李理

环信人工智能研发中心

BERT 理论与实战

内容提要

Deep Learning 在 NLP 领域的发展

三个阶段：

- Word Embedding
  - Word2Vec
  - GloVe
- RNN 改进和扩展
  - LSTM/GRU
  - Seq2Seq
  - Attention/Self-Attention
- Contextual Word Embedding
  - ELMo
  - OpenAI GPT
  - BERT

Deep Learning 在 NLP 领域的发展

三个阶段：

- Word Embedding
  - Word2Vec
  - GloVe
- RNN 改进和扩展
  - LSTM/GRU
  - Seq2Seq
  - Attention/Self-Attention
- Contextual Word Embedding
  - ELMo
  - OpenAI GPT
  - BERT

Deep Learning 在 NLP 领域的发展

三个阶段：

- Word Embedding
  - Word2Vec
  - GloVe
- RNN 改进和扩展
  - LSTM/GRU
  - Seq2Seq
  - Attention/Self-Attention
- Contextual Word Embedding
  - ELMo
  - OpenAI GPT
  - BERT

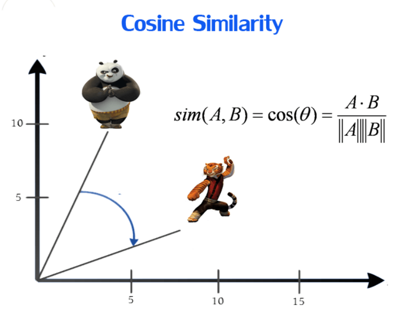
Deep Learning 在 NLP 领域的发展

三个阶段：

- Word Embedding
  - Word2Vec
  - GloVe
- RNN 改进和扩展
  - LSTM/GRU
  - Seq2Seq
  - Attention/Self-Attention
- Contextual Word Embedding
  - ELMo
  - OpenAI GPT
  - BERT

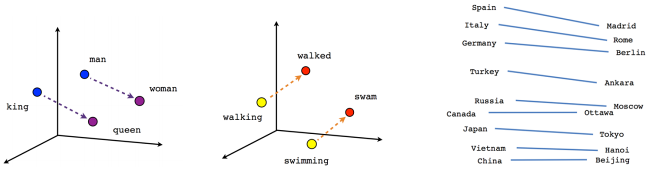
Word Embedding

把词映射为”语义”空间的点：



Word Embedding

效果：



Male-Female Verb tense Country-Capital

RNN/LSTM/GRU

语义是上下文相关的：

He deposited his money in this bank .

His soldiers were arrayed along the river bank .

RNN/LSTM/GRU

语义是上下文相关的：

He deposited his money in this bank .

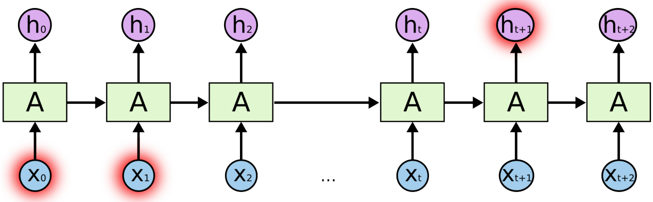
His soldiers were arrayed along the river bank .

RNN/LSTM/GRU

语义是上下文相关的：

He deposited his money in this bank .

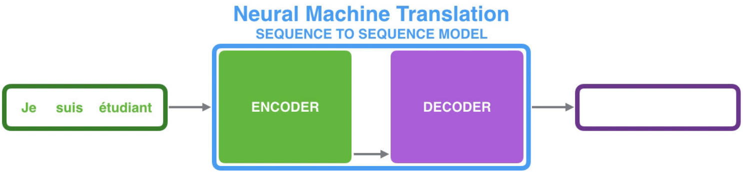
His soldiers were arrayed along the river bank .



Seq2Seq

Seq2Seq 由两个 RNN 组成

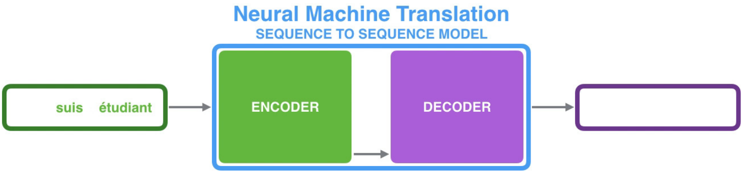
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

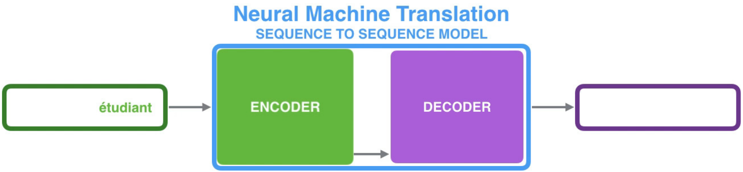
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

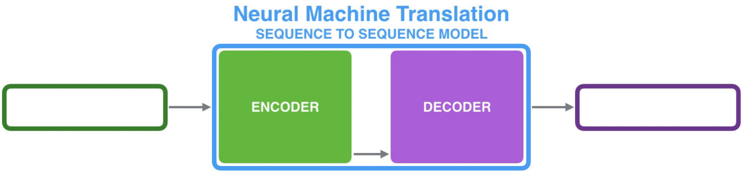
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

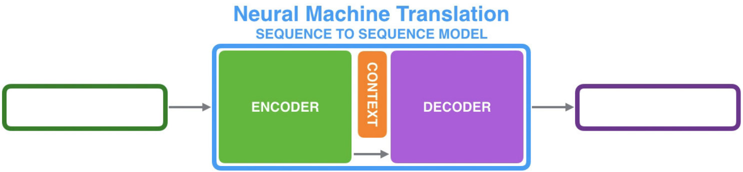
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

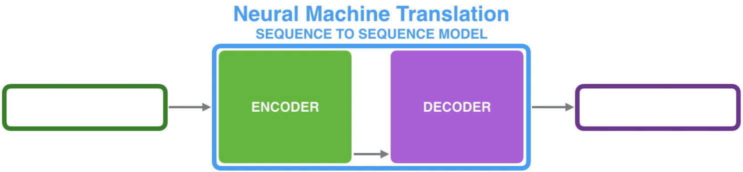
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

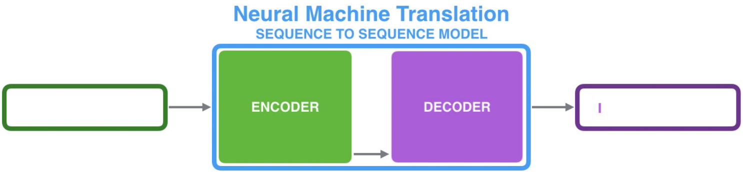
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

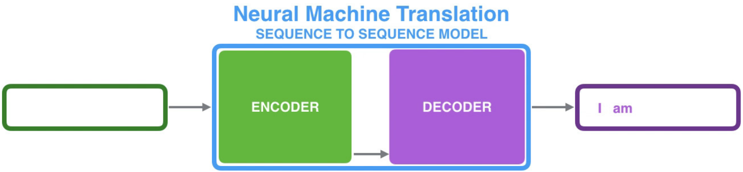
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

