预训练在自然语言处理的发展：从Word Embedding到BERT模型

张俊林

新浪微博

2018.11

TABLE OF CONTENTES

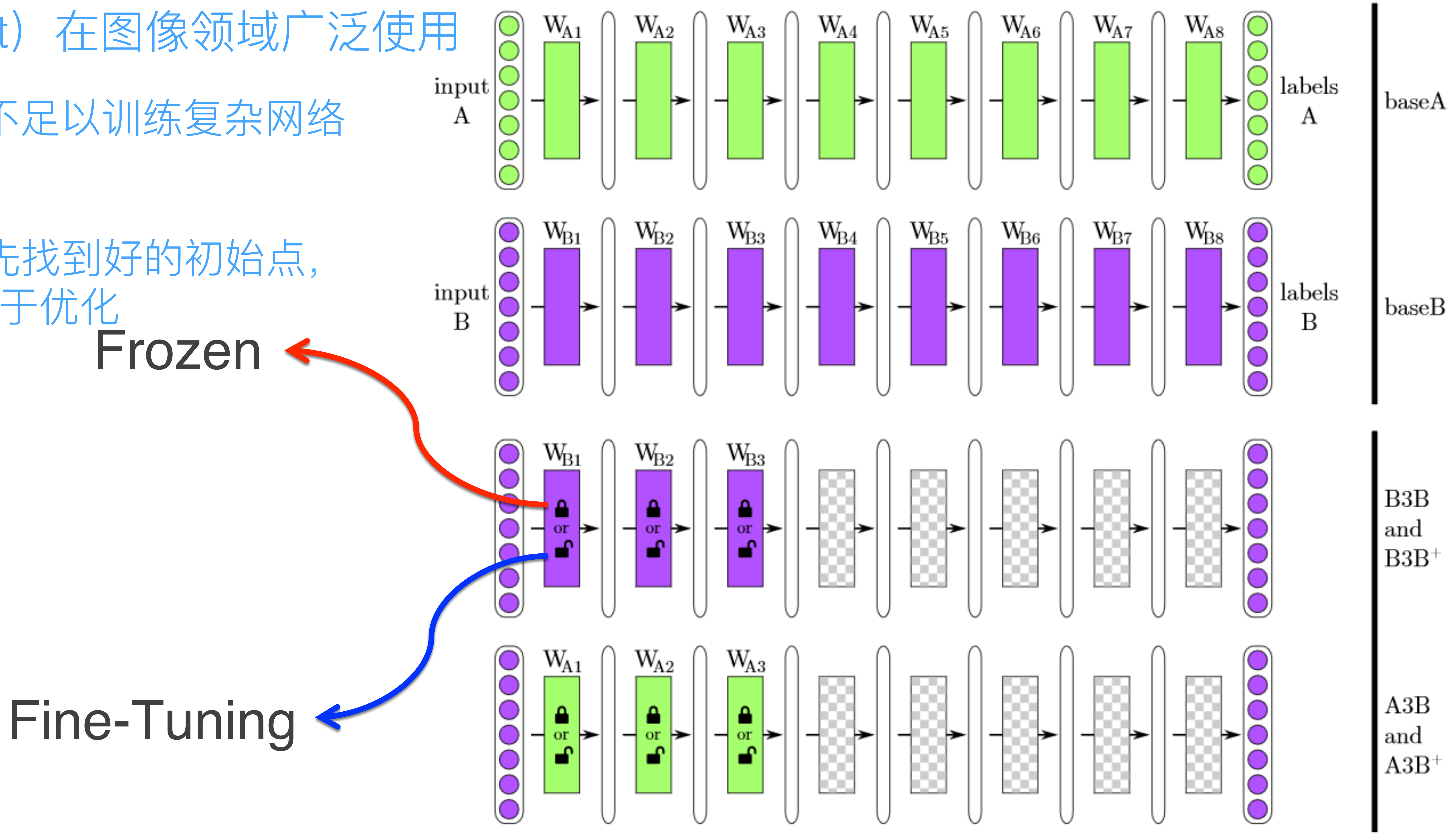
预训练在图像领域的应用

预训练（say,imagenet）在图像领域广泛使用

1.训练数据小，不足以训练复杂网络

2.加快训练速度

3.参数初始化，先找到好的初始点，有利于优化

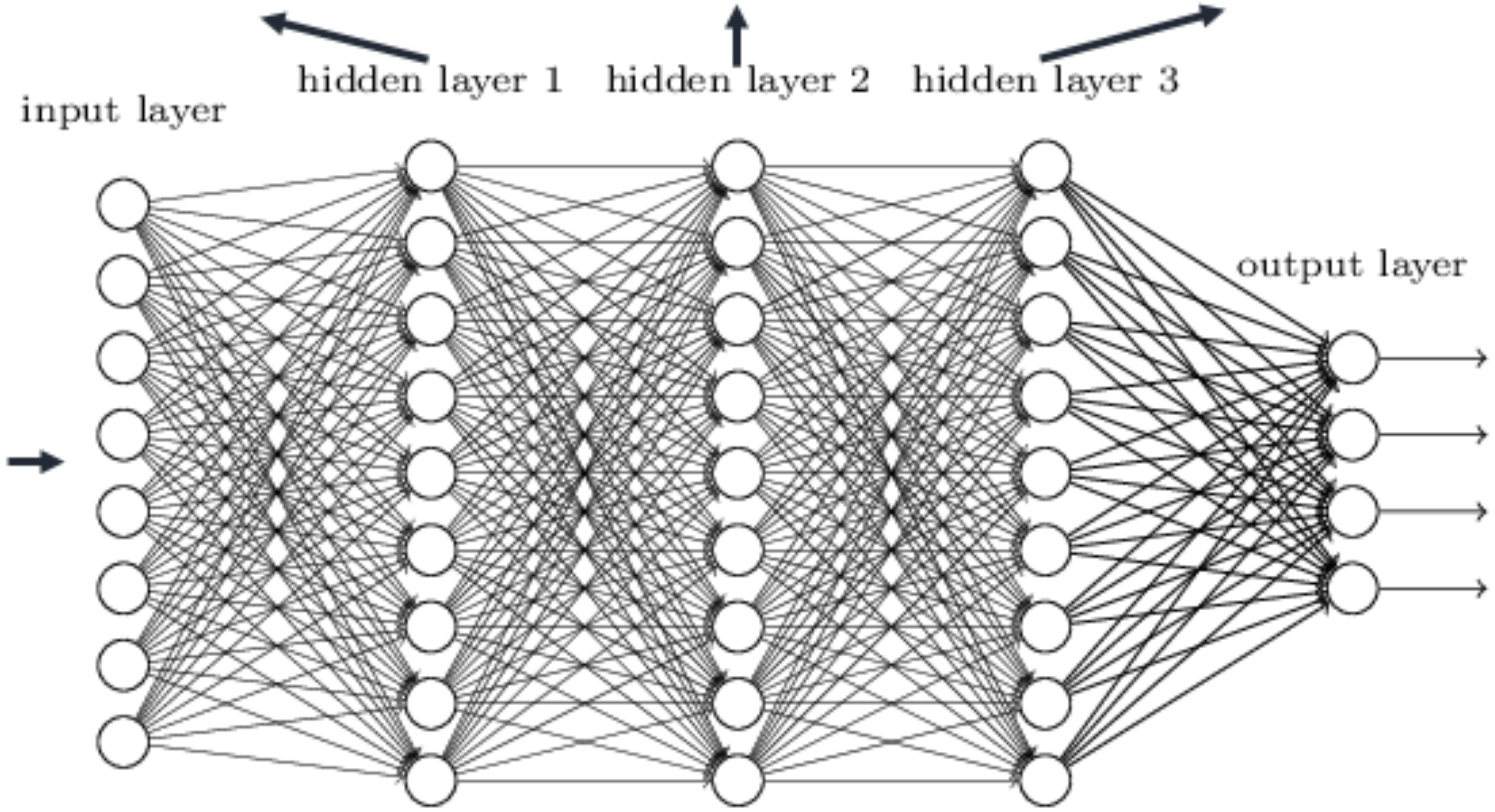


预训练在图像领域的应用

Deep neural networks learn hierarchical feature representations

为什么要复用?

底层特征的可复用性



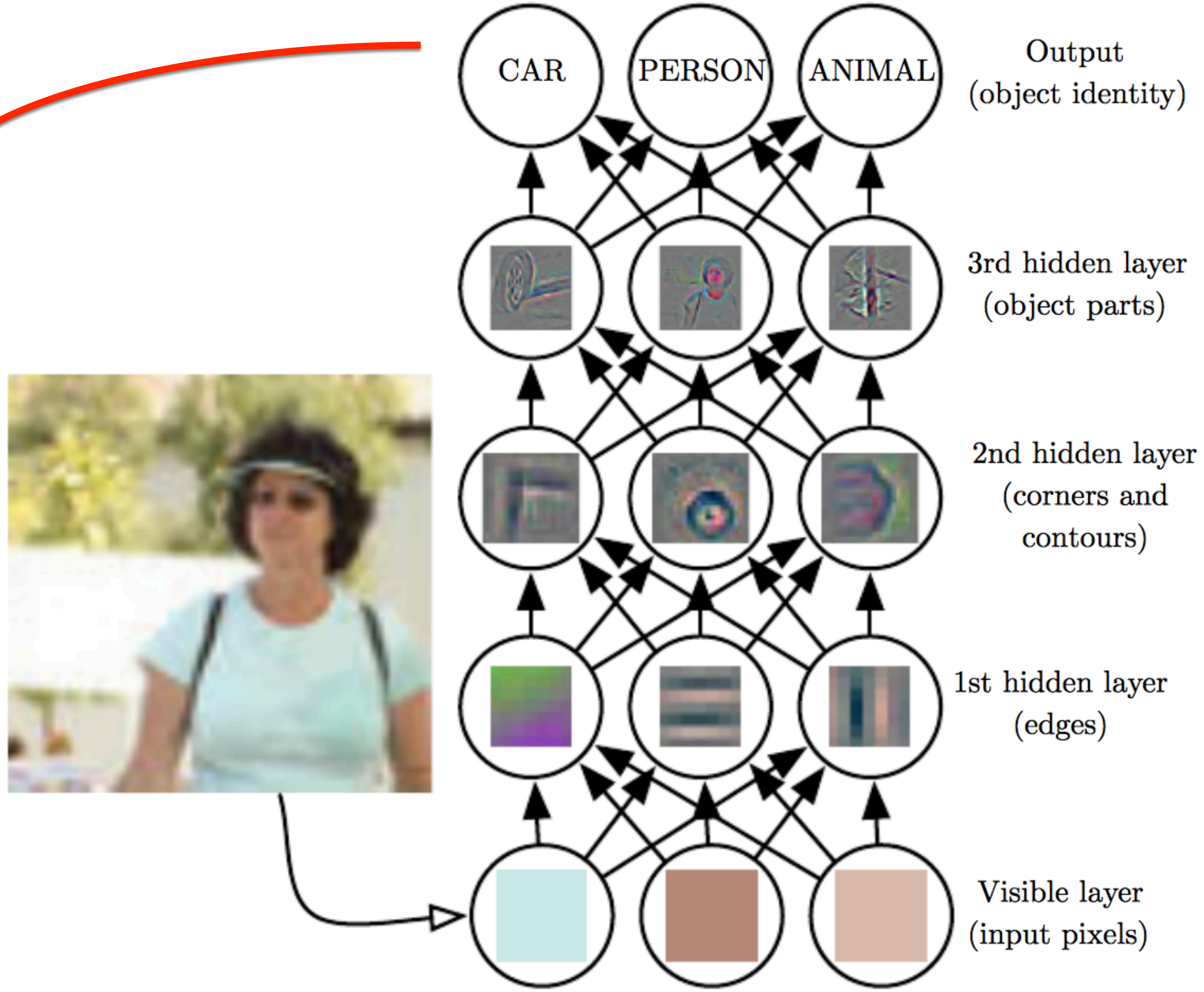
为什么要fine-tuning?

高层特征任务相关性

微博
weibo.com

预训练在图像领域的应用

Imagenet具备任务通用性



预训练在图像领域的应用

keras Home Articles Learn Tools Examples Reference News

Using Pre-Trained Models

Applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning.

Weights are downloaded automatically when instantiating a model. They are stored at `~/.keras/models/`.

