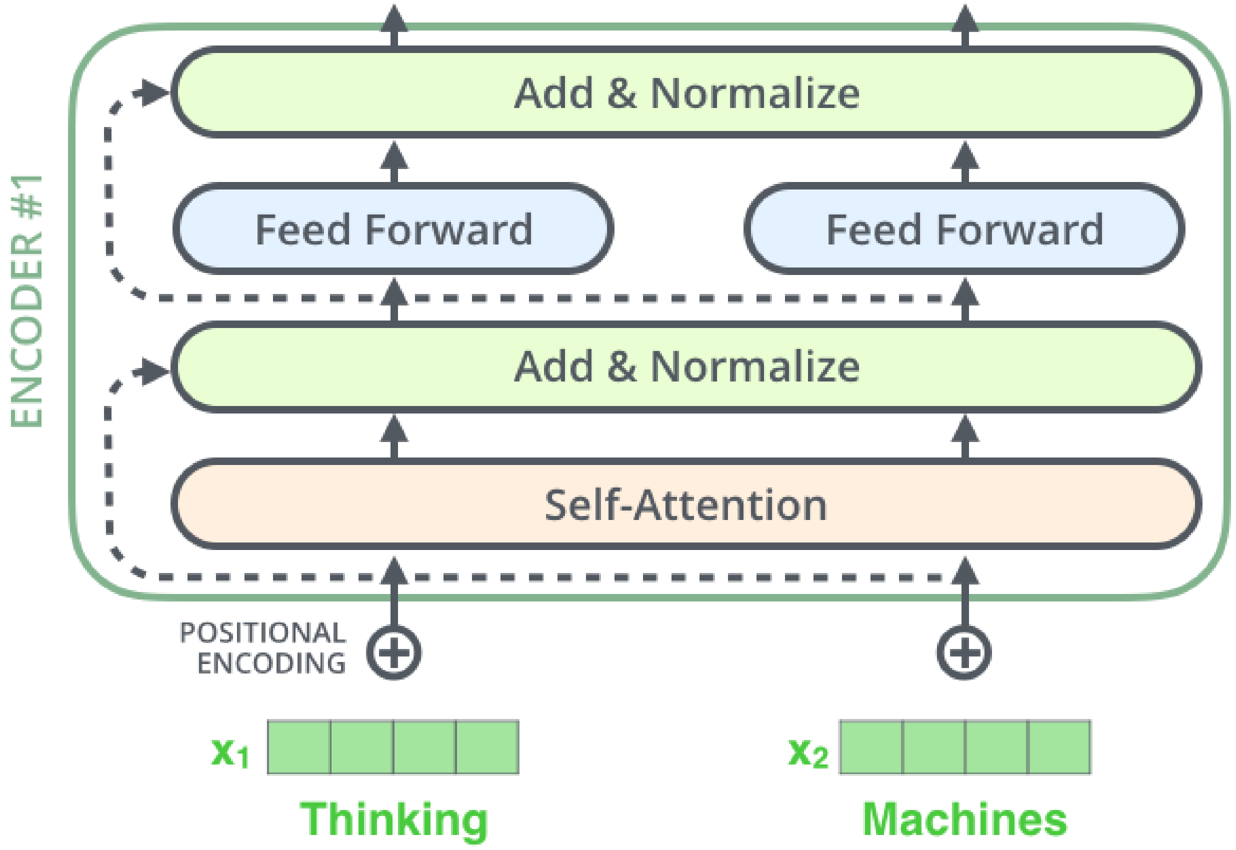


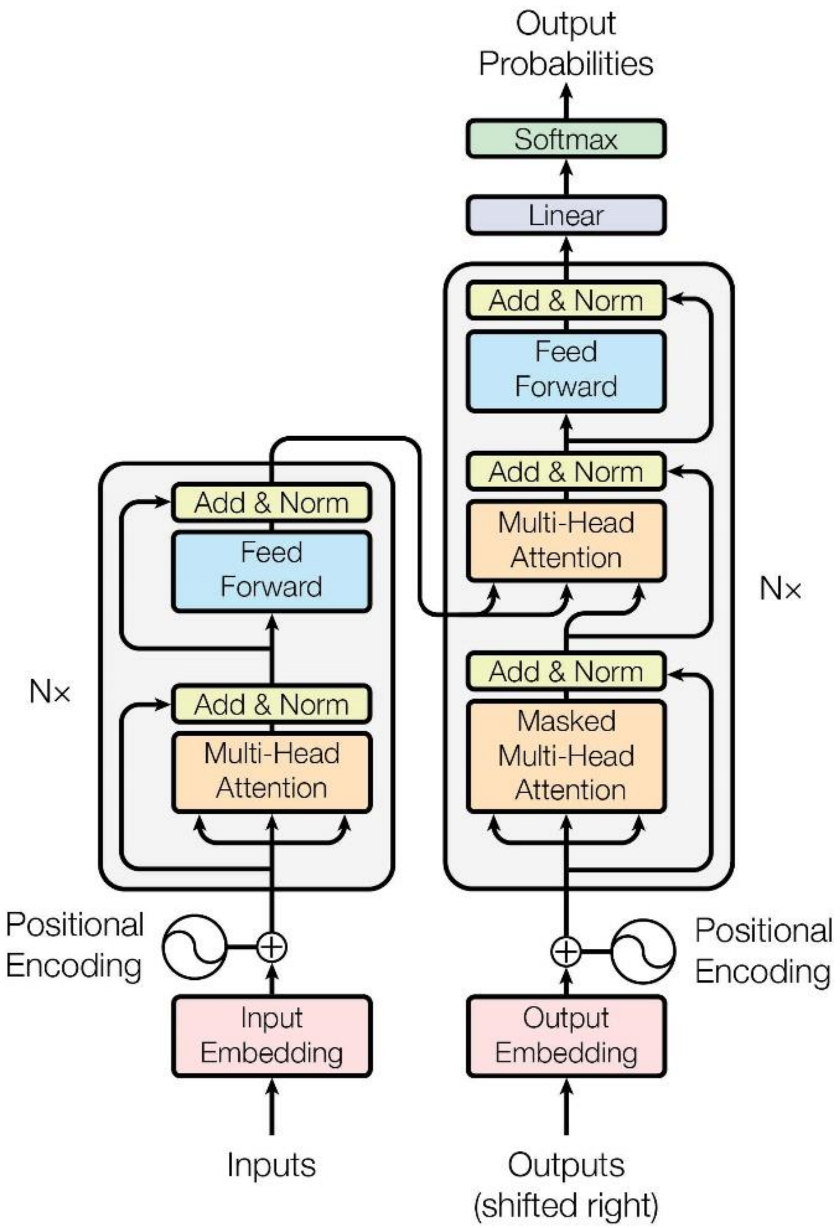


