深度之眼
deepshare.net

Attention2Bert

导师：Duke

深度之眼
deepshare.net

目录

Overview

Dialogue tasks:

Natural Language Understand

Natural Language Generation

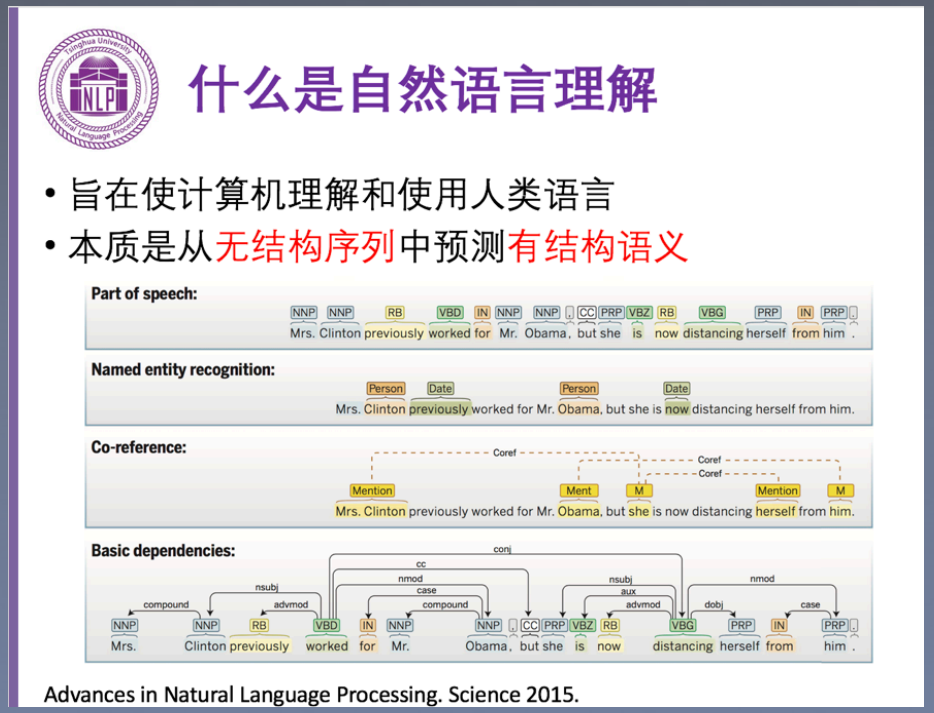
Dialogue Management

我之所以借助对话来做这个任务分类，是因为我们可以整体上把NLP任务分到三大类别。

我们是存在很多不同的任务：

文本分类、阅读理解、序列标注等都可以列为NLU的序列

稍微复杂一点的机器翻译、QA等可以划定为NLU和NLG的集合DM，即涉及到了理解并作出决策，来进行NLG，就更复杂了一些。当然，我们存在新的策略就是直接把NLU和NLG建立联系做对话，这样，相当于是让NN直接作为DM来，把策略的工作交给网络来做



——清华大学-刘知远老师的ppt中的一页

Guideline2NLP

NLP相关从业常用技能

1.序列标注（HMM+viterbi，CRF，Bilstm+CRF，Bert）
2.分类任务，bert
3.阅读理解，bert
4.智能问答，检索式 vs 生成式
5.信息抽取，规则，NN
6.召回与排序，推荐常用，文本作为重要特征
7.机器翻译，transformers
8.多轮对话，multi-turn dialogue
9.机器学习，Basic knowledge
10.深度学习，BP, optimizer etc.
11.强化学习，Interaction
12.GAN/Multi-task，Electra (stanford)

