Progress of Pre-training in NLP: from word embedding to BERT

Jun Chen

2018/11/10

Table of contents

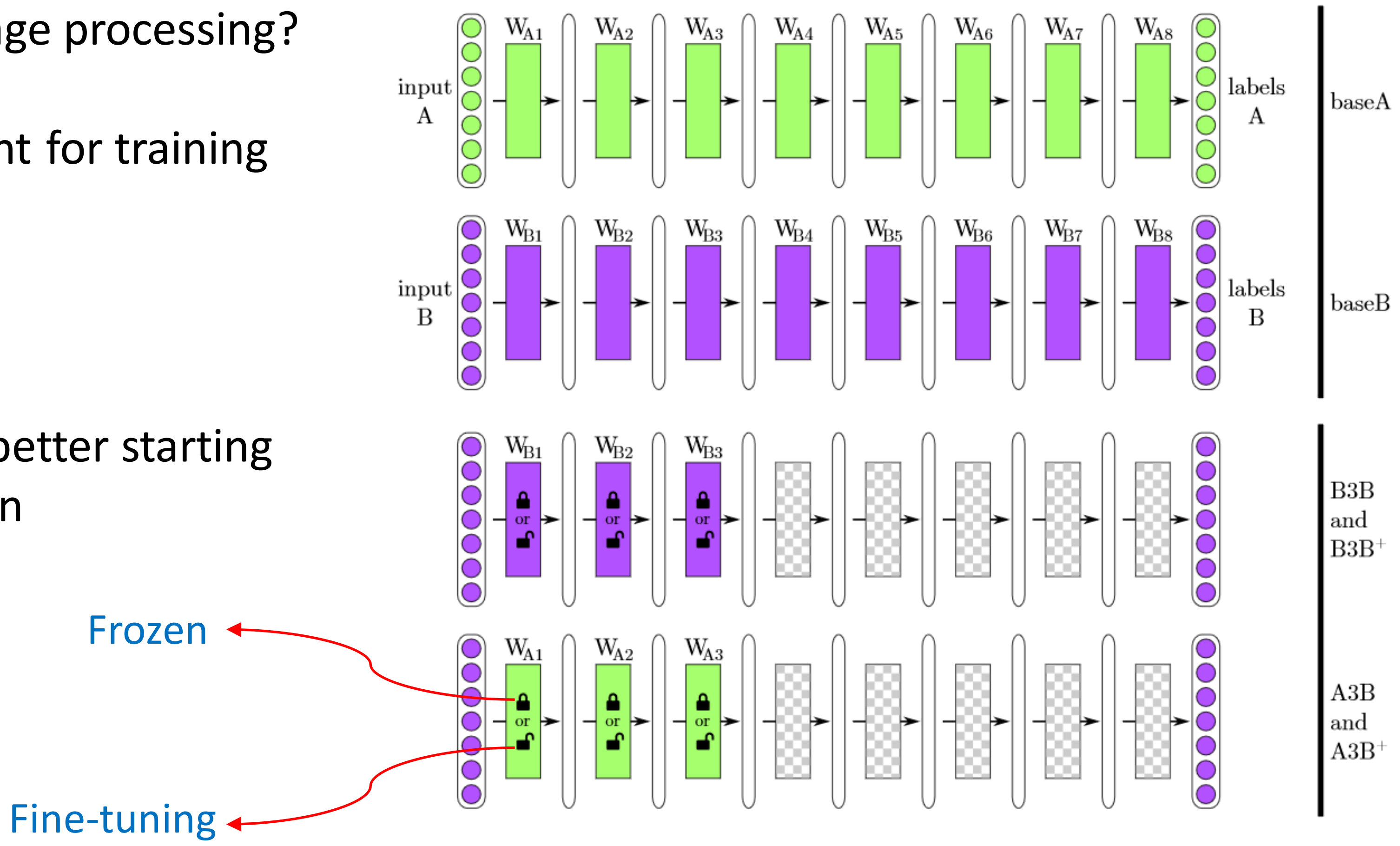
Pre-training in image processing

Why pre-training is popular in image processing?

1. Small training set is not sufficient for training complex neural network

2. Speed up the training process

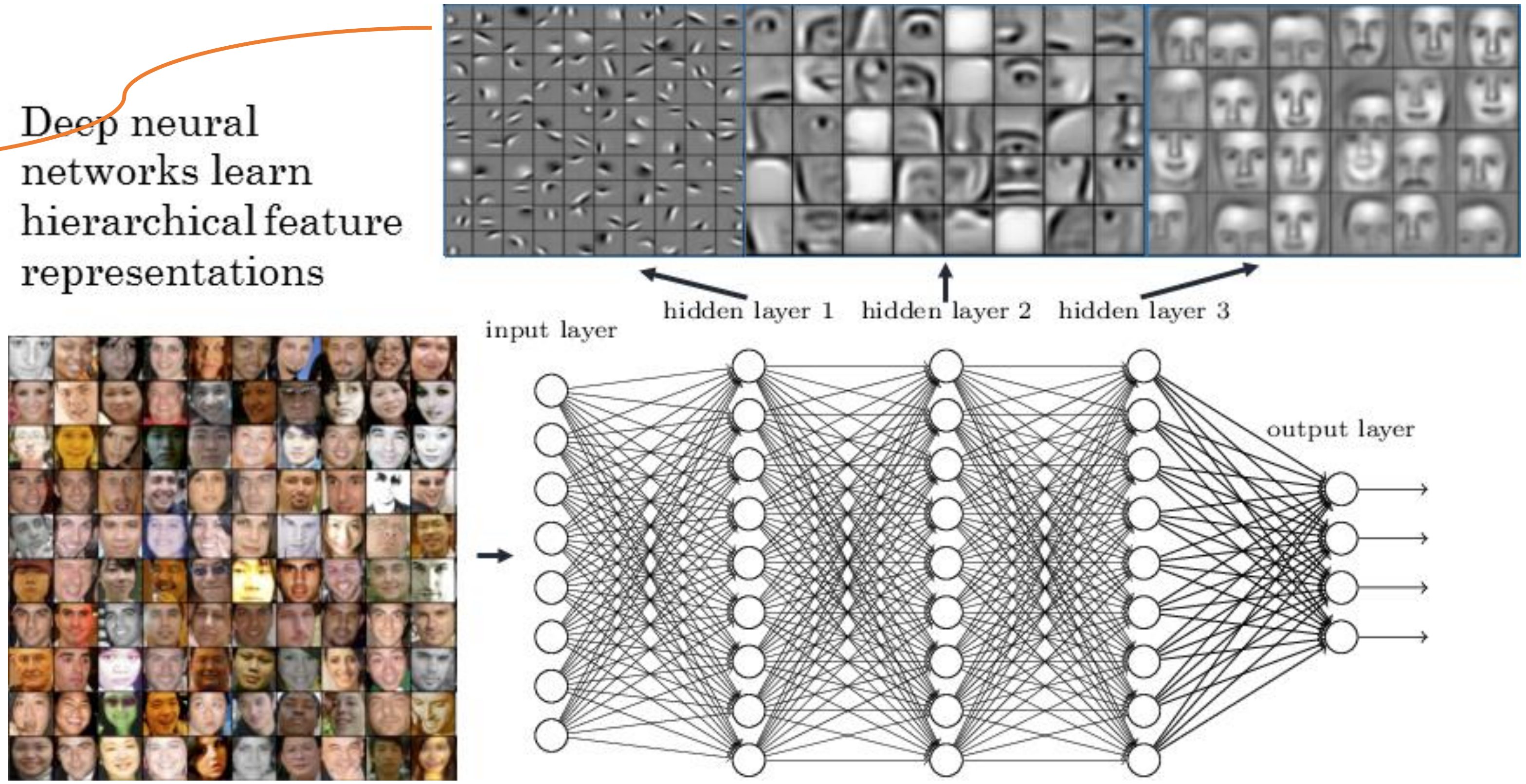
3. Parameter initialization: find a better starting point; be beneficial to optimization



Pre-training in image processing

Why freeze the weights?
First few layers capture universal features like curves and edges

Why fine-tuning?
Subsequent layers capture features more specific to task/dataset



Pre-training in image processing

Pre-trained Imagenet models have good generalization ability.