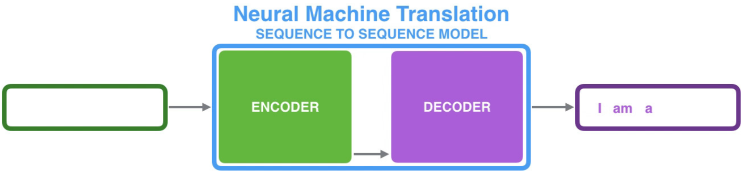
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

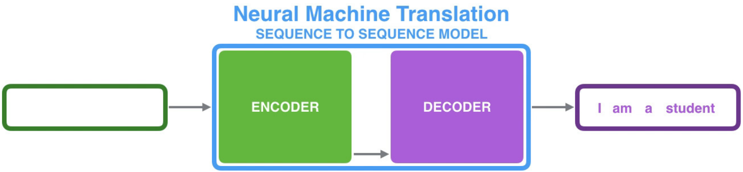
可用于翻译、摘要、问答和对话系统



Seq2Seq

Seq2Seq 由两个 RNN 组成

可用于翻译、摘要、问答和对话系统



Contexual Word Embedding

问题

监督数据量不足
难以学到复杂的上下文表示

解决方案

无 监 督的 Contextual Word Embedding

- ELMo
- OpenAI GPT
- BERT

Contexual Word Embedding

问题

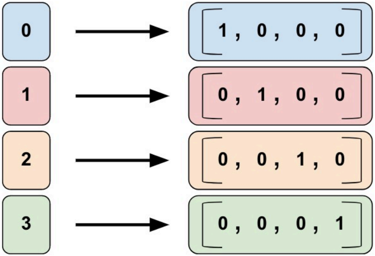
监督数据量不足
难以学到复杂的上下文表示

解决方案

无监督的 Contextual Word Embedding

- ELMo
- OpenAI GPT
- BERT

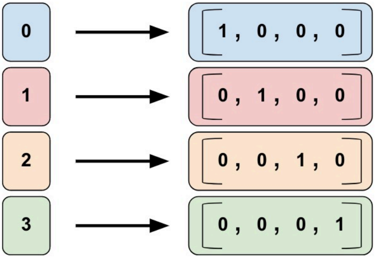
One-Hot Encoding



问题

- 高维
- 稀疏
- 正交

One-Hot Encoding



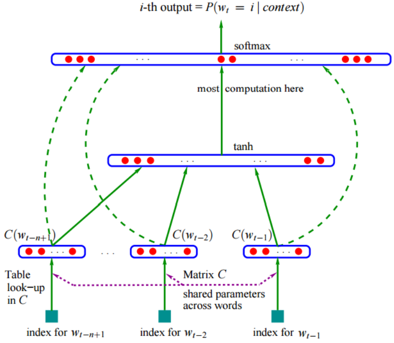
问题

- 高维
- 稀疏
- 正交

Neural Network Language Model

语言模型，预测句子概率：

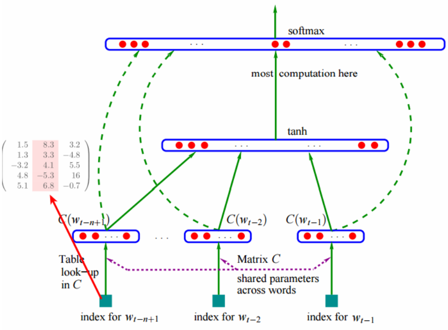
$$P(w) = P(w_1, ..., w_K) = \prod_{k=1}^{K} P(w_k | w_{k-1}, ..., w_1)$$



Neural Network Language Model

语言模型，预测句子概率：

$$P(w) = P(w_1, ..., w_K) = \prod_{k=1}^{K} P(w_k | w_{k-1}, ..., w_1)$$



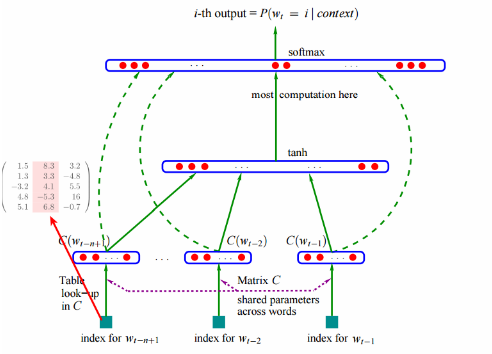
$$\begin{pmatrix} 1.5 & 8.3 & 3.2 \\ 1.3 & 3.3 & -4.8 \\ -3.2 & 4.1 & 5.5 \\ 4.8 & -5.3 & 16 \\ 5.1 & 6.8 & -0.7 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 8.3 \\ 3.3 \\ 4.1 \\ -5.3 \\ 6.8 \end{pmatrix}$$

- TensorFlow tf.nn.embedding_lookup
- PyTorch torch.nn.Embedding

Neural Network Language Model

语言模型，预测句子概率：

$$P(w) = P(w_1, ..., w_K) = \prod_{k=1}^{K} P(w_k | w_{k-1}, ..., w_1)$$



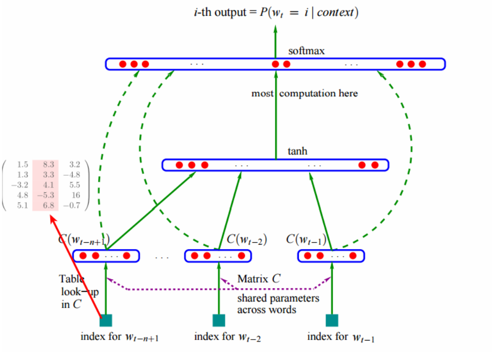
$$\begin{pmatrix} 1.5 & 8.3 & 3.2 \\ 1.3 & 3.3 & -4.8 \\ -3.2 & 4.1 & 5.5 \\ 4.8 & -5.3 & 16 \\ 5.1 & 6.8 & -0.7 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 8.3 \\ 3.3 \\ 4.1 \\ -5.3 \\ 6.8 \end{pmatrix}$$

- TensorFlow tf.nn.embedding_lookup
- PyTorch torch.nn.Embedding

Neural Network Language Model

语言模型，预测句子概率：

$$P(w) = P(w_1, ..., w_K) = \prod_{k=1}^{K} P(w_k | w_{k-1}, ..., w_1)$$



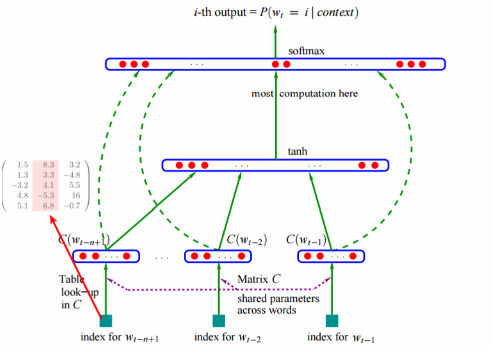
$$\begin{pmatrix} 1.5 & 8.3 & 3.2 \\ 1.3 & 3.3 & -4.8 \\ -3.2 & 4.1 & 5.5 \\ 4.8 & -5.3 & 16 \\ 5.1 & 6.8 & -0.7 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 8.3 \\ 3.3 \\ 4.1 \\ -5.3 \\ 6.8 \end{pmatrix}$$