Guideline2NLP

深度之眼
deepshare.net

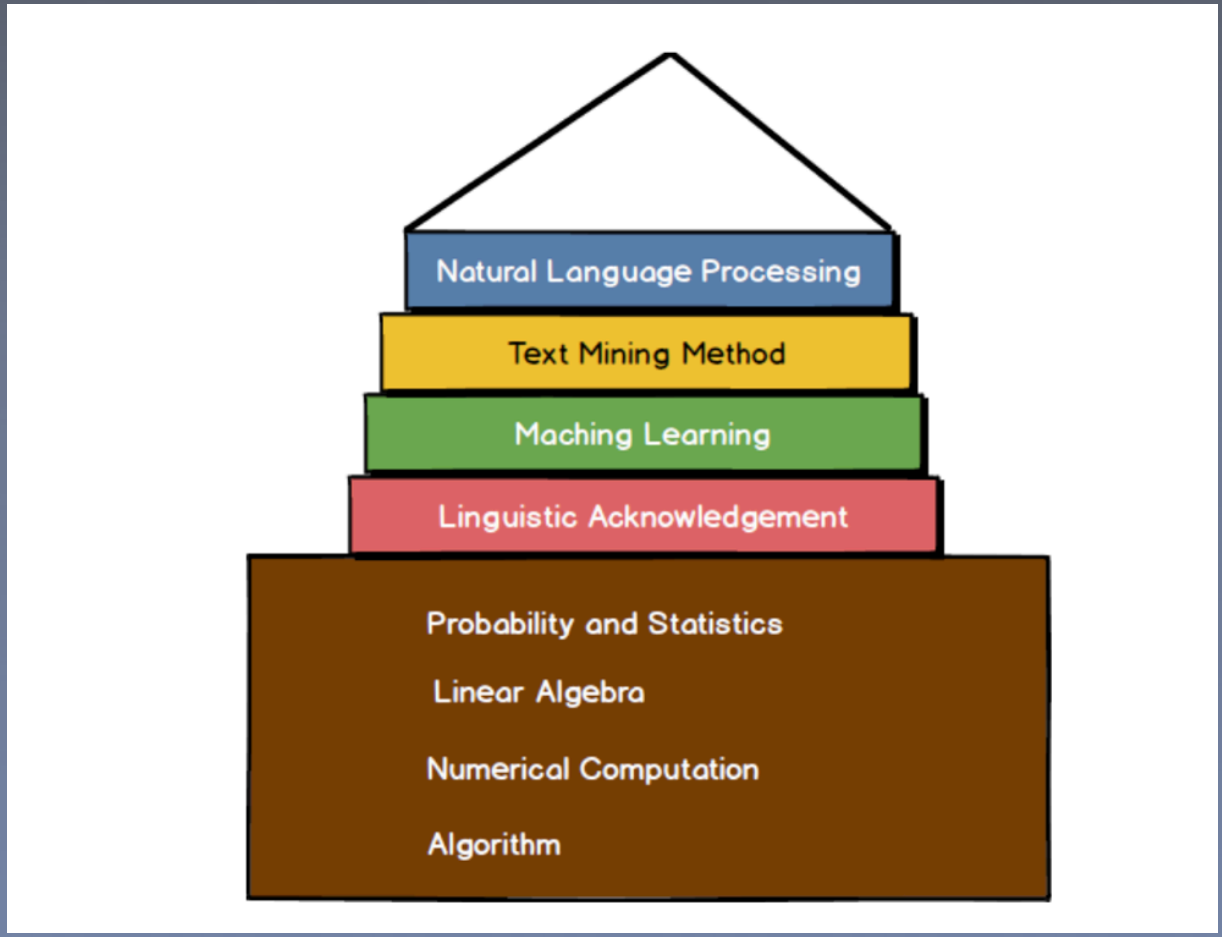
1. 要具备一定的工程能力，对想法进行验证起码，这叫做工程师；
2. 要有问题的解析能力，基础的算法数据结构要有，这叫做算法工程师；
3. 要有NLP的相关知识，包括但不限于语言相关的知识以及对NLP相关的工具包的使用和理解；
4. 不断的学习能力和一定的英文能力，能够follow最新的研究；
5. 有一定的数学能力，对一些公式可以进行推理验证；
6. 再有就是创造能力了，这个就是以上的能力具备之后，可以自己考虑创造。