Can be used for:

Prediction(use pretrained model directly)

Feature Extraction(only replace and retrain the classifier on top of the ConvNet)

Fine-tuning(fine-tune all the layers of the ConvNet, or keep some of the earlier layers fixed and fine-tune some higher-level layers)

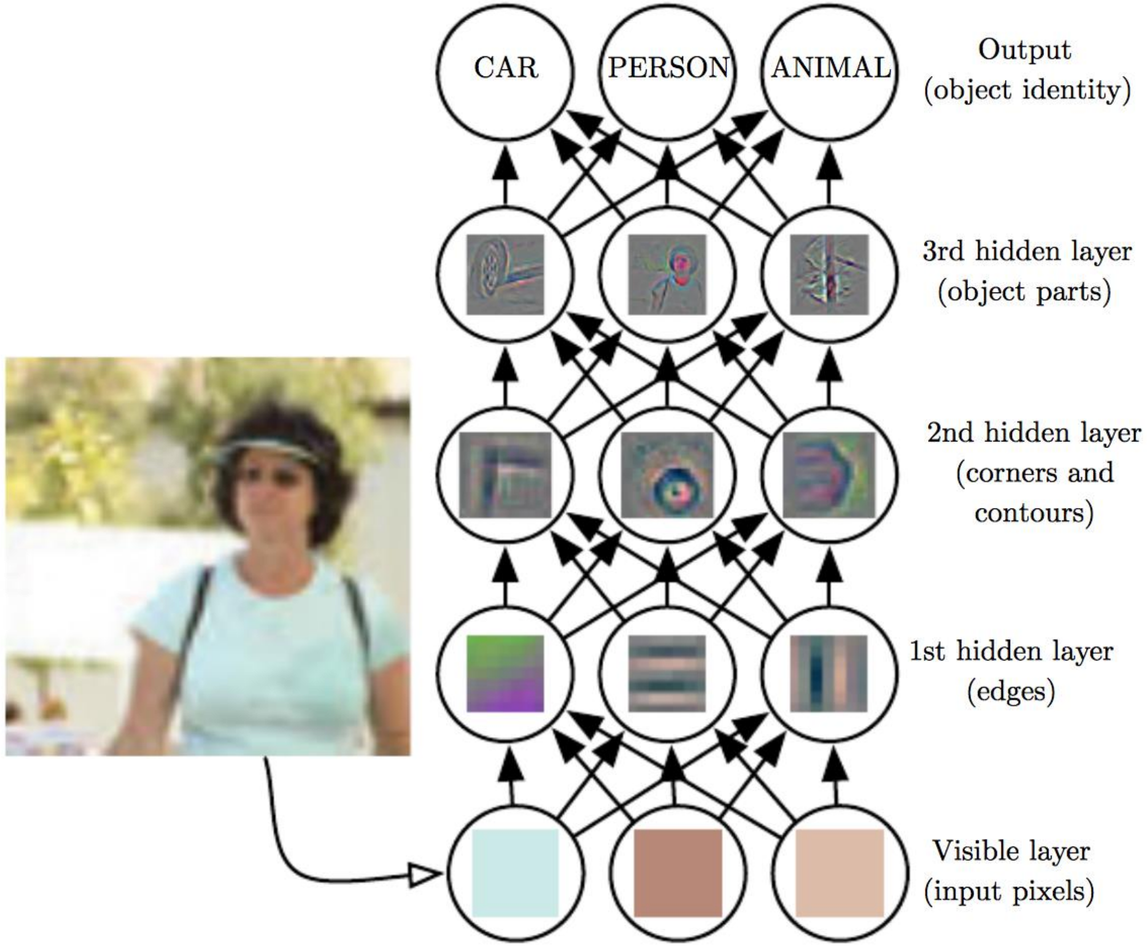


Table of contents

Language model

- The goal of language modelling is to estimate the probability distribution of various linguistic units, e.g., words, sentences etc

- Machine Translation:
  - P(**high** winds tonite) > P(**large** winds tonite)
- Spell Correction
  - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)

- Two categories: count-based and continuous-space LM.

Probabilistic language model(count-based)

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

- Related task: probability of an upcoming word:

  $P(w_5 | w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n | w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

Chain rule

$$P(S) = P(w_1, w_2, \ldots, w_n) \Rightarrow P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \cdots P(w_n|w_1, w_2, \cdots w_{n-1})$$

$$\Downarrow$$

$$L = \sum_{w \in C} logP(w|context(w))$$

Probabilistic language model(count-based)

Example: N-gram

- The LM probability $p(w_1,w_2,\ldots,w_n)$ is a product of word probabilities based on a history of m preceding words:

$$p(w_n|w_1, w_2, \cdots, w_{n-1}) \approx p(w_n|w_{n-m}, \cdots, w_{n-2}, w_{n-1})$$

- The estimation of a trigram word prediction probability:

$$p(w_3|w_1, w_2) = \frac{count(w_1,w_2,w_3)}{\sum_w count(w_1,w_2,w)}$$

- Drawbacks:
  - Sparsity->smoothing
  - curse of dimensionality
  - rely on exact pattern(not linguistically informed)
  - dependency beyond the window is ignored

Neural language models

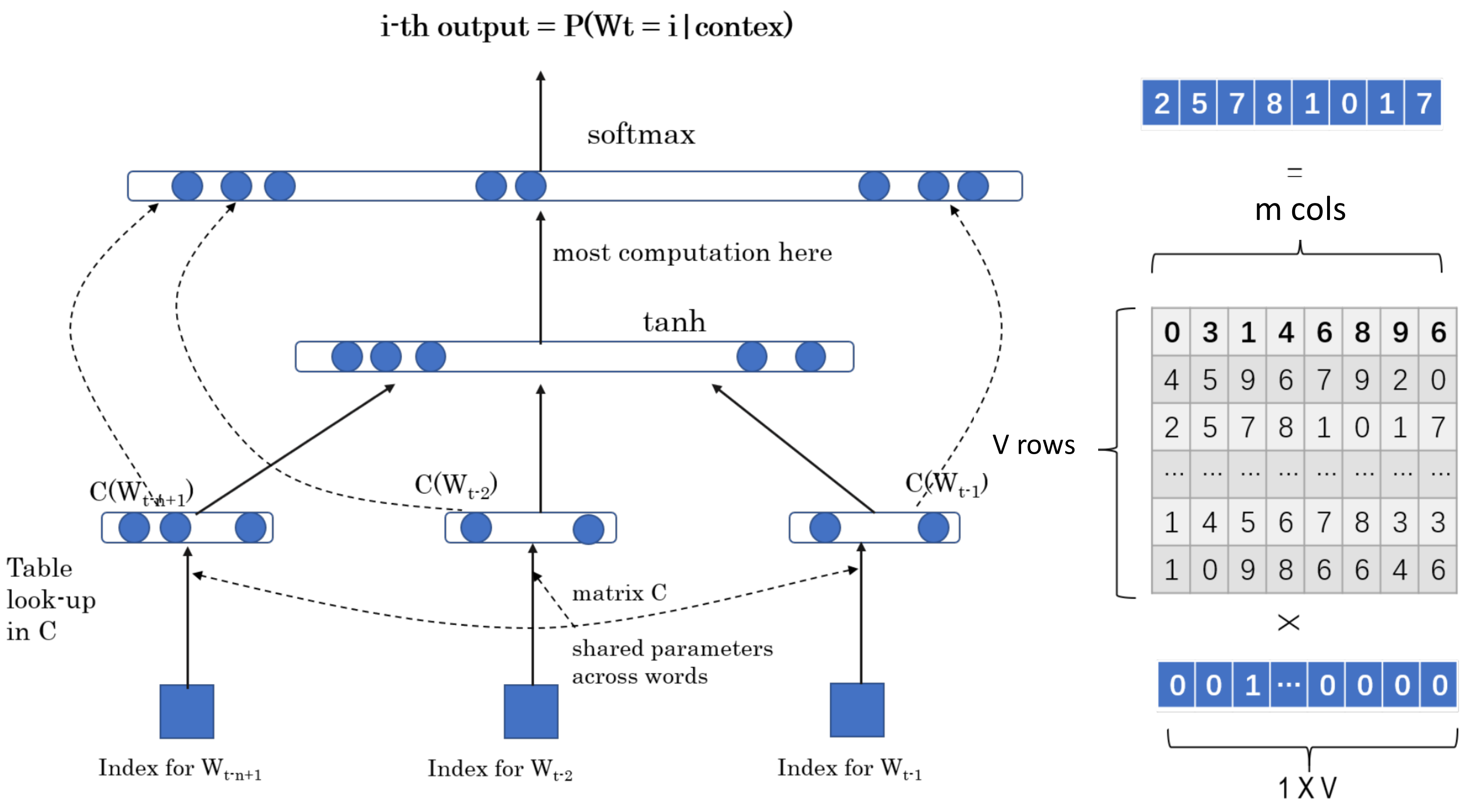
Two main forms:

- Feed-forward neural network based LM
  -to tackle the problems of data sparsity

- recurrent neural network based LM
  -to address the problem of limited context.

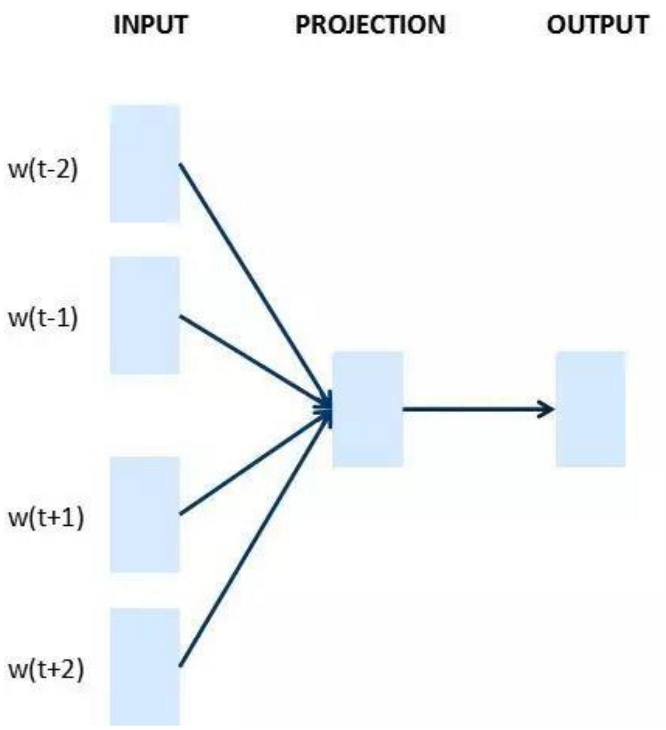
Neural language model(Continuous-space)