A Transformer Model

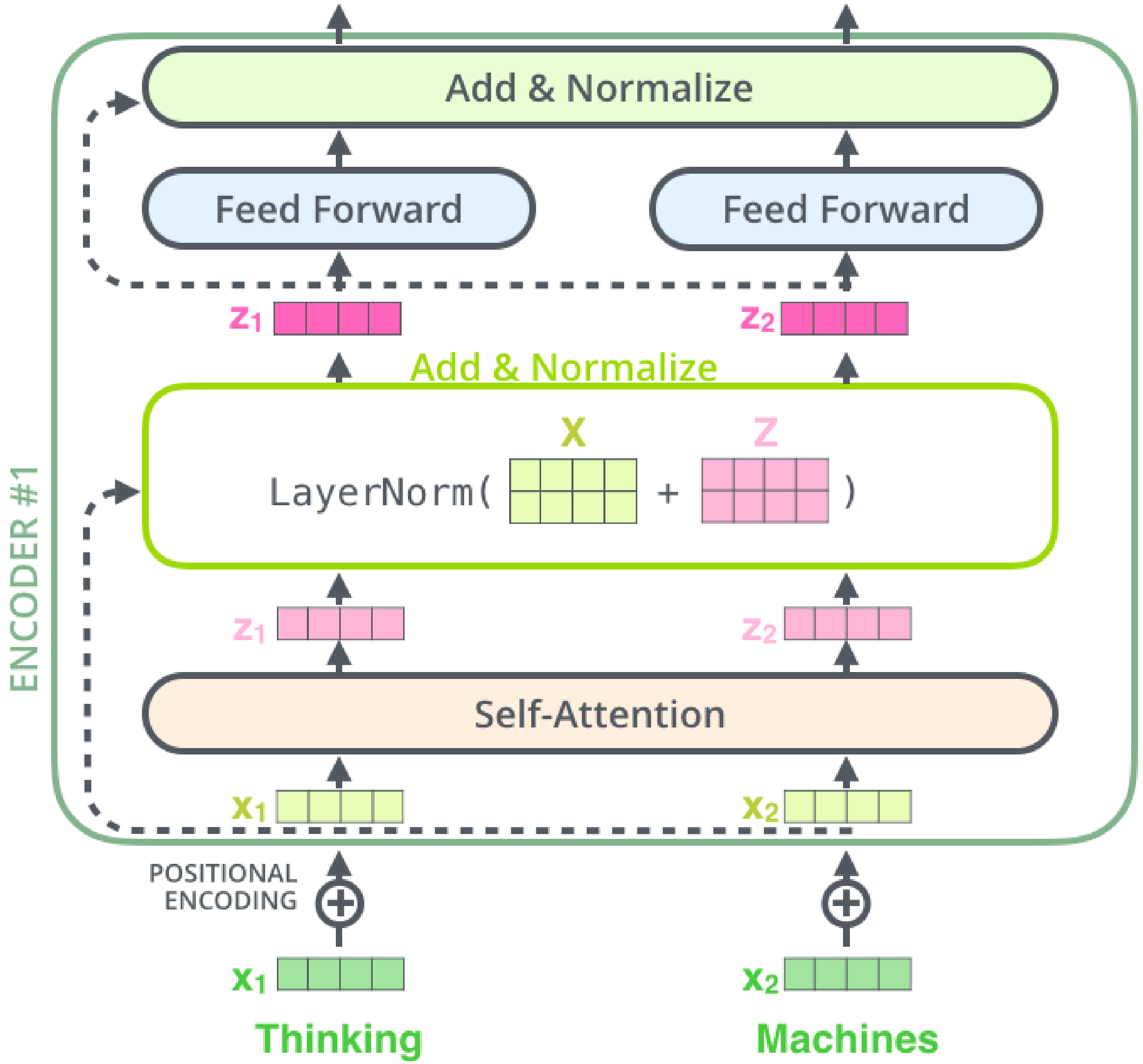


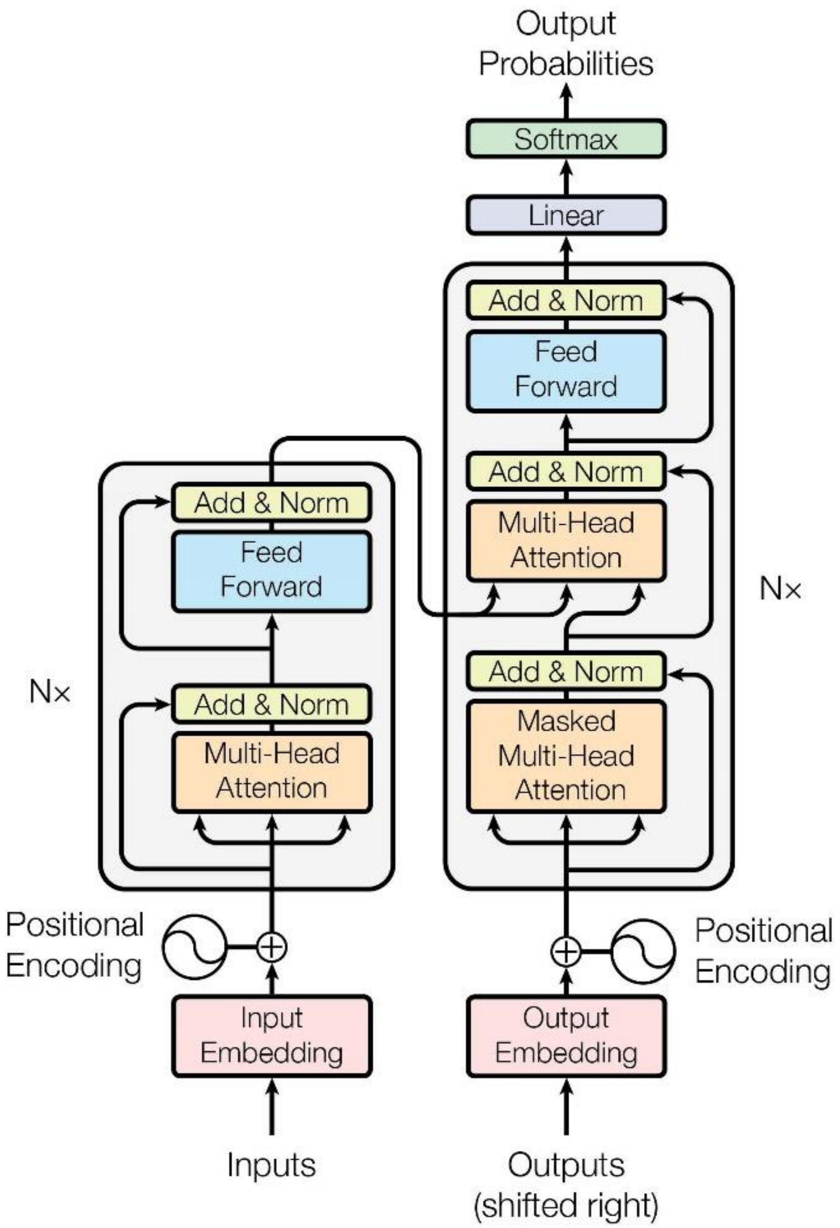
A Transformer Model





A Transformer Model





A Transformer Model