https://zhuanlan.zhihu.com/p/112066313
NLP不同岗位的介绍

Guideline2NLP

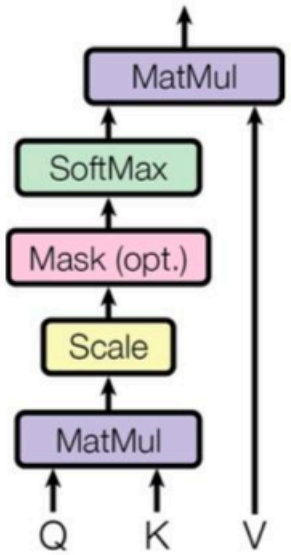
https://github.com/graykode/nlp-roadmap

很有趣的一个图

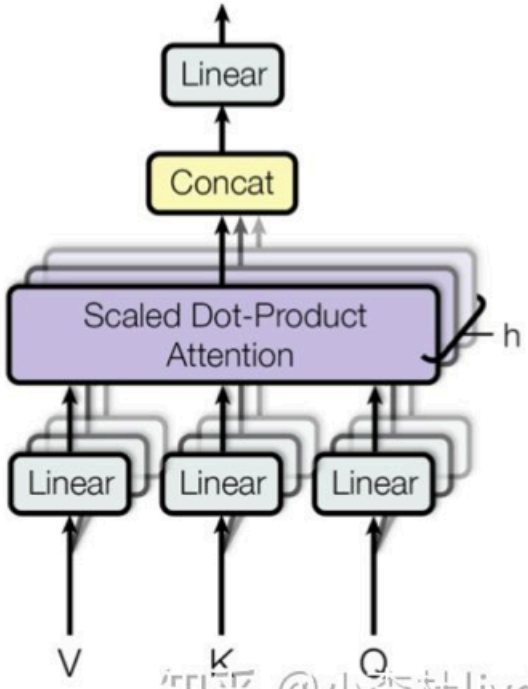


Attention

Scaled Dot-Product Attention



Multi-Head Attention



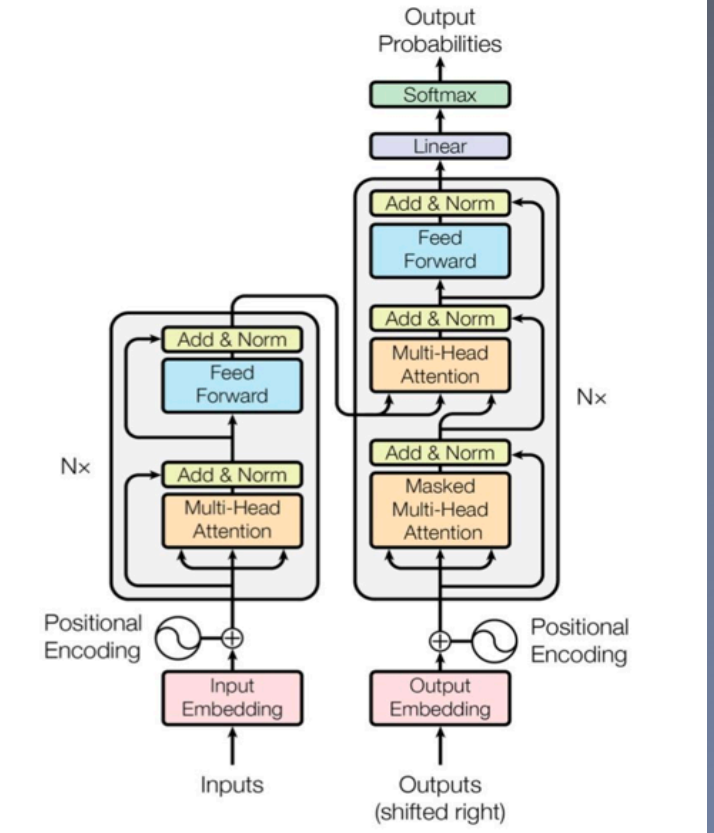


Figure 1: The Transformer - model architecture.

