# bert 在抽取任务中的效果

## 一、抽取式摘要

### 1、前期介绍

在预训练模型（elmo）出现之前，采用的 nlp 任务往往是：分词--->word2vec--->model--->loss，对于中文的分词，是为了能更好的学习词汇级别的特征；分字则对模型的效果有一定影响，因为输入没有包含词的特征。

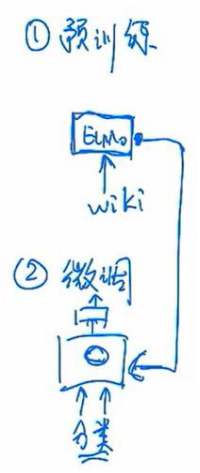
分词工具的效果会不会影响模型的效果？

答：会，但影响不大，例如：做情感分析，分词=(xxx,xxx,好看) or (xxx,xxx,好，看)，这两种分词效果导致“好看”比“好”的积极影响更浓，而“看”没有实际作用，会影响效果，但效果却不大。

预训练模型 elmo 为什么存在？

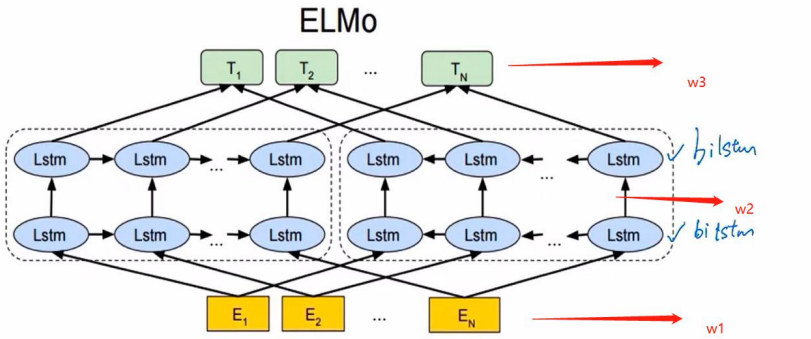
答：因为早期使用的 word2vec 不能解决词向量中一词多义的问题。

预训练模型出现后，往往采用：pre_training + finetune 的方式。由于小厂的数据少，标注成本又很高，所以采用预训练模型来微调，往往更为合算。




预训练+微调图

Elmo 模型介绍




Elmo 网络图

Elmo 采用两层的 bilstm,预训练数据为 wiki 数据，训练任务是自回归任务，即上一个词预测下一个词。

动态词向量构建：采用任务数据来进行微调，得到 w=a1*w1+a2*w2+a3*w3 作为词

向量来解决一词多义的问题。

取哪一层的向量较为合适？

答：如上图，根据对 elmo 的理解，任务是 LM，则 w3 向量则更倾向于 LM 任务，例如句子通顺特征、句子级别特征等；w2、w1 更倾向于底层信息，例如词性特征、字特征等基本特征。因此分类任务喜欢拿顶层特征，ner 喜欢拿底层特征。

为什么要有词向量？

答：以前的模型提取特征不太好，词向量则是给模型一定的先验知识，让模型学得更好，自 transformer 出来后，就不需要词向量，因此，该模型本来很深，能学到所有得知识。

## 2、Bert 结构

### (1) Bert Input

Bert 输入采用 subwords 的形式，英文拆成字根，例如 playing-->play、##ing;中文拆成一个个字。Subwords 的方式：BPE、wordpiece,基本原理都是字母拼接的最好概率。