The following image classification models (with weights trained on ImageNet) are available:

- Xception
- VGG16
- VGG19
- ResNet50
- InceptionV3
- InceptionResNetV2
- MobileNet
- MobileNetV2
- DenseNet
- NASNet

Prediction

Feature Extraction

Fine-tuning

预训练在图像领域的应用

Classify ImageNet classes with ResNet50

```r
# instantiate the model
model <- application_resnet50(weights = 'imagenet')

# load the image
img_path <- "elephant.jpg"
img <- image_load(img_path, target_size = c(224,224))
x <- image_to_array(img)

# ensure we have a 4d tensor with single element in the batch dimension,
# the preprocess the input for prediction using resnet50
x <- array_reshape(x, c(1, dim(x)))
x <- imagenet_preprocess_input(x)

# make predictions then decode and print them
preds <- model %>% predict(x)
imagenet_decode_predictions(preds, top = 3)[[1]]
```

```
  class_name class_description      score
1  n02504013    Indian_elephant 0.90117526
2  n01871265             tusker 0.08774310
3  n02504458   African_elephant 0.01046011
```

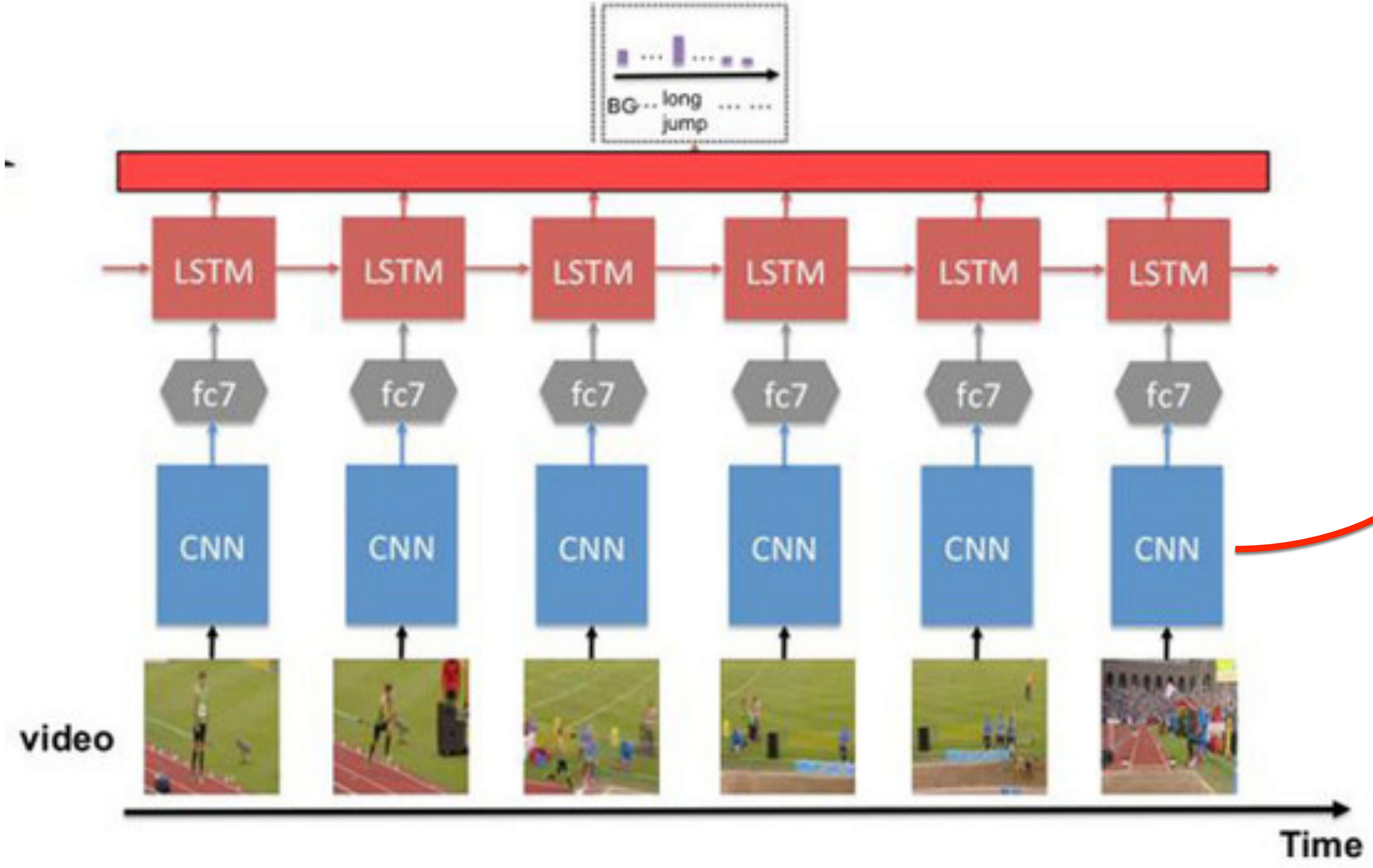
Prediction

Fine-tune InceptionV3 on a new set of classes

```r
# create the base pre-trained model
base_model <- application_inception_v3(weights = 'imagenet', include_top = FALSE)

# add our custom layers
predictions <- base_model$output %>%
  layer_global_average_pooling_2d() %>%
  layer_dense(units = 1024, activation = 'relu') %>%
  layer_dense(units = 200, activation = 'softmax')

# this is the model we will train
model <- keras_model(inputs = base_model$input, outputs = predictions)

# first: train only the top layers (which were randomly initialized)
# i.e. freeze all convolutional InceptionV3 layers
freeze_weights(base_model)

# compile the model (should be done *after* setting layers to non-trainable)
model %>% compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy')

# train the model on the new data for a few epochs
model %>% fit_generator(...)

# at this point, the top layers are well trained and we can start fine-tuning
# convolutional layers from inception V3. We will freeze the bottom N layers
# and train the remaining top layers.

# let's visualize layer names and layer indices to see how many layers
# we should freeze:
layers <- base_model$layers
for (i in 1:length(layers))
  cat(i, layers[[i]]$name, "\n")

# we chose to train the top 2 inception blocks, i.e. we will freeze
# the first 172 layers and unfreeze the rest:
freeze_weights(base_model, from = 1, to = 172)
unfreeze_weights(base_model, from = 173)
```

Fine-tune

预训练在图像领域的应用



Pretraining+Fine-tuning

微博
weibo.com

TABLE OF CONTENTES

语言模型

Sentence 1:美联储主席本·伯南克昨天告诉媒体7000亿美元的救助资金

Sentence 2:美主席联储本·伯南克告诉昨天媒体7000亿美元的资金救

Sentence 3:美主车席联储本·克告诉昨天公司媒体7000伯南亿美行元

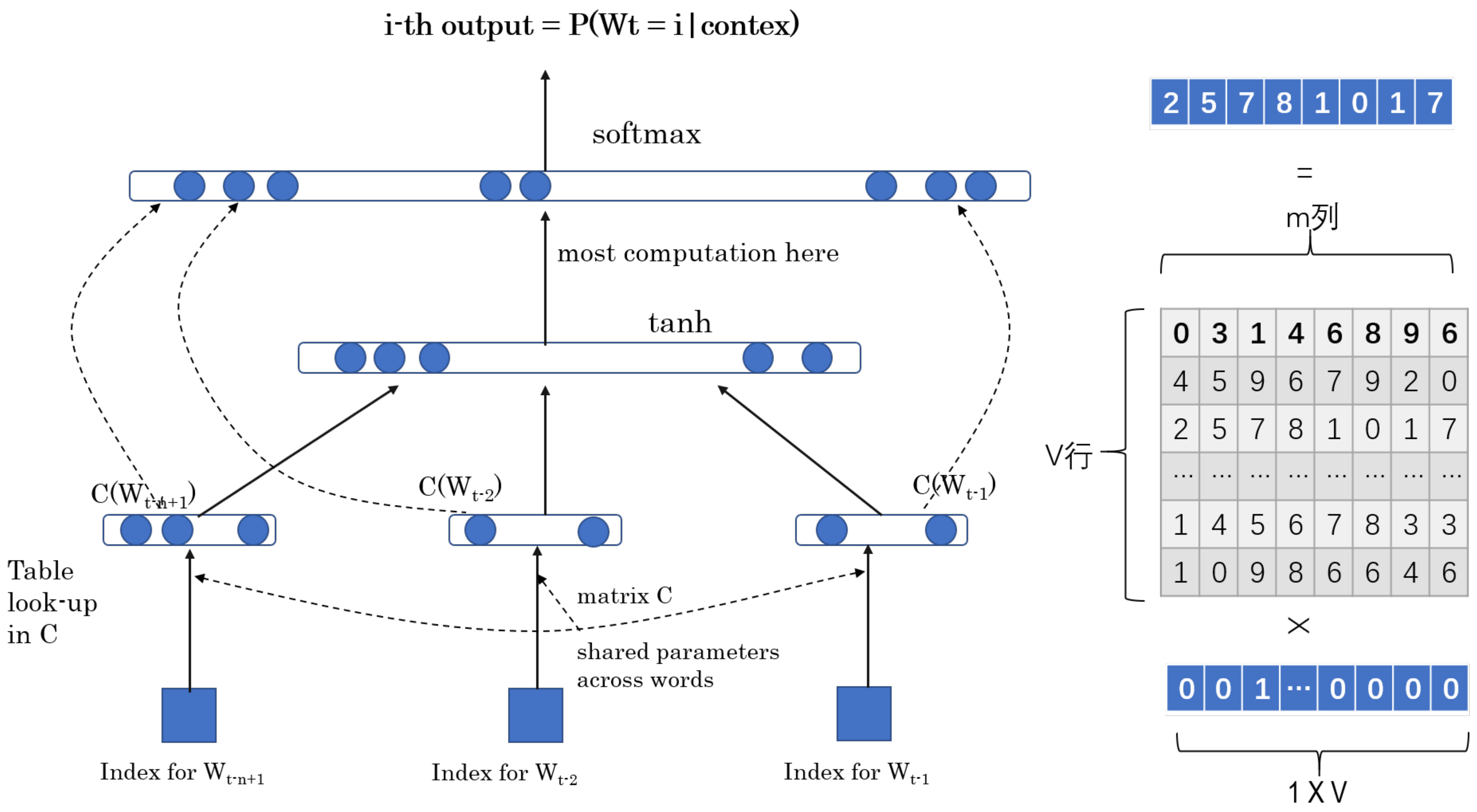
哪个句子更像一个合理的句子？如何量化评估?

$$P(S) = P(w_1, w_2, \ldots, w_n)$$

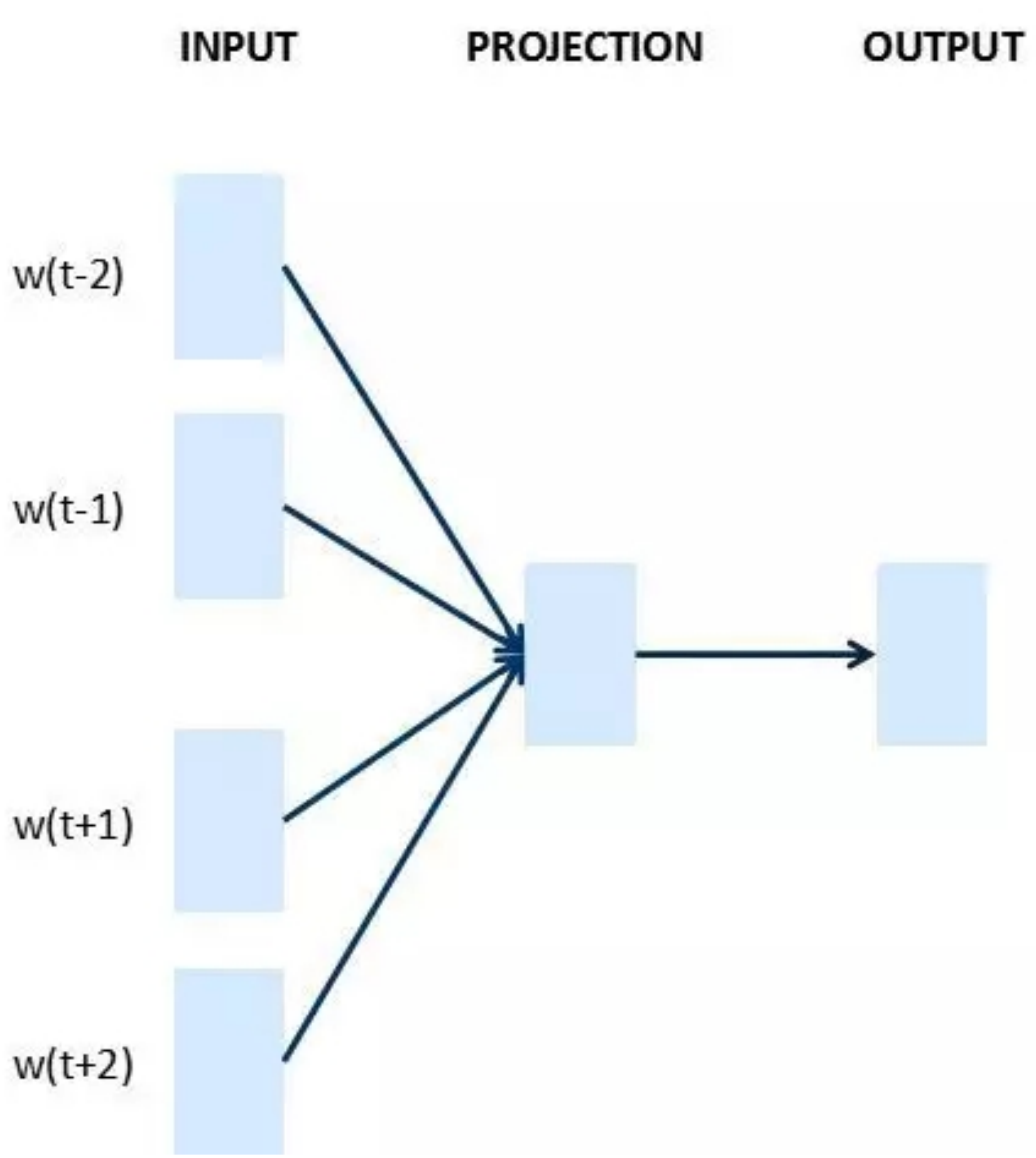
$$P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\cdots P(w_n|w_1, w_2, \cdots w_{n-1})$$