- TensorFlow tf.nn.embedding_lookup
- PyTorch torch.nn.Embedding

Neural Network Language Model

语言模型，预测句子概率：

$$P(w) = P(w_1, \ldots, w_K) = \prod_{k=1}^{K} P(w_k | w_{k-1}, \ldots, w_1)$$



$$\begin{pmatrix} 1.5 & 8.3 & 3.2 \\ 1.3 & 3.3 & -4.8 \\ -3.2 & 4.1 & 5.5 \\ 4.8 & -5.3 & 16 \\ 5.1 & 6.8 & -0.7 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 8.3 \\ 3.3 \\ 4.1 \\ -5.3 \\ 6.8 \end{pmatrix}$$

- TensorFlow tf.nn.embedding_lookup
- PyTorch torch.nn.Embedding

Word2Vec

Distributional Hypothesis:

两个词上下文相似，则它们的语义也相似

Word2Vec

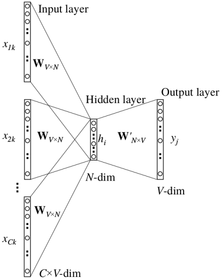
Distributional Hypothesis:

两个词上下文相似，则它们的语义也相似

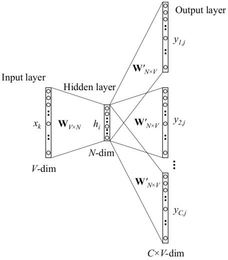
Figure: 词的上下文

Word2Vec

CBOW: Context 预测中心词

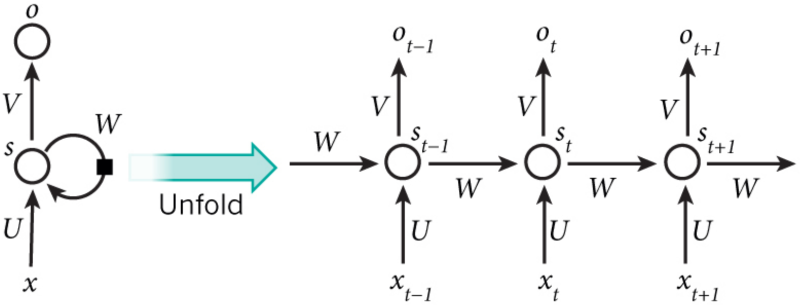


Skip-Gram: 中心词预测 Context



Vanilla RNN

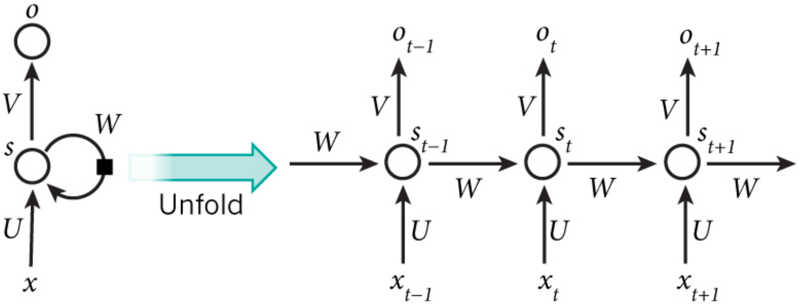
RNN 有” 记忆” 能力



$$s_t = f(Ux_t + Ws_{t-1})$$

Vanilla RNN

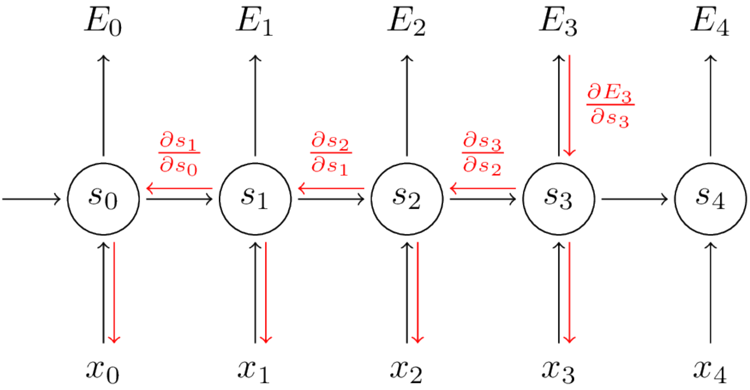
RNN 有" 记忆" 能力



$$s_t = f(Ux_t + Ws_{t-1})$$

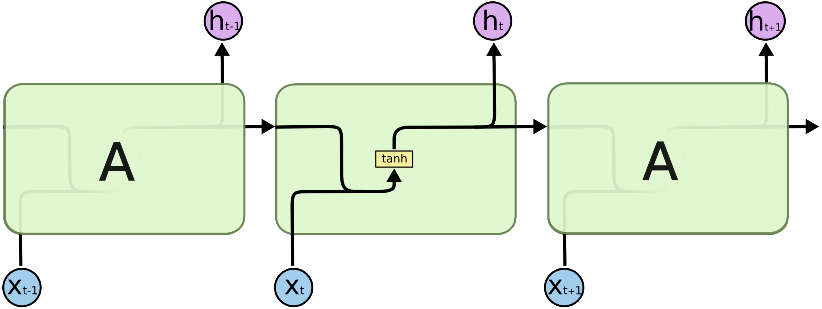
Vanilla RNN

t 时刻的 Loss 要往前传递：



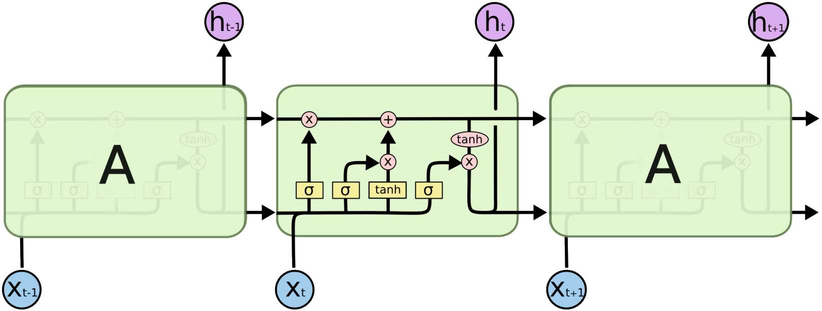
LSTM

LSTM 通过门的机制来避免梯度消失



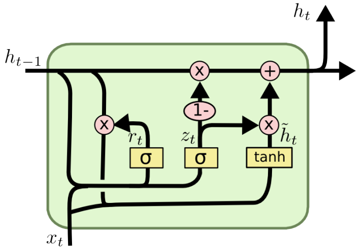
LSTM

LSTM 通过门的机制来避免梯度消失



GRU

GRU 把遗忘门和输入门合并成一个更新门



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

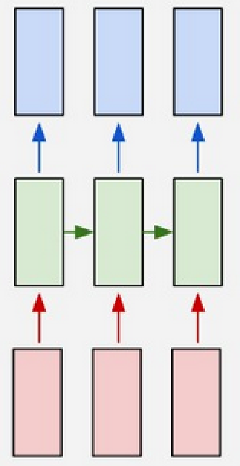
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