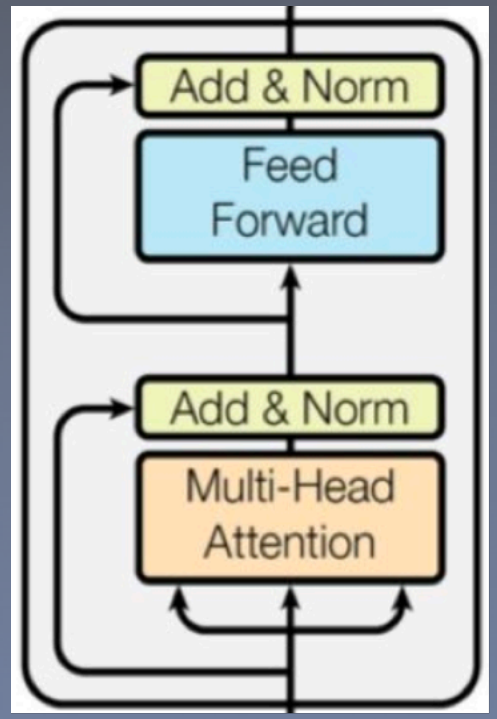
Attention

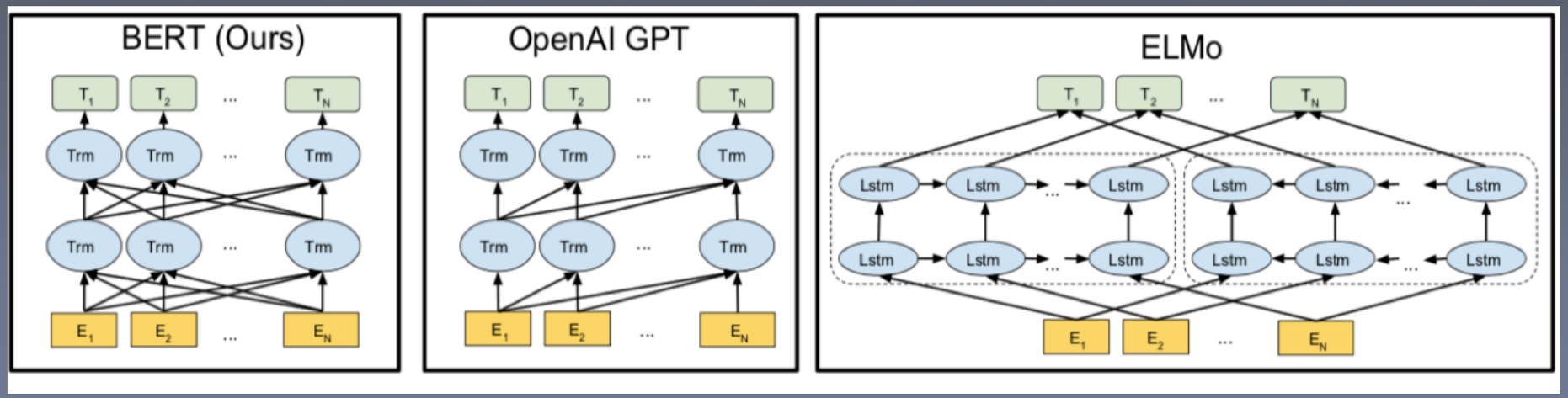
自注意力

所谓的自注意力就是找到一个输入中不同部分之间的相关关系的一种注意力机制。这种机制在machine reading, abstractive summarization, image description generation中比较有用。

Self-attention, also known as **intra-attention**, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence. It has been shown to be very useful in machine reading, abstractive summarization, or image description generation.

Attention-transformer





Bert-how to use

大部分笔记在：https://zhuanlan.zhihu.com/p/113326366

https://zhuanlan.zhihu.com/p/113326366

要么把token的表示送入到output layer做token-level的任务，例如ner或者qa任务；

要么就把cls的表示送入到output来做分类，例如en-tailment或者情感分析。

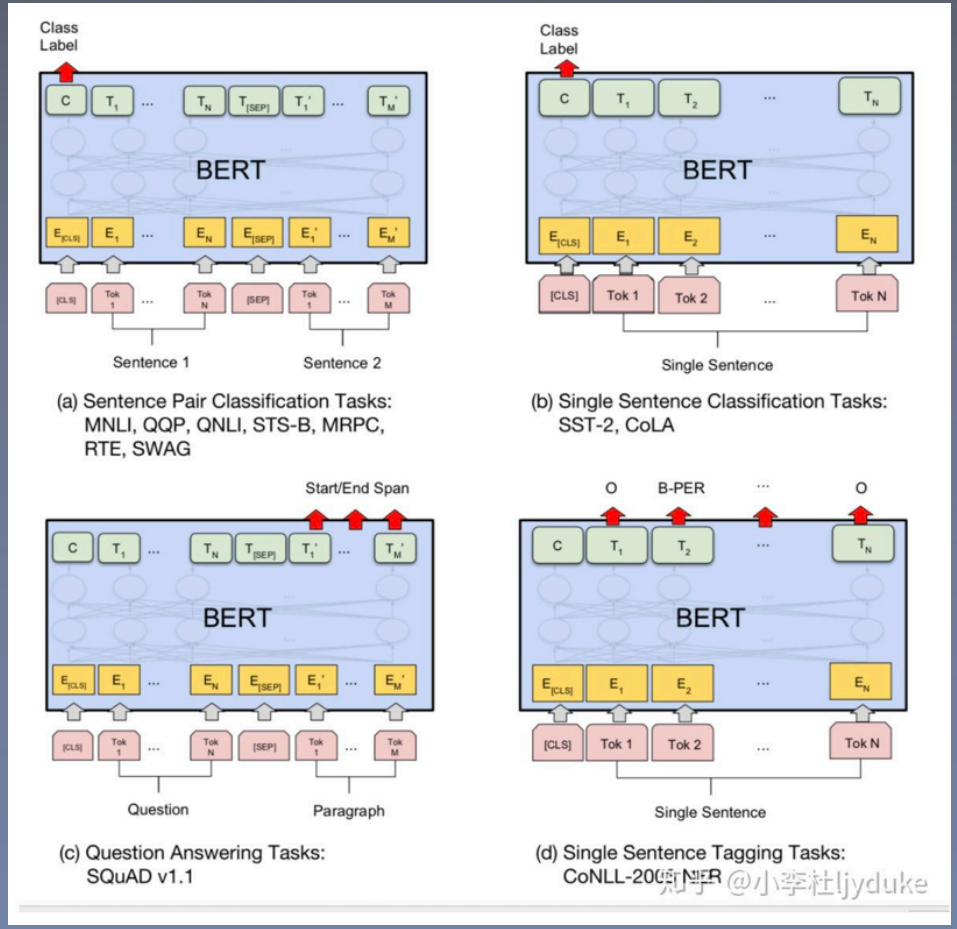
GPT可以做生成任务，但是BERT不可以。因为BERT就不是一个正常的语言模型（从左往右或者反过来）

https%3A//mp.weixin.qq.com/s/CUVWDkuQ5ygtELqDGJo4sg

Bert-how to use

要么把token的表示送入到output layer做token-level的任务，例如ner或者qa任务；

要么就把cls的表示送入到output来做分类，例如en-tailment或者情感分析。



Bert-paper

Introduction

强调了语言模型pre训练的重要性，列出了几种任务：

sentence-level tasks NLI and paraphrasing,目的是通过整体分析预测sentence之间的关系

token-level。NER 细粒度的分析

运用预训练模型到下游任务的策略

Feature-based: ELMO, 使用task-specific 结构，其中包含了pretrained的表示作为**additioal features**。