语言模型：$L = \sum_{w \in C} logP(w|context(w))$

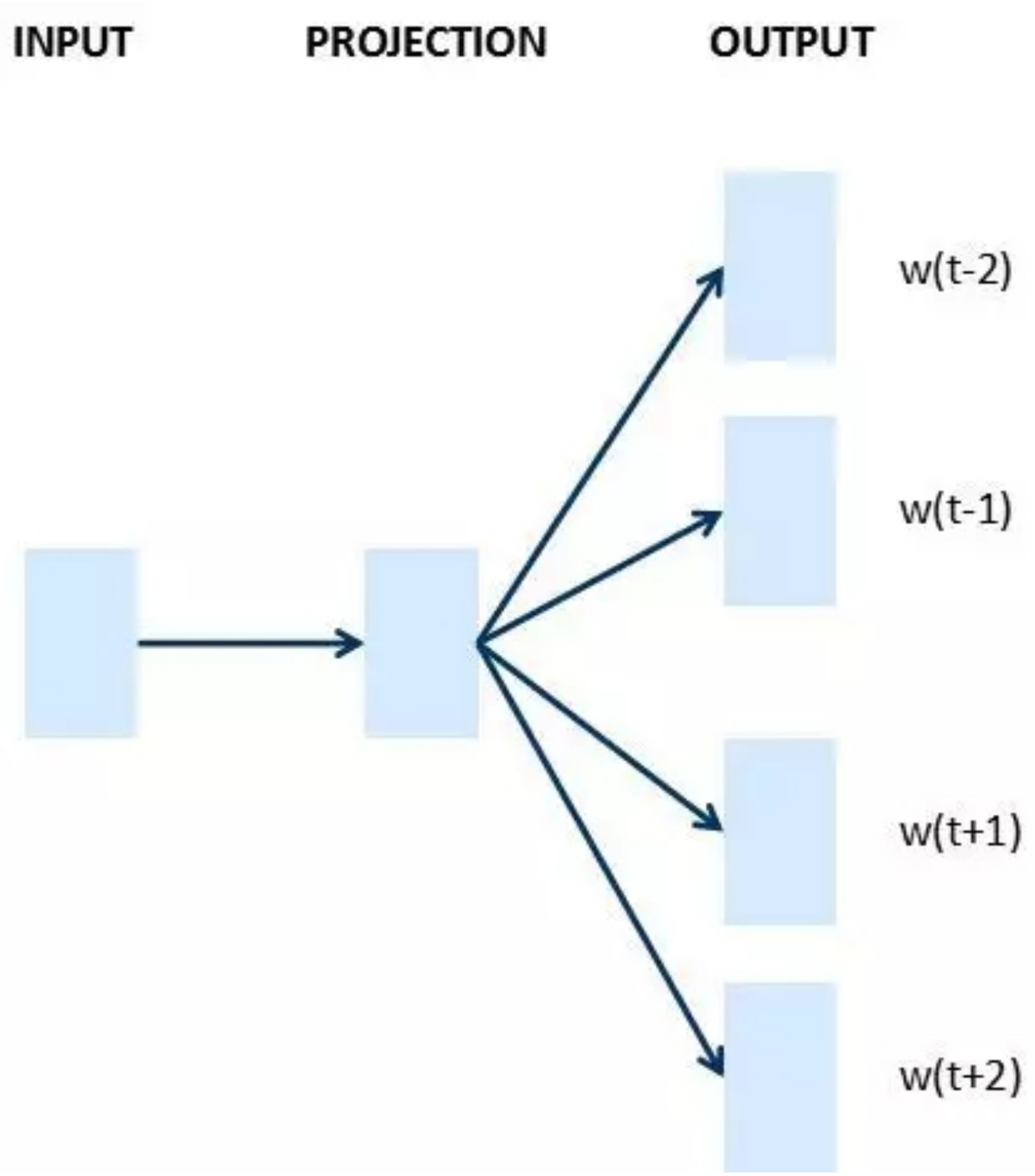
神经网络语言模型（NNLM）



Word2vec



CBOW (Continuous Bag-of-Words Model)



Skip-gram (Continuous Skip-gram Model)

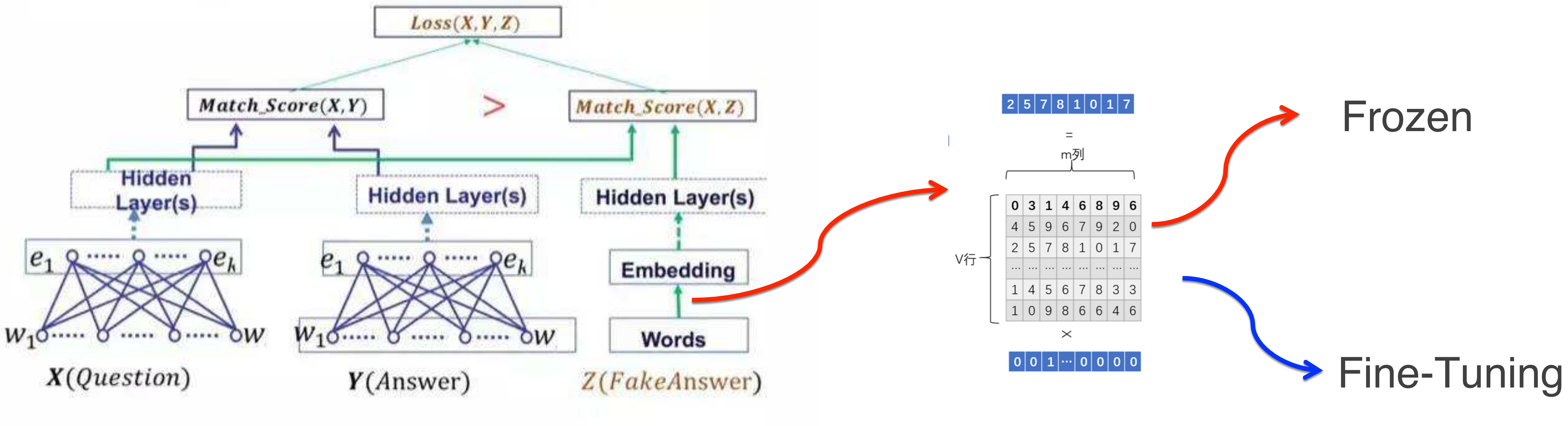
Word Embedding例子

```
In [10]: result = model.most_similar(u"青蛙")

In [11]: for e in result:
    print e[0], e[1]
   ....:
老鼠 0.559612870216
乌龟 0.489831030369
蜥蜴 0.478990525007
猫 0.46728849411
鳄鱼 0.461885392666
蟾蜍 0.448014199734
猴子 0.436584025621
白雪公主 0.434905380011
蚯蚓 0.433413207531
螃蟹 0.4314712286
```

```
In [20]: result = model.most_similar(u"清华大学

In [21]: for e in result:
    print e[0], e[1]
   ....:
北京大学 0.763922810555
化学系 0.724210739136
物理系 0.694550514221
数学系 0.684280991554
中山大学 0.677202701569
复旦 0.657914161682
师范大学 0.656435549259
哲学系 0.654701948166
生物系 0.654403865337
中文系 0.653147578239
```

```
In [24]: result = model.most_similar(u"习近平")

In [25]: for e in result:
    print e[0], e[1]
   ....:
胡锦涛 0.809472680092
江泽民 0.754633367062
李克强 0.739740967751
贾庆林 0.737033963203
曾庆红 0.732847094536
吴邦国 0.726941585541
总书记 0.719057679176
李瑞环 0.716384887695
温家宝 0.711952567101
王岐山 0.703570842743
```

学会了单词的WE， 怎么用?



QA

Word Embedding Matrix

这是18年之前NLP中典型的预训练模式！

有什么问题值得改进?

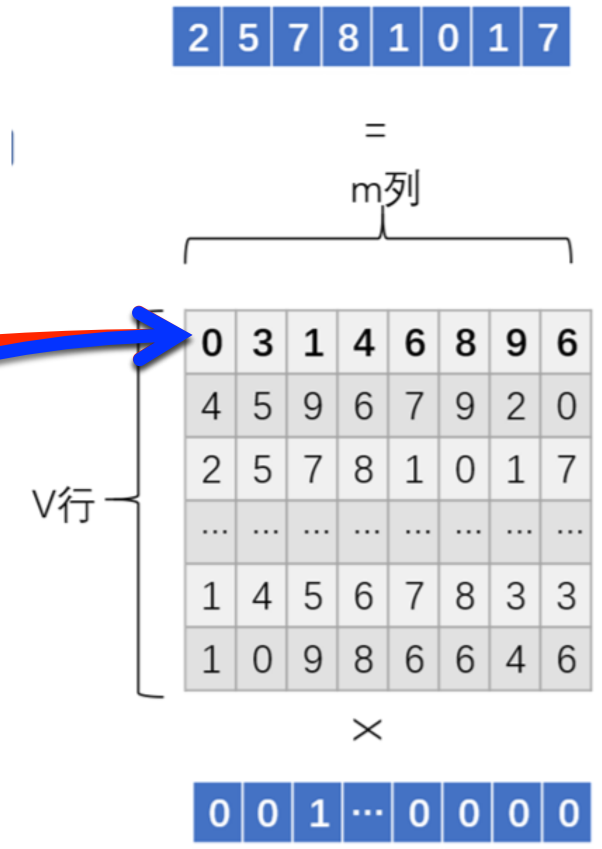
…very useful to protect banks or slopes from being washed away by river or rain…

…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…

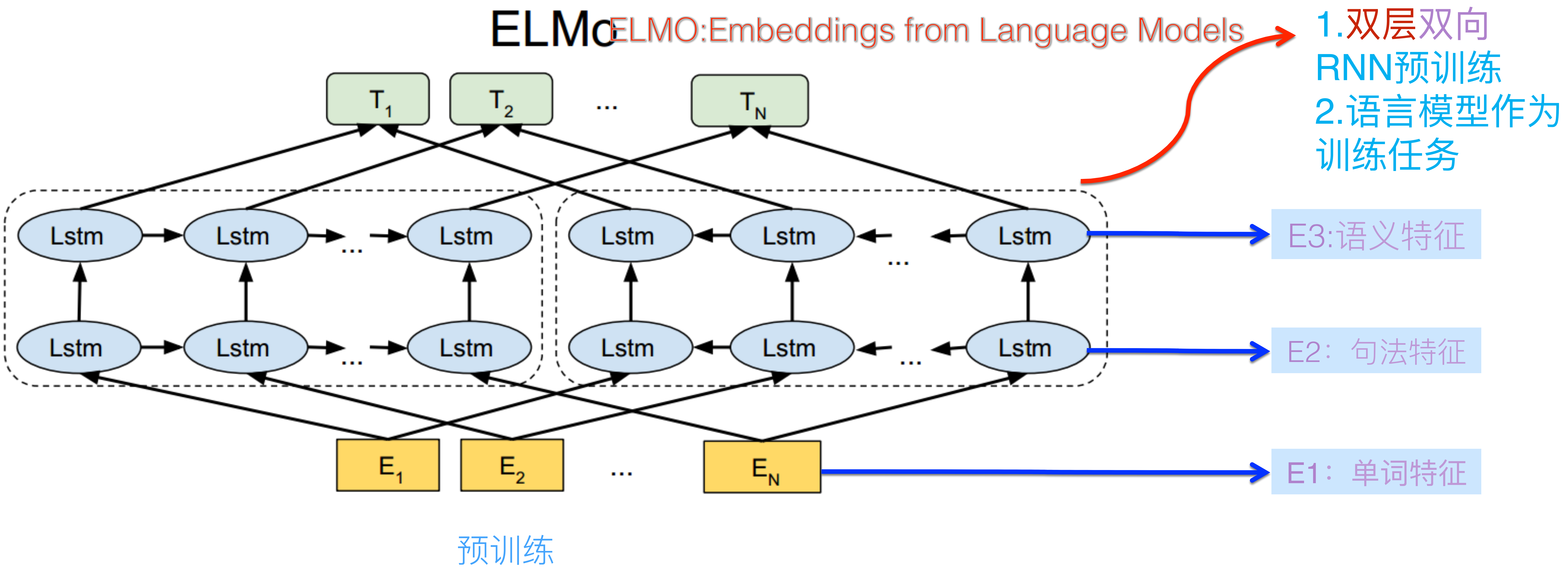
(多义词)Bank:
1.河岸
2.银行



静态的Word Embedding!