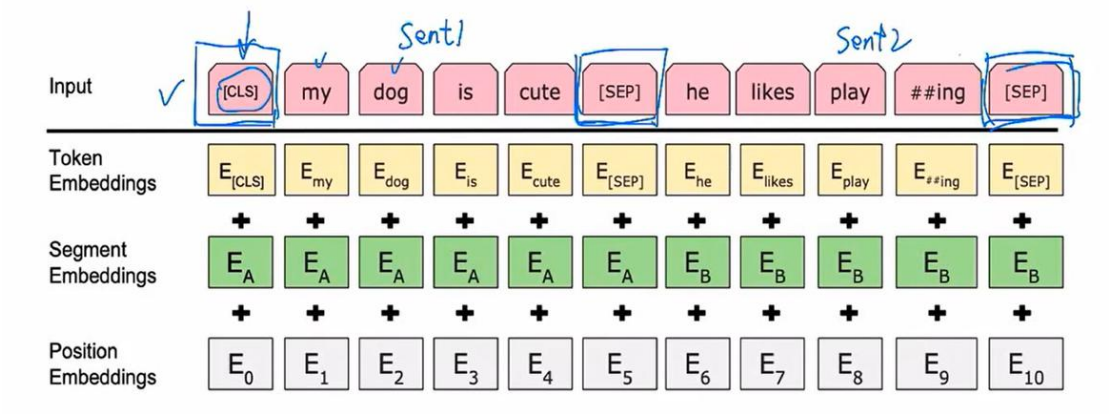
拆成 subwords 的好处?

答：字典减少、避免 OOV。

Input 的预训练输入基本格式：[cls] sent1 [SEP] sent2 [SEP]

微调的 Input 可以为：[cls] sent1 [SEP] sent2 [SEP] 时间特征 [SEP] 其他特征 [SEP]

### (2) Bert Embedding




embedding 图

根据 input 构建三个重要的 embedding：token_embedding、segment_embedding、position_embedding。Bert 不能很好的学到位置信息，因此加入位置编码，位置编码采用 0--->max_len 来对应，表示绝对位置编码。Segment_embedding 是用来训练 nsp 任务。

位置编码都有哪些？

答：

### (3) Bert 预训练任务

① MLM

任务描述：

MLM 是一个类似于完形填空的遮蔽语言模型，任务是对 input 进行 mask,从而预测出原本的 input 的任务。

输入数据：将 input 中的 15% MASK 操作

MASK 操作：

- 80%---> [mask]代替
- 10%---> 其他词代替
- 10%---> 不变

输出数据：原始的 input

② NSP

任务描述：判断两个句子是否是同一上下文。

输入数据：正样本：[sent1,sent2]

负样本：[sent1,sent3]