Fine-tuning：GPT （Generative Pre-trained Transformer），介绍了最小的task-specific参数，并且在下游任务上做fine-tuning来微调参数，就可以得到不错的效果。

以上的两个pretraining都用了同样的**objective function【语言模型】**，且都用了unidirectional language models来学习general的语言表示。

Bert-paper

重点来了：

提出了当前的瓶颈一是单向的语言模型作为学习。这个一个是限制了网络结构，另一个就是在sentencelevel是sub-optimal的，对于句子级别的人物来说。那么对于token级别（QA）的来说就更差劲了，因为双向上下文的信息对这些任务很重要。

提出了BERT作为解决方案，改进了fine-tuning的预训练方法。和GPT的结构很像，但是应用了双向，并且想出了怎么用，这个是关键。双向的transformer。本身transformer是一个很有力的特征提取器。（三层，语义信息，位置信息，句子信息）。

Bert-paper

应用方法：使用MLM “Masked language model” 作为与训练的objevtive， 说是被Cloze task任务inspired的【BOW也是类似的方法】。
The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context.
这样的方法就可以把左右的context都放进去

第二个任务叫做NSP。Next sentence prediction。a “next sentence prediction” task that jointly pre- trains text-pair representations. 文本对的表示。这个怎么理解呢？Specifically, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext). 这个在后面有介绍原因，这里先提一下。【cls】 0 1 分别代表了是后面的句子，或者不是
The contributions of our paper are as follows:
MLM来训练，做到了双向transformer NSP，做到句子级别的训练，特征捕捉
fine-tuning的模型，sota结果。

Bert-how to pretrain

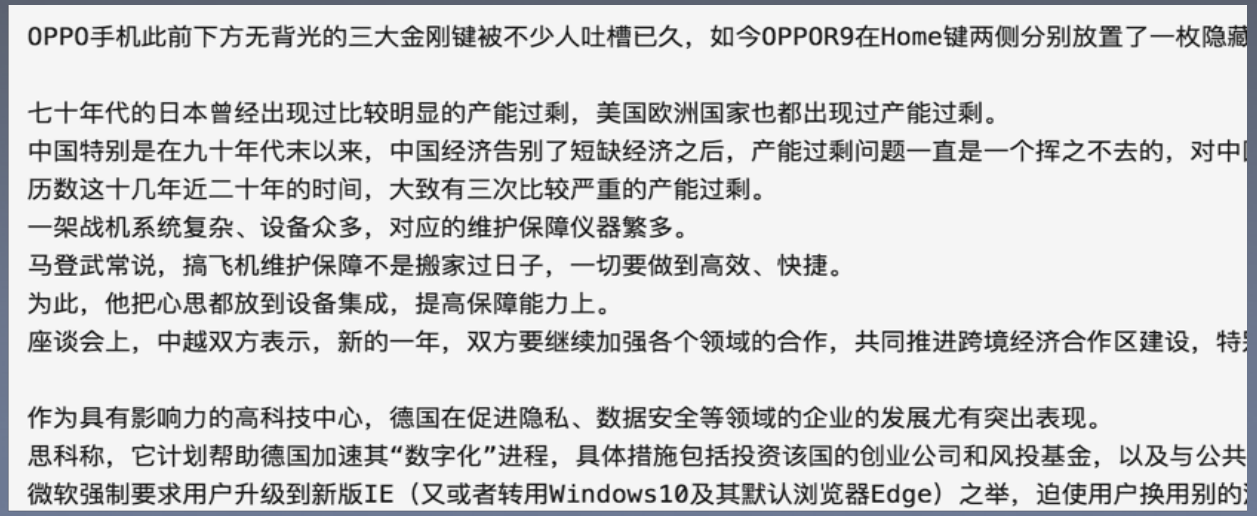
如何从0开始训练一个完整的bert

1. 将网络上的demo跑通

下载网络上的预训练模型，然后使用里面附带的vocab.txt进行。

2. 等到这个demo通了之后，我们就需要利用自己的数据，来重新训练。【TPU训练是不需要管内存的】

wordpiece意义在于，比如"loved", "loving", "loves"这三个单词。其实本身的语义都是“爱”的意思，但是如果我们以单词为单位，那它们就算不一样的词，在英语中不同后缀的词非常的多，就会使得词表变的很大，训练速度变慢，训练的效果也不是太好。 BPE算法通过训练，能够把上面的3个单词拆分成"lov","ed", "ing", "es"几部分，这样可以把词的本身的意思和时态分开，有效的减少了词表的数量。 并且一定程度上可以避免oov的情况。【没有见过loved，但是见过lov，有见过ed，那不就结了】



