

第十章 试题知识点多标签分类-BERT

试题知识点标注

Lesson-04

BERT与文本分类

风老师

2020.7

目录

- BERT介绍
- BERT与文本分类

1/2 BERT 介绍

词向量生成方法

缺点: 这几种词向量生成方法都不能有效解决一词多义问题

回顾: 语言模型

- ngram语言模型: 基于统计概率的语言模型: $P(w) = P(w_1|w_0) \cdot P(w_2|w_1) \dots$
- 预训练语言模型: word2vec, glove, fasttext
- 高级语言模型: ELMo, GPT 等
- 各种语言模型的问题 (每种语言模型都有自己的问题)
如 ELMo 的两个 LSTM 会分开训练, 没有交叉

word2vec: 局部的, 周边词预测/中间词或中间词预测/周边词

glove: 基于全局统计的词向量生成方法 (公式推导生成词向量)

fasttext: 局部的, 词向量生成工具

ELMo 和 GPT 能解决一词多义问题。因为网络的输入为一个句子, 每个词对应的词向量依赖上下文进行输出。

注: ① bert 使“预训练+微调”方式成为目前文本处理的常用方式

② ELMo 和 GPT 的发现都早于 bert

回顾: transformer

- Encoder-Decoder 结构
- Self-Attention: 计算句子中词的权重
- Multi-head attention: 增加模型的表达能力, (表达更多的依赖)
- Positional Encoding
- 解码器 Self-Attention 输入层与编码器区别: 只允许关注输出序列中较前的位置 (mask 遮罩)

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

位置编码, 保持词的位置关系, 形成位置向量, 加入到 embedding 中。

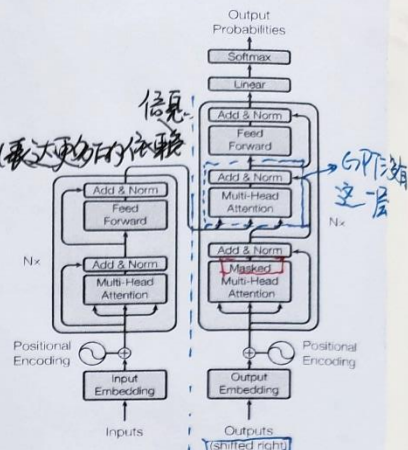


Figure 1: The Transformer - model architecture.

① GPT 基于 decoder 进行计算, bert 基于 encoder 进行计算。

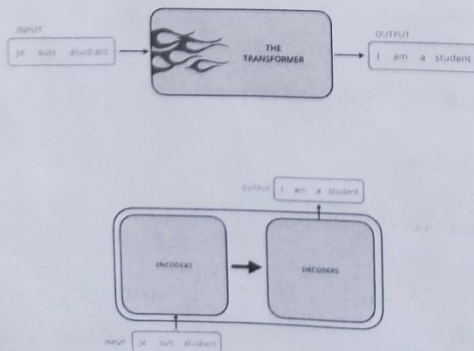
② GPT 是单向的, bert 是双向的。→ 你能看见我, 我也能看见你 (因为没有遮罩)

你可以看见我, 我不能看见你
(后面可以看见前面的信息, 前面不能看见后面的)

bert 使用
encoder 结构

GPT 使用 decoder
的结构

Encoder-Decoder 结构拆解

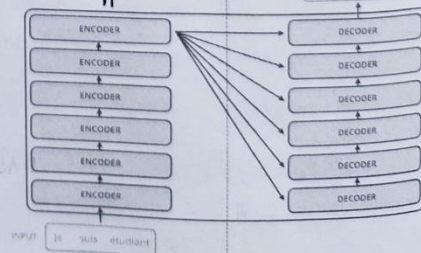


Encoder 是双向的

bert

Decoder 是单向的

GPT



BERT概述

基于transformer的双向预训练语言模型。

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Google AI Language) >>
- BERT: Bidirectional Encoder Representations from Transformer
- GLUE基准80.04% (7.6%绝对提升), MultiNLI准确率86.7% (5.6%绝对提升)
- <https://gluebenchmark.com/leaderboard>