- Feed-Forward Neural Network Based Models

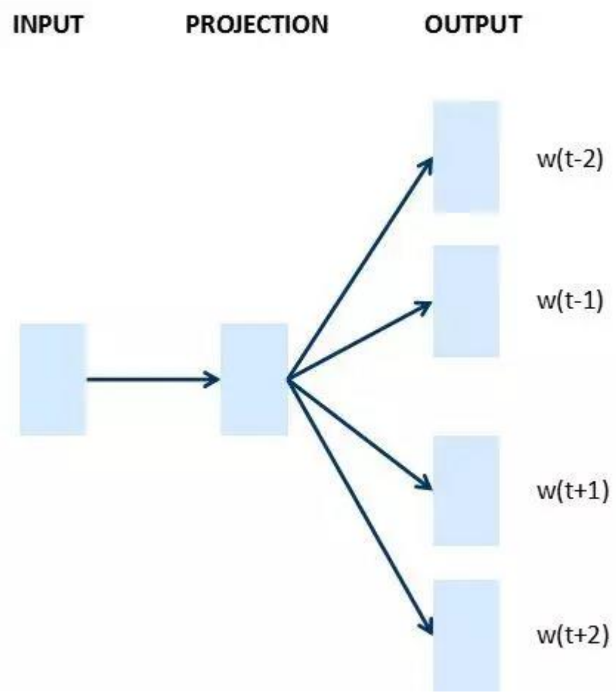


word2vec

A framework for learning word vectors

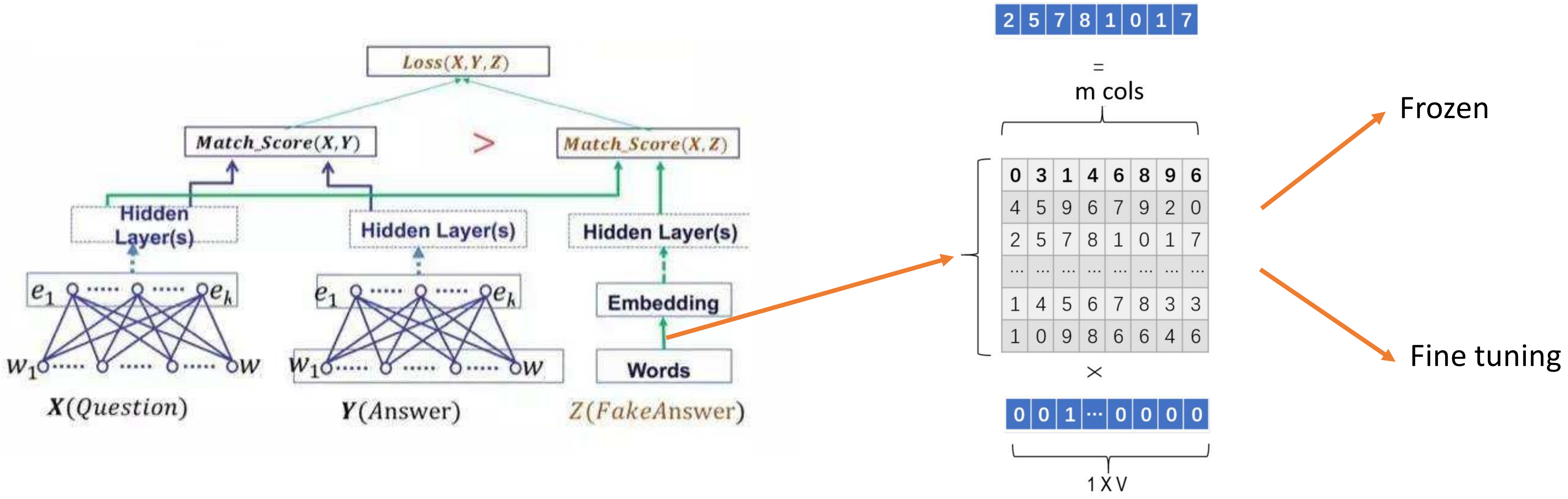


CBOW (Continuous Bag-of-Words Model)



Skip-gram (Continuous Skip-gram Model)

How to use word embedding?



QA

Word Embedding Matrix

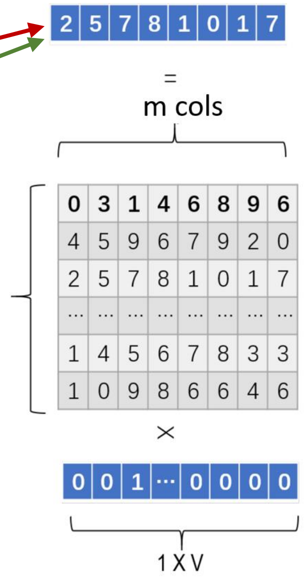
Classical usage of pretraining before 2018

Limit of word2vec

…very useful to protect banks or slopes from being washed away by river or rain…
…the location because it was high, about 100 feet above the bank of river…
…The bank has plan to branch throughout the country…
…They throttled the watchman and robbed the bank

Polysemy: Bank

1. ORGANIZATION
2. RAISED GROUND

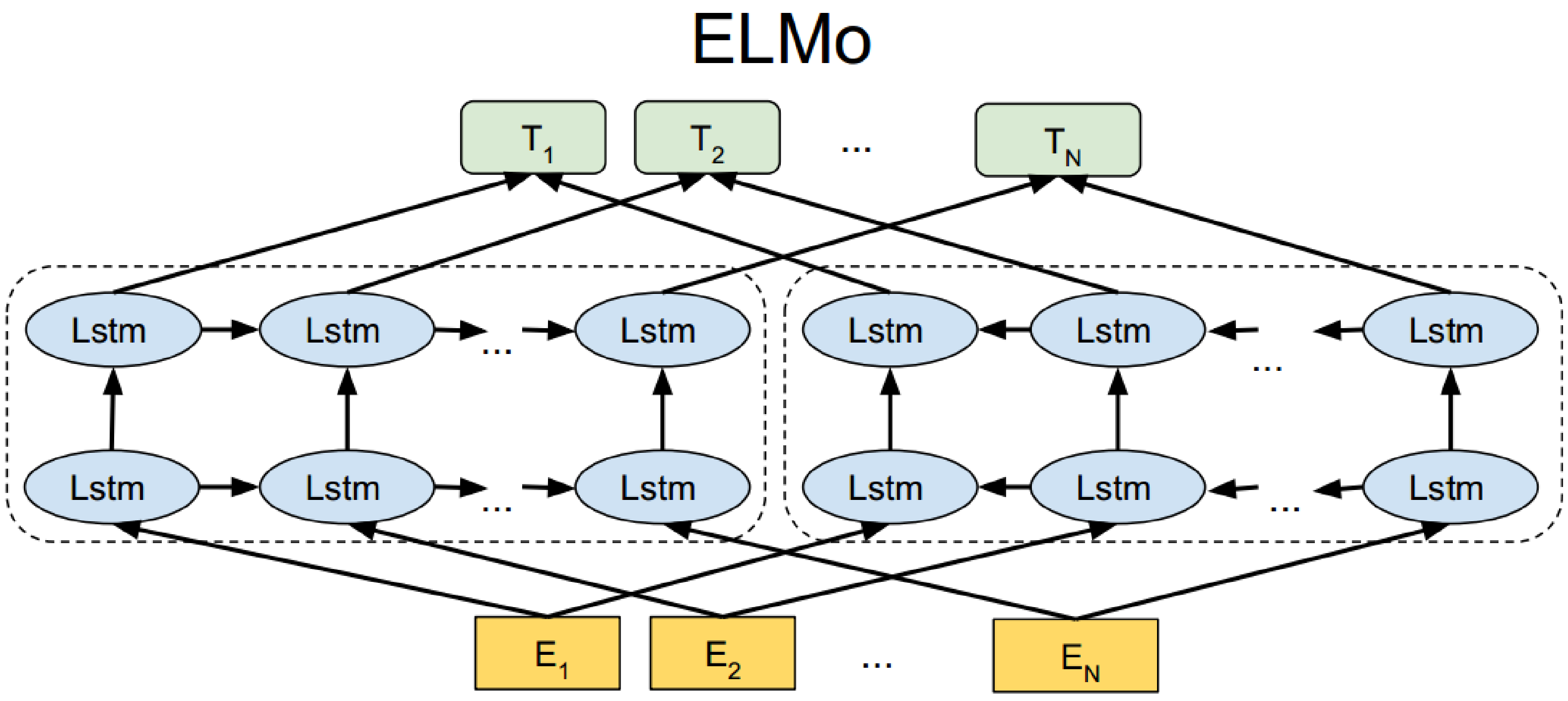


Static word embedding:
one vector for each word and smooshes all the context into that vector

