# boolqQA Report

赵妍，朱大卫，王之睿

January 2021

## 目录

# 1 组员信息

- 组长：王之睿，1800017714
- 组员：赵妍，1800017790
- 组员：朱大卫，1800012713

# 2 任务描述与数据集分析

## 2.1 任务描述

Yes/No 问题是 QA 中的经典问题。特别的，本任务的每条数据包含文本、文章标题、基于文本的问题，以及问题的答案（yes/no）。在每个测试样例中，要求根据文本内容、文章标题（不必须）、基于文本的问题，给出问题的答案 yes/no 。参照现有的 NLP 技术，对本任务的抽象大致可以分为以下几类：

1. 基于问答系统（Question & Answering）的策略。
2. 基于文本蕴含（Textual Entailment）的策略。
3. 基于文本分类（Text Classification）的策略。

我们的实验主要是基于文本分类模型和文本蕴含模型。

## 2.2 数据集分析

数据数量：train.jsonl: 9427，dev.jsonl: 3270 ，test.jsonl: 1000
准确率：majority-baseline：62.17%，human acc：90%，bert-mnli + finetune：80.4%
篇章平均长度: 103.48，最大长度: 981，最小长度: 2.
问题平均长度: 8.89，最大长度: 21，最小长度: 2.

| Question Topic | | | |
|---|---|---|---|
| Category | Example | Percent | Yes% |
| Entertainment Media | Is You and I by Lady Gaga a cover? | 22.0 | 65.9 |
| Nature/Science | Are there blue whales in the Atlantic Ocean? | 22.0 | 56.8 |
| Sports | Has the US men's team ever won the World Cup? | 11.0 | 54.5 |
| Law/Government | Is there a seat belt law in New Hampshire? | 10.0 | 70.0 |
| History | Were submarines used in the American Civil War? | 5.0 | 70.0 |
| Fictional Events | Is the Incredible Hulk part of the avengers? | 4.0 | 87.5 |
| Other | Is GDP per capita same as per capita income? | 26.0 | 65.4 |
| **Question Type** | | | |
| Category | Example | Percent | Yes% |
| Definitional | Is thread seal tape the same as Teflon tape? | 14.5 | 55.2 |
| Existence | Is there any dollar bill higher than a 100? | 14.5 | 69.0 |
| Event Occurrence | Did the great fire of London destroy St. Paul's Cathedral? | 11.5 | 73.9 |
| Other General Fact | Is there such thing as a dominant eye? | 29.5 | 62.7 |
| Other Entity Fact | Is the Arch in St. Louis a national park? | 30.0 | 63.3 |

图 1: Question Statistics

# 3 数据处理

## 3.1 词表构建与词嵌入

常见的 Tokenization 的方法有：bpe、unigram、bigram 等，bpe 算法的切割粒度在词根词缀级别，能够有效降低词表大小，减小 embedding 层的参数数量，提高训练速度，且能更好地应对词表外的词汇。unigram 策略则控制切割的粒度在单词级别，并可能伴随有取词干等策略。本实验中为使用 GloVe 预训练词向量（具体是 globe.6B.100d，即词向量维度为 100 维），而采用了 unigram 的切词方法。

在下述实验中，词表大小为 43068，不同于常规的基于频率降序的筛选方式，对于训练集中出现的 token，只要它在 GloVe 中有对应的词向量，我们就将其纳入到词表中。经统计分析，该词表对于文本的覆盖率接近 100%。

## 3.2 标题信息的使用

本实验就标题的使用对实际结果是否有提升做了消融实验（Ablation Experiment），详见 6.1 节。特别地，在需要使用标题信息时，我们将其与文本拼接，放在文本的第一句。

# 4 封闭式模型实验

所谓封闭式模型实验，在训练阶段模型所用的参数不能是由其他数据集和预训练好的，也不能引入其他的数据集。在封闭式环境下，我们组探究了 LSTM_ATTN，BIMPM,ESIM 和 ABCNN 四种相关的模型，并做了相应的结果分析。

## 4.1 实验环境

- 计算资源：1080Ti * 1
- 实验工具：pytorch 1.4.0, NLTK, GloVe

## 4.2 LSTM-ATTN

基于文本蕴含策略，LSTM-ATTN 是 seq2seq 的代表模型。encoder 端利用 attention 机制，分别对文本和问题过一层 Bi-LSTM 进行编码：

$$passage\_hidden_1, ..., passage\_hidden_n = Bi - LSTM(p_1, p_2, ..., p_n)$$

$$question\_hidden_2, ..., question\_hidden_m = Bi - LSTM(q_1, q_2, ..., q_m)$$

经过 lstm 编码之后再用 attention 求每个 token 的加权分数:

$$\alpha_i = softmax(tanh(M * passage\_hidden_i))$$

$$\beta_j = softmax(tanh(M * question\_hidden_j))$$

并将分数与它们的向量做加权求和, 加权求和得到的分数与句末得到的 hidden 拼接之后，然后将问题和文本的编码向量拼接起来:

$$passage\_hidden = concat(\sum_{i=1}^{n} \alpha_i * passage\_hidden_i, passage_hidden_n)$$