评估自然语言处理任务效果的网站。

BERT的贡献

BERT的比较标准为GPT, GPT为单向的。

- 证明了双向模型对文本特征表示的重要性
- 证明了预训练模型能够消除很多繁重的任务相关的网络结构
- 在11个NLP任务上, 提升了state of the art水平

增加各种层

采用预训练+微调。

BERT特点

如“苹果手机”与“吃苹果”中“苹果”的词向量是相同的, [0.1, 0.2, 0.7, ..., 0] 为同一个值。

- 与其他词向量的关系:
 - Word2vec等词向量是词维度, 训练好就确定了。
 - BERT句子维度的向量表示, 依赖上下文构建结果。
 - 苹果 “苹果”的词向量不相同
 - bank

BERT和Word2vec都是无监督训练, 不需要标签信息

通过无监督训练方式定义了一种有监督任务。

如周围词预测中间词

000 0 000
J J J

BERT 模型架构

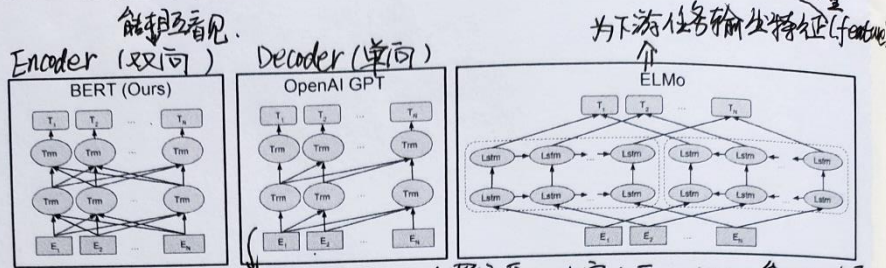


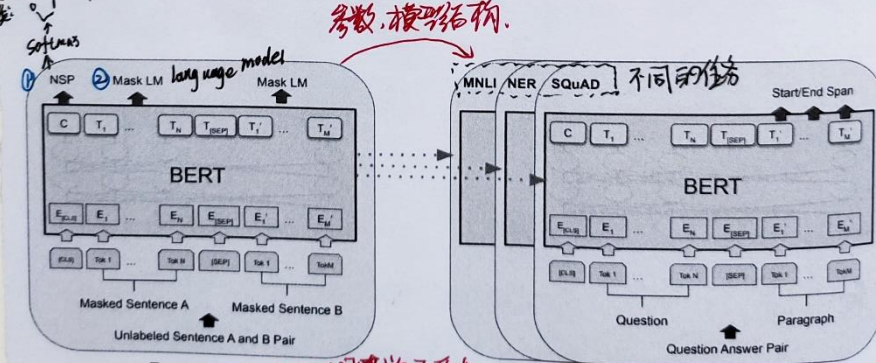
Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

微调模型的权重
词向量不同. 两个“苹果”
对应的词向量不同.

词向量是固定的, 如“苹果手机”与“吃苹果”中苹果的向量是相同的

BERT 核心思想

预训练和微调是两个不同的任务, 有自己不同的损失函数. 模型结构.



Pre-training (google 完成) 目的是学习两大任务 (NSP, Mask LM)

Fine-Tuning (需要我亲自完成)

预训练两大任务:

- ① 预测下一个句子 (句子B是不是句子A的下一个句子)
- ② 预测被 mask 的词.

模型架构, 参数是一样的, 输入和输出不同

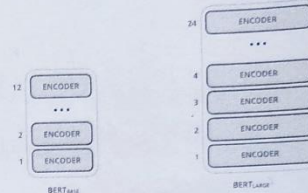
输入词级的是句子级

fasten

BERT

- 特点：跨越不同任务的统一架构
 - 预训练架构与最终下游架构之间的差异很小。

- 参数：
 - layer: 层 hidden ~~state~~ attention: 12个头 Million
 - 基础模型 BERT_BASE (L=12, H=768, A=12, 总参数=110M)
 - BERT_LARGE (L=24, H=1024, A=16, 总参数=340M)
 - In all cases we set the feed-forward/filter size to be 4H, i.e., 3072 for the H=768 and 4096 for the H=1024.