输出数据：正样本--->1

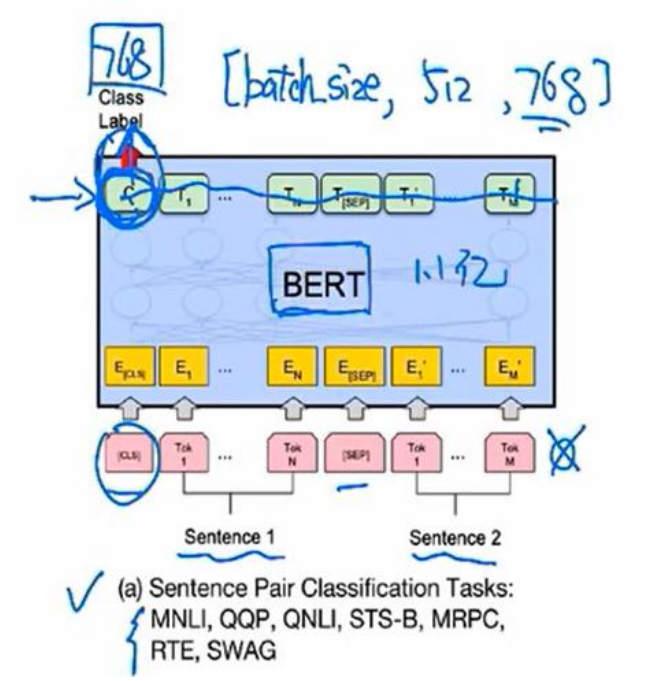
负样本--->0

③ Loss

Loss=loss_mlm+loss_nsp

## (4) Fine-Tuning

①句子对分类任务




输入 Input：[CLS] sent1 [SEP] sent1

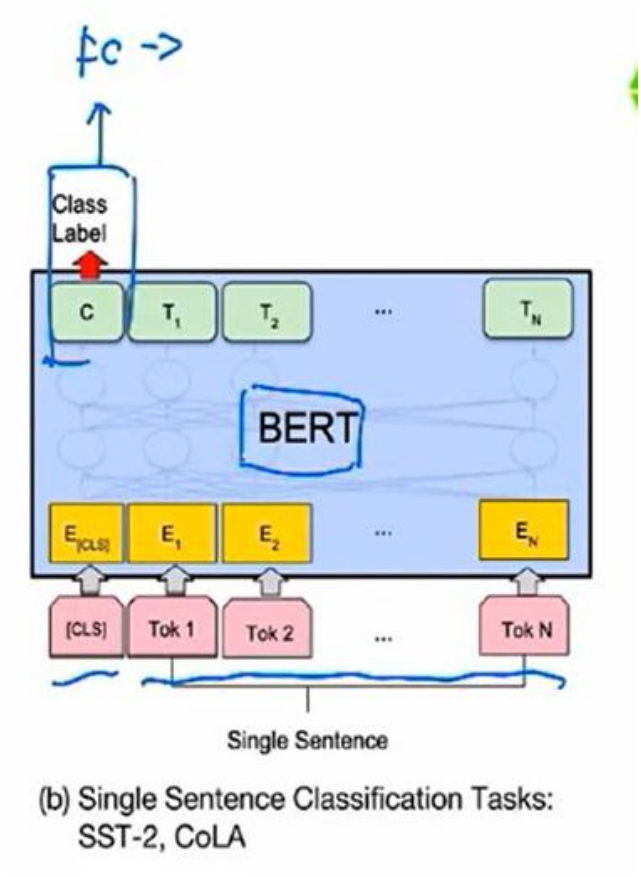
输出：类别 label

任务：采用[CLS]来进行分类 or 所有输出接 textcnn，因为[CLS]能表示句子的含义。

为什么[CLS]能表示句子的含义？

答：因为 transformer 的结构的关系，[CLS]可以和任何 token 关联，更何况预训练采用[CLS]来进行 NSP 任务。

②单文本分类任务



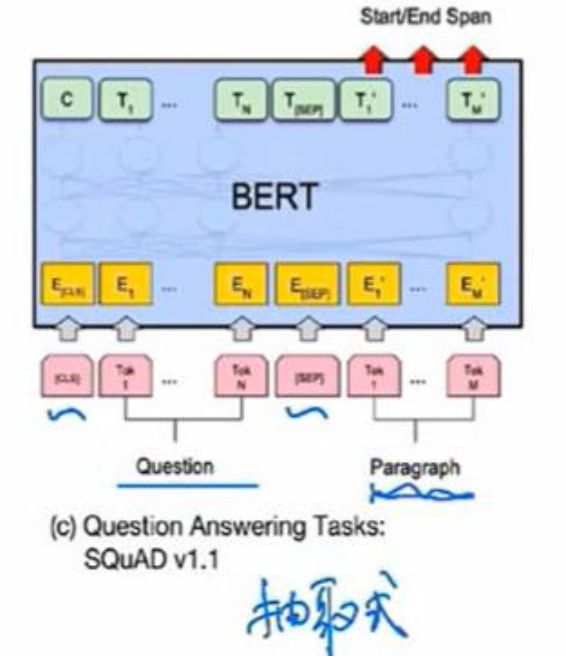

(b) Single Sentence Classification Tasks: SST-2, CoLA

输入 input：[CLS] sent1

输出：类别 label

任务：采用[CLS]来进行分类 or 所有输出接 textcnn

③抽取式 QA 任务




(c) Question Answering Tasks: SQuAD v1.1

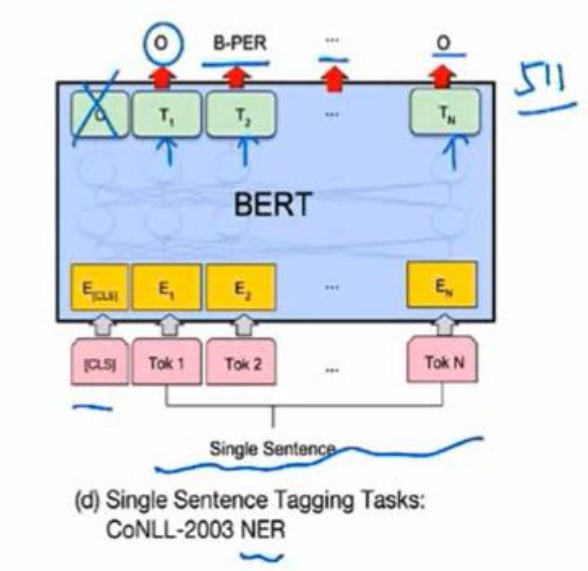
抽取式

输入 input：[CLS] question [SEP] paragragh

输出：(start、answer_len)或者(start,end)

