ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

word2vec

https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/30/word2vec/
https://mc.ai/deep-nlp-word-vectors-with-word2vec/
https://wikidocs.net/22660

Bert

http://jalammar.github.io/illustrated-transformer/
https://medium.com/dissecting-bert/dissecting-bert-part-1-d3c3d495cdb3

1. BackGround(NLP, Text Mining)

1. Text embedding(encoding, representation vector)
    a. How Text can be represented to vector(number) for feeding to model

2. Text tokenization
    a. rule 기반(ex konlp기반 형태소 분석기)
    b. wordpiece 기반(ex sentence2vec)
    c. 나는 오늘 식당에 가서 밥을 먹었다 -> 나 는 오늘 식당 에 가 서 밥 을 먹었 다

3. one hot vector(sparse vector)
    a. 나는 오늘 식당에서 밥을 먹었다. -> 1 2 3 4 5
    그리고 나는 식당에서 책도 읽었다. -> 6 1 3 7 8

1. BackGround(NLP, Text Mining)

1. Bag of words
   a. 나는(1) 오늘(2) 식당에서(3) 밥을(4) 먹었다(5) 그리고(6) 책도(7) 읽었다(8)
   b. 1 1 1 1 1 0 0 0
   c. 1 0 1 0 0 1 1 1
2. tfidf
   a. 단어별 중요도에 따른 가중치 추가
   b. 0.5 0.3 1.5 1.2 1 0 0 0
3.

1. BackGround(NLP, Text Mining)

신경망을 이용한 text embedding (word2vec, fasttext, glove (using NN)

중심 단어

주변 단어

The fat cat sat on the mat

The fat cat sat on the mat

The fat cat sat on the mat

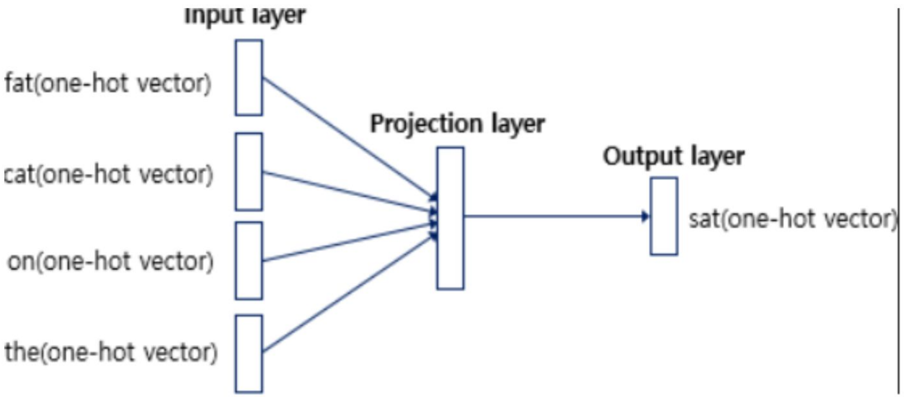
The fat cat sat on the mat

The fat cat sat on the mat

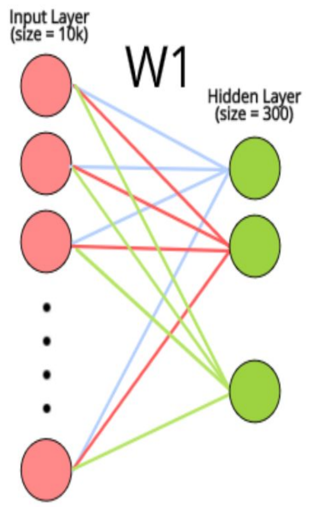
The fat cat sat on the mat

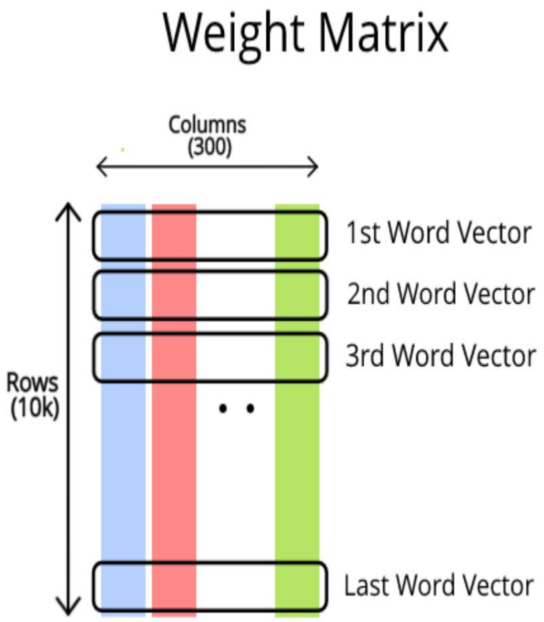
The fat cat sat on the mat

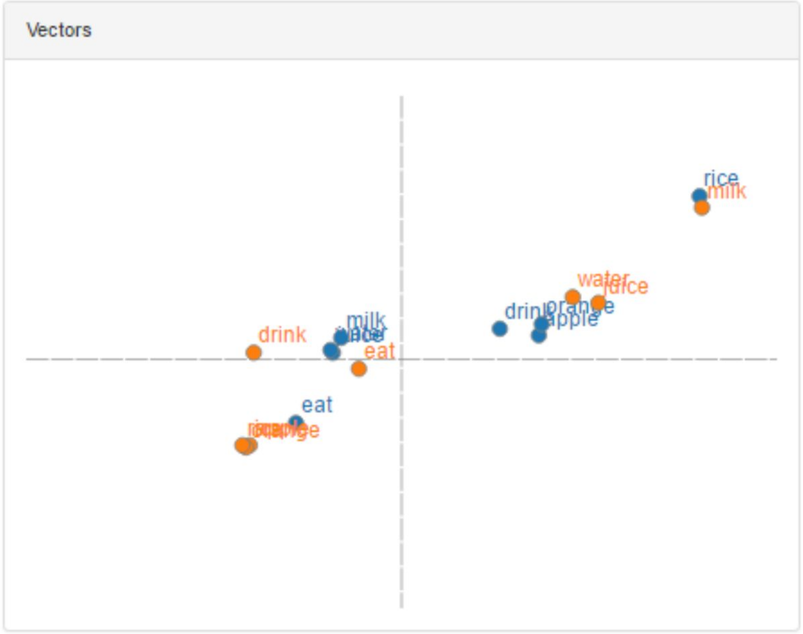
중심 단어	주변 단어
[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 1, 0, 0]	[0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0]