BERT

- 输入&输出：

输入为单句或句对组合

- WordPiece embeddings (英文)
- 单字拆分 (中文)
- [CLS]: 起始标记
- [SEP]: 句对分割标记

分隔符, SEP参与计算, 但对应的输出没有意义

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

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

CLS对应的词向量没有意义, 但CLS在下游任务中是有意义的 (起始标志)

Sum of embedding

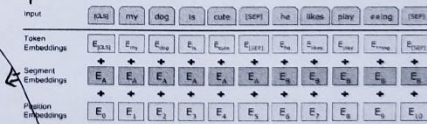


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embedding and the position embeddings.

使用单字或 Word piece 可以使词表大小变得较小 (字是有限的, 词是无限的)

lower x2 次, newest x6, widest x3 low x5
单字拆分 lower newest widest low

lower newest widest
↓
lower newest widest

Wordpiece 与 Subwords 区别: Wordpiece 是基于统计的拆分 (将经常使用的拼分在一起, 如词缀)

Subwords 是基于 n-gram 的拆分, fasttext 中的

BERT pre-train(Task 1): Masked LM

- 从左到右训练 or 从右到左训练
- BiLSTM等双向训练的问题 \rightarrow 如ELMO暴露来信息的问题
- masked LM: 随机MASK 15% 的word piece (存在的问题?)
 - (1) [MASK] token 替换 (80%)
 - (2) 随机token 替换 (10%)
 - (3) 不变 (10%)

句子中词先后组合的关系
(预测当前词)

预训练的任务在预训练被mask的词, (句子的任务, 单词的word piece)
mask是遮罩自己, shifted right 是遮罩后面的。

BERT pre-train(Task 2): Next Sentence Prediction (NSP)

- 目的: 问答, 推理等句子对之间的关系
- 训练数据构建: ~~句子A~~ 句子A + 50% 真实句子B + 50% 随机句子
 - 50%: B是跟随A的实际下一个句子 (标记为IsNext)
 - 50%: 来自语料库的随机句子 (标记为NotNext)
- 对比:
 - BERT: 传输所有参数初始化最终任务模型参数
 - 其他: 句子嵌入被转移到下游任务

- The final model achieves 97%-98% accuracy on NSP.
- The vector C is not a meaningful sentence representation without fine-tuning, since it was trained with NSP.

将预训练模型输出的feature 传给下游任务

后
将预训练后的embedding 模型的结构和参数都传给下游任务(微调)

预训练数据:

• 英文:

- BooksCorpus (800M words)

- 英语维基百科 (2,500M字) 提取文本段落, 忽略列表, 表格和标题。

• 汉语

- 中文维基百科

Fine-tuning: 需要做调的原因是预训练任务和下游任务不一致。

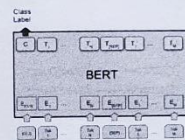
- Transformer中的自动注意机制允许BERT通过交换适当的输入和输出来模拟许多下游任务

- 将任务特定的输入和输出插入到BERT中, 并对端到端的所有参数进行微调。

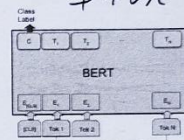
- 为什么要微调?

• 参数:

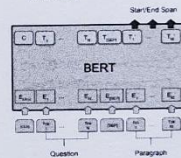
- Batch size: 16, 32
- Learning rate (Adam): $5e-5$, $3e-5$, $2e-5$
- Number of epochs: 2, 3, 4



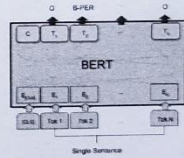
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



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



(c) Question Answering Tasks:
SQuAD v1.1



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

微调是一个有监督学习过程。

Fine-tuning 效果比 feature 好。

BERT & GPT: 对比

- ^{双向} Encoder VS. ^{单向} Decoder

bert ^{使用} 更多的数据: BooksCorpus and Wikipedia VS. BooksCorpus

- 在预训练中采用 SEP CLS (GPT: Fine-tuning)

- 每个 batch size 词用的更多

- (5e-5, 4e-5, 3e-5, and 2e-5) VS. 5e-5

^{rate} 基于不同的任务采用不同的 learning

在生成问题中, 单向模型比双向的好。像生 bert 不适合生成式问题。

BERT 每一层学到了什么:

- ACL 2019: What does BERT learn about the structure of language?