$$question\_hidden = concat(\sum_{j=1}^{m} \beta_j * question\_hidden_j, question\_hidden_m)$$

decode 端是简单的二分类器。以下是网络概念图与 Loss & Accu 随 Epoch 变化的趋势图：

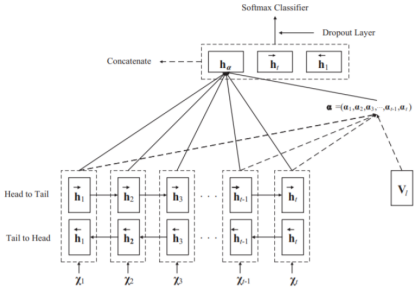


图 2: LSTM-ATTN

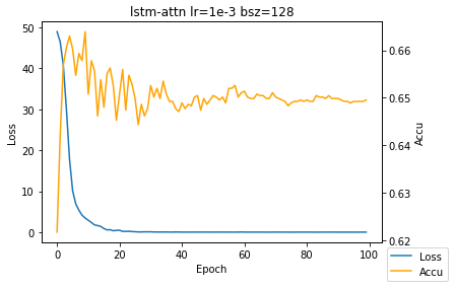


图 3: LSTM-ATTN Loss & Accu with Epoch

### 4.3 ABCNN (Attention-Based Convolutional Neural Network)

基于文本蕴含策略，ABCNN 是 Siamese Network 的一种，它用带 Attention 机制的 Bi-CNN 来分别编码 passage 与 question，得到两个编码向量。再利用分类器和这两个向量输出预测结果。

我们的项目采用了 ABCNN-3 模型，这个模型共在两处引入了 Attention 机制：

1) 首先在 input 层，即在卷积操作前就进行 Attention，若用 $F_{i,r} \in R^{d*s}$ 表示句子向量，则 attention 矩阵计算公式如下：

$$A_{i,j} = match\_score\left(F_{0,r}[:,i], F_{1,r}[:,j]\right)$$

令 x,y 分别为 question 与 passage 的 embedding，我们的模型采取的 match-score 计算方式为：

$$\frac{1}{1+|x-y|}$$

2）其次是在 pooling 层，即对卷积层的输出结果进行加权。该 attention 矩阵的计算方式与前面类似，对于第一个句子 (question)，第 j 个词向量的权重为 $a_{0,j} = \sum A[j,:]$；对于第二个句子 (passage)，第 j 个词向量的权重为 $a_{1,j} = \sum A[:,j]$. 在进行池化的时候对应词的词向量需要乘上权重取和。

在具体使用中，我们可以将 k 个结构相同的 ABCNN 进行堆叠训练，即将 k 个堆叠 ABCNN 的输出向量拼接起来，作为最终的特征向量。

图 4 是 ABCNN 网络概念图，图 5 是其 Loss & Accu 随 Epoch 变化的趋势图。

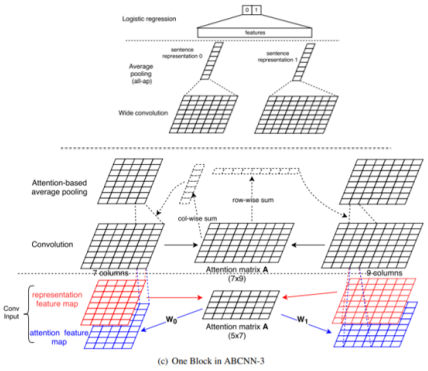


图 4: ABCNN

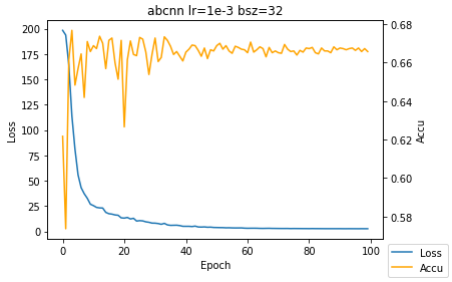


图 5: ABCNN Loss & Accu with Epoch

### 4.4 ESIM (Enhanced Sequential Inference Model)

基于文本蕴含策略，采用 matching-aggregation 结构，这种结构可以捕捉到两个句子之间的交互特征。ESIM 是一种专为自然语言推断而生的加强版 LSTM。用句子间的注意力机制 (intra-sentence attention)，来实现局部的推断，进一步实现全局的推断。

模型具体包括四层：输入编码层（Input Encoding Layer，是个双向 LSTM）、局部推理层（Local Inference Layer）、推理组合层（Inference Composition Layer，也是个双向 LSTM）和预测层（Prediction Layer）。

(1) 输入编码层：将 question 与 passage 输入到同一个 BiLSTM 中，使句子中的学每个单词具有其上下文含义。

(2) 局部推理层：采用点积，求解两个输入句子的相似度矩阵。令 $\bar{a}$ 和 $\bar{b}$ 分别表示编码后的 question 与 passage，令 $e_{i,j}$ 表示相似度矩阵，则计算两个句子间的相互表示，从而进行局部推理：

$$\tilde{\mathbf{a}}_i = \sum_{j=1}^{\ell_b} \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{\ell_b} \exp\left(e_{ik}\right)} \overline{\mathbf{b}}_j, \forall i \in [1, \ldots, \ell_a]$$

$$\tilde{\mathbf{b}}_j = \sum_{i=1}^{\ell_a} \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{\ell_a} \exp\left(e_{kj}\right)} \overline{\mathbf{a}}_i, \forall j \in [1, \ldots, \ell_b]$$