1. BackGround(NLP, Text Mining)



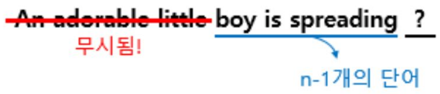




1. BackGround(NLP, Text Mining)

1. 언어 모델링
   a. 이전 단어들이 주어졌을 때 다음 단어를 예측
   b. 통계를 이용한 방법과 인공 신경망을 이용한 방법
   c. Ngram vs 신경망(LSTM)

n-gram을 통한 언어 모델에서는 다음에 나올 단어의 예측은 오직 n-1개의 단어에만 의존합니다. 예를 들어 **'An adorable little boy is spreading'** 다음에 나올 단어를 예측하고 싶다고 할 때, n=4라고 한 4-gram을 이용한 언어 모델을 사용한다고 합시다. 이 경우, spreading 다음에 올 단어를 예측하는 것은 n-1에 해당되는 앞의 3개의 단어만을 고려합니다.



$$P(w|\text{boy is spreading}) = \frac{\text{count}(\text{boy is spreading } w)}{\text{count}(\text{boy is spreading})}$$

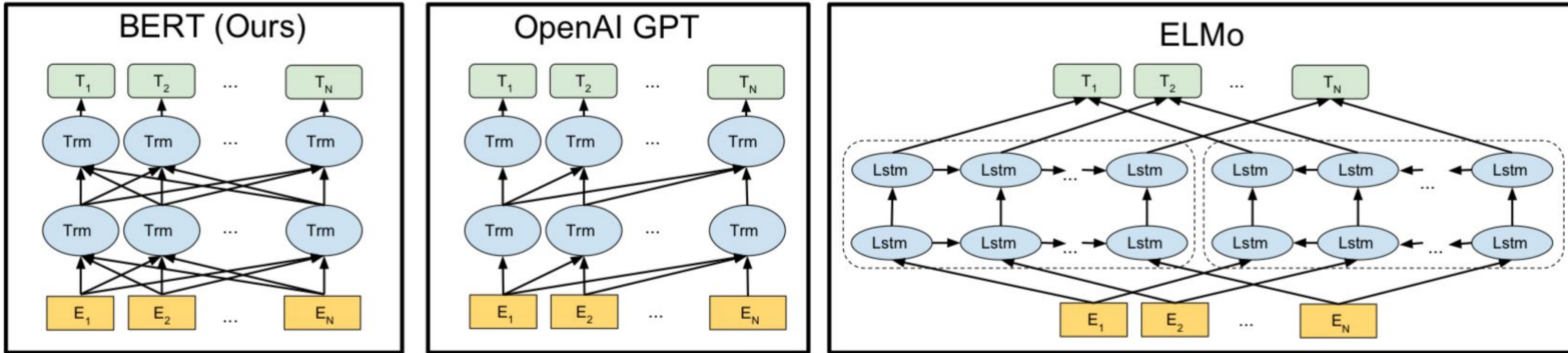
만약 갖고있는 코퍼스에서 boy is spreading가 1,000번 등장했다고 합시다. 그리고 boy is spreading insults가 500번 등장했으며, boy is spreading smiles가 200번 등장했다고 합시다. 그렇게 되면 boy is spreading 다음에 insults가 등장할 확률은 50%이며, smiles가 등장할 확률은 20%입니다. 확률적 선택에 따라 우리는 insults가 더 맞다고 판단하게 됩니다.

$$P(\text{insults}|\text{boy is spreading}) = 0.500$$

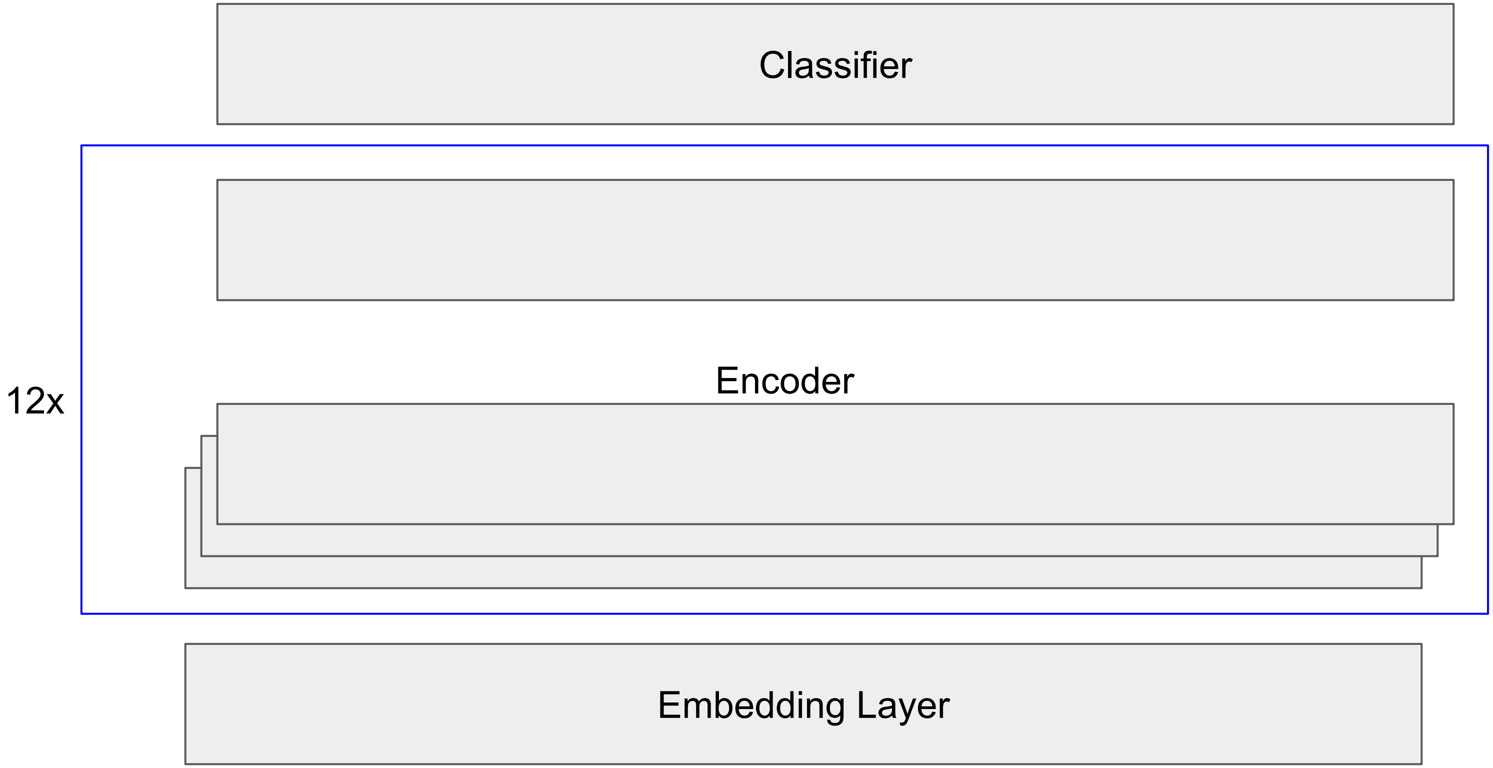
$$P(\text{smiles}|\text{boy is spreading}) = 0.200$$