Table of contents

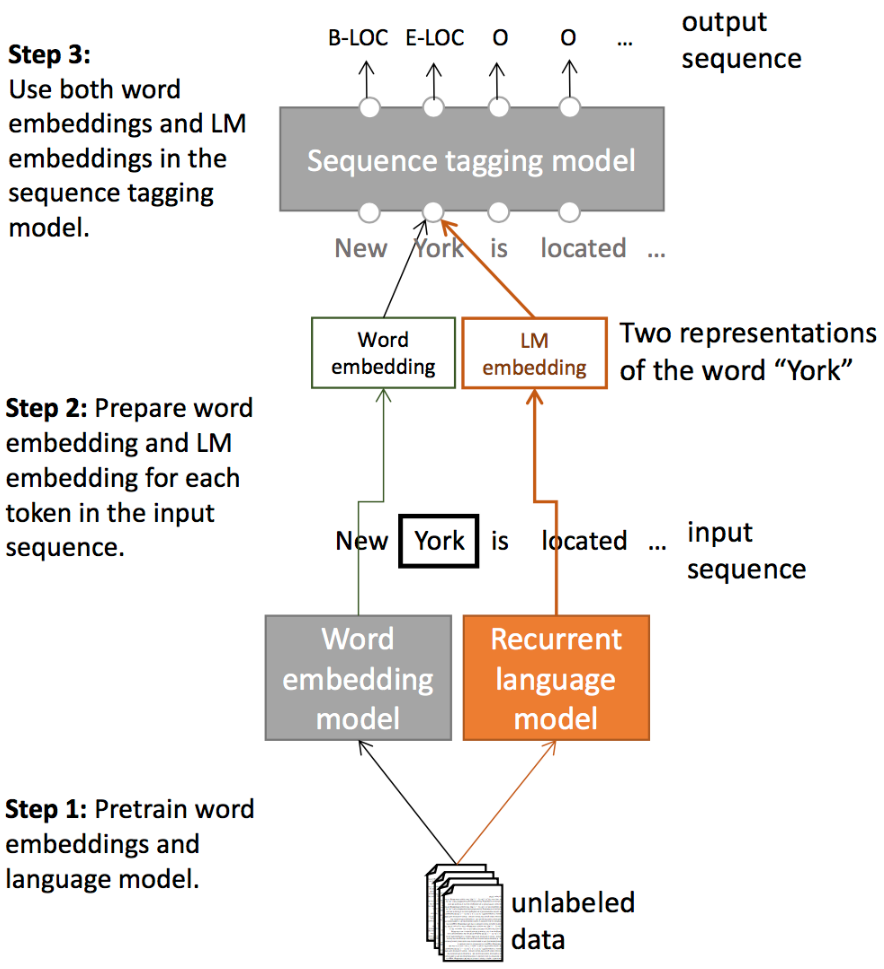
ELMo:Embeddings from Language Models (NAACL 2018)

Deep contextualized word representations



Pre-training

ELMo: How to use?



ELMo: Polysemy?

Play:

1. exercise for recreation
2. drama

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

- Solved!

ELMo: Performance

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

6 NLP tasks: improved by 5 ~ 25%

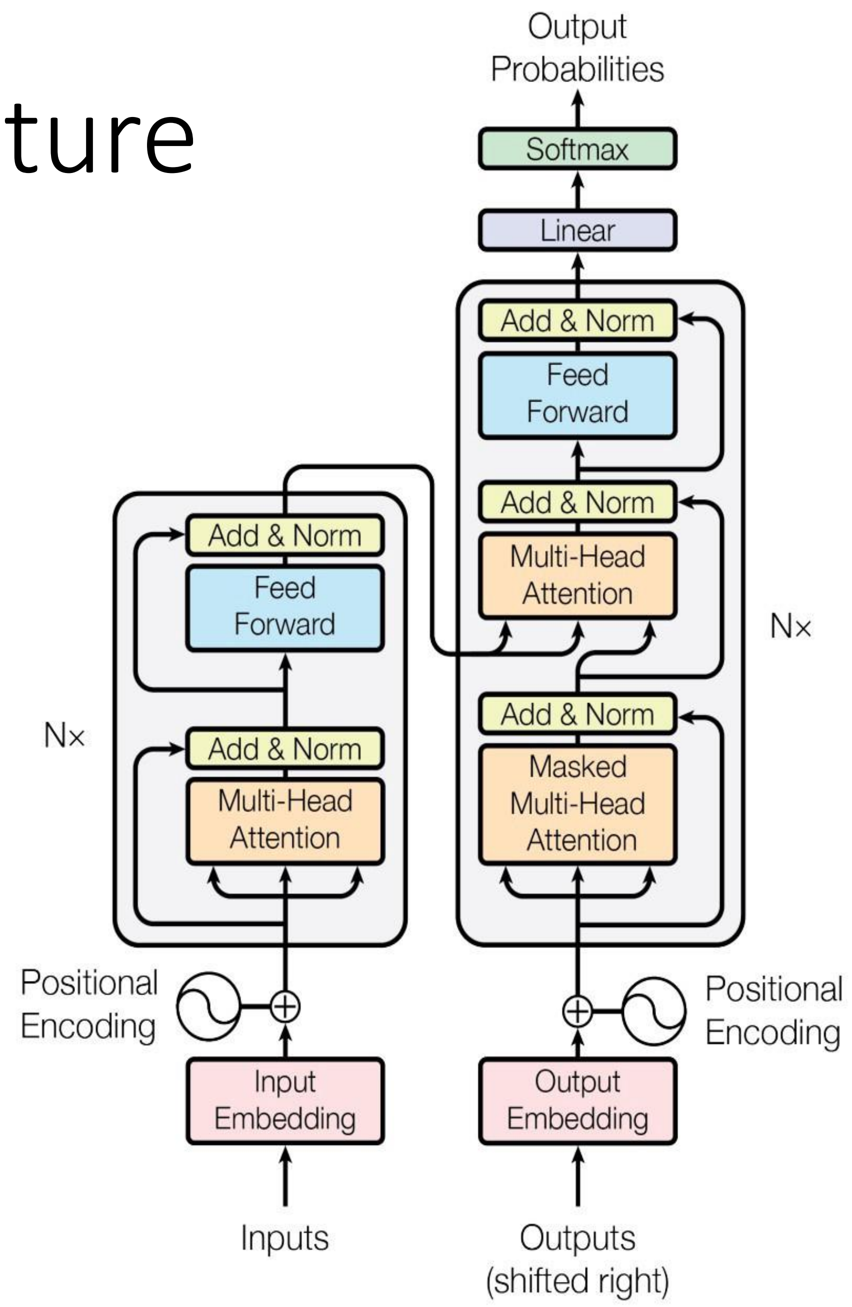
ELMo: disadvantages

- LSTM is not as powerful as transformer in feature extraction
- Concatenation is not a ideal way to fuse the bi-directional information

Table of contents

Transformer: overall architecture

- A sequence-to-sequence model based entirely on attention
- Strong results on standard WMT datasets
- Fast: only matrix multiplications



Transformer: Attention Tricks

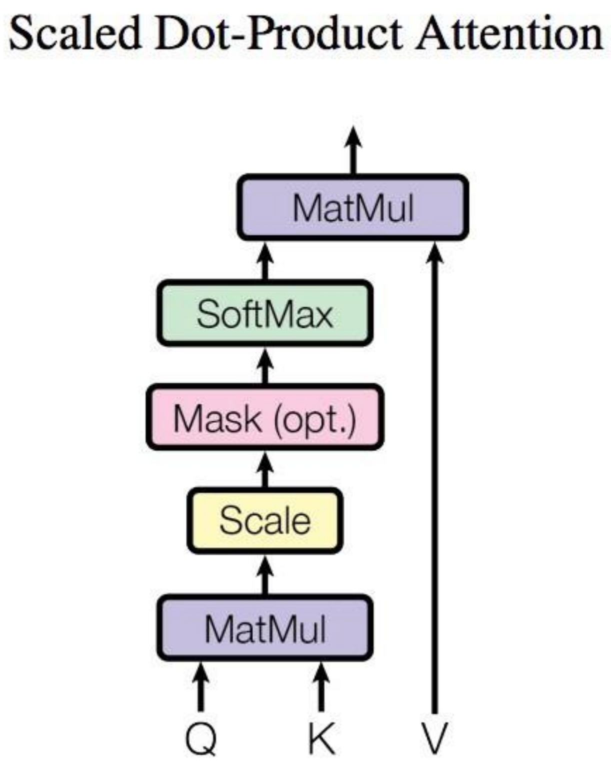
- **Self Attention**: Each layer combines words with Others

- **Multi-headed Attention**: 8 attention heads learned Independently

- **Normalized Dot-product Attention**: Remove bias in dot product when using large networks

- **Positional Encodings**: Make sure that even if we don’t have RNN, can still distinguish positions