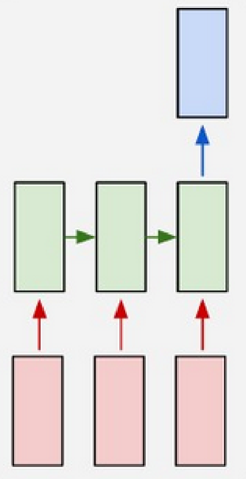
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

一个 RNN 的输出

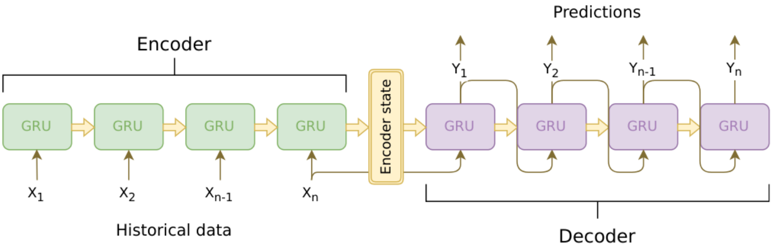
一个 RNN：





Seq2Seq

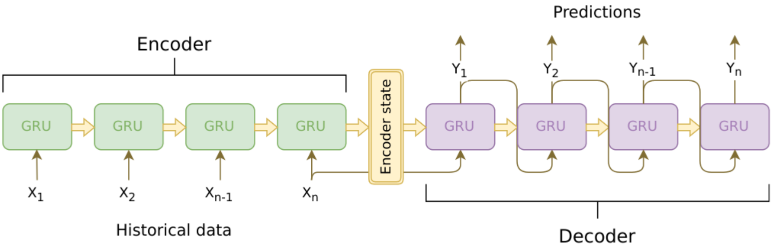
使用两个 RNN，Enocder 和 Decoder



问题：定长的 context 向量

Seq2Seq

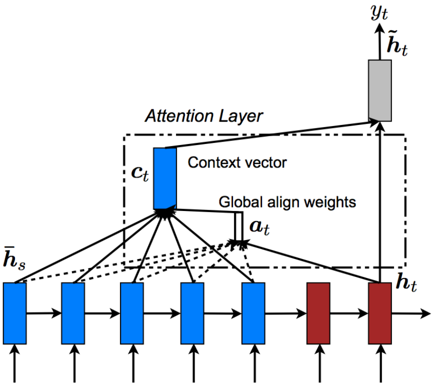
使用两个 RNN，Enocder 和 Decoder



问题：定长的 context 向量

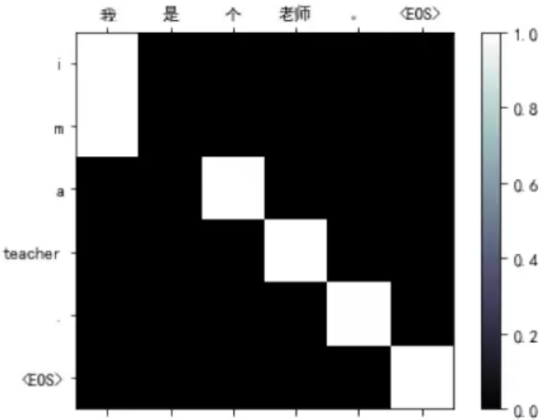
Attention 机制

翻译某个词时 Pay Attention to 相关词：



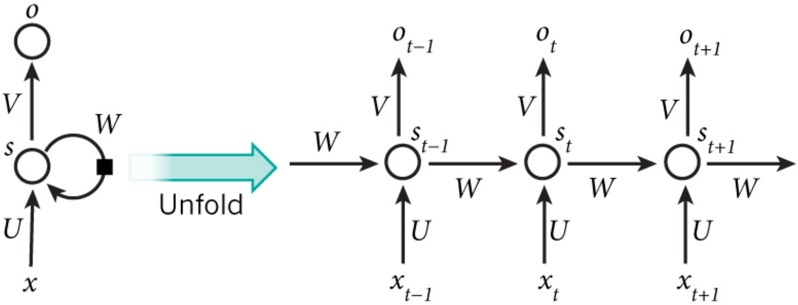
Attention 机制

Soft 对齐：



RNN 的问题

顺序依赖，无法并行。



RNN 的问题

The animal didn’t cross the street because it was too tired.

The animal didn’t cross the street because it was too narrow.

- The animal didn’t cross the street because it?
- it? was too tired.
- Attention 考虑整句，需要 Decoder
- Self-Attention

RNN 的问题

The animal didn’t cross the street because it was too tired.

The animal didn’t cross the street because it was too narrow.

- The animal didn’t cross the street because it?
- it? was too tired.
- Attention 考虑整句，需要 Decoder
- Self-Attention

RNN 的问题

The animal didn't cross the street because it was too tired.

The animal didn't cross the street because it was too narrow.

- The animal didn't cross the street because it?
- it? was too tired.
- Attention 考虑整句，需要 Decoder
- Self-Attention

RNN 的问题

The animal didn’t cross the street because it was too tired.

The animal didn’t cross the street because it was too narrow.

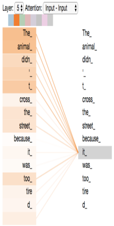
- The animal didn’t cross the street because it?
- it? was too tired.
- Attention 考虑整句，需要 Decoder
- Self-Attention

RNN 的问题

The animal didn’t cross the street because it was too tired.

The animal didn’t cross the street because it was too narrow.

- The animal didn’t cross the street because it?
- it? was too tired.
- Attention 考虑整句，需要 Decoder
- Self-Attention

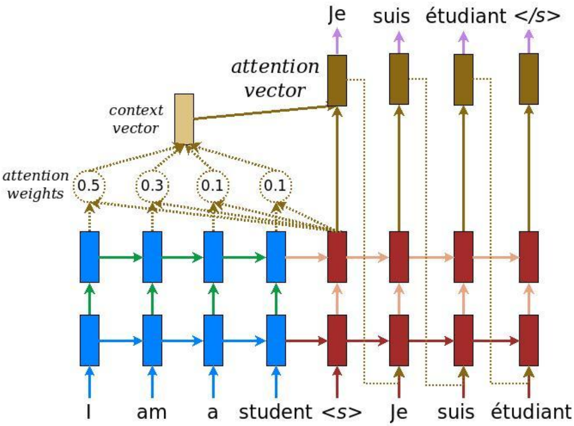


普通 Attention

普通的 Attention 需要外部的” 驱动”:

普通 Attention

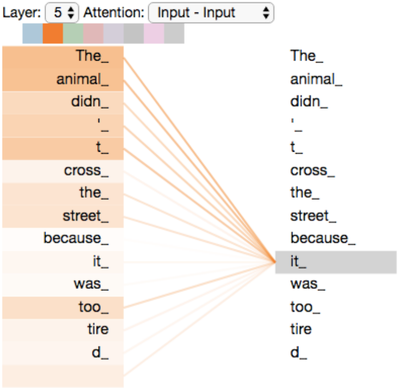
普通的 Attention 需要外部的” 驱动”:



Self-Attention 自驱动

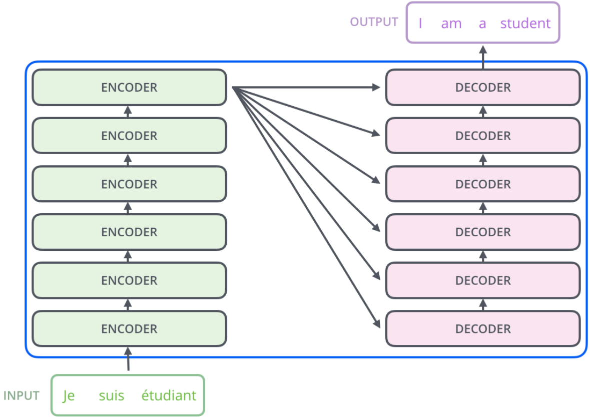
编码第 t 个词时

用当前状态去驱动：



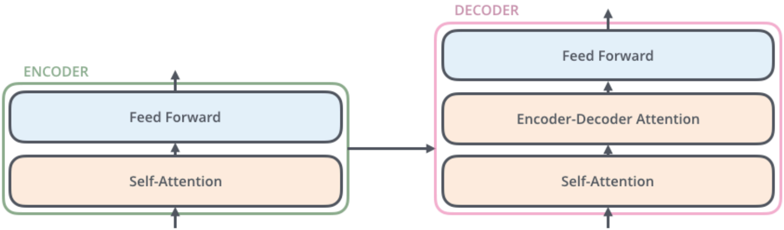
Transformer 结构

多层的 Encoder-Decoder



Transformer 结构

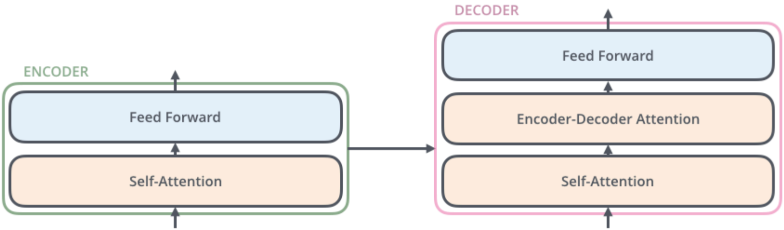
一层 Encoder 和 Decoder



Decoder 还有" 普通" 的 Attention 输入来自 Encoder

Transformer 结构

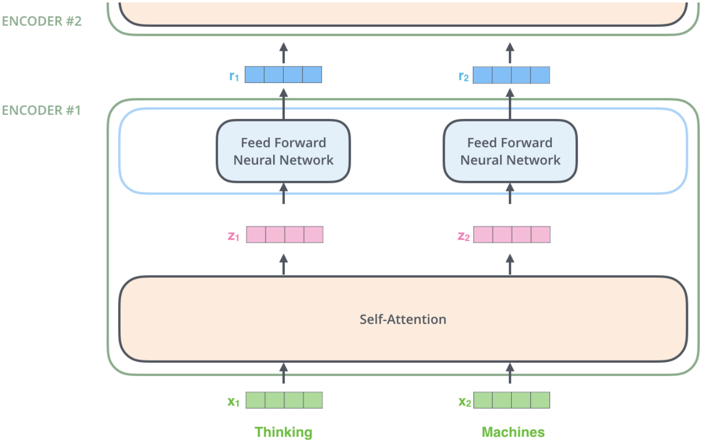
一层 Encoder 和 Decoder



Decoder 还有”普通”的 Attention 输入来自 Encoder

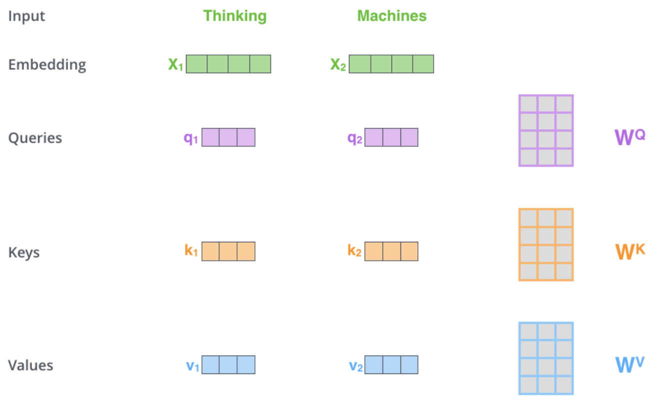
Transformer 结构

Encoder 详细结构，注意 Self-Attention 和 FNN 的区别



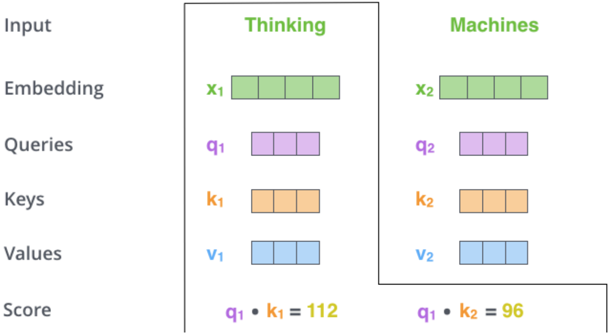
Self-Attention 计算

把每个词变换成三个向量 Q、K 和 V



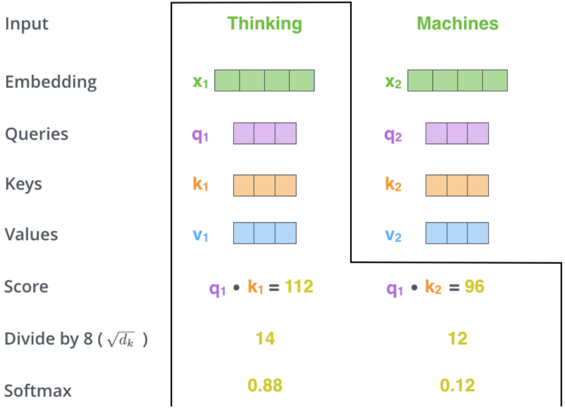
Self-Attention 计算

计算 q_1 和 k_1, k_2 的 score



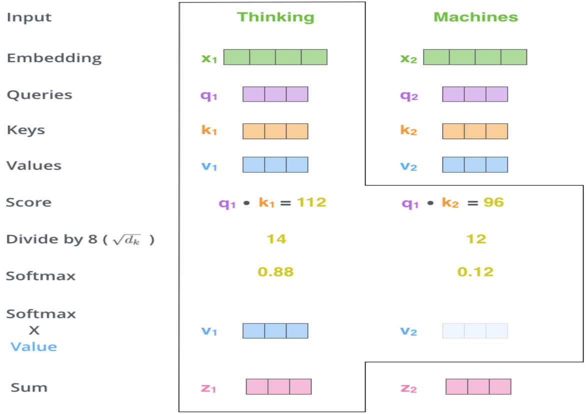
Self-Attention 计算

score 变成概率



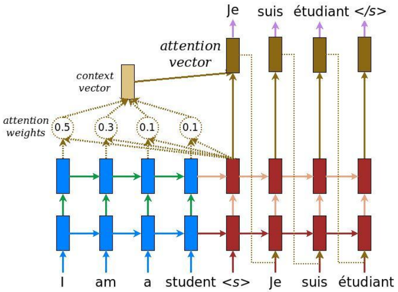
Self-Attention 计算

加权计算



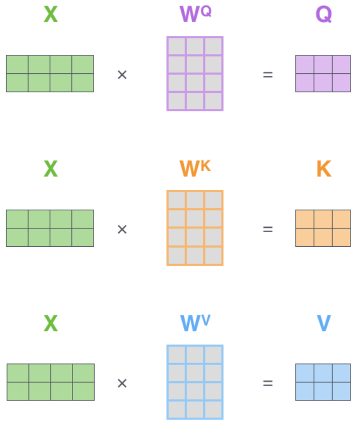
普通 Attention 的对比

- query 是 decoder 的隐状态
- key 是 encoder 的输出
- value 也是 encoder 的输出

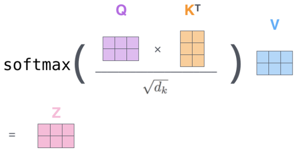


矩阵计算

一次计算所有的 Q、K 和 V

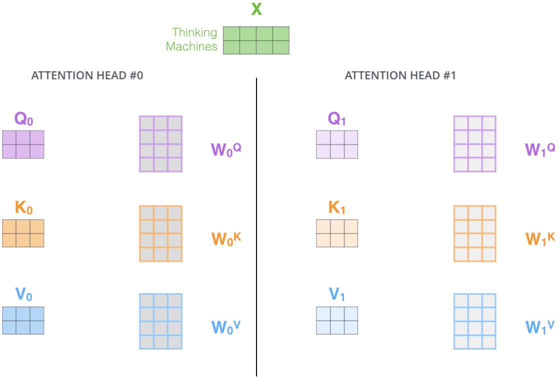


一次计算输出



Multi-Heads

多个 Attention(Q、K 和 V)