任务：针对 answer 在 paragragh 对应的开始位置的 token 进行 start 预测，结束位置进行 end 预测。

④NER 任务




输入 input：[CLS] sent1

输出：非[CLS]的对应的 token 位置接 crf，进行 ner 任务

⑤经验

- Bert 上面可以加 cnn、pooling 结构，最后再接 fc
- bert 上面不建议使用 rnn、attention(因为 bert 用了 attention 就能学到这些结构,加了意义不大)
- Bert 的输入可以为:[cls][cls1] xxx，来进行多重分类，CLS 用来预测最外层分类，CLS1 用来预测分完类之后内层的分类。
- 微调：1、bert 模型不更新，只外层更新；2、bert 模型选择部分更新；3、bert 的 1-6 层是偏词等基本特征，7-12 层是偏任务特征，可以选择某几层来连接最外层

(5) Bert 代码运用

- tranformers 包调用，主要三大部分：config（模型参数）、tokenization(将输入文本转 bert 格式的 id,token 处理)、model（bert 模型结构实例化对象）

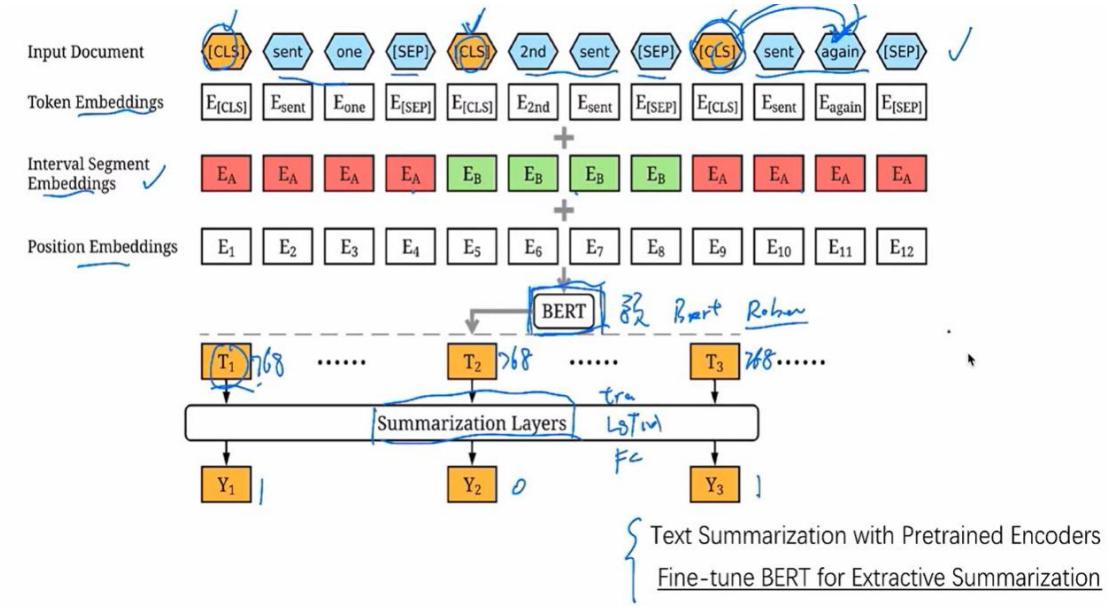
```
--->(bert):BertModel(
            (embedding):BertEmbedding(
                (word_embedding):Embedding(30000,768))
            )
```