Self Attention

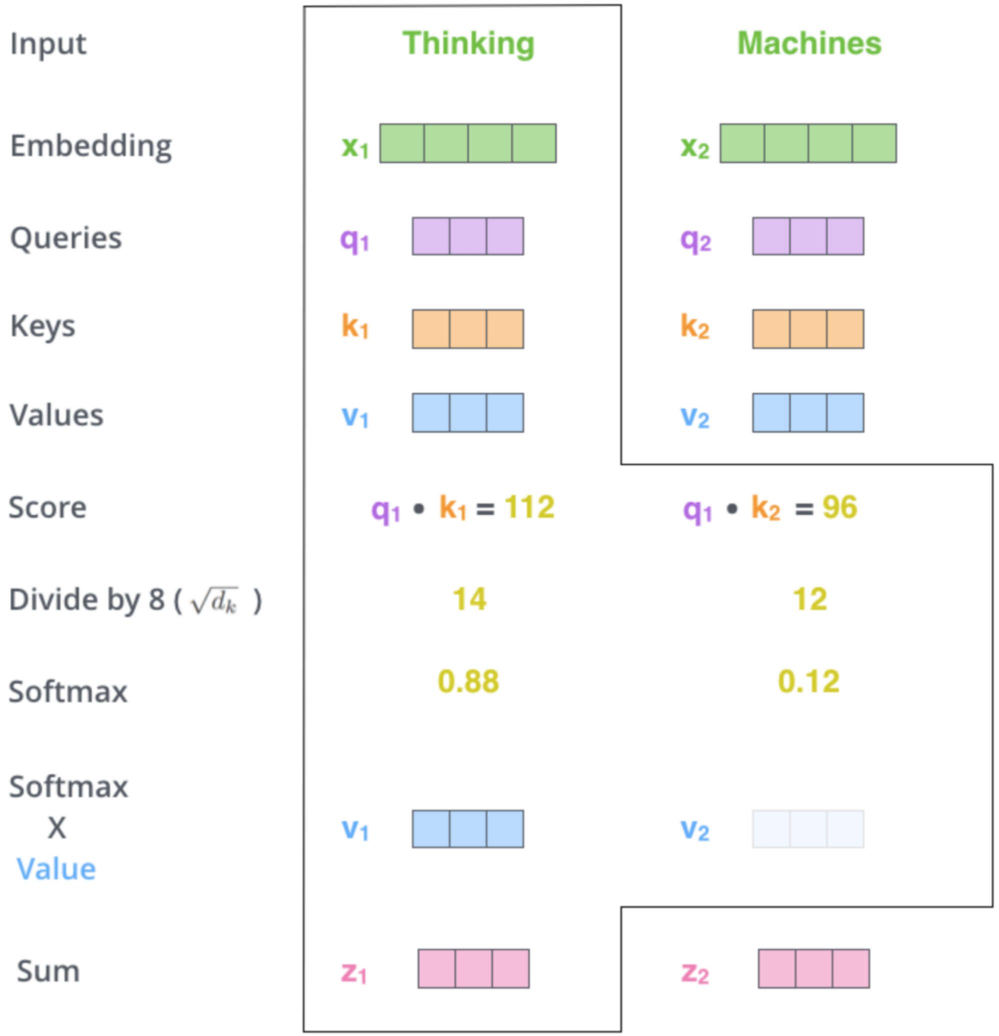


- Mapping a query and a set of key-value pairs to an output

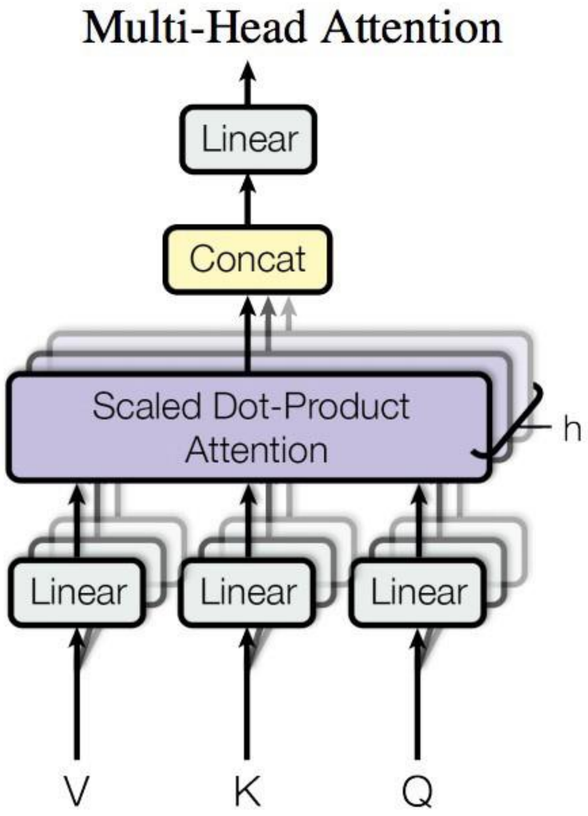
- The output is a weighted sum of the values

- The weight assigned to each value is computed by a compatibility function of the query with the corresponding key

Self Attention



Multi-Head Attention

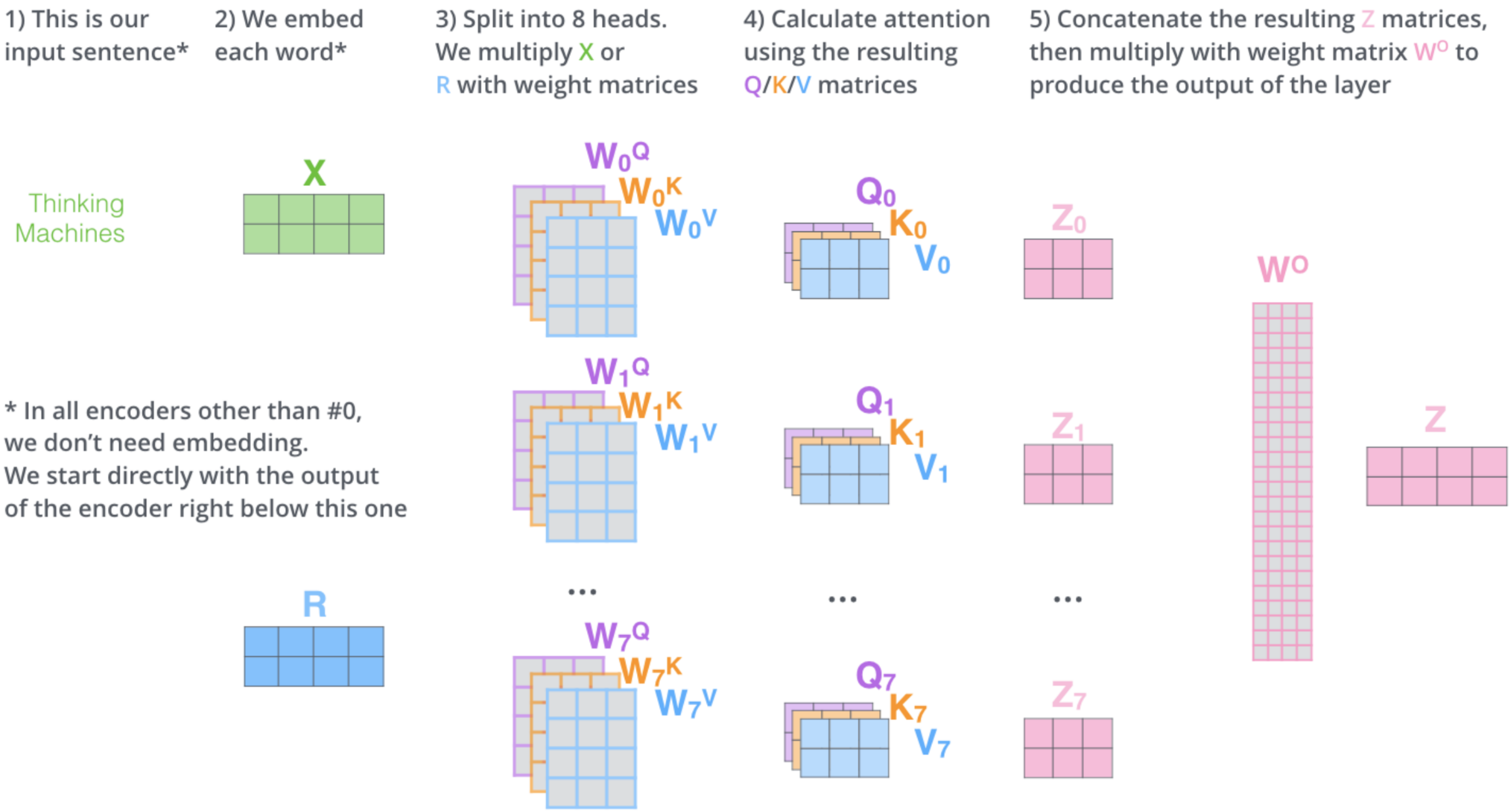


- It expands the model's ability to focus on different positions.