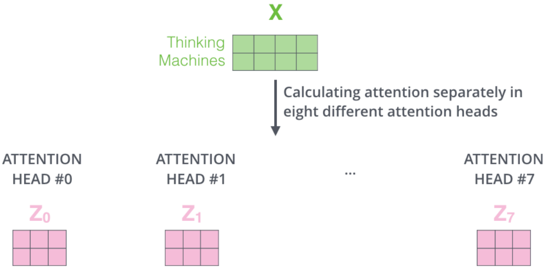


Mult-Heads

Multi-Heads 输出多个 z：

Mult-Heads

Multi-Heads 输出多个 z：



Mult-Heads

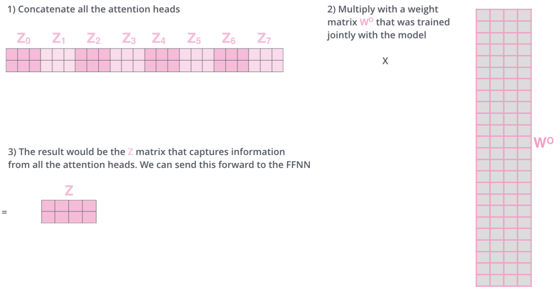
Multi-Heads 输出多个 z：

组合多个 z：

Mult-Heads

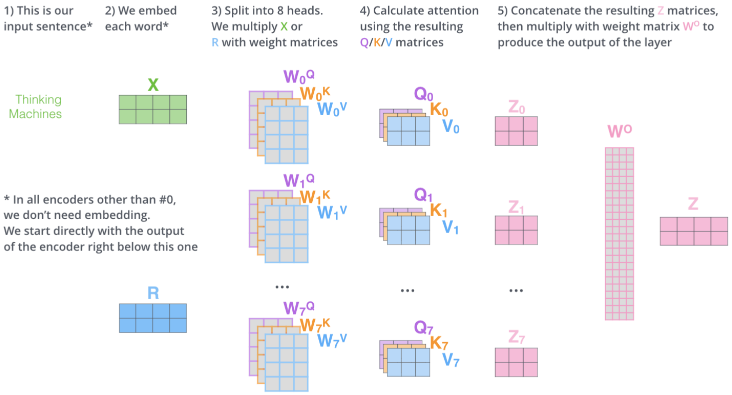
Multi-Heads 输出多个 z：

组合多个 z：



Multi-Heads

完整过程为：



位置编码

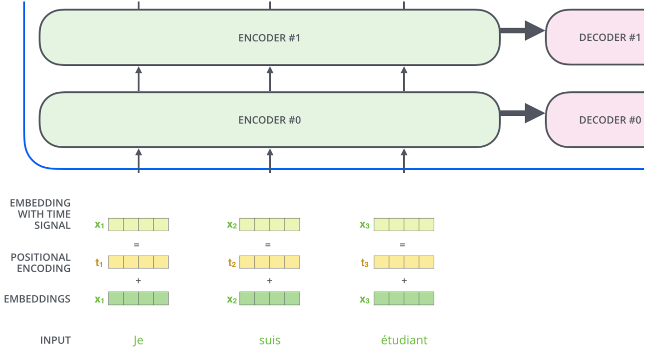
北京 到 上海 的机票

上海 到 北京 的机票

位置编码

北京 到 上海 的机票

上海 到 北京 的机票



位置编码

绝对位置编码，每个位置一个 Embedding

位置编码

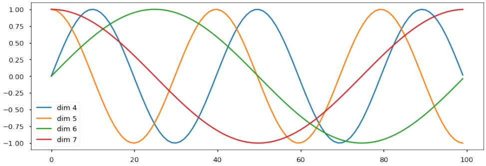
绝对位置编码，每个位置一个 Embedding

北京到上海的机票 vs 你好，我要北京到上海的机票

位置编码

绝对位置编码，每个位置一个 Embedding

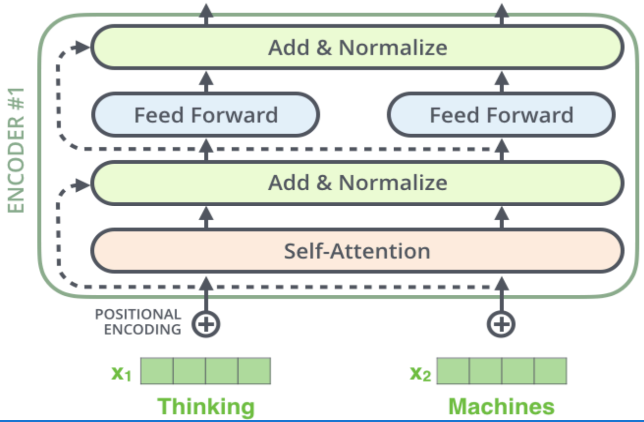
相对位置编码



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

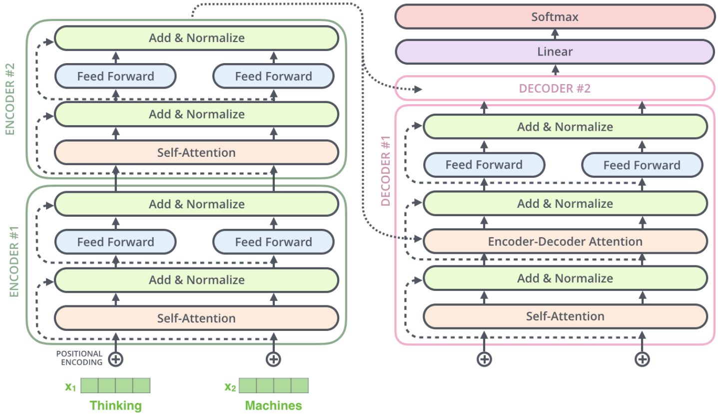
Encoder 完整结构

加上残差连接和 LayerNorm



Decoder 完整结构

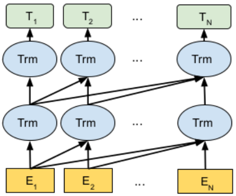
再加上普通 Attention



Decoder Mask

Decoder 不能利用未知信息
Mask Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$



Contextual Word Embedding

问题

- Word Embedding 无上下文
- 监督数据太少

解决方法

Contextual Word Embedding

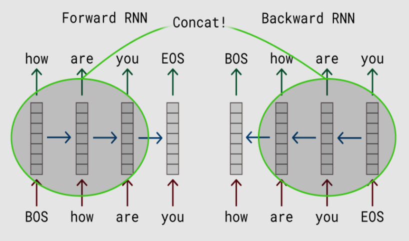
- 无监督
- 考虑上下文的 Embedding

ELMo


Hey ELMo, what's the embedding of the word "stick"?
There are multiple possible embeddings! Use it in a sentence.
Oh, okay. Here: "Let's stick to improvisation in this skit"
Oh in that case, the embedding is: -0.02, -0.16, 0.12, -0.1 ....etc

ELMo

多层双向的 LSTM 的 NNLM



$$ELMo_k^{task} = E(R_k; \Theta_{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{kj}^{LM}$$

OpenAI GPT

问题

- Contextual Word Embedding 作为特征
- 不适合特定任务

OpenAI GPT 的改进

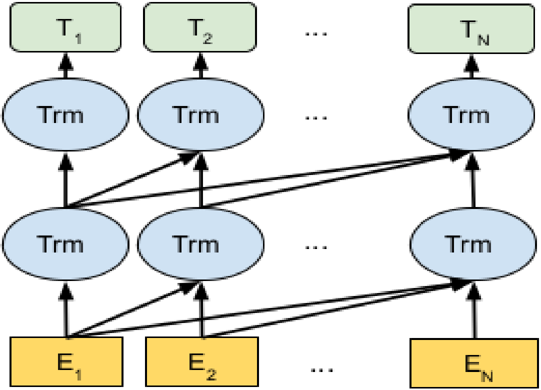
- 根据任务 Fine-Tuning
- 使用 Transformer 替代 RNN/LSTM

OpenAI GPT

没有 Encoder 的 Transformer？

OpenAI GPT

没有 Encoder 的 Transformer？

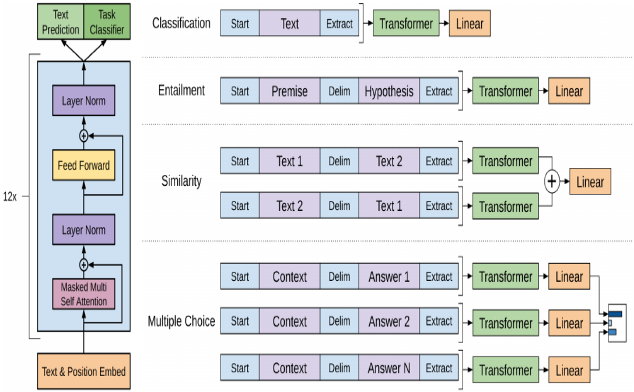


OpenAI GPT

怎么 Fine-Tuning？

OpenAI GPT

怎么 Fine-Tuning？



BERT

OpenAI GPT 的问题

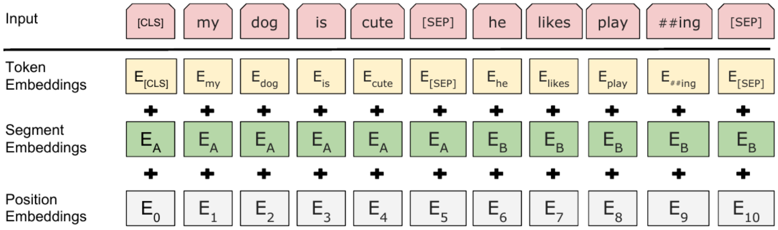
- 单向 The animal didn’t cross the street because it was too tired.
- Pretraining(1) 和 Fine-Tuning(2) 不匹配