佻
佼
使
侃
侄
侈
例
注意其中英文 会使用wordpiece。

HOW TO CONSTRUCT WORD
github.com/kwonmha/bert-vocab-builder

Bert-how to pretrain

预训练

在warmup 阶段学习率会比之后低，之后学习率恢复正常。学习率的改变一般有以下几种方式。

"warmup_cosine": WarmupCosineSchedule,

"warmup_constant": WarmupConstantSchedule,

"warmup_linear": WarmupLinearSchedule

```
python3 create_pretraining_data.py  \
  --input_file=gs://clue_storage/clue_pretrain_corpus/raw_txt_corpus/train/c
  --output_file=gs://clue_storage/clue_pretrain_corpus/tfrecords/bert_base_1
  --vocab_file=./bert_base/vocab_clue.txt  \ # 我们使用的词表
  --t2s=True  \ # 我们是否要进行简体和繁体的转化
  --do_lower_case=True \ # 是否要做大写转小写
  --max_seq_length=128 \ # 句子的最大输入长度
  --max_predictions_per_seq=20 \ # 下一句话的预测
  --masked_lm_prob=0.15 \ # mask单词的概率
  --random_seed=12345 \
```

Bert-how to pretrain

这个max_seq 和 maxprediction 在createdata以及做pretrain的时候必须要保持一致。

The max_predictions_per_seq is the maximum number of masked LM predictions per sequence. You should set this to around max_seq_length * masked_lm_prob (the script doesn't do that automatically because the exact value needs to be passed to both scripts).

from: [What is suitable max_predictions_per_seq size when max_seq_length set to 512? · Issue #516 · google-research/bert]

```
python3 run_pretraining.py \
  --input_file=$INPUT_DIR \
  --output_dir=$OUT_PUT_DIR \
  --do_train=True \
  --do_eval=True \
  --bert_config_file=./bert_base/bert_config.json \ # 输入的config参数配置
  --train_batch_size=4096 \ batch。batch理论上是越大越好的。但是放不下，所以就用mini-b
  --max_seq_length=512 \
  --max_predictions_per_seq=51 \
  --num_train_steps=500,500 \ # 这里意思是 一共训练多少步。具体的steps和epochs之间的计
  --num_warmup_steps=10,000 \ # 预热的步骤
  --learning_rate=0.00176  \ # 学习率
  --save_checkpoints_steps=2000 \
```

paper的建议

> We train with batch size of 256 sequences (256 sequences * 512 tokens = 128,000 tokens/batch) for **1,000,000 steps, which is approximately 40 epochs over the 3.3 billion word corpus**. We use Adam with learning rate of 1e-4, β1 = 0.9, β2 = 0.999, L2 weight decay of 0.01, learning rate warm-up over the first 10,000 steps, and linear decay of the learning rate

Bert-how to pretrain

mlm最好要达到0.70以上，
nsp基本接近满分。可以不用，
参加robert。

```
#!/usr/bin/env bash
# convert c5 txt to tfrecords with clue vocab(8k), set t2s to True
# add a hyper-parameter: t2s, traditional chinese to simplified chinese
echo $1,$2
PWD=$(cd -P -- "$(dirname -- "$0")" && pwd -P)
j=0
if [ ! -d $PWD/log_c5_vocab8k ];then
  mkdir $PWD/log_c5_vocab8k
fi
for((i=$1;i<=$2;i++));
do
  #random_seed=`expr 12345 + $i `
  #echo $random_seed
  printf -v l "%07d" $i
  echo $l                                  # original: bert_base_128_vocab8k
  python3 create_pretraining_data.py  \
    --input_file=gs://clue_storage/clue_pretrain_corpus/raw_txt_corpus/train/clue_pretrain_$l.txt   \
    --output_file=gs://clue_storage/clue_pretrain_corpus/tfrecords/bert_base_128_c5_vocab8k/clue_pretrain128_$i.tfr
ecord \
    --vocab_file=./bert_base/vocab_clue.txt  \
    --t2s=True  \
    --do_lower_case=True \
    --max_seq_length=128 \
    --max_predictions_per_seq=20 \
    --masked_lm_prob=0.15 \
    --random_seed=12345 \
    # --dupe_factor=5 >$PWD/log_c5_vocab8k/tfrecord_$i.log 2>&1 &
  j=$[j+1]
  if [ $j -eq 20 ];then
    wait
    j=0
  fi
done
wait
echo "Finish"
```