- It gives the attention layer multiple representation subspaces

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

Multi-Head Attention



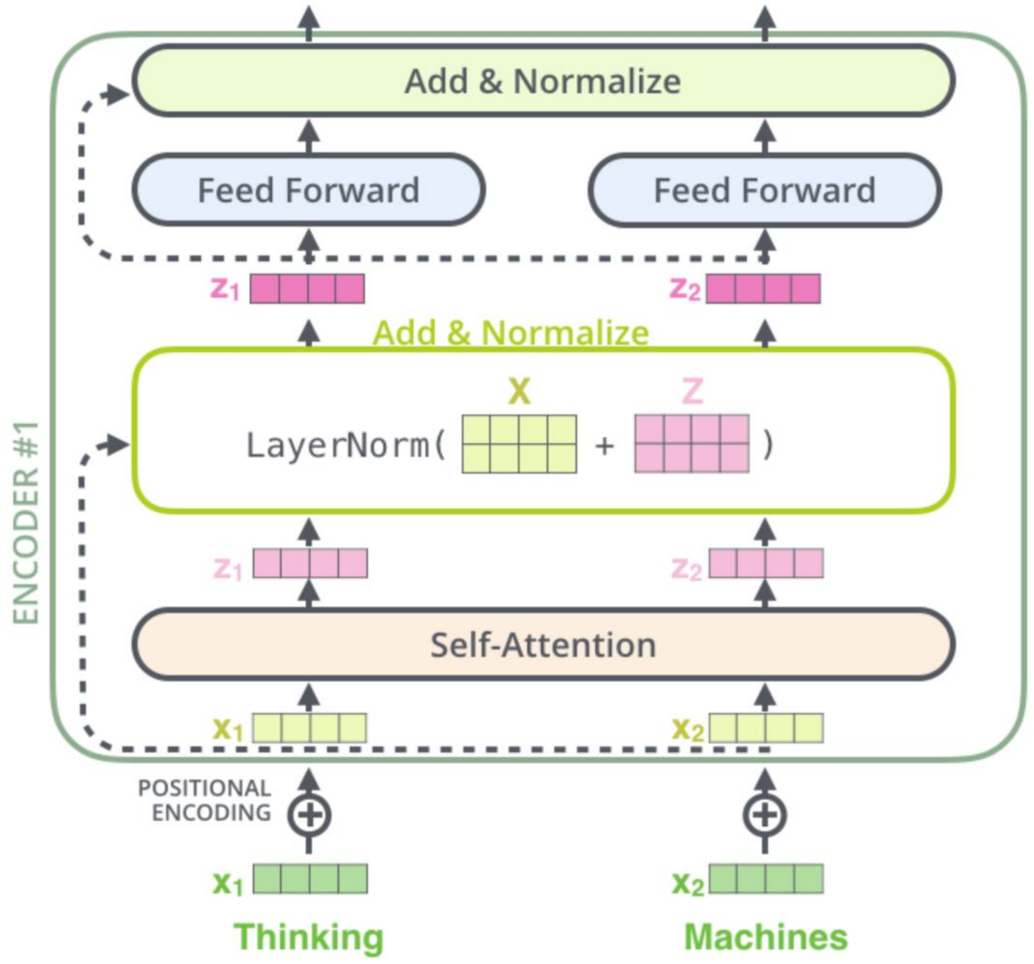
Positional Encodings

- Encode the positions of the inputs as vectors and are then added to the input embeddings
- Each dimension of the positional encoding is a wave with a different frequency

$$PE[pos, 2i] = sin(pos/10000^{2i/d_model})$$

$$PE[pos, 2i+1] = cos(pos/10000^{2i/d_model})$$

Encoder



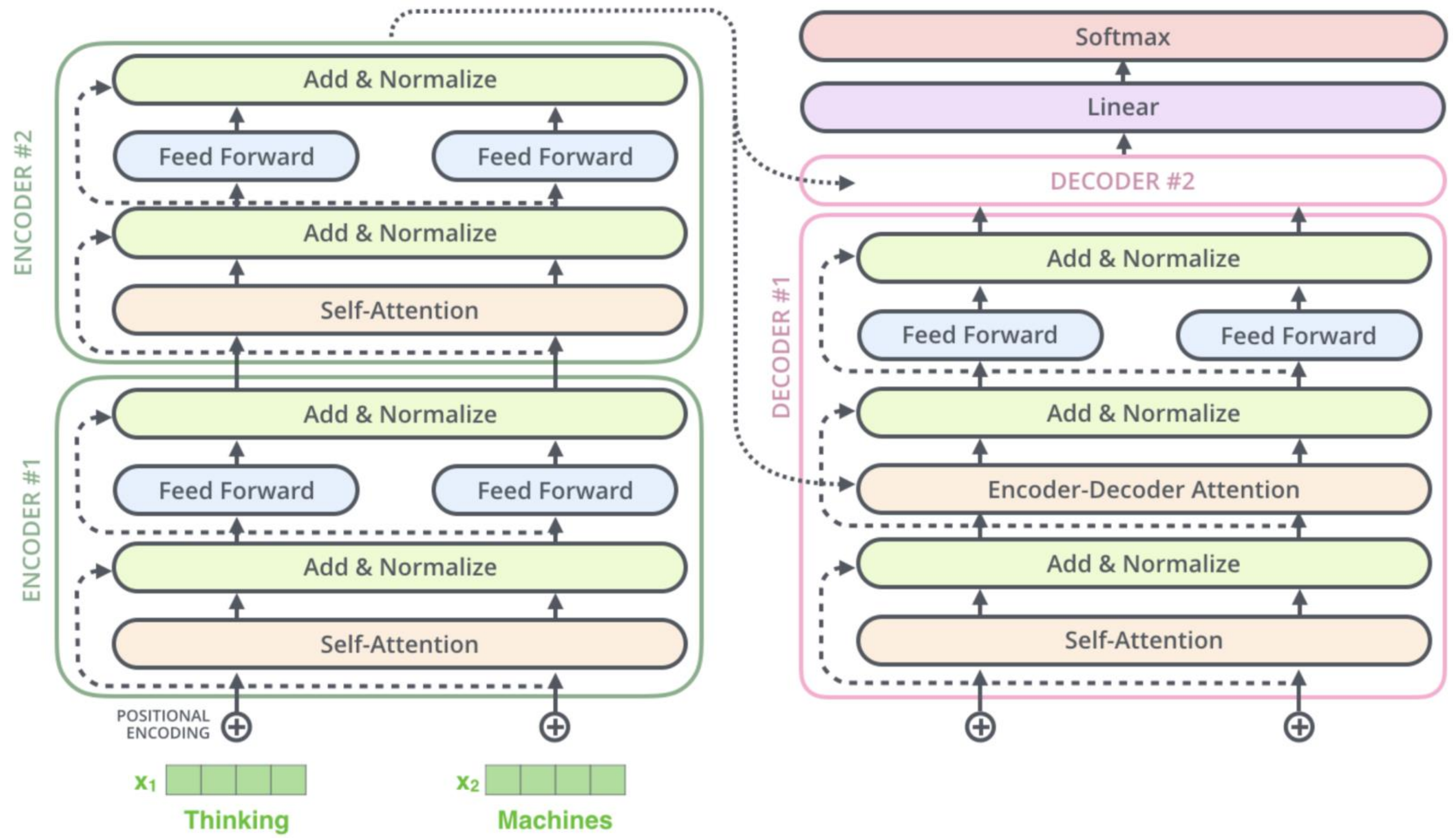
Add & Normalize
Feed Forward
Feed Forward
z1
z2
Add & Normalize
X
Z
LayerNorm(+)
z1
z2
Self-Attention
x1
x2
ENCODER #1
POSITIONAL
ENCODING
x1
x2
Thinking
Machines

Decoder

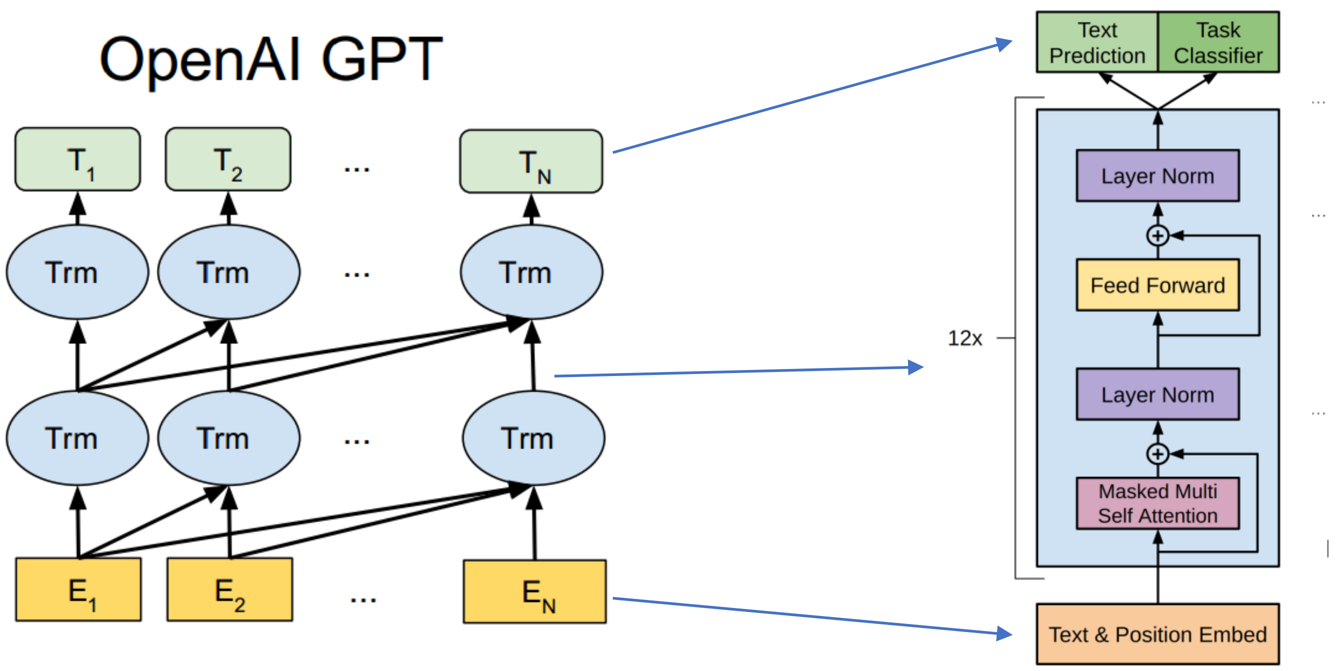
Masked multi-head self attention: self-attention layer is only allowed to attend to earlier positions in the output sequence