1. BackGround(NLP, Text Mining)

1. 사전학습 (word2vec etc)
    a. Use more Deep NN -> ELMO, OPEN-GPT, BERT
    b. unidirectional vs bidirectional (Open GPT vs ELMO, BERT)
    c. feature based vs fine tuning (ELMO vs OPEN GPT, BERT)
2. ALBERT
    a. BERT is very Good, but too much computing resource, long training time
    b. minimized the parameter with maintaining performance

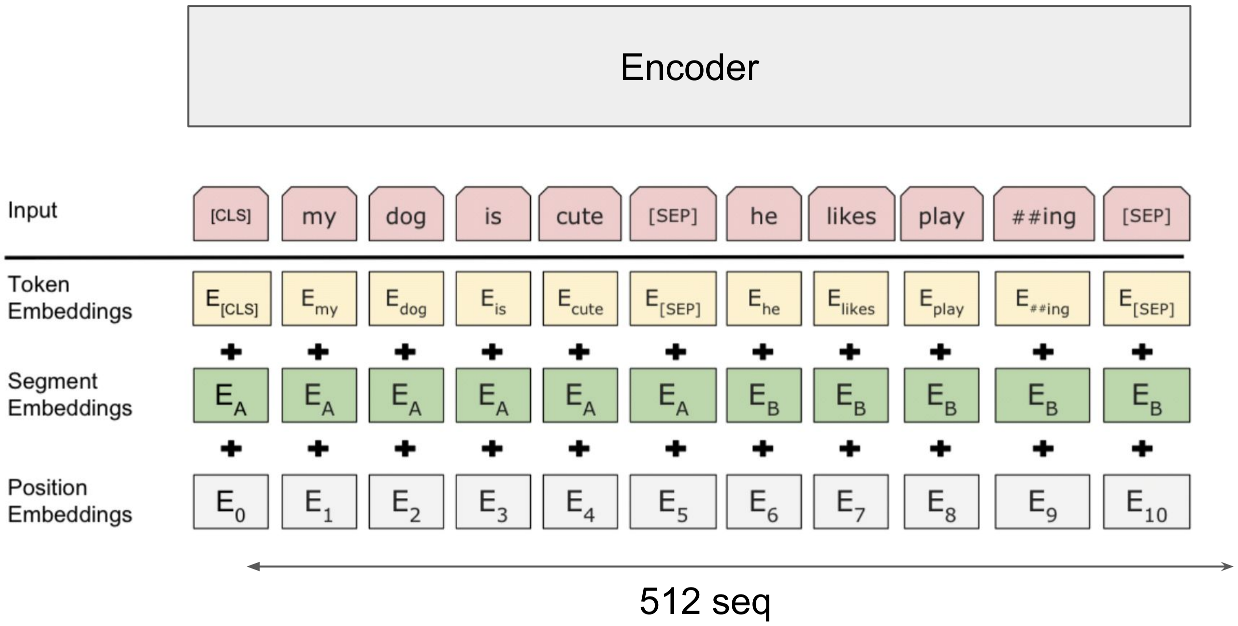


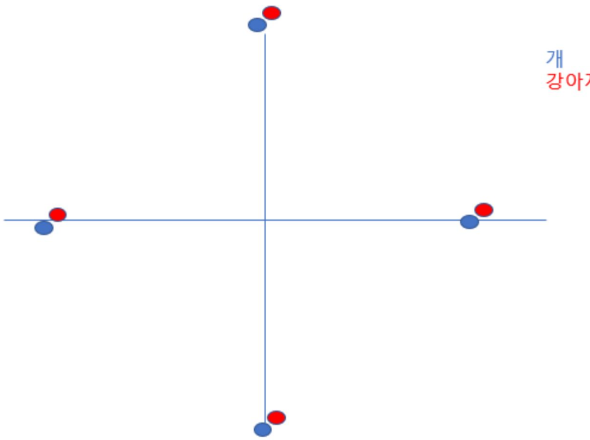
1. BERT(Bidirectianal Encoder representation of Transformer)