解决方法

- Masked LM
- NSP Multi-task Learning
- Encoder again

BERT 输入表示

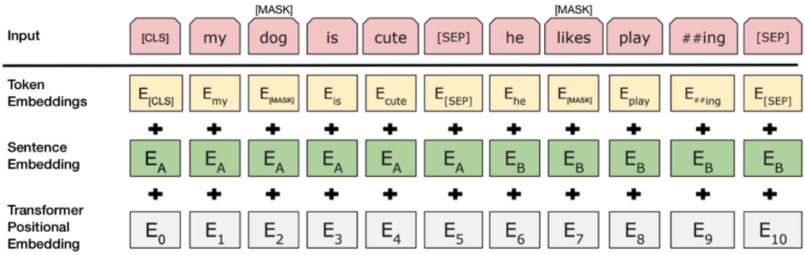
- 输入分两段
- BPE 编码



Masked LM

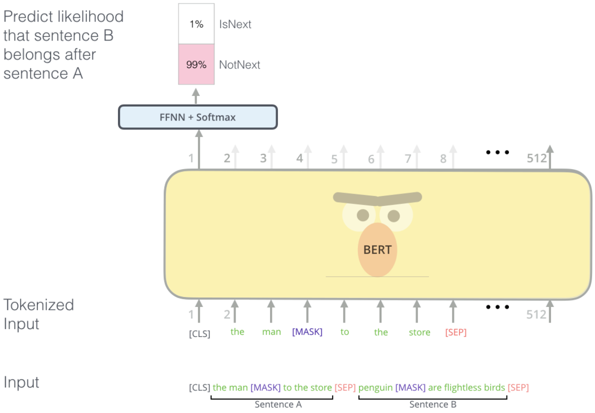
类似于完形填空

随机 Mask 掉 15% 的词，让 BERT 来预测



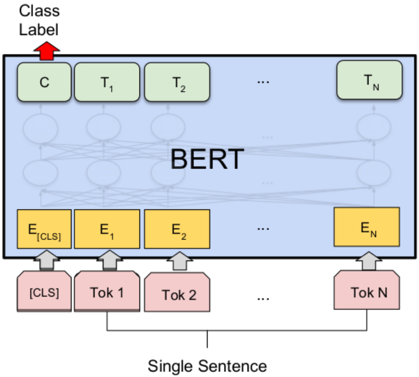
预测句子关系

引入新任务解决 Pretraining 和 Fine-Tuning 不匹配



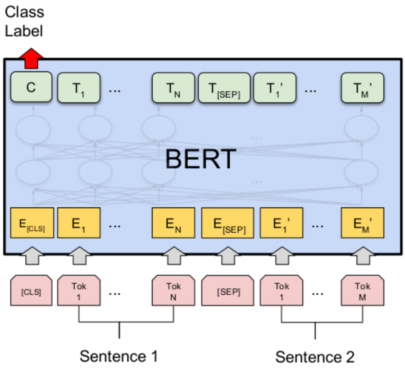
Fine-Tuning

单个句子的任务



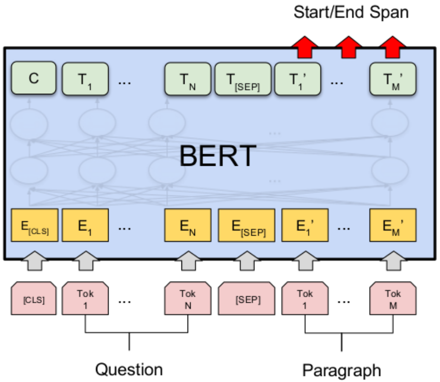
Fine-Tuning

两个句子的任务



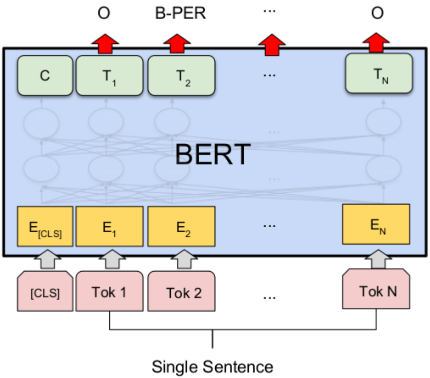
Fine-Tuning

问答类的任务



Fine-Tuning

序列标注



Pretrained Models

模型	层数	隐单元	head 数	总参数
BERT-base-uncased	12	768	12	110M
BERT-base-cased	12	768	12	110M
BERT-large-uncased	24	1024	16	340M
BERT-large-cased	24	1024	16	340M
BERT-large-ml-cased	12	768	12	110M
BERT-base-chinese	12	768	12	110M

Fine-Tuning

```
python run_classifier.py \
    --task_name=MRPC \
    --do_train=true \
    --do_eval=true \
    --data_dir=$GLUE_DIR/MRPC \
    --vocab_file=$BERT_BASE_DIR/vocab.txt \
    --bert_config_file=$BERT_BASE_DIR/bert_config.json \
    --init_checkpoint=$BERT_BASE_DIR/bert_model.ckpt \
    --max_seq_length=128 \
    --train_batch_size=8 \
    --learning_rate=2e-5 \
    --num_train_epochs=3.0 \
    --output_dir=/tmp/mrpc_output/
```

Pretraining

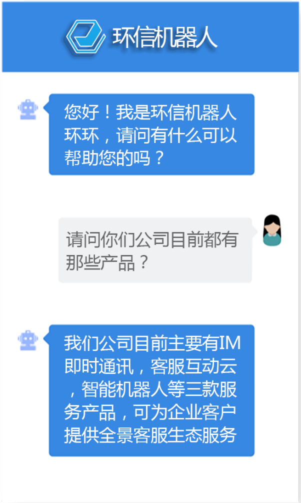
数据预处理：

```
python create_pretraining_data.py \
    --input_file=./sample_text.txt \
    --output_file=/tmp/tf_examples.tfrecord \
    --vocab_file=$BERT_BASE_DIR/vocab.txt \
    --do_lower_case=True \
    --max_seq_length=128 \
    --max_predictions_per_seq=20 \
    --masked_lm_prob=0.15 \
    --random_seed=12345 \
    --dupe_factor=5
```

Pretraining

```
python run_pretraining.py \
    --input_file=/tmp/tf_examples.tfrecord \
    --output_dir=/tmp/pretraining_output \
    --do_train=True \
    --do_eval=True \
    --bert_config_file=$BERT_BASE_DIR/bert_config.json \
    --init_checkpoint=$BERT_BASE_DIR/bert_model.ckpt \
    --train_batch_size=32 \
    --max_seq_length=128 \
    --max_predictions_per_seq=20 \
    --num_train_steps=20 \
    --num_warmup_steps=10 \
    --learning_rate=2e-5
```

案例分析



环信机器人，高频常见问题 (FAQ)

两种解决方法