Encoder-Decoder Attention: attention layer works just like multiheaded self-attention, except it creates its Queries matrix from the layer previous to it, and takes the Keys and Values matrix from the output of the encoder stack

Encoder-Decoder



GPT: Generative Pre-Training (OpenAI 2018)

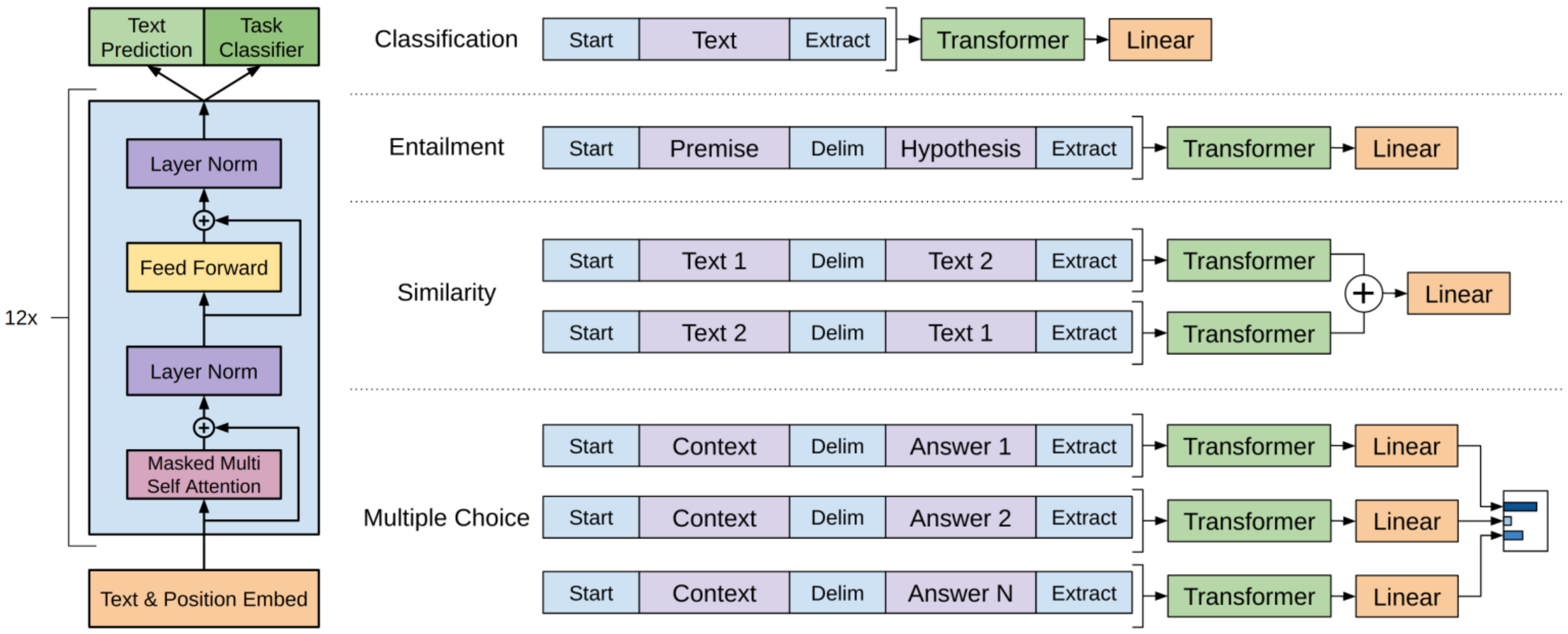


Pre-training

1. Transformer as feature extractor

2. pre-train a neural network using a language modeling objective (monodirectional)

GPT: how to use?



GPT: performance

Table 2: Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

Table 3: Results on question answering and commonsense reasoning, comparing our model with current state-of-the-art methods.. 9x means an ensemble of 9 models.

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

Table 4: Semantic similarity and classification results, comparing our model with current state-of-the-art methods. All task evaluations in this table were done using the GLUE benchmark. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

Method	**Classification**		**Semantic Similarity**			**GLUE**
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3	68.9
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**	**72.8**

Achieve best performance in 9 of 12 NLP tasks

GPT: Analysis of various model ablations

Table 5: Analysis of various model ablations on different tasks. Avg. score is a unweighted average of all the results. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

Method	Avg. Score	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	MNLI (acc)	QNLI (acc)	RTE (acc)
Transformer w/ aux LM (full)	74.7	45.4	91.3	82.3	82.0	**70.3**	**81.8**	**88.1**	**56.0**
Transformer w/o pre-training	59.9	18.9	84.0	79.4	30.9	65.5	75.7	71.2	53.8
Transformer w/o aux LM	**75.0**	**47.9**	**92.0**	**84.9**	**83.2**	69.8	81.1	86.9	54.4
LSTM w/ aux LM	69.1	30.3	90.5	83.2	71.8	68.1	73.7	81.1	54.6

1. Transformer is better feature extractor than LSTM
2. Pre-training is critical

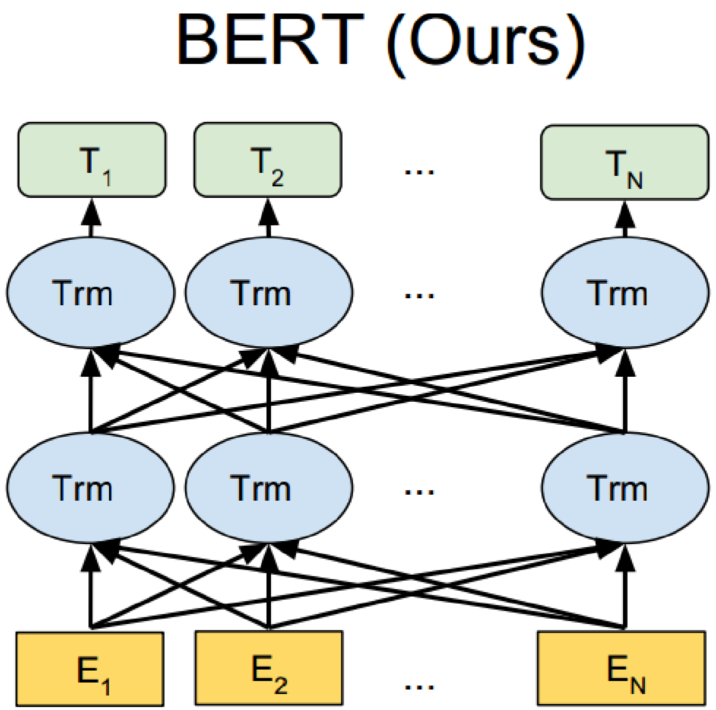
GTP: disadvantage

- Monodirectional model

Table of contents

- Pre-training in image processing
- Language model to word embedding
- Word embedding to ELMO
- Word embedding to GTP
- The birth of BERT

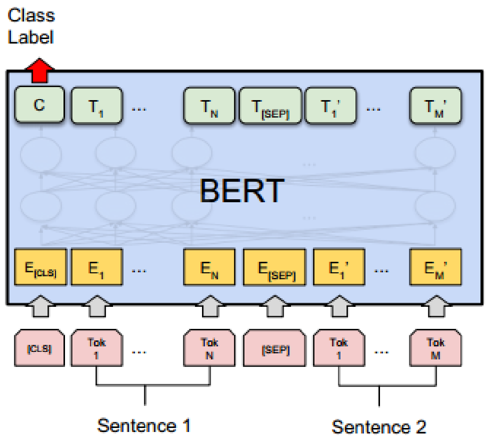
BERT: Bidirectional Encoder Representations from Transformers(Google AI 2018)



1. Transformer as feature extractor
2. Language modeling objective (bi-directional)

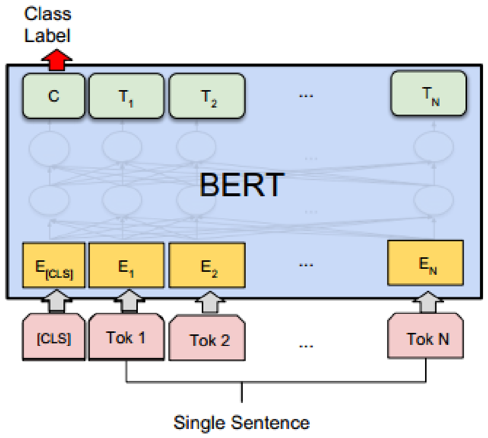
BERT: how to use?

Relationship between sentences



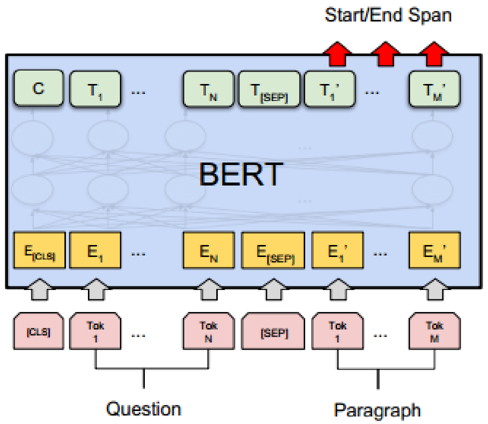
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

Single sentence classification



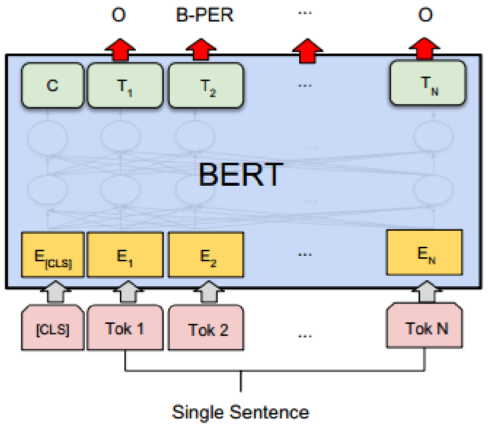
(b) Single Sentence Classification Tasks:
SST-2, CoLA

Reading comprehension



(c) Question Answering Tasks:
SQuAD v1.1

Sequence tagging



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

BERT: performance

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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

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.