## 二、Fine_tuning BERT for summarization

### 1、Bertsum

bertsum 做摘要是采用抽取的方式，bertsum 主要在英文上做过，中文也

可以尝试，具体结构如下图分析：




输入 input:将原文按每一个句子进行划分，输入格式：[CLS] sent1 [SEP] [CLS] sent2 [CLS] sent3 [SEP] [CLS] sent3 [SEP]

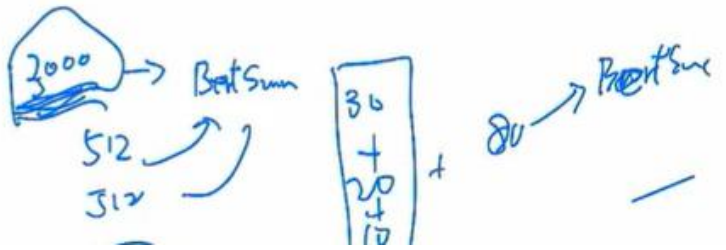
Embedding： token_embedding 与原来一致， segment_embedding 采用交叉式， postion_embedding 和原来一致。

外层： 只取[CLS]对应部分的输出，接到 summarization 层后进行分类，判断每一个句子是否为摘要。

效果：r1=43.25 , r2=20.24 , rl=39.63

句子长度>512，该什么解决？

答：采用递归的方式处理，例如文本长度 3000，先将 30003 拆分成多个 512,放入 bertsum 中，得到多个 bertsum 的结果再整合，放入 bertsum 再摘要一次。