Embedding, positional encoding

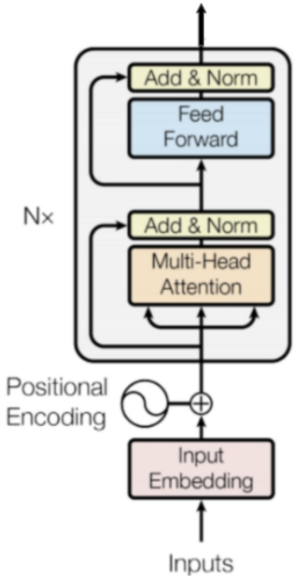
give offset according to words position

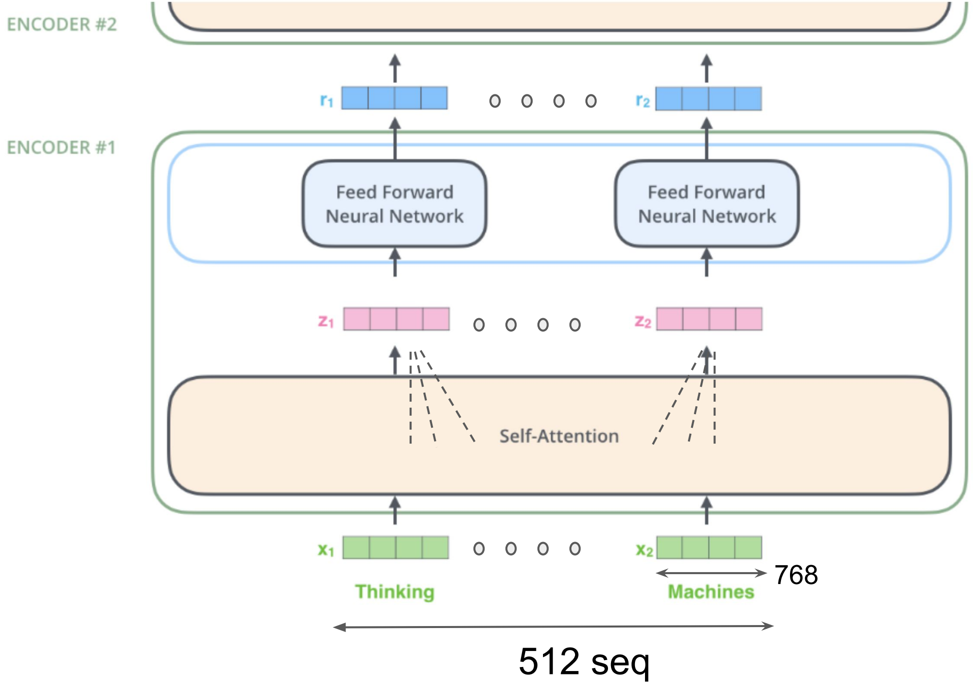


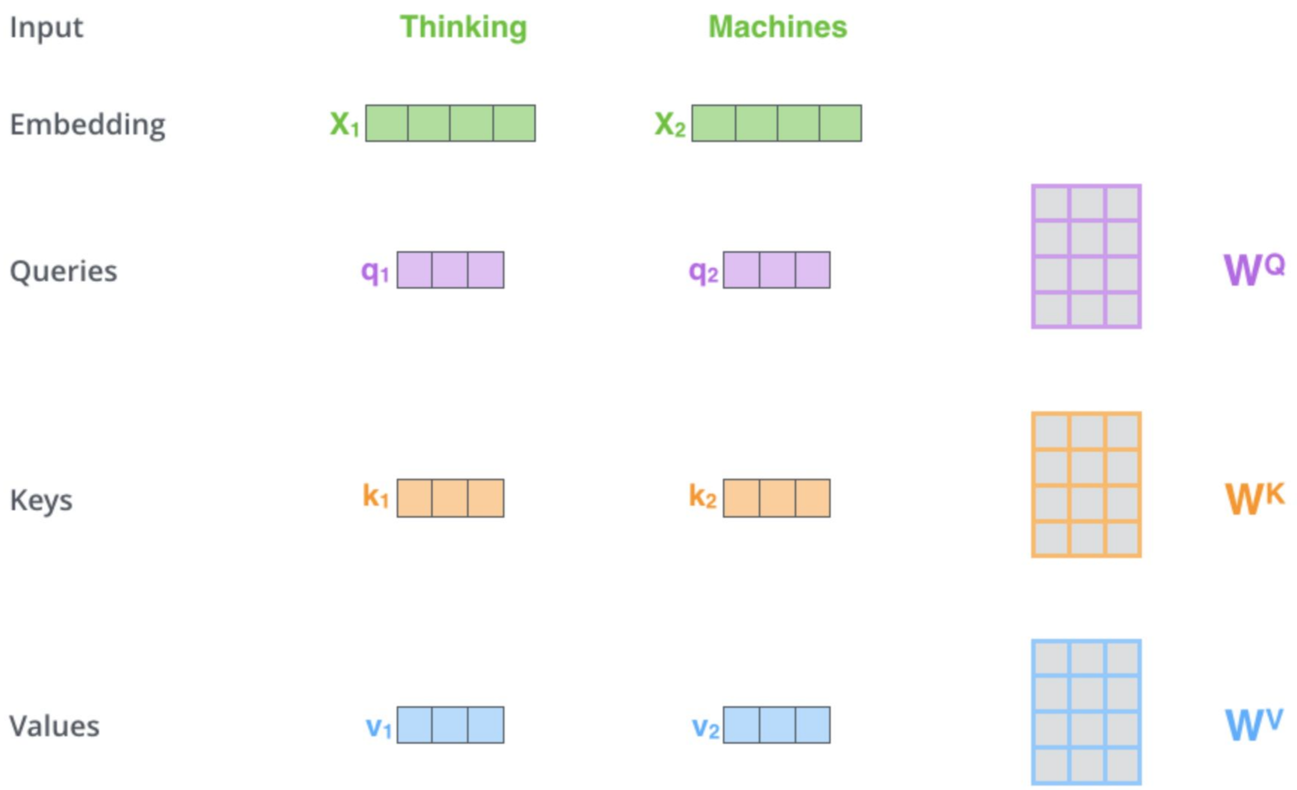


1. BERT(Bidirectianal Encoder representation of Transformer)

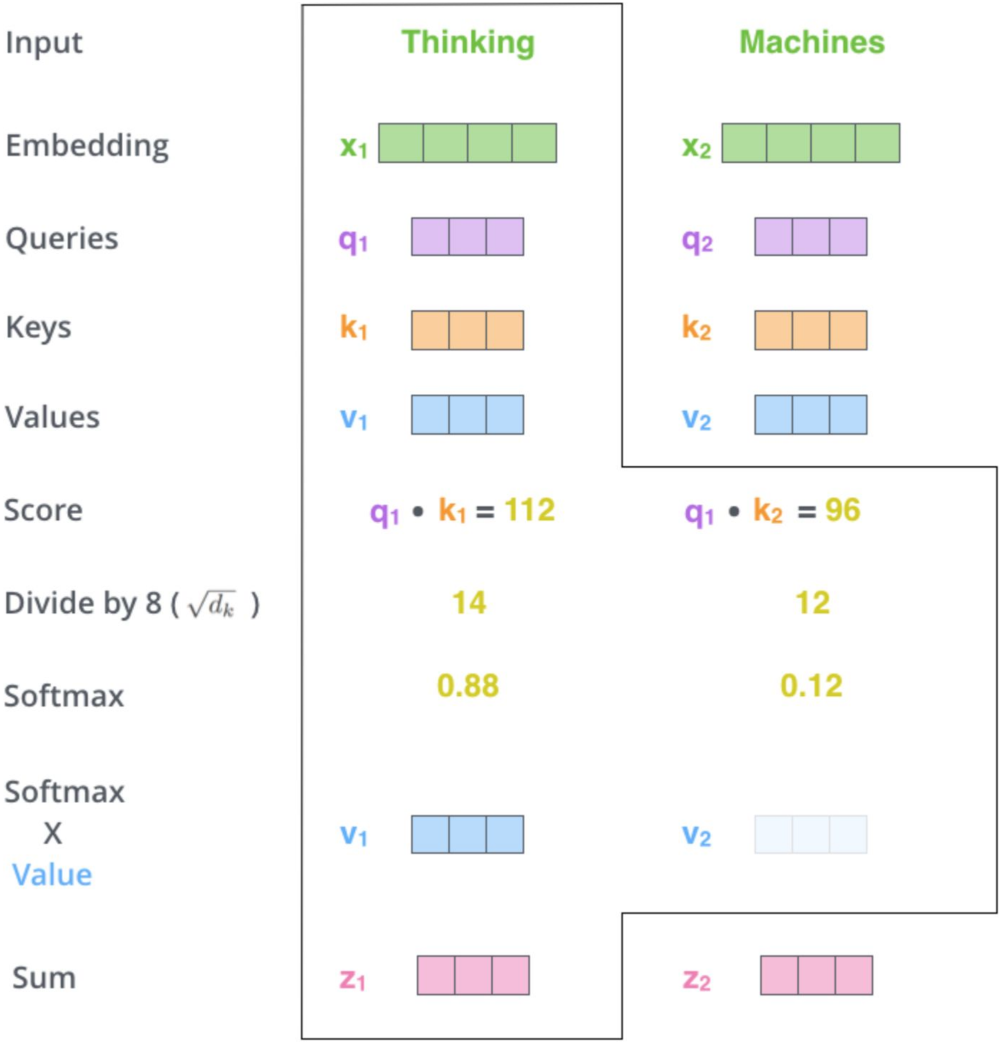
It simiar to FCN (not scalar but vector, how to calculate similarity ? use dot product