- <https://hal.inria.fr/hal-02131630/document>

前1,2层 ^{前面是什么,后面是什么} 低层网络捕捉了短语级别的结构信息

- ^{单词长度, 直观的特征} 表层信息特征在底层网络 (3, 4), 句法信息特征在中间层网络 (6~9), ^{如的在} 语义信息特征在高层网络。 (9~12)

- 主谓一致表现在中间层网络 (8, 9)

BERT 变体:

• ROBERTA

- 静态mask → 动态mask *bert的mask是动态的.*
- 去除句对NSP任务, 输入连续多个句子 *改为*
- 更多数据 更大batch size 更长时间 *句对占用的空间是有限的, 单句子可以*

• ALBERT

- 减少参数 (瘦身): $V \times H \rightarrow V \times E, E \times H$ *训练时间更长 占满整个空间 (512)*
- 在词表 V 到隐层 H 的中间, 插入一个小维度 E *所有层的 Attention 参数是统一的*
- 共享所有层的参数: Attention FFN
- SOP 替换 NSP: 负样本换成了同一篇文章中的两个逆序的句子
- BERT对MASK 15% 的词来预测. ALBERT 预测的是 n-gram 片段, 包含更完整的语义信息 *词组*

训练数据长度: 90%取512 BERT 90% 128

对应BERT large: H:1024 → 4096 L:24 → 12 *窄而深 → 宽而浅*

• 百度 ERNIE

ALBERT 训练中 90% 采用 512 的数
据量, 10% 采用 128.

句A. 句B
正样本: $A+B$
负样本: $A+Random$

$\Rightarrow \begin{cases} A+B \\ B+A \end{cases}$

2/2 BERT与文本分类

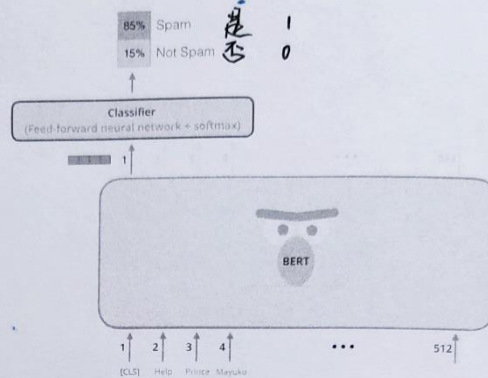
BERT可以解决的问题

- 序列标注
- 分类任务
- 句子关系 A与B是否相似
- 生成任务 (表现不好, 因为bert的训练中没有顺序生成的概念, 没有前后预测的因果关系)