## 2、传统方式

缺点：

①繁杂的数据预处理

需要加入更多的特征，例如字符级别特征，词级别特征等

②基于 word_embedding

太大（词向量）、太稀疏

③模型能力较低

④需要大量训练数据

优点：

推理阶段：速度快

3、Bert 方式

优点：

①先进的 Tokenizer

WordPiece

②模型能力更强

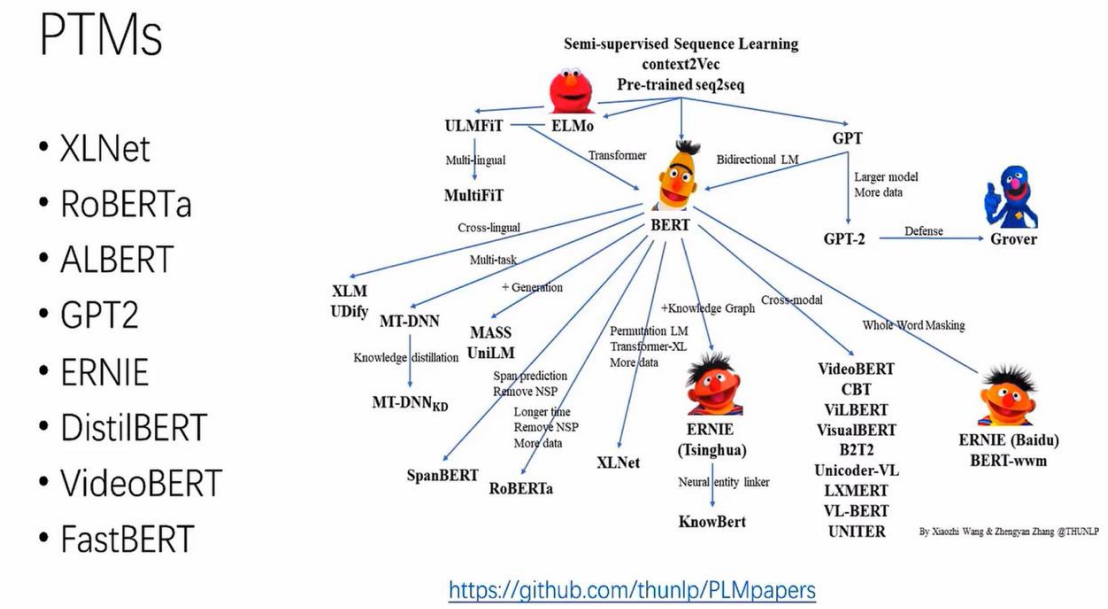
Transformer 的 encode 部分

③解决数据问题

预训练任务已经喂入大量数据训练

## 三、PTMs 代码应用

最近的预训练模型：




https://github.com/thunlp/PLMpapers

各个预训练模型的任务等数据：

| PTMs | Architecture† | Input | Pre-Training Task | Corpus | Params | GLUE‡ | FT?§ |
|---|---|---|---|---|---|---|---|
| ELMo [129] | LSTM | Text | BiLM | WikiText-103 | | | No |
| GPT [136] | Transformer Dec. | Text | LM | BookCorpus | 117M | 72.8 | Yes |
| GPT-2 [137] | Transformer Dec. | Text | LM | WebText | 117M ~ 1542M | | No |
| BERT [35] | Transformer Enc. | Text | MLM & NSP | WikiEn+BookCorpus | 110M ~ 340M | 81.9* | Yes |
| InfoWord [88] | Transformer Enc. | Text | DIM+MLM | WikiEn+BookCorpus | ≈BERT | 81.1* | Yes |
| RoBERTa [111] | Transformer Enc. | Text | MLM | BookCorpus+CC-News+OpenWebText+ STORIES | 355M | 88.5 | Yes |
| XLNet [202] | Two-Stream Transformer Enc. | Text | PLM | WikiEn+ BookCorpus+Giga5 +ClueWeb+Common Crawl | ≈BERT | 90.5§ | Yes |
| ELECTRA [24] | Transformer Enc. | Text | RTD+MLM | same to XLNet | 335M | 88.6 | Yes |
| UniLM [38] | Transformer Enc. | Text | MLM& NSP | WikiEn+BookCorpus | 340M | 80.8 | Yes |
| MASS [154] | Transformer | Text | Seq2Seq MLM | *Task-dependent | | | Yes |
| BART [98] | Transformer | Text | DAE | same to RoBERTa | 110% of BERT | 88.4* | Yes |
| T5 [138] | Transformer | Text | Seq2Seq MLM | Colossal Clean Crawled Corpus (C4) | 220M ~ 11B | 89.7* | Yes |
| ERNIE(THU) [207] | Transformer Enc. | Text+Entities | MLM+NSP+dEA | WikiEn + Wikidata | 114M | 79.6 | Yes |
| KnowBERT [130] | Transformer Enc. | Text | MLM+NSP+EL | WikiEn + WordNet/Wiki | 253M ~ 523M | | Yes |
| K-BERT [107] | Transformer Enc. | Text+Triples | MLM+NSP | WikiZh + WebtextZh + CN-DBpedia + HowNet + MedicalKG | ≈BERT | | Yes |
| KEPLER [189] | Transformer Enc. | Text | MLM+KE | WikiEn + Wikidata/WordNet | | | Yes |
| WKLM [195] | Transformer Enc. | Text | MLM+ERD | WikiEn + Wikidata | ≈BERT | | Yes |