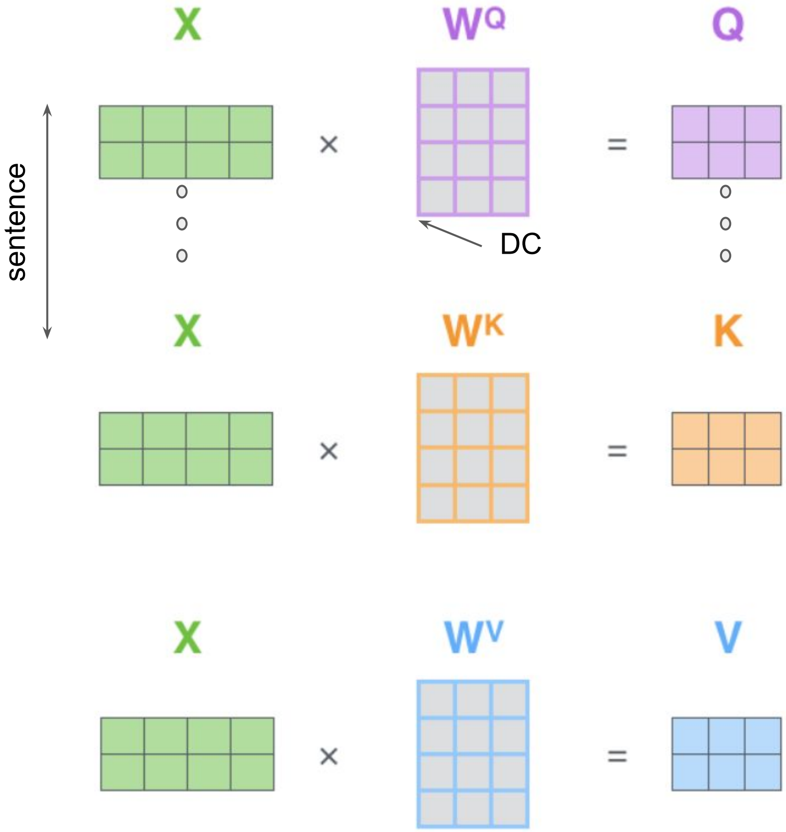


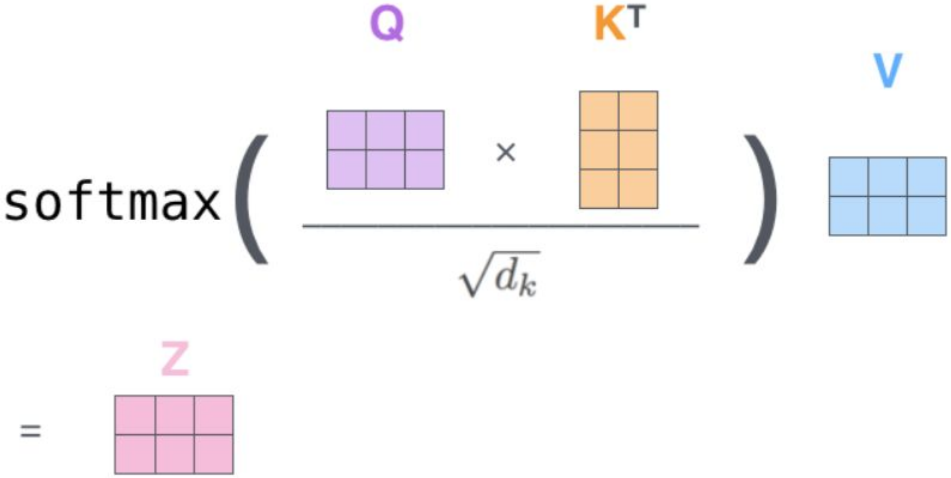
Input
Thinking
Machines
Embedding
X1
X2
Queries
q1
q2
WQ
Keys
k1
k2
WK
Values
v1
v2
WV



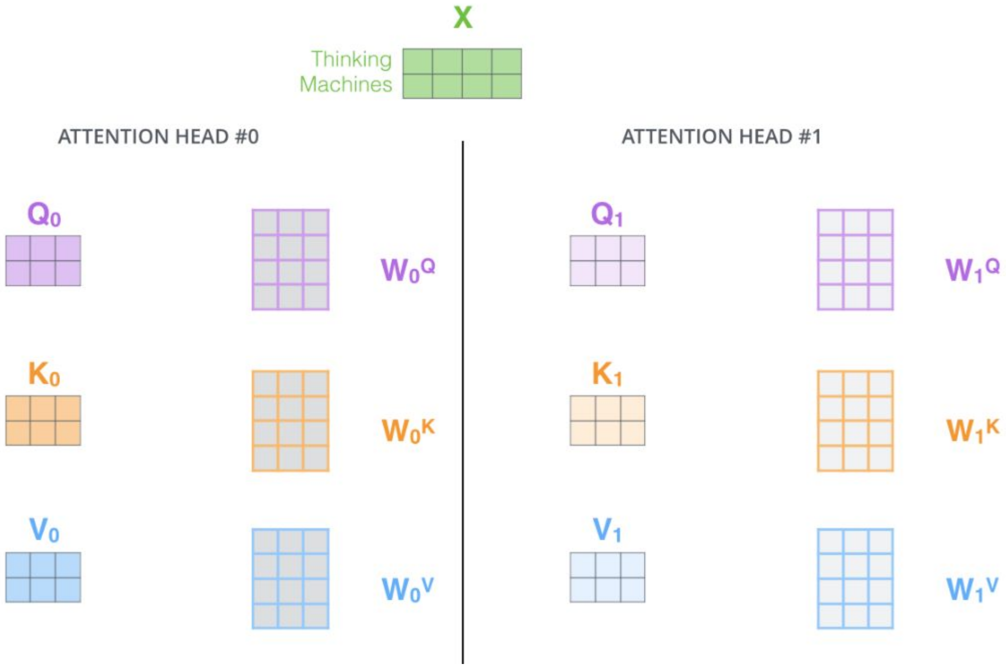
how about offset or activation ft?