最后对局部推理进行增强，$m_a$ 与 $m_b$ 分别表示两个句子增强后的局部信息表示：

$$\mathbf{m}_a = [\overline{\mathbf{a}}; \tilde{\mathbf{a}}; \overline{\mathbf{a}} - \tilde{\mathbf{a}}; \overline{\mathbf{a}} \odot \tilde{\mathbf{a}}]$$

$$\mathbf{m}_b = [\overline{\mathbf{b}}; \tilde{\mathbf{b}}; \overline{\mathbf{b}} - \tilde{\mathbf{b}}; \overline{\mathbf{b}} \odot \tilde{\mathbf{b}}]$$

(3) 推理组合层：用 BiLSTM 将已经得到的局部信息 $m_a$ 与 $m_b$ 进行编码，得到其局部信息的上下文表示向量，对输出进行最大池化与平均池化操作，再将两个特征向量进行拼接，得到最终的特征向量。

(4) 预测层：将得到的最终特征向量输入到全连接层，这里有两层全连接层：第一层采用 tanh 激活函数，第二层采用 softmax 激活函数。

图 6 是 ESIM 的网络概念图，图 7 是其 Loss & Accu 随 Epoch 变化的趋势图。

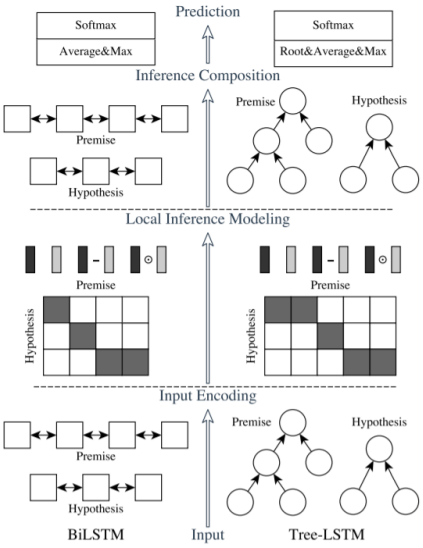


图 6: ESIM (Left)

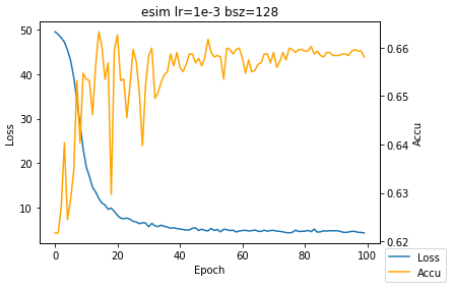


图 7: ESIM Loss & Accu with Epoch

### 4.5 BIMPM (bilateral multi-perspective matching)

基于文本蕴含策略，与 ESIM 一样采用 matching-aggregation 结构。BIMPM 同样是分别编码文本和问题，但 ABCNN 只是针对两个句子向量做匹配，对于两个句子之间的交互信息利用的很少，而 BIMPM 首先对两个句子之间的单元做匹配 (比如各自经过 LSTM 处理后得到的不同 time step 的输出)，匹配结果通过一个神经网络 (CNN 或 LSTM) 转化为一个向量，然后再做分类，这种方式能更好的捕捉到两个句子之间的交互特征。

BIMPM 的具体网络结构如下：

(1)Word Representation Layer 采用预训练的 word embedding 将每个词转化为一个向量 word embedding，同时还将每个词中的每个字母输入 LSTM 得到 char embedding,。将 word embedding 与 char embedding 拼接作为最后的 word representation。

(2)Context Representation Layer：将 question 与 passage 输入同一个 Bi-LSTM，得到每时间步的上下文向量。

(3)Matching Layer：采用了双向多角度匹配（Bilateral multi-perspective matching），从而比较两个句子的每个上下文向量，获取两个句子细粒度的联系信息。首先参考论文 [5] 中定义了新的 cosine function：

$$m = f_m(v_1, v_2; W)$$

$$m_k = cosine(Wk \circ v1, Wk \circ v2)$$

$$W \in R^{l \times d}$$

其中，W 就是 multi-perspective，维度为 l×d，l 是 perspective 个数，d 是向量 $v_1$ 与 $v_2$ 的维度。在计算两个向量的 cosine 相似度时，需要分别乘向量 $W_k$ 作为权重；一共有 l 个 perspective，则共需要做 l 次 cosine 相似度计算，最后生成 $m = [m_1, ..., m_k, ..., m_l]$。的向量，向量中的每个元素都是特定 perspective 下的结果。

之后，参考论文 [5] 中提出了 4 种匹配策略：(a)Full-Matching，将一个句子的每个 Bi-LSTM 时间步与另一个句子的最后一个时间步计算相似度；(b)Maxpooling-Matching：