Bert-how to pretrain

training的脚本，注意如果输入的是folder怎么办

```
# INPUT_DIR=gs://clue_storage/clue_pretrain_corpus/tfrecords/tiny_train/*.tfrecord
INPUT_DIR=gs://clue_storage/clue_pretrain_corpus/tfrecords/tiny_train/clue_pretrain128_
#INPUT_DIR=gs://clue_storage/clue_pretrain_corpus/tfrecords/bert_base_128_c5_vocab8ki/
clue_pretrain128_1.tfrecord
OUT_PUT_DIR=gs://clue_storage/clue_pretrain_corpus/
pretraining_output_310_300tiny_berttest_1tfrecord_res
TPU_IP=10.240.1.2
# bert_base/vocab_clue.txt
MODEL_DIR=$1
nohup python3 run_pretraining.py \
  --input_file=$INPUT_DIR \
  --output_dir=$OUT_PUT_DIR \
  --do_train=True \
  --do_eval=True \
  --bert_config_file=./$MODEL_DIR/bert_config.json \
  --train_batch_size=128 \
  --max_seq_length=128 \
  --max_predictions_per_seq=20 \
  --num_train_steps=1000000 \ # 这个是global step，如果要follow着之前或者follow别人的训练，
# 记得要把这个改的比别人大 . 1G的数据在70w mlm只到了0.52左右。但是5m左右的数据的话，基本上50w就够了
  --num_warmup_steps=10000 \
  --learning_rate=2e-5  \
  --save_checkpoints_steps=2000 \
  --num_tpu_cores=8 --use_tpu=True --tpu_name=grpc://$TPU_IP:8470
```

Tips：
如果模型不收敛，请用小数据，小学习率尝试一下，可能是max的数据太多了；
可能是你生成的数据有问题，请仔细检查（我因为这个浪费了好几天）
wc -l 计算的是\n 所以请仔细核对词表。
实在是没有机器。。。。。。忧桑

Bert-how to pretrain


```
{
  "attention_probs_dropout_prob": 0.1, attention 中的dropout
  "directionality": "bidi", # 双向
  "hidden_act": "gelu", # 激活函数
  "hidden_dropout_prob": 0.1, # hidden dropout
  "hidden_size": 768, # hidden size
  "initializer_range": 0.02, # 所有权重矩阵的
truncated_normal_initializer的stdev
  "intermediate_size": 3072, # 中间的大小 4* 768 ffn 的大小就这样设置了
  "max_position_embeddings": 512, # position embedding size
  "num_attention_heads": 12, # Transformer encoder的每个attention
layer的attention heads数
  "num_hidden_layers": 12, # Transformer encoder的hidden layer数
  "pooler_fc_size": 768, # pool
  "pooler_num_attention_heads": 12,
  "pooler_num_fc_layers": 3,
  "pooler_size_per_head": 128,
  "pooler_type": "first_token_transform",
  "type_vocab_size": 2, # 就是nsp的任务的参数 句子a和句子b
  "vocab_size": 21128 # inputs_ids的vocabulary size
}
```

Bert-how to pretrain


```
I0317 10:56:51.714832 140381245654784 error_handling.py:96] evaluation_loop marked as finished
INFO:tensorflow:***** Eval results *****
I0317 10:56:51.715177 140381245654784 run_pretraining.py:484] ***** Eval results *****
INFO:tensorflow:  global_step = 1000000
I0317 10:56:51.715279 140381245654784 run_pretraining.py:486]   global_step = 1000000
INFO:tensorflow:  loss = 7.419402e-08
I0317 10:56:51.715584 140381245654784 run_pretraining.py:486]   loss = 7.419402e-08
INFO:tensorflow:  masked_lm_accuracy = 1.0
I0317 10:56:51.715685 140381245654784 run_pretraining.py:486]   masked_lm_accuracy = 1.0
INFO:tensorflow:  masked_lm_loss = 7.987938e-08
I0317 10:56:51.715753 140381245654784 run_pretraining.py:486]   masked_lm_loss = 7.987938e-08
INFO:tensorflow:  next_sentence_accuracy = 0.54625
I0317 10:56:51.715817 140381245654784 run_pretraining.py:486]   next_sentence_accuracy = 0.54625
INFO:tensorflow:  next_sentence_loss = 0.6892878
I0317 10:56:51.715880 140381245654784 run_pretraining.py:486]   next_sentence_loss = 0.6892878
```

Bert-how to use it

官方给出了推荐的参数，我们也可以使用试试看：

Batch size: 16, 32

Learning rate (Adam): 5e-5, 3e-5, 2e-5

Number of epochs: 2, 3, 4

We also observed that large data sets (e.g., 100k+ labeled training examples) were far less sensitive to hyperparameter choice than small data sets. Fine-tuning is typically very fast, so it is rea- sonable to simply run an exhaustive search over the above parameters and choose the model that performs best on the development set.

Bert-how to use it

How to finetune bert: https://arxiv.org/pdf/1905.05583.pdf

To adapt BERT to a target task, we need to consider several factors: 1) The first factor is the preprocessing of long text since the maximum sequence length of BERT is 512. 2) The second factor is layer selection. The official BERT-base model consists of an embedding layer, a 12-layer encoder, and a pooling layer. We need to select the most effective layer for the text classification task. 3) The third factor is the overfitting problem. A better optimizer with an appropriate learning rate is desired.