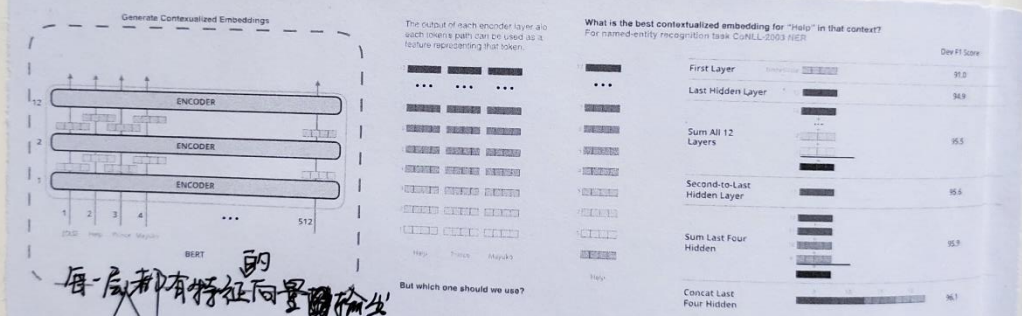
对于生成式问题, 建议采用transformer的seq2seq框架。

Bert文本分类方法

从结构上看
bert进行文本
分类的方式



从结构上看



每一层都有特征向量输出
对于单个的单词

↑ 取不同层的输出进行组合测试

文本分类方法1: 特征抽取+其他模型

特征抽取流程①

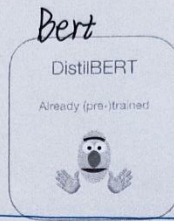
Feature-based

对词向量加权求和得到

Step #1: Use DistilBERT to embed all the sentences

Sentence embeddings

	Sentence	Label
0	a scarring, funny and finally transporting re-imagining of beauty and the beast and 1930s apparently reassembled from the cutting room floor of any given daytime soap	1
1	the movie is undone by a 1,999 filmmaking methodology that's just experimental enough	0



	Sentence Embeddings				Label
	0	1	...	767	
0	-0.215	-0.1402	...	0.201	1
1	-0.172	-0.154	...	0.371	0
...
1,999	0.124	0.014	...	0.274	1

接其他模型
如 Logistic

特征抽取

其他模型

文本分类方法1: 特征抽取+其他模型

特征抽取流程②

train/test 流程

Step #2: Test/Train Split for model #2, logistic regression

	Sentence Embeddings				Label
	0	1	...	767	
0	-0.215	-0.1402	...	0.201	1
1
2
3
4
5
6
7
8
9
...
1,999	0.124	0.014	...	0.274	1

Training set
75% of examples

Testing set
25% of examples

	Sentence Embeddings				Label
	0	1	...	767	
0	-0.215	-0.1402	...	0.201	1
1
2
3
4
5
6
...
1,499

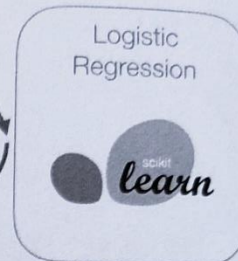
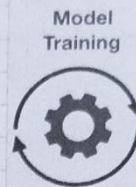
	Sentence Embeddings				Label
	0	1	...	767	
1,500
...
1,999	0.124	0.014	...	0.274	1

文本分类方法1: 特征抽取+其他模型

特征抽取流程③

Step #3: Train the logistic regression model using the training set

	Sentence Embeddings				Label
	0	1	...	767	
0	-0.215	-0.1492	...	0.201	1
1					
2					
3					
4					
5					
6					
...					
1,499					



文本分类方法2: bert fine-tuning

微调流程

- 修改run_classifier.py
 - 构建自己的processor 继承DataProcessor
 - processors 增加新构建的DataProcessor
 - Run!

注: bert的作用是预训练语言模型, 通过海量数据进行学习, 解决了标注样本少的问题。

源码地址

- <https://github.com/google-research/bert>
- <https://github.com/hanxiao/bert-as-service>

附论文地址:

BERT

- <https://arxiv.org/pdf/1810.04805.pdf>

作业

- 使用bert完成试题知识点标注项目。
 - 基于上节课的tf2 transformer结构实现bert(可选)
 - 尝试使用fine-tuning直接分类
 - 尝试使用extract features或 bert-as-service与其他模型结合使用

附：

- (1) attention 综述: <https://zhuanlan.zhihu.com/p/62136754>
- (2) bert 简介: <https://zhuanlan.zhihu.com/p/92849070>
- (3) 自然语言处理任务评价网站: <https://gluebenchmark.com/leaderboard>
- (4) bert 源码 github: <https://github.com/google-research/bert>
- (5) bert-as-service 包: <https://github.com/hanxiao/bert-as-service>
- (6) bert 论文: <https://arxiv.org/pdf/1810.04805.pdf>
- (7) bert 解读博客: https://blog.csdn.net/weixin_42001089/article/details/97657149
- (8) ai 竞赛开放平台: <https://www.flyai.com/>
- (9) 图神经网络工具: <https://github.com/thunlp/OpenKE>