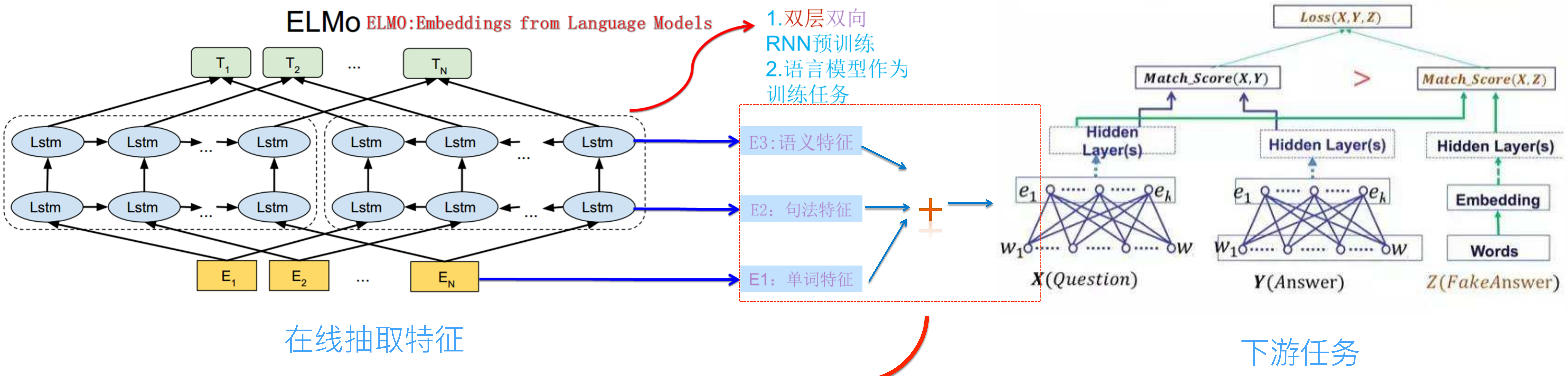
TABLE OF CONTENTES

从WE到ELMO：基于上下文的Embedding



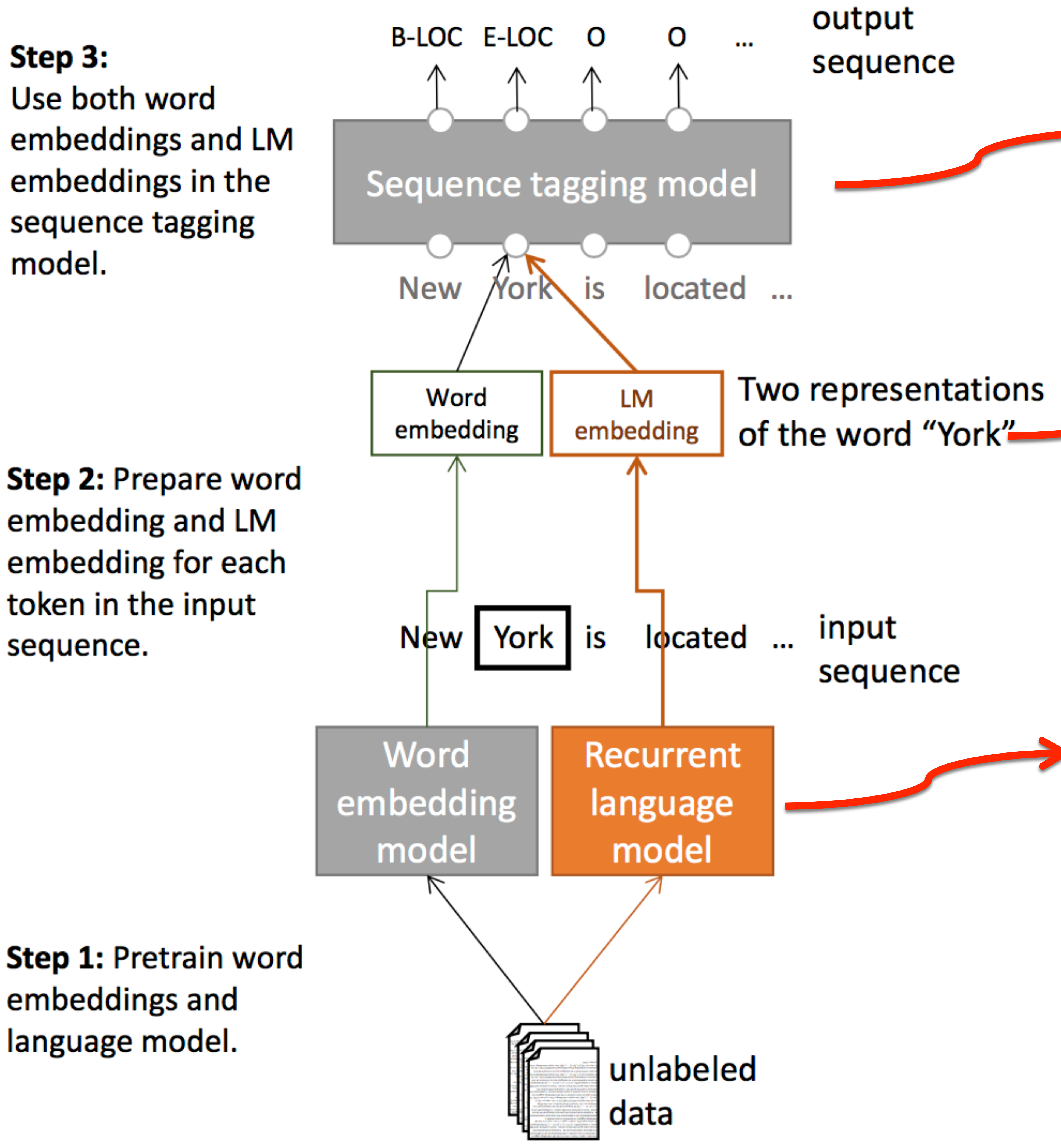
NAACL 2018 最佳论文：Deep contextualized word representations

ELMO：训练好之后如何使用？



Feature-based Pre Training

ELMO：训练好之后如何使用？



下游任务：特征集成

在线抽取特征

预训练

Feature-based Pre Training：实例

（TagLM，与ELMO思路类似，细节不同，提出稍早些，和ELMO是同一批作者）

ELMO：多义词问题解决了吗?

(多义词)Play:
1.运动
2.音乐

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular <u>play</u> 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 <u>play</u> .
	Olivia De Havilland signed to do a Broadway <u>play</u> for Garson {...}	{...} they were actors who had been handed fat roles in a successful <u>play</u> , 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.

不仅解决了，词性也能对应起来！

ELMO：效果如何?

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个NLP任务：5%到25%的提高!

ELMO：有什么缺点?

事后看（GPT和Bert出来之后对比）：

1.LSTM抽取特征能力远弱于Transformer

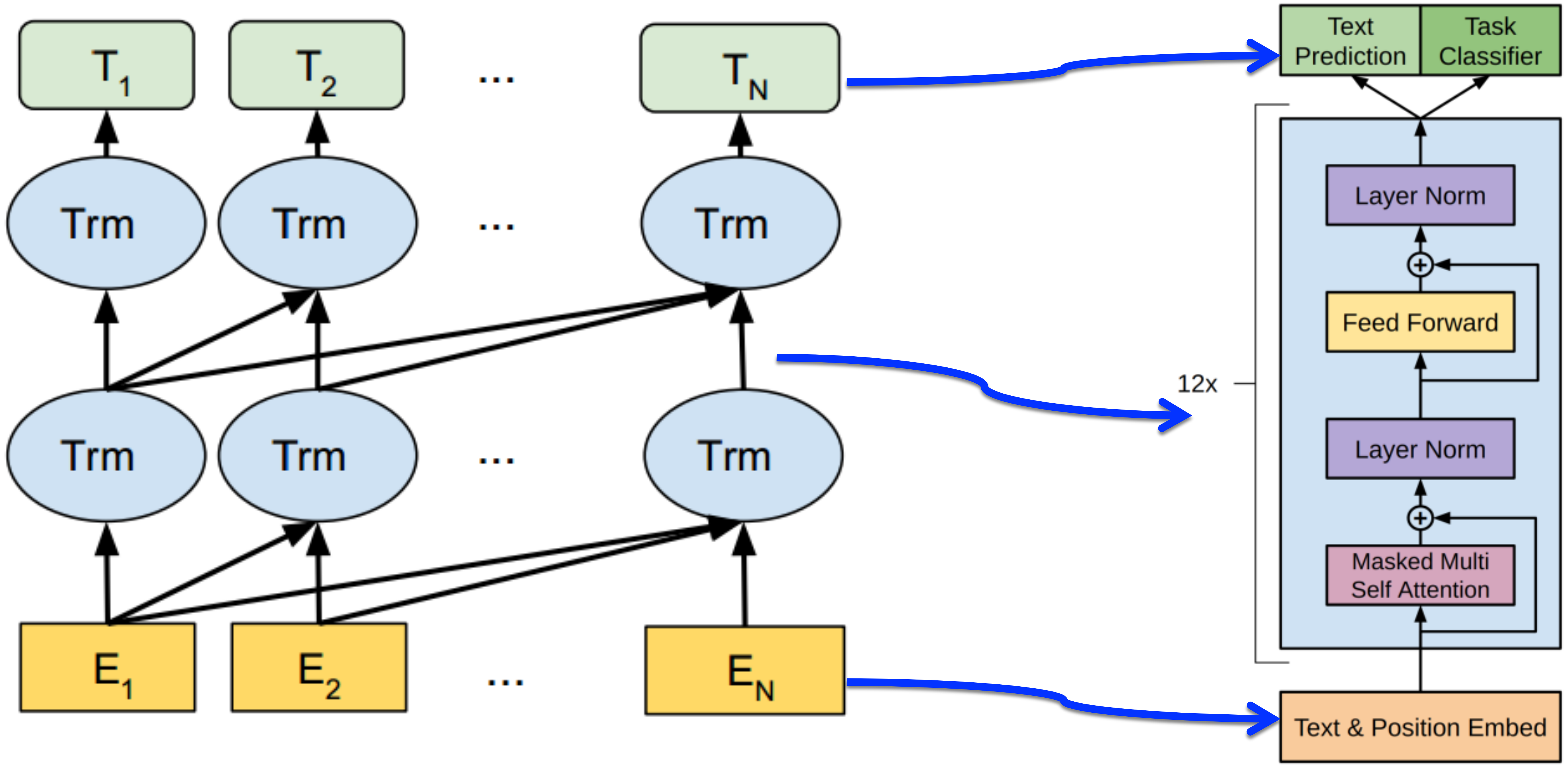
2.拼接方式双向融合特征融合能力偏弱

TABLE OF CONTENTES

从WE到GPT：Pretrain+Finetune两阶段过程

OpenAI GPT

1.Transformer特征抽取器
2.语言模型作为训练任务（单向）



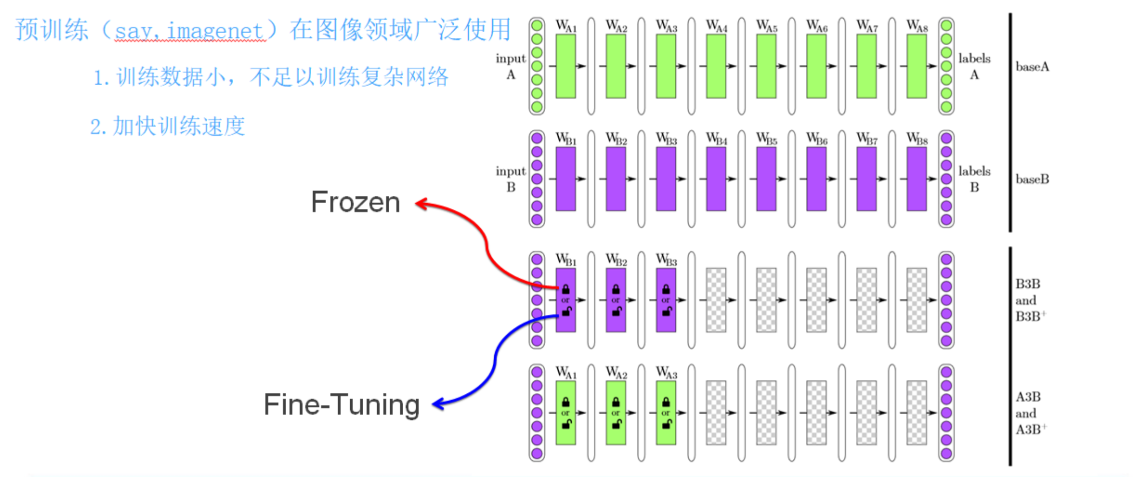
预训练

1.Transformer由Google2017年提出
2.本质上是个self attention叠加结构
3.目前是效果最好的特征抽取器，RNN在未来要被替代
4.优点：易于并行（RNN弱项）；捕获长距离特征能力强（Transformer>LSTM>CNN）;有深度有内涵

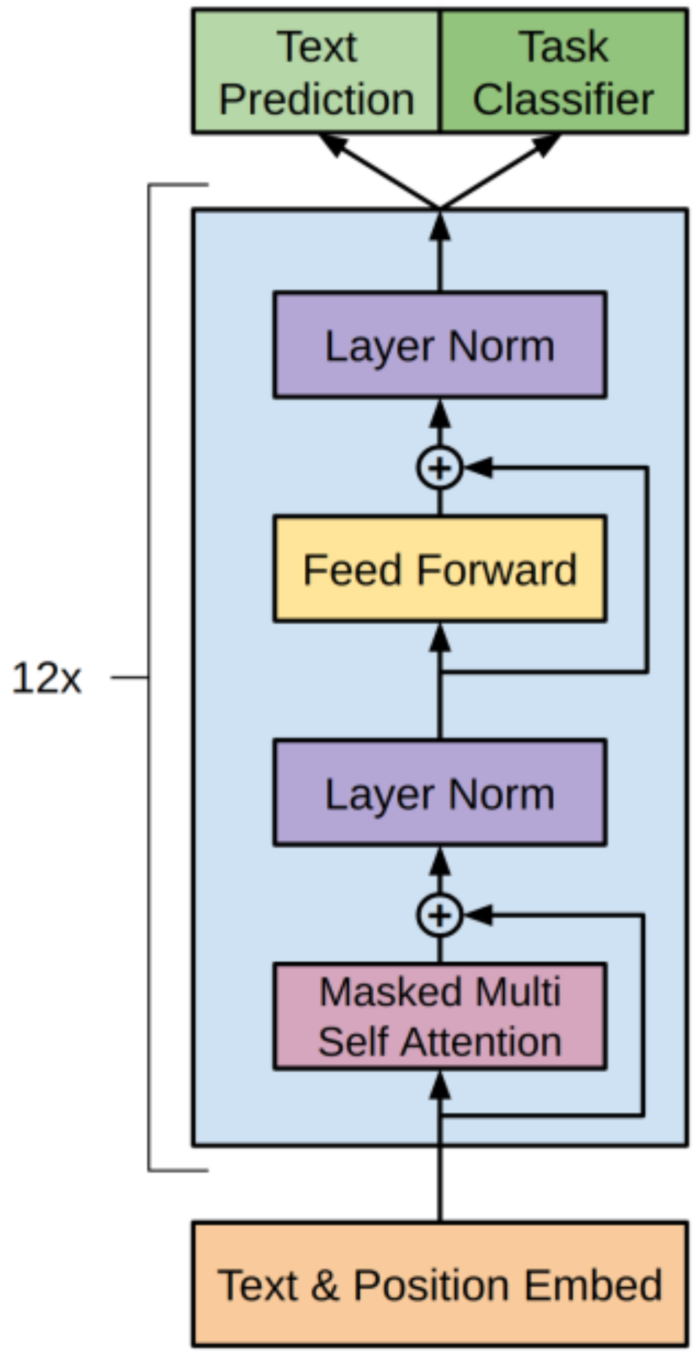
论文：Improving Language Understanding by Generative Pre-Training