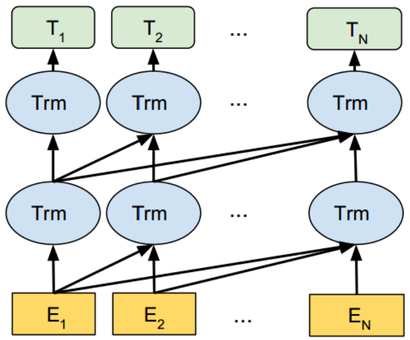
System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	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.

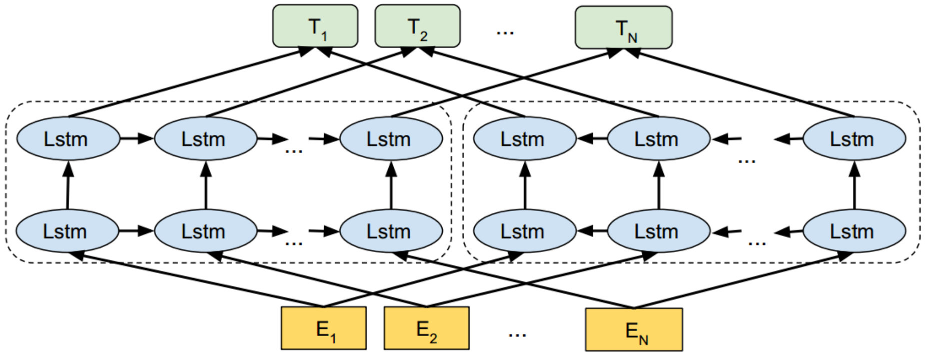
Improve the performance of all 11 tasks

From word embedding to BERT

OpenAI GPT



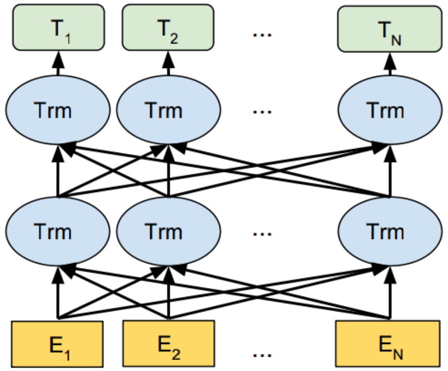
ELMo



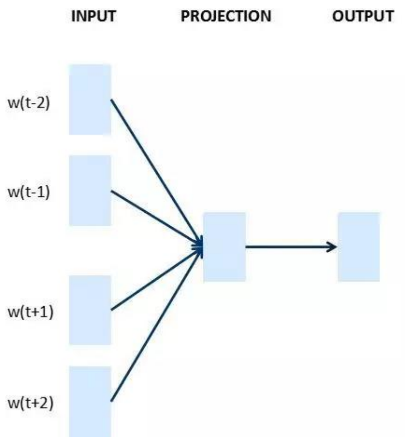
Transformer

bidirectional LM

BERT (Ours)

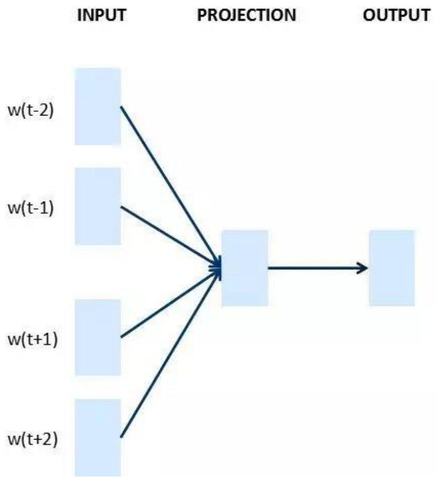


?



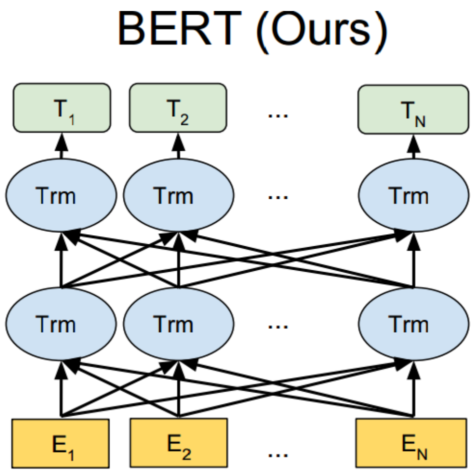
Word2Vec:CBOW

BERT: how to construct bi-directional LM?

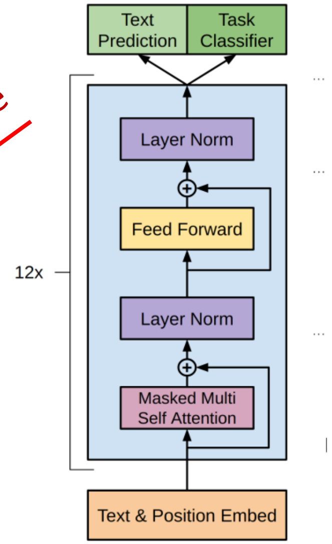


Word2Vec:CBO

Similar idea



But want Trm architecture



Next sentence prediction

Masked LM

BERT: how to construct bi-directional LM?

token. Instead, the training data generator chooses 15% of tokens at random, e.g., in the sentence `my dog is hairy` it chooses `hairy`. It then performs the following procedure:

- Rather than *always* replacing the chosen words with `[MASK]`, the data generator will do the following:
- 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`. The purpose of this is to bias the representation towards the actual observed word.

→ Masked LM

BERT: how to construct bi-directional LM?

not directly captured by language modeling. In order to train a model that understands sentence relationships, we pre-train a binarized *next sentence prediction* task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus. For example:

Next sentence prediction

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

`he bought a gallon [MASK] milk [SEP]`

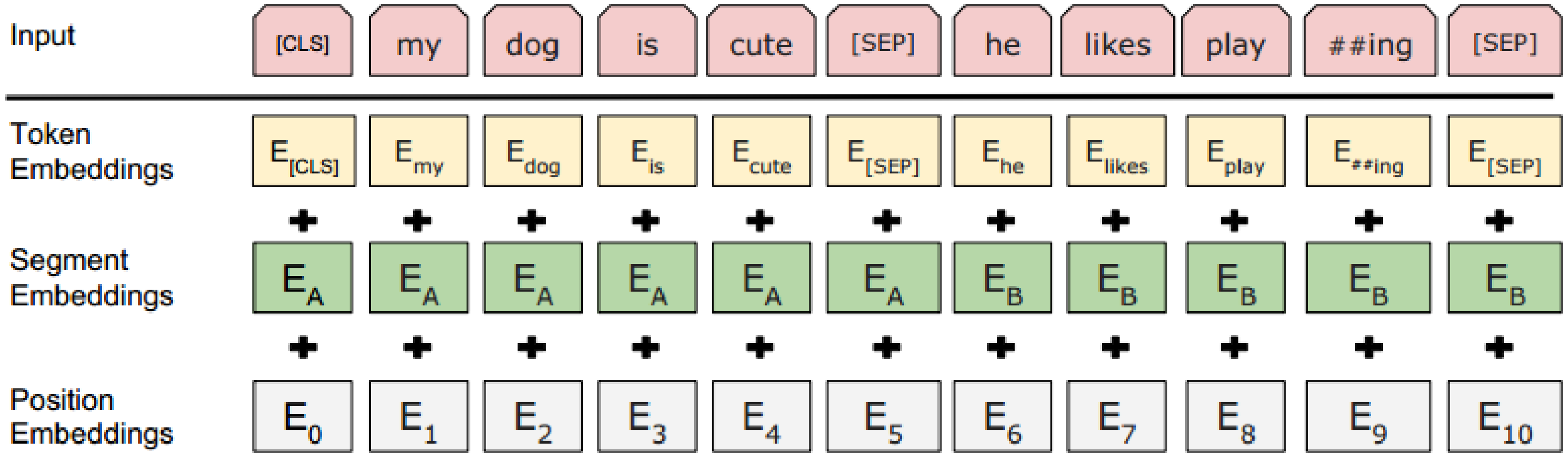
Label = `IsNext`

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

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

Label = `NotNext`

BERT: input processing



BERT: Ablation analysis

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

Table 5: Ablation over the pre-training tasks using the $\text{BERT}_{\text{BASE}}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. "+ BiLSTM" adds a randomly initialized BiLSTM on top of the "LTR + No NSP" model during fine-tuning.

1. Bidirectional language model is important

2. Next sentence prediction is critical for certain tasks