- 相似度计算 (KNN)
- 意图分类

相似度计算

几十万标注的训练数据：

手机号码注销了，怎么换手机号吗？	如何修改手机号	1
我都不敢充值了	我充值不了	0
支付宝怎么充值	微信怎么充值	0.5

Baseline 是 DSSM，F1 得分提高10%

相似度计算

几十万标注的训练数据：

手机号码注销了，怎么换手机号吗？	如何修改手机号	1
我都不敢充值了	我充值不了	0
支付宝怎么充值	微信怎么充值	0.5

Baseline 是 DSSM，F1 得分提高10%

意图分类

问题和方法

问题：给定一个句子，判断其意图分类 几万训练数据，几百个类别，数据分布不均衡

BaseLine 系统

- 多层 LSTM
- 多个模型 Ensembling
- 上百个人工特征

BERT 分类器

- 中文模型
- 进行 Fine-Tuning
- 没有任何特殊处理

F1 得分提高3%！

意图分类

问题和方法

问题：给定一个句子，判断其意图分类 几万训练数据，几百个类别，数据分布不均衡

BaseLine 系统

- 多层 LSTM
- 多个模型 Ensembling
- 上百个人工特征

BERT 分类器

- 中文模型
- 进行 Fine-Tuning
- 没有任何特殊处理

F1 得分提高3%！

Tips

- 使用中文模型，不要使用多语言模型！
- max_seq_length 可以小一点，提高效率
- 内存不够，需要调整 train_batch_size
- 有足够多的领域数据，可以尝试 Pretraining

Tips

- 使用中文模型，不要使用多语言模型！
- max_seq_length 可以小一点，提高效率
- 内存不够，需要调整 train_batch_size
- 有足够多的领域数据，可以尝试 Pretraining

Tips

- 使用中文模型，不要使用多语言模型！
- max_seq_length 可以小一点，提高效率
- 内存不够，需要调整 train_batch_size
- 有足够多的领域数据，可以尝试 Pretraining

Tips

- 使用中文模型，不要使用多语言模型！
- max_seq_length 可以小一点，提高效率
- 内存不够，需要调整 train_batch_size
- 有足够多的领域数据，可以尝试 Pretraining

总结

- Word Embedding
- RNN/LSTM/GRU
- Seq2Seq、Attention 和 Self-Attention
- Contextual Word Embedding
  - ELMo
  - OpenAI GPT
- BERT 原理
- BERT 实战

进阶阅读和主要参考资料 I

- 作者博客
  http://fancyerii.github.io/
- Xin Rong.
  word2vec Parameter Learning Explained, 2014;
  arXiv:1411.2738.
- Colah
  Understanding LSTM Networks
  http://colah.github.io/posts/2015-08-Understanding-LSTMs/

进阶阅读和主要参考资料 II

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin.
  Attention Is All You Need, 2017;
  arXiv:1706.03762.
- Jay Alammar
  The Illustrated Transformer
  http://jalammar.github.io/illustrated-transformer/
- Alexander Rush
  The Annotated Transformer
  http://nlp.seas.harvard.edu/2018/04/03/attention.html

进阶阅读和主要参考资料 III

- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer.
  Deep contextualized word representations, 2018;
  arXiv:1802.05365.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever.
  Improving language understanding with unsupervised learning, 2018;
  Technical report, OpenAI.

进阶阅读和主要参考资料 IV

Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018;
arXiv:1810.04805.

谢谢！