GPT：训练好之后如何使用？

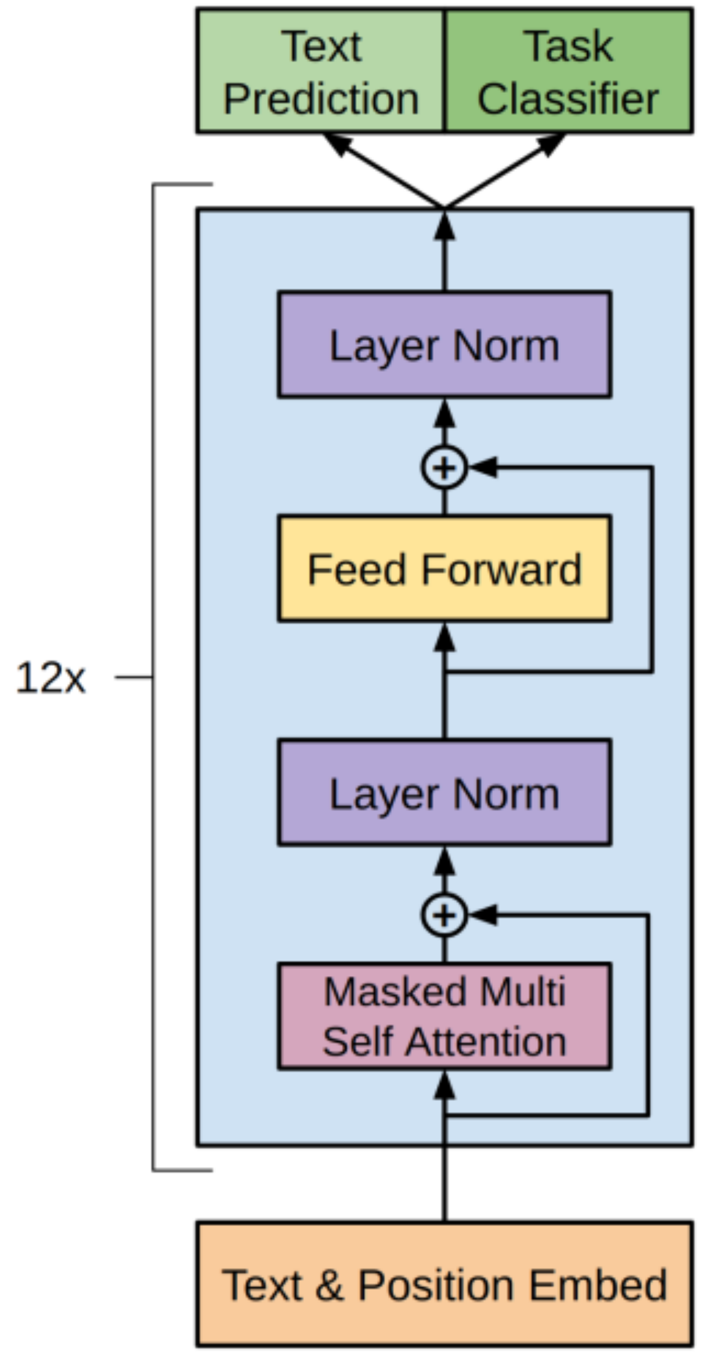
预训练在图像领域的应用



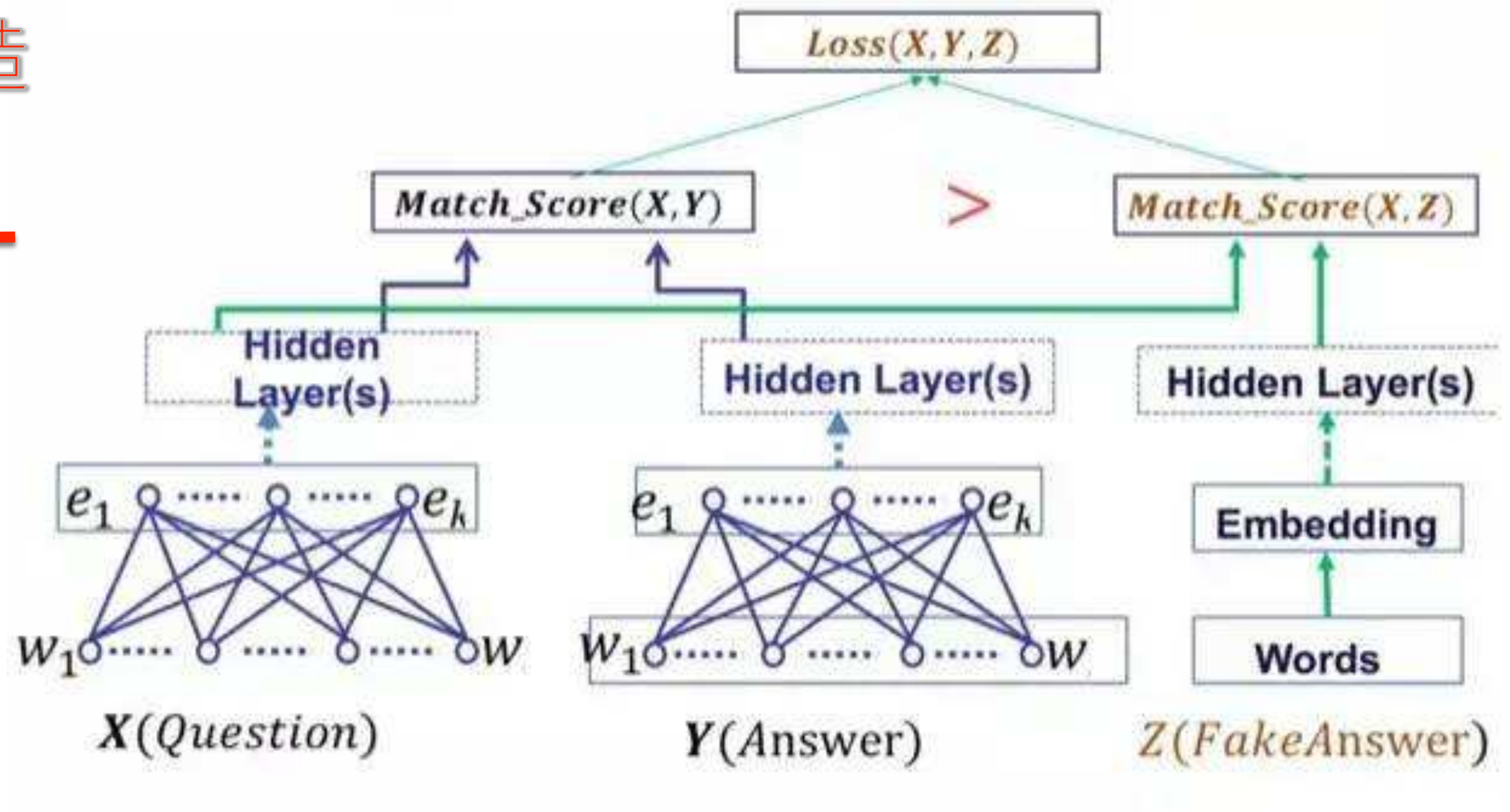
在这样的夜里
你有没有想起我



初始化参数



结构改造

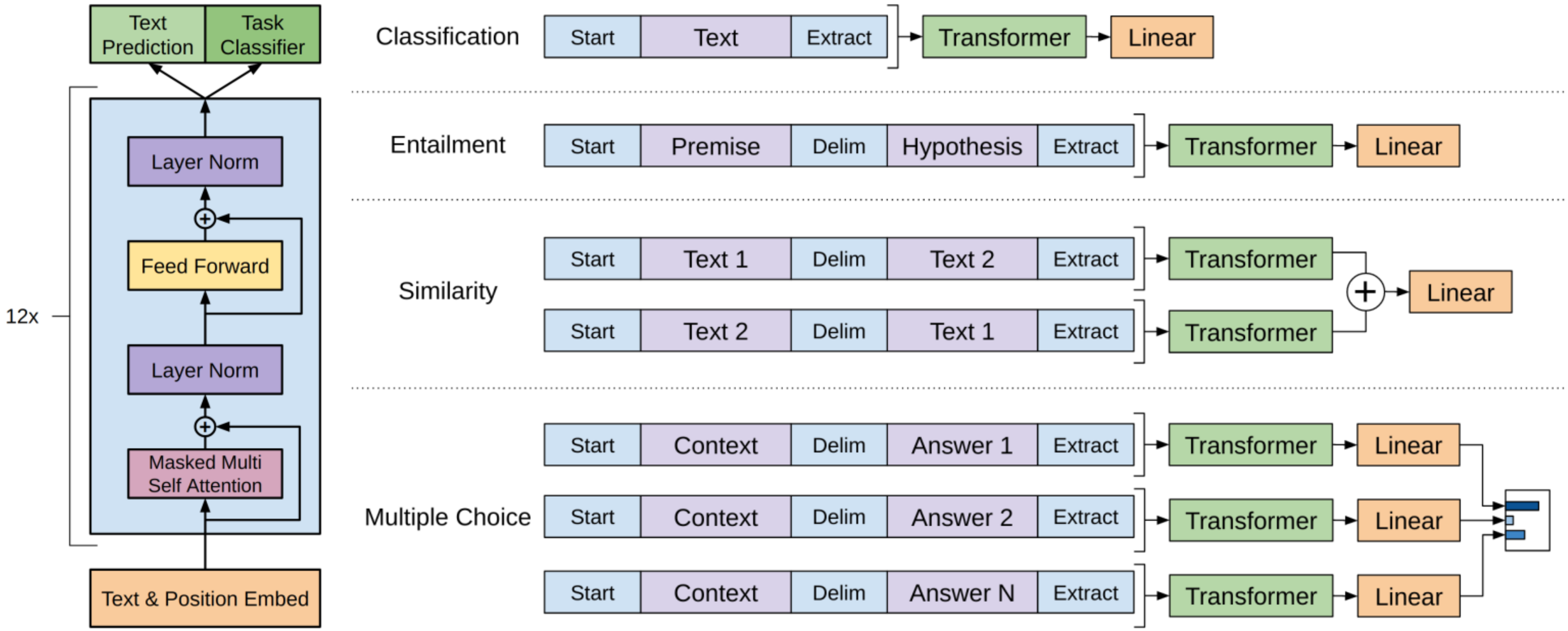


预训练

Fine-Tuning

下游任务

GPT：如何改造下游任务?



GPT：效果如何?

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**

12个NLP任务：9个达到最好效果！

GPT：有效因子分析

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特征抽取能力远强于LSTM

2.预训练至关重要

GPT：有什么缺点?

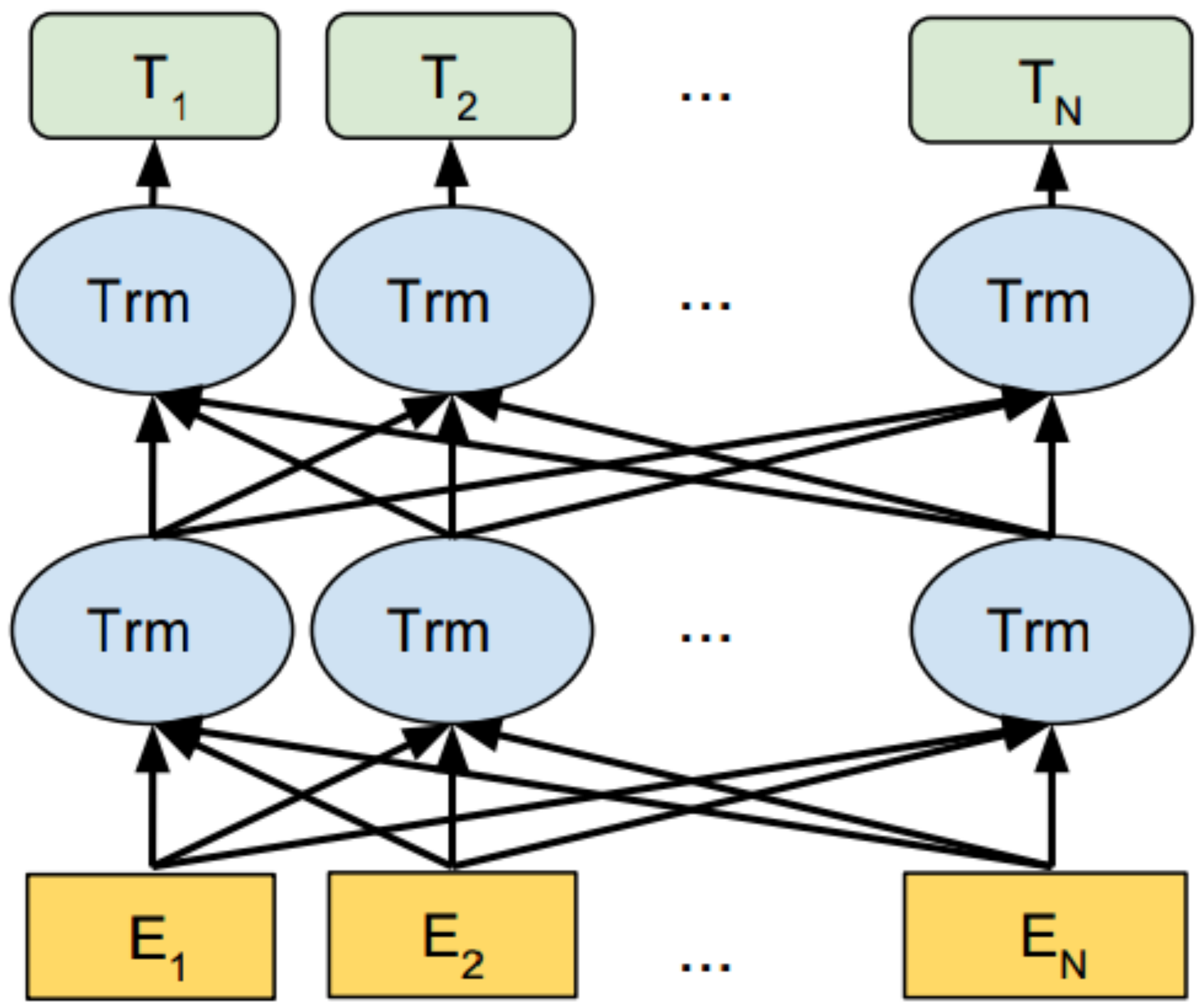
事后看（和Bert出来之后对比）：
1.要是把语言模型改造成双向的就好了
2.不太会炒作，GPT也是非常重要的工作.