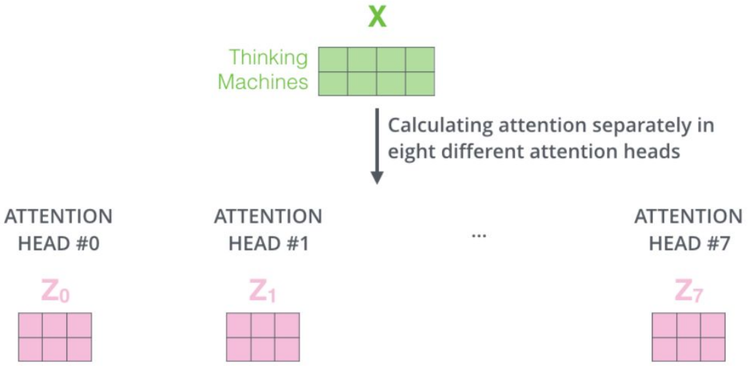
similar to convolution in image
1d convolution





multi head attention
768 -> 68 * 8
embedding vector : 768
attention vector : 68
for computing resource





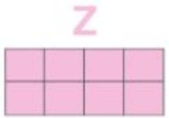
1) Concatenate all the attention heads

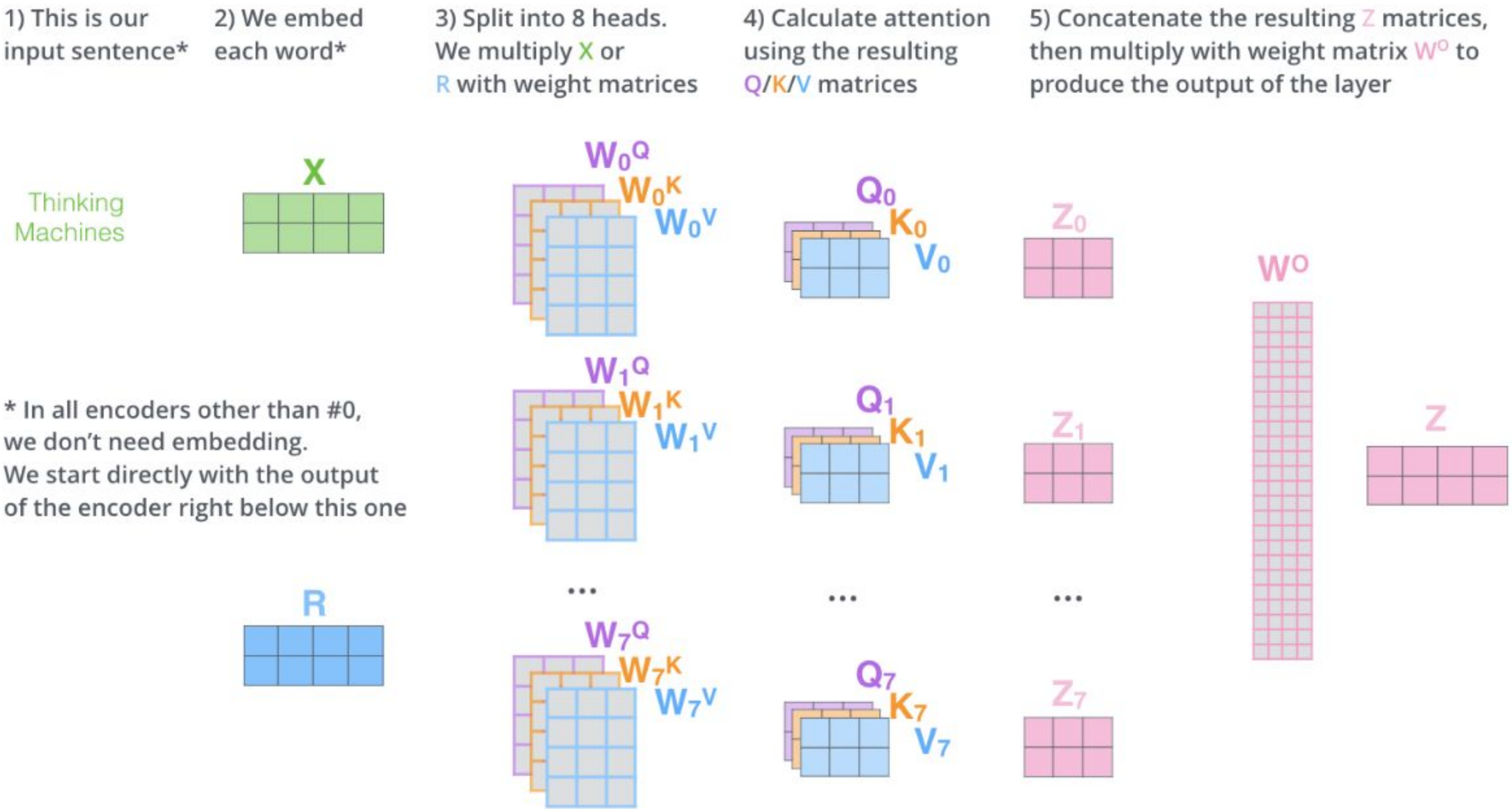
2) Multiply with a weight matrix W^O that was trained jointly with the model