Intuitively, the lower layer of the BERT model may contain more general information. We can fine-tune them with different learning rates.

Bert-how to use it

4.2 Further Pre-training

The BERT model is pre-trained in the general-domain corpus. For a text classification task in a specific domain, such as movie reviews, its data distribution may be different from BERT. Therefore, we can further pre-train BERT with masked language model and next sentence prediction tasks on the domain-specific data. Three further pre-training approaches are performed:

4.3 Multi-Task Fine-Tuning

Multi-task Learning is also an effective approach to share the knowledge obtained from several related supervised tasks. Similar to Liu et al. (2019), we also use fine-tune BERT in multi-task learning framework for text classification.

All the tasks share the BERT layers and the embedding layer. The only layer that does not share is the final classification layer, which means that each task has a private classifier layer. The experimental analysis is in Sec. 5.5.

Bert-how to use it

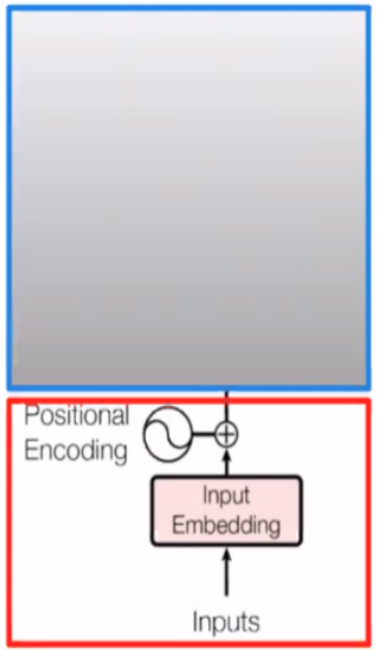
a
输入[CLS]sentence A [sep] sentence B [sep]
输出[CLS]进入softmax
b
输入[CLS]sentence A [sep]
输出[CLS]进入softmax
c
输入[CLS]sentence A(Question) [sep] sentence B (paragraph)[sep]
输出[sentence B 的 token representation]进入softmax
d
输入[CLS]sentence A [sep]
输出sentence A的token representation进入softmax

Albert

Method 1: factorized embedding parametrization

- Token embeddings are context independent while hidden layer embeddings are context dependent.
- Token embeddings are sparsely updated.

$O(V \times H) \rightarrow O(V \times E + E \times H)$ where $E << H$



Copy from Dr.Lan

Albert

原来是每个block都用自己的参数，

现在改成了所有的block都用一样的参数

参数共享的一个大优势就是模型可以更深

3:

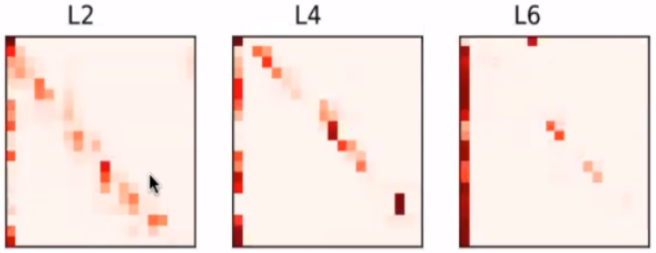
NSP: Next Sentence Prediction

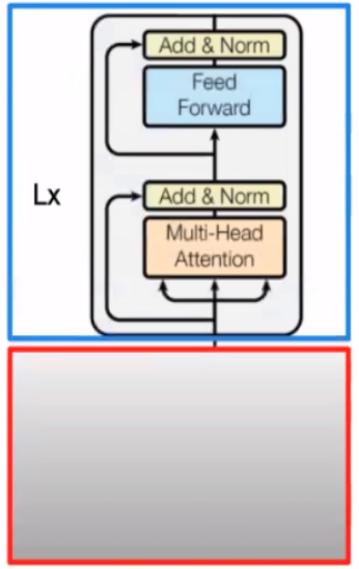
SOP: Sentence Order Prediction

4:

Remove Dropout

Method 2: Cross-layer parameter sharing





- The attention-feedforward operations are repeated, can we share their parameters?
- Visualization of attention map show similar operation across layers

O (12 x L x H x H) → O (12 x H x H)

Gong, Linyuan, et al. "Efficient Training of BERT by Progressively Stacking." *ICML*. 2019.

结 语

The birth of Albert and Electra shows the popularity of Light language model. If you are NLP researchers, you could focus on this kind of problem.

Welcome to the NLPCC-2020 task1: Light Pre-Training Chinese Language Model for NLP Task

www.cluebenchmarks.com

http://tcci.ccf.org.cn/conference/2020/cfpt.php

dukeenglish.github.io

深度之眼
deepshare.net

联系我们：

电话：18001992849

邮箱：service@deepshare.net

Q Q： 2677693114

公众号

客服微信