TABLE OF CONTENTES

从GPT和ELMO及Word2Vec到Bert：新星的诞生

1.Transformer特征抽取器
2.语言模型作为训练任务（双向）

BERT (Ours)

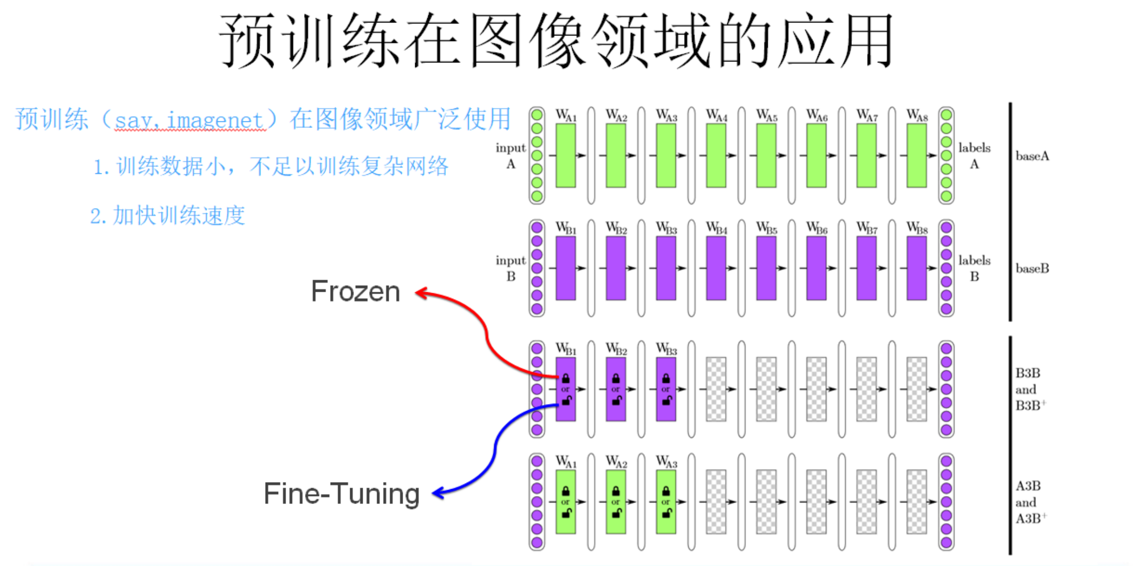


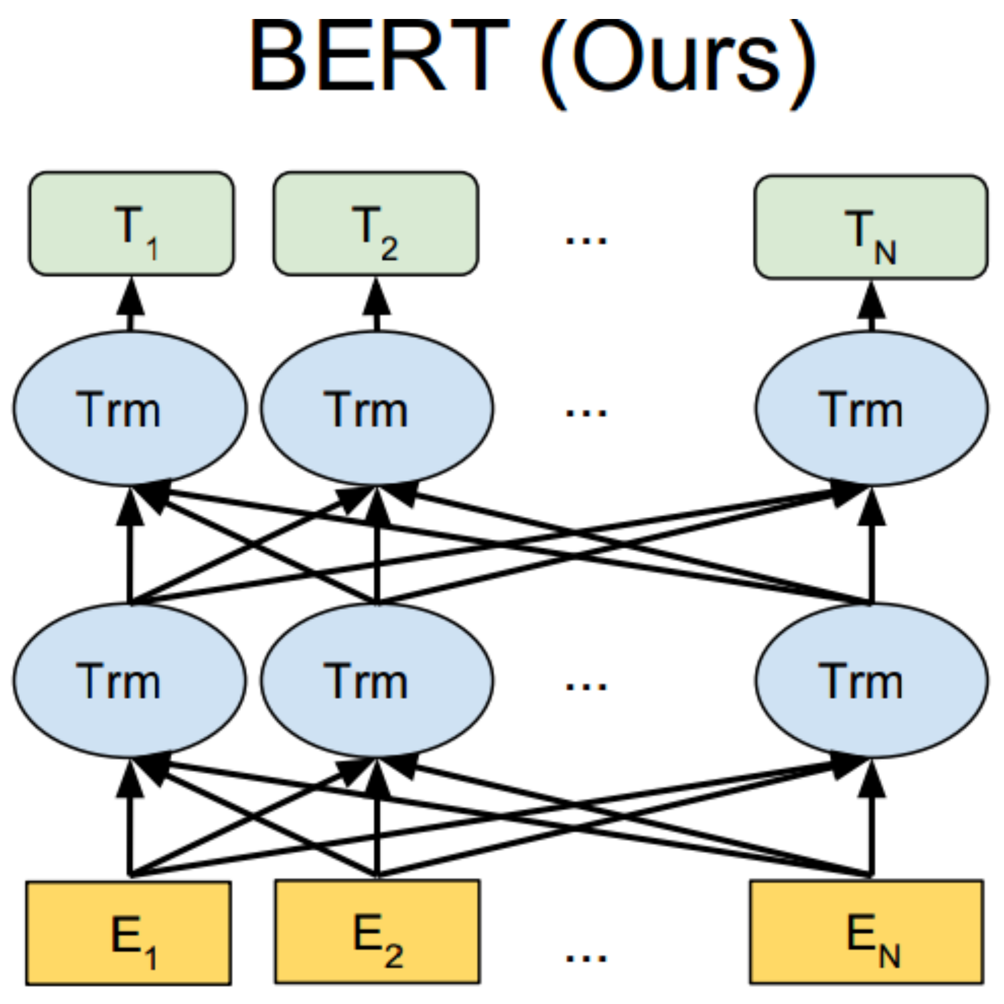
预训练

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

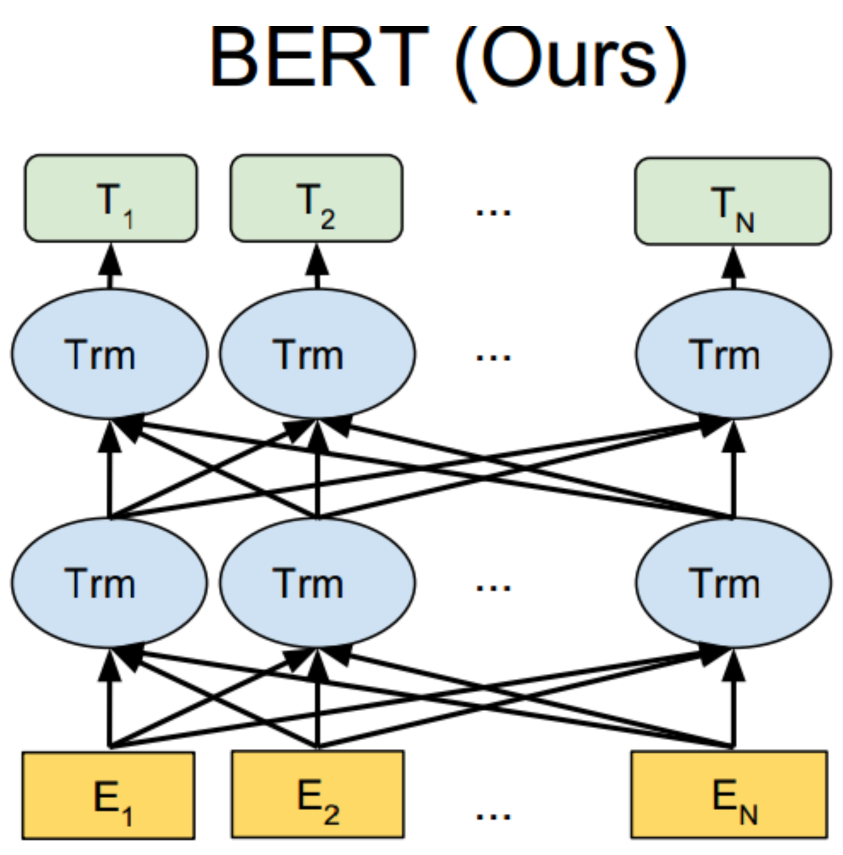
Bert：训练好之后如何使用？

在这样的夜里
你有没有想起我

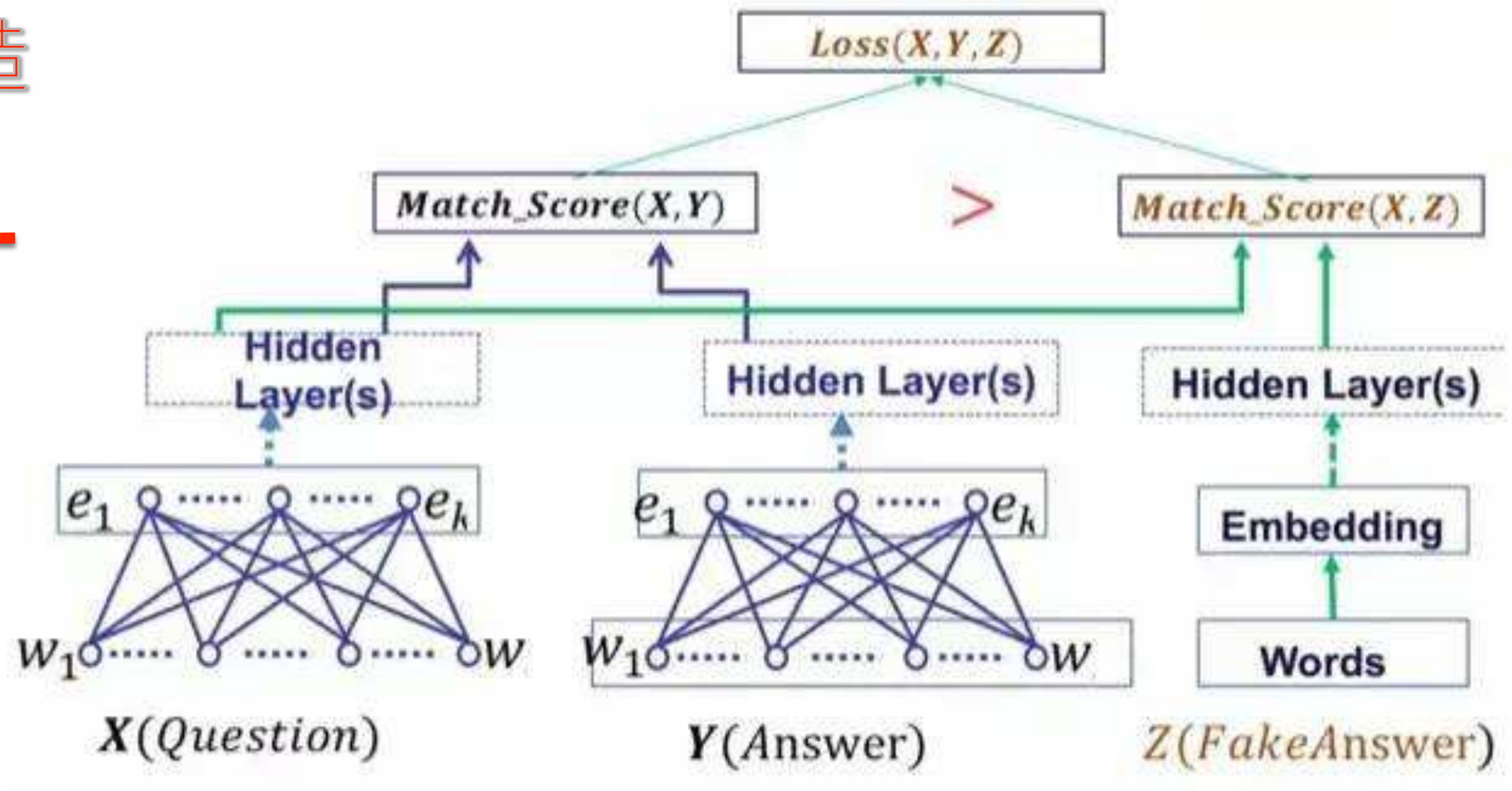




初始化参数



结构改造



预训练

Fine-Tuning

下游任务

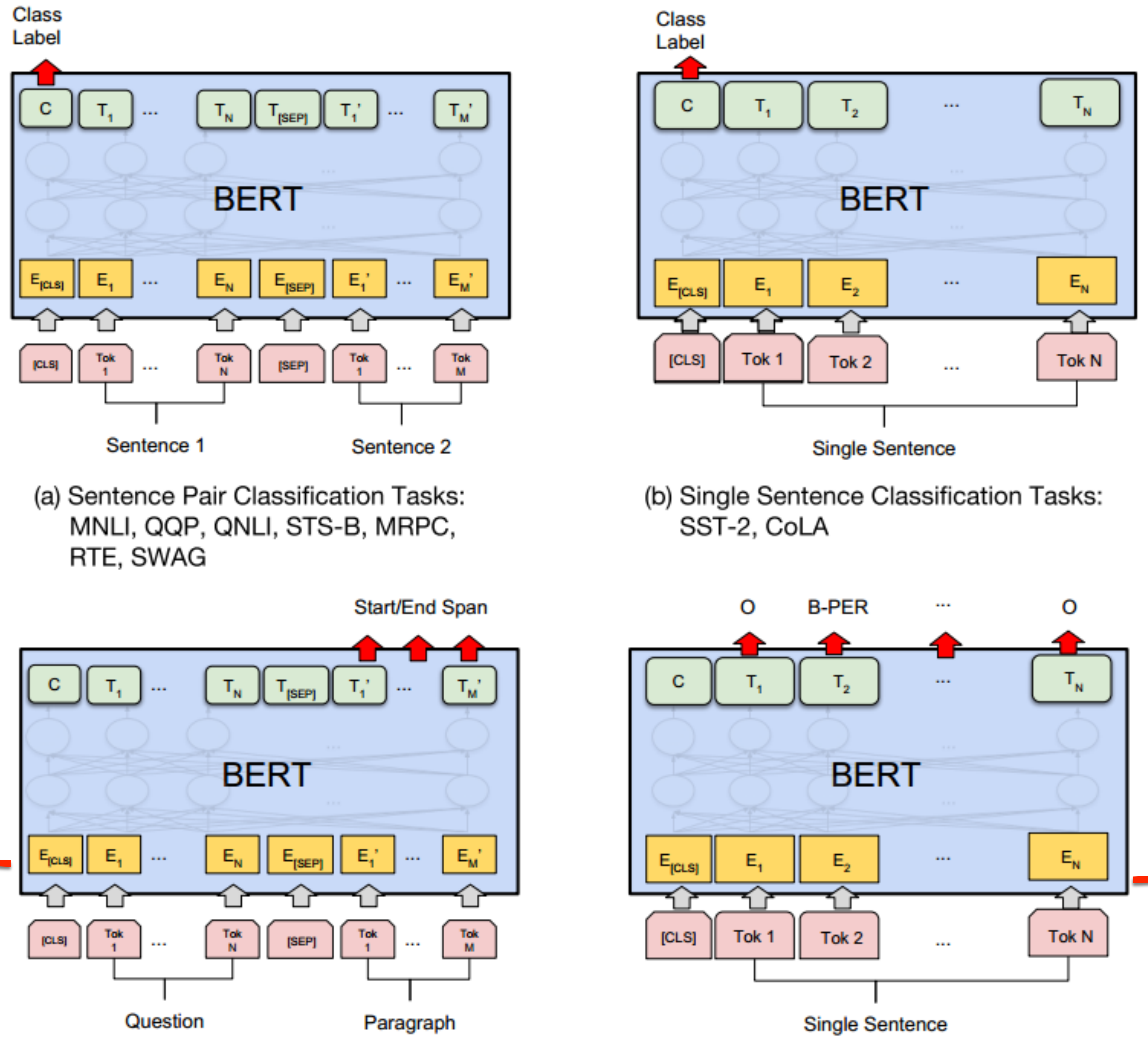
NLP的四大类任务

- 序列标注：分词/POS Tag/NER/语义标注.....
- 分类任务：文本分类 / 情感计算......
- 句子关系判断：Entailment/QA/自然语言推理....
- 生成式任务：机器翻译 / 文本摘要.....