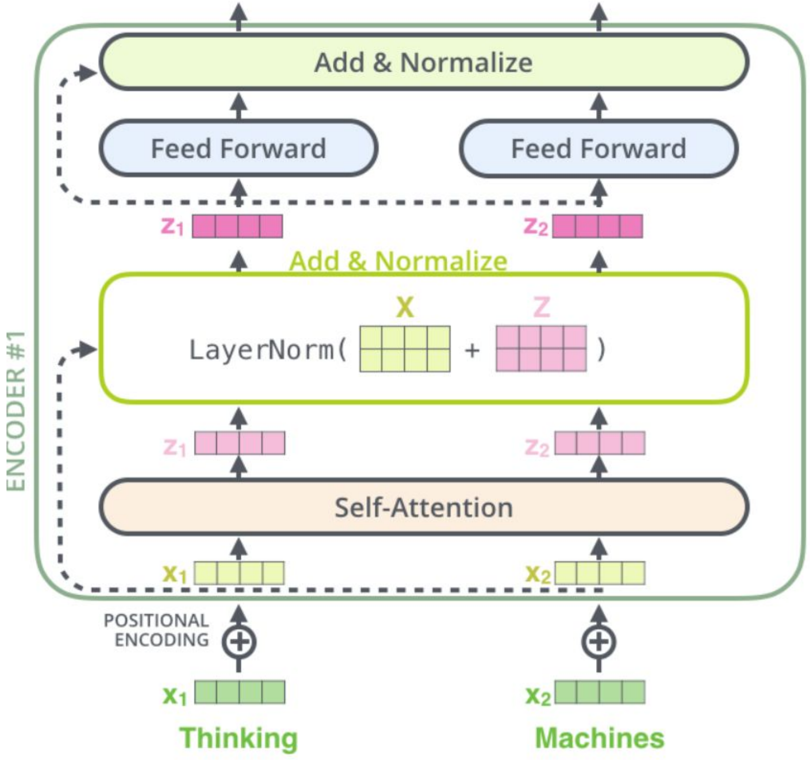
X

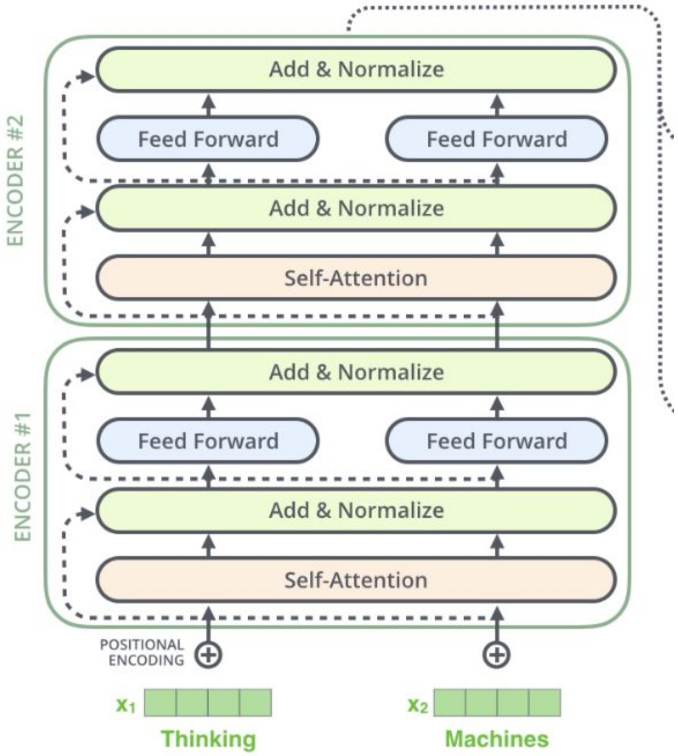
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



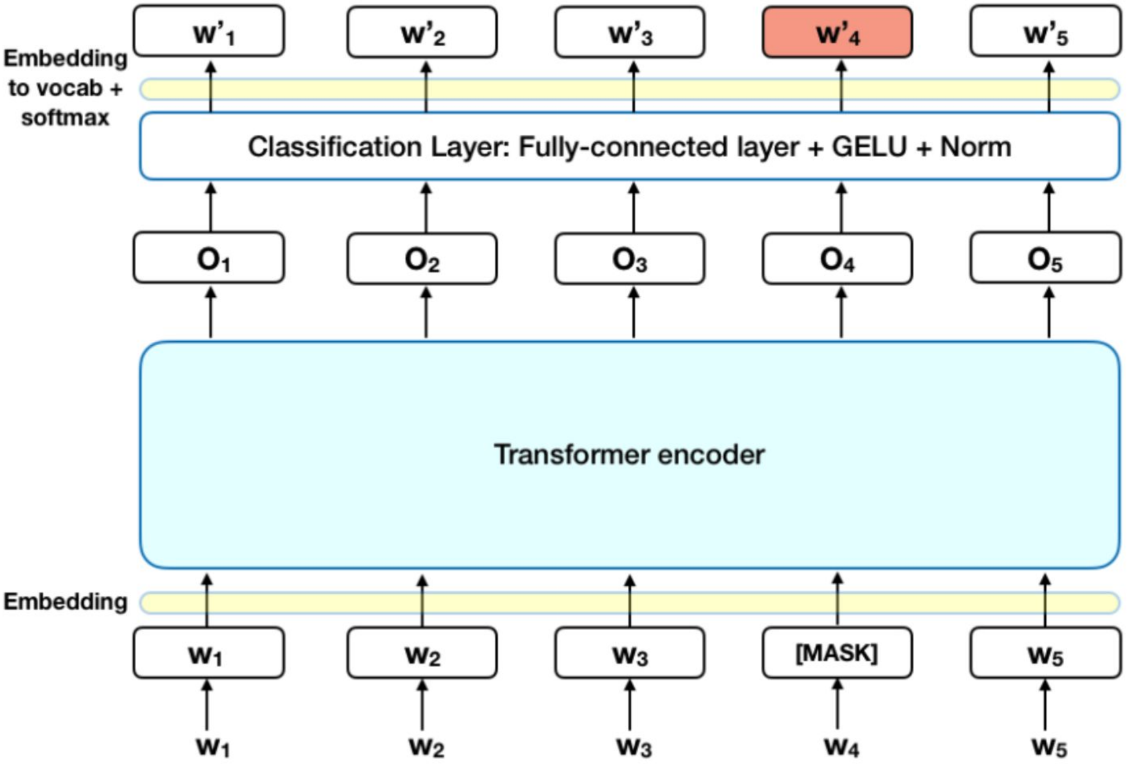

1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W_0^Q W_0^K W_0^V
W_1^Q W_1^K W_1^V
...
W_7^Q W_7^K W_7^V
Q_0 K_0 V_0
Q_1 K_1 V_1
...
Q_7 K_7 V_7
Z_0
Z_1
...
Z_7
W^O
Z


ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
z_1
z_2
Add & Normalize
X
Z
LayerNorm(+)
z_1
z_2
Self-Attention
x_1
x_2
POSITIONAL ENCODING
x_1
x_2
Thinking
Machines


ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x_1
x_2
Thinking
Machines

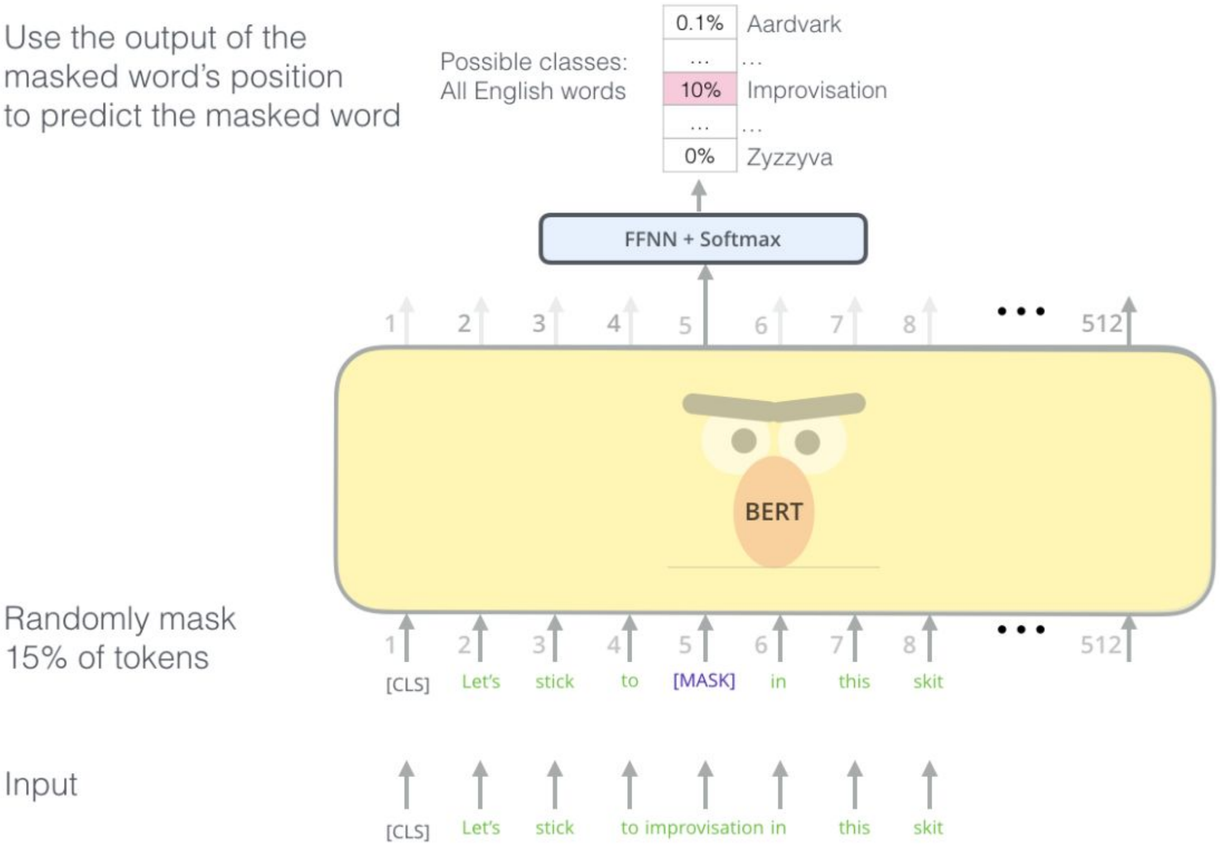
How to train bert?

1. MLM(masked Language model)

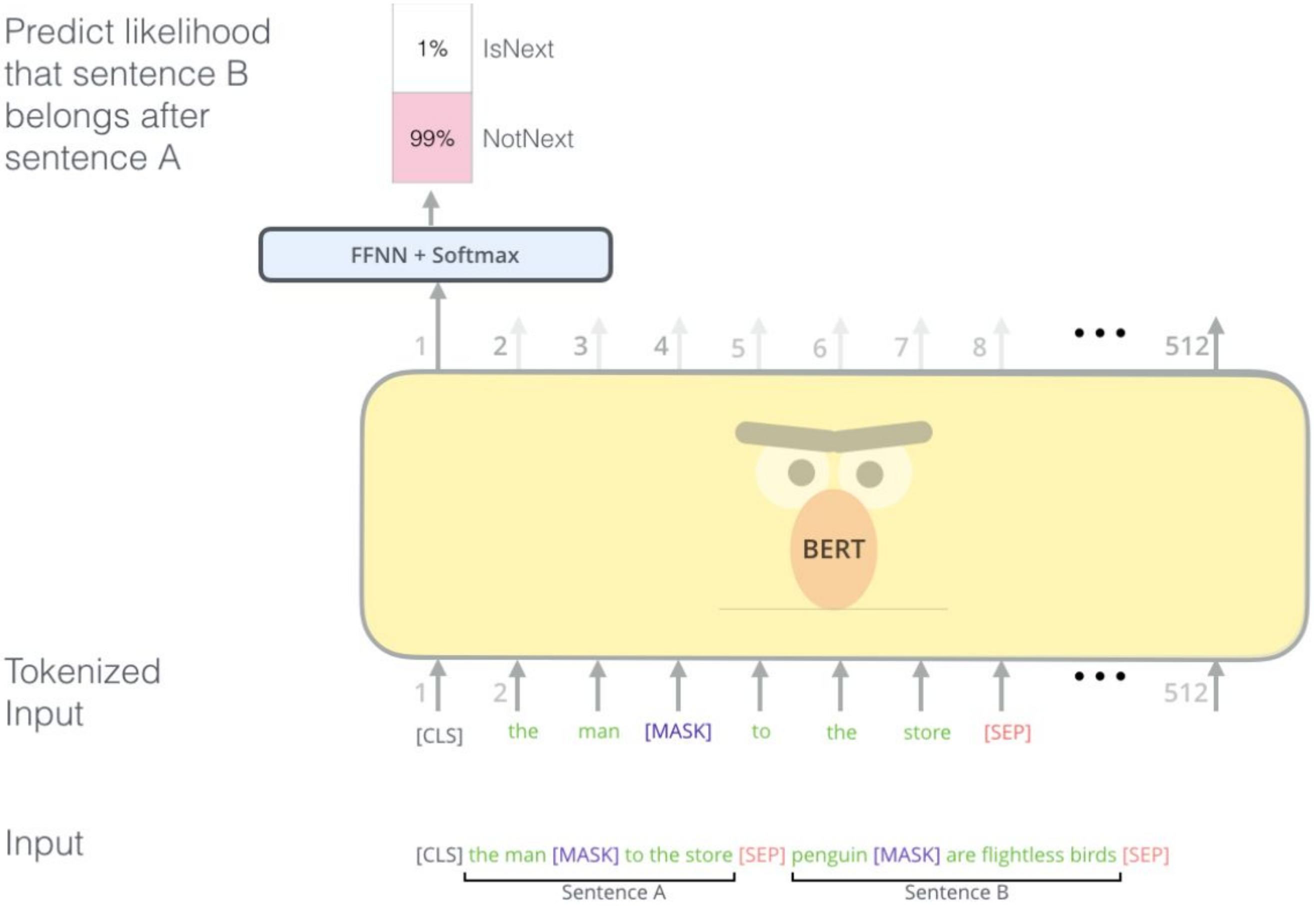


making rate : 15%
making : 80, wrong, 10, correct 10

Masked language model



NSP(next sentence prediction)



ALBERT

Factorized embedding parameterization

Parameter sharing

NSO

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

ALBERT

Factorized embedding parameterization

bert : embeding == hidden size(768)

albert : embedding =128, hidden size :768

1. wordpiece embedding meant to learn context independent representation, hidden size embedding meant to learn context dependent representation
2. Parameter decrease : 30000(vocab size) * 768 -> 30000*128 + 128 *768

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

ALBERT

CROSS-LAYER PARAMETER SHARING

group sharing(for Layer)

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

ALBERT

SENTENCE ORDER PREDICTION (SOP)

	Intrinsic Tasks			Downstream Tasks					
SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

ALBERT

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**