Bert：如何改造下游任务?

句子关系类任务

单句分类任务



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

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

阅读理解任务

序列标注类任务

(c) Question Answering Tasks: SQuAD v1.1

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

Bert：效果如何?

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

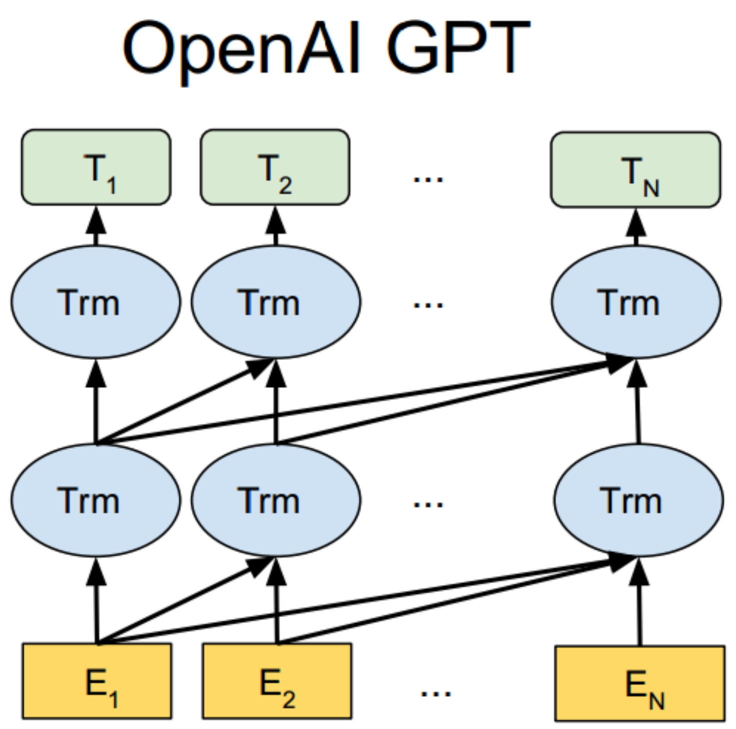
Table 4: SWAG Dev and Test accuracies. Test results were scored against the hidden labels by the SWAG authors. †Human performance is measure with 100 samples, as reported in the SWAG paper.

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.

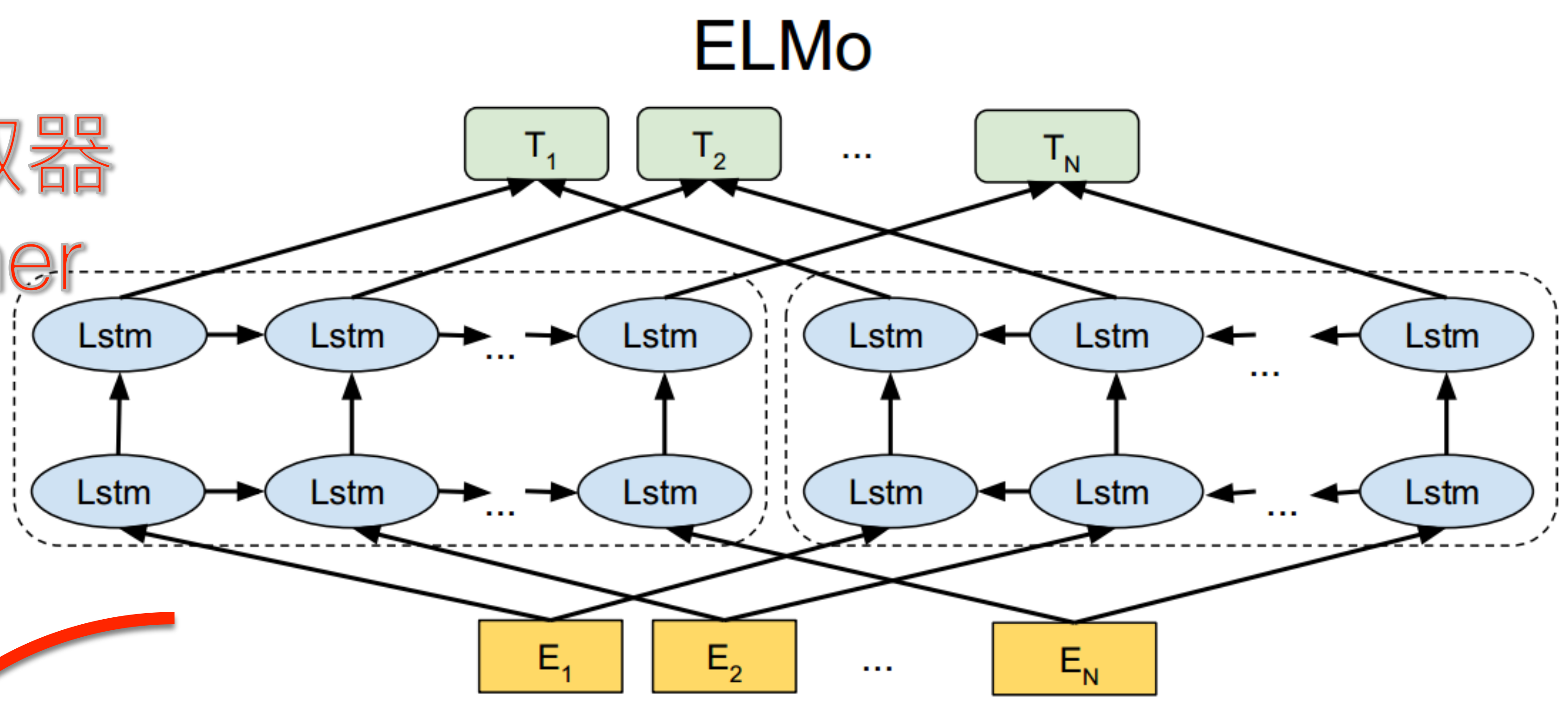
11个NLP任务：全面提升效果！

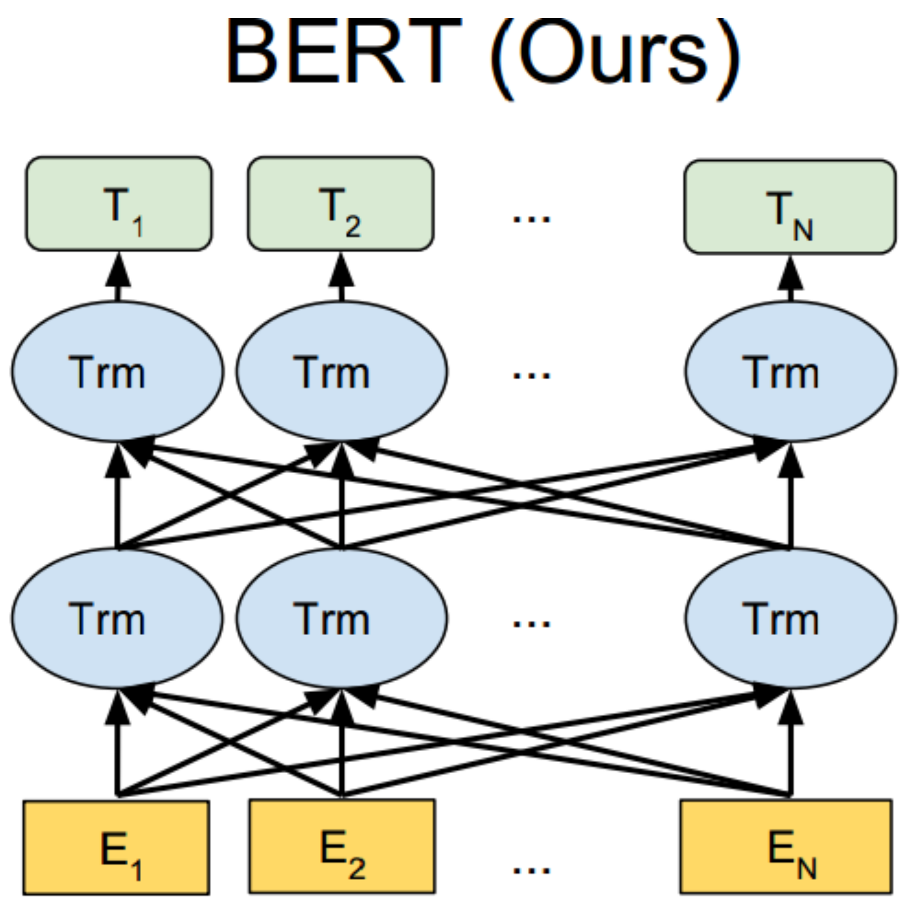
从GPT和ELMO及Word2Vec到Bert：四者的关系

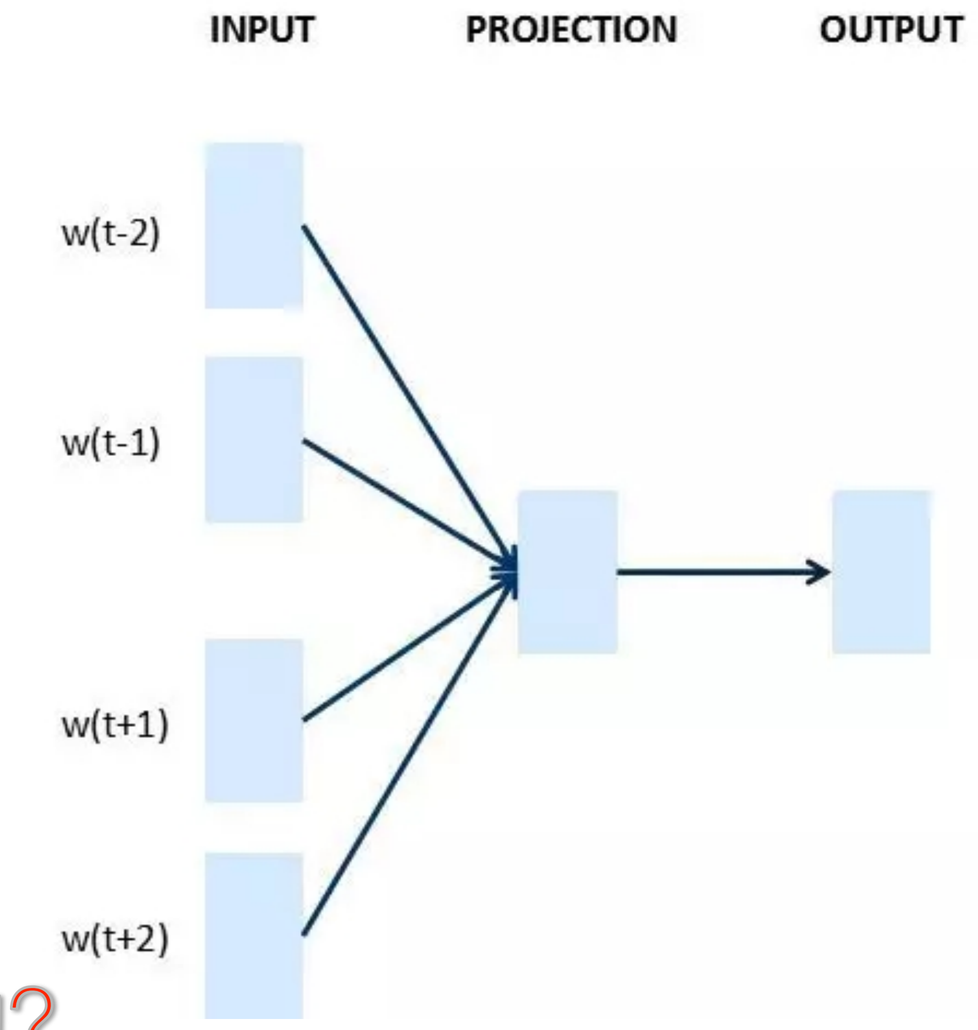


如果：语言模型改成双向的

如果：特征抽取器
换成Transformer



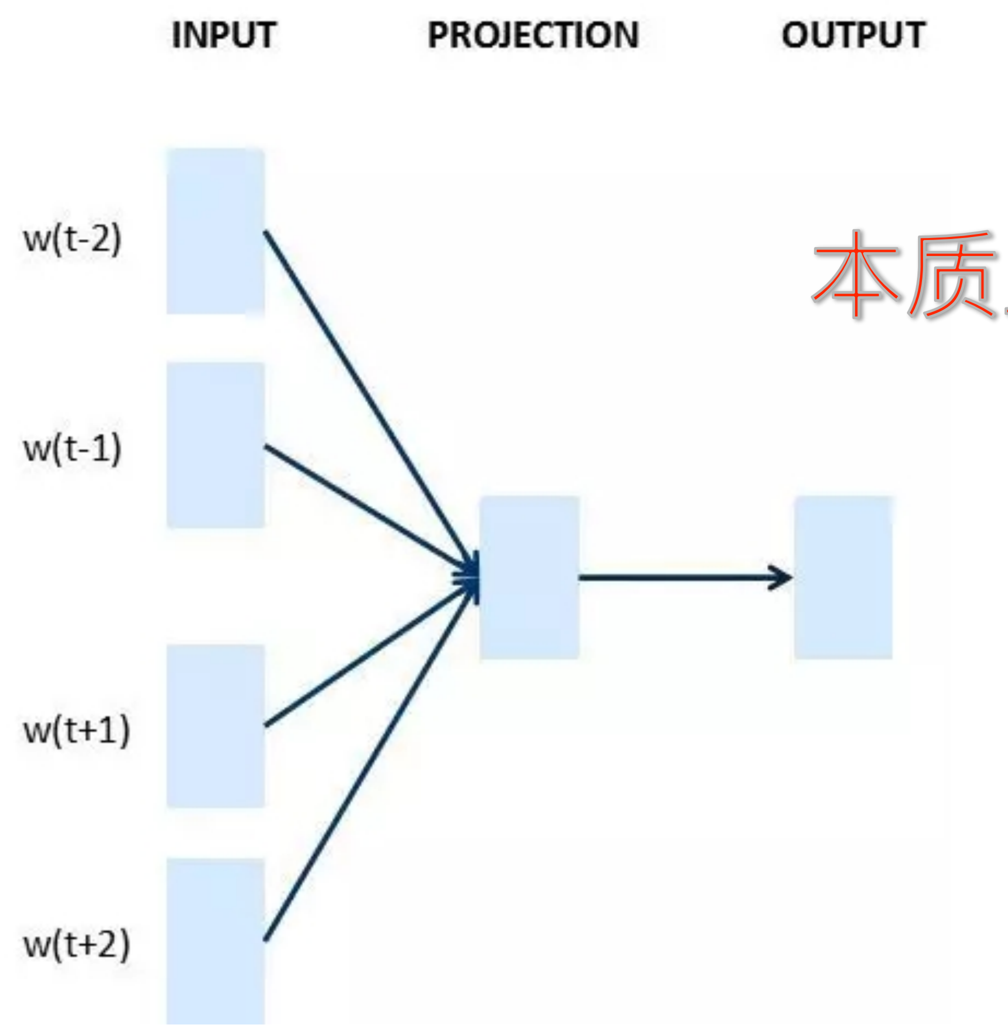




于是：怎么把语言模型改成双向的?

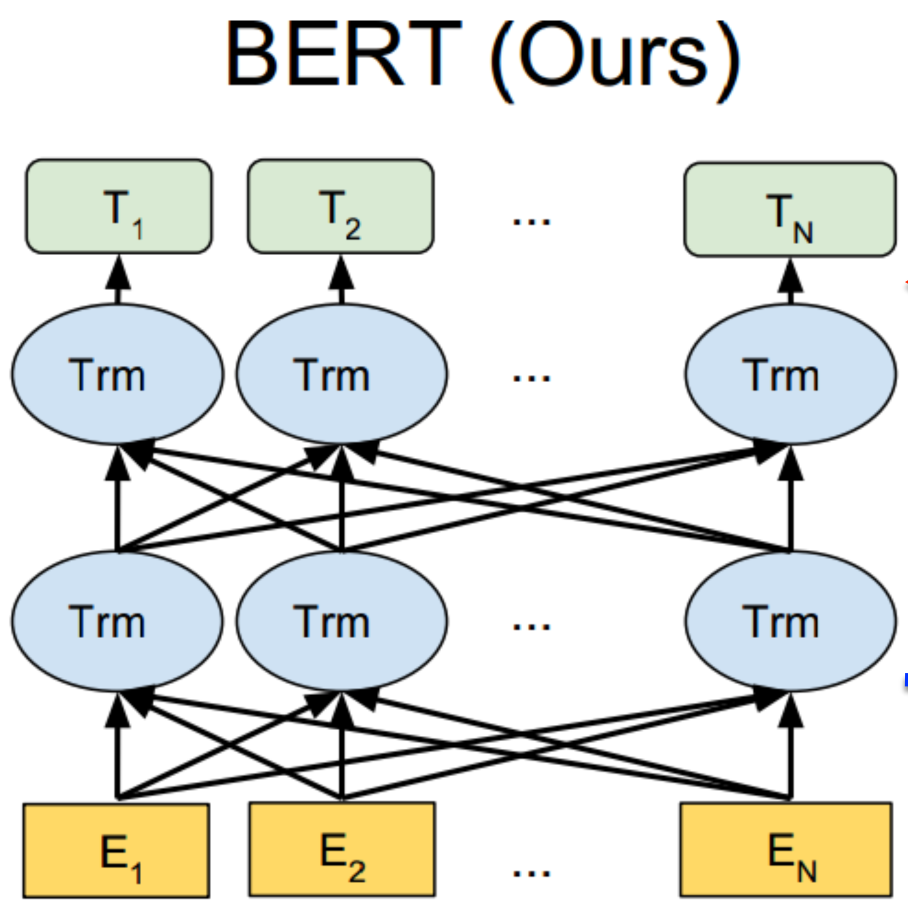
Word2Vec:CBOW

Bert：如何构造双向语言模型？

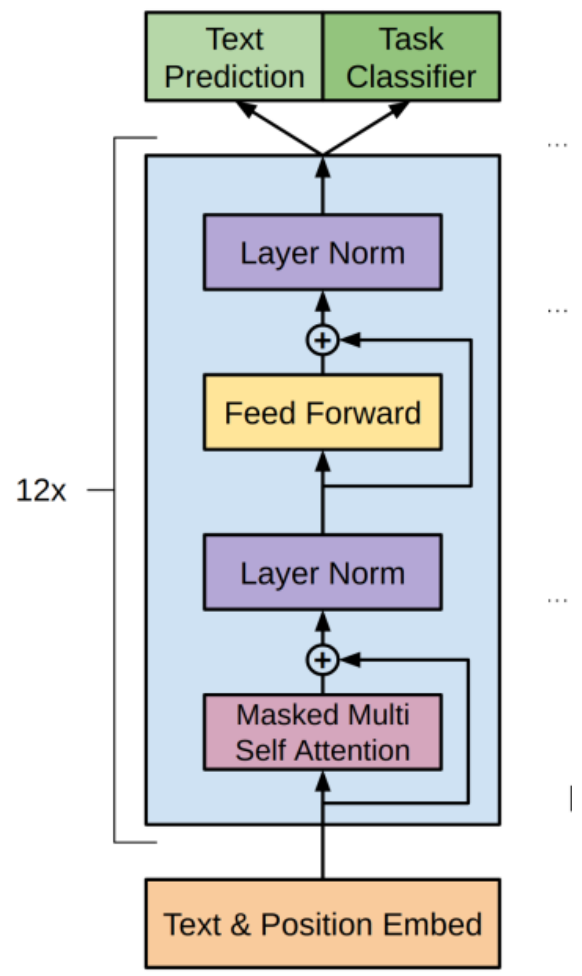


Word2Vec:CBOW

本质上是这个思想



但要这个网络结构



于是提出了：Masked LM

另外提出了：Next Sentence Prediction

Bert：如何构造双向语言模型?

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：如何构造双向语言模型?

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:

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`

Next Sentence Prediction

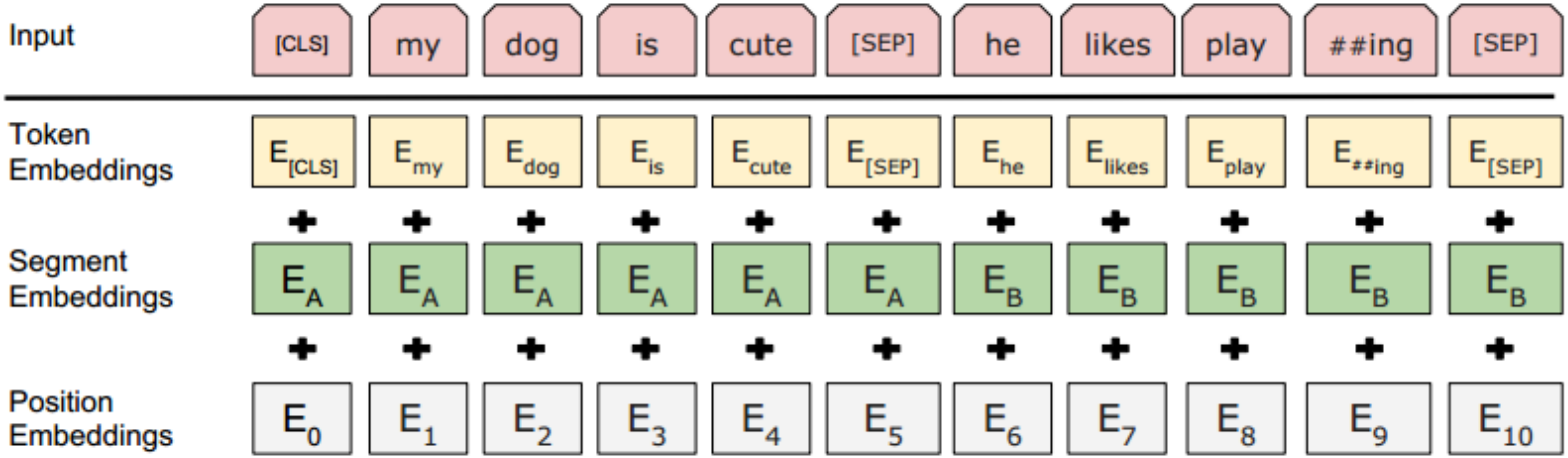
Bert：如何构造双向语言模型?

AB句子分隔符

```
INFO:tensorflow:*** Example ***
INFO:tensorflow:tokens: [CLS] [MASK] 夏 季 养 生 小 知 识 # 多 吃 瓜 类 、 多 吃 凉 性 蔬 菜 、 多 吃 “ 杀 菌 ” 蔬 菜 、 补 充 充 [MASK] 维 生 素 、 [MASK] [MASK] 水 和 >
无 机 硒 [MASK] 适 量 地 补 充 蛋 白 质 [SEP] 由 于 这 款 拍 [MASK] 能 让 使 用 者 万 一 跌 落 山 崖 [MASK] [MASK] [MASK] 的 伤 [MASK] 减 到 最 低 [MASK] 因 此 能 增 加 >
使 用 者 的 [MASK] 生 机 会 [SEP]
INFO:tensorflow:input_ids: 1 3 774 662 338 39 71 230 497 9 77 409 1157 530 17 77 409 1409 220 1979 1159 17 77 409 51 1017 1630 53 1979 1159 17 865 778 778 3 740 39 739 17
 3 3 179 50 219 128 2354 3 674 357 99 865 778 1122 386 439 2 464 122 44 453 560 3 66 265 427 87 206 312 12 1395 735 292 2867 3 3 3 7 718 3 942 63 107 752 3 259 394 66 535
 176 427 87 206 7 3 39 128 59 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
INFO:tensorflow:input_mask: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
INFO:tensorflow:segment_ids: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
INFO:tensorflow:masked_lm_positions: 1 3 34 39 40 45 46 60 61 73 74 75 78 83 93 0 0 0 0 0
INFO:tensorflow:masked_lm_ids: 9 662 783 865 778 1791 17 887 437 49 221 385 779 5 435 0 0 0 0 0
INFO:tensorflow:masked_lm_weights: 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0
INFO:tensorflow:next_sentence_labels: 1
```

Mask单词的位置信息，预训练输出时需要

Bert：输入部分的处理



Bert：输出部分的处理（预训练阶段）

```python
def get_masked_lm_output(bert_config, input_tensor, output_weights, positions,
                         label_ids, label_weights):
  """Get loss and log probs for the masked LM."""
  input_tensor = gather_indexes(input_tensor, positions)

  with tf.variable_scope("cls/predictions"):
    # We apply one more non-linear transformation before the output layer.
    # This matrix is not used after pre-training.
    with tf.variable_scope("transform"):
      input_tensor = tf.layers.dense(input_tensor,
          units=bert_config.hidden_size,
          activation=modeling.get_activation(bert_config.hidden_act),
          kernel_initializer=modeling.create_initializer(
              bert_config.initializer_range))
      input_tensor = modeling.layer_norm(input_tensor)

    # The output weights are the same as the input embeddings, but there is
    # an output-only bias for each token.
    output_bias = tf.get_variable("output_bias", shape=[bert_config.vocab_size],
        initializer=tf.zeros_initializer())
    logits = tf.matmul(input_tensor, output_weights, transpose_b=True)
    logits = tf.nn.bias_add(logits, output_bias)
    log_probs = tf.nn.log_softmax(logits, axis=-1)

    label_ids = tf.reshape(label_ids, [-1])
    label_weights = tf.reshape(label_weights, [-1])

    one_hot_labels = tf.one_hot(
        label_ids, depth=bert_config.vocab_size, dtype=tf.float32)

    # The `positions` tensor might be zero-padded (if the sequence is too
    # short to have the maximum number of predictions). The `label_weights`
    # tensor has a value of 1.0 for every real prediction and 0.0 for the
    # padding predictions.
    per_example_loss = -tf.reduce_sum(log_probs * one_hot_labels, axis=[-1])
    numerator = tf.reduce_sum(label_weights * per_example_loss)
    denominator = tf.reduce_sum(label_weights) + 1e-5
    loss = numerator / denominator

  return (loss, per_example_loss, log_probs)
```

0.第一个字符位置CLS对应Transformer输出分类结果

1. 获得Masked word 位置信息，掩码取出，不定长，最长20；

2.套上隐层

3.结果和词典对应单词矩阵乘法，其实是算e[masked]*e[dictionary word i]的内积，相似性

4.套上softmax预测最可能的单词

5.累加loss

Bert：有效因子分析

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.双向语言模型是最重要的

2.预测下个句子对个别任务有明显影响

BERT的评价及意义

- 由上述过程可知：Bert并未有重大模型创新，更像个最近两年NLP进展的集大成者
- 关键效果太好了，这将影响未来NLP的研究和应用模式
- 充分利用大量无监督NLP数据，将语言学知识隐含地引入特定任务中；
- 未来两年的NLP研究及应用模式可以预见如下
    - Transformer无处不在，逐步替代RNN和CNN；
    - 两阶段模型：超大规模预训练+具体任务FineTuning （两阶段模型结构相近）
    - or 超大规模预训练+具体任务特征补充（两阶段模型结构可以不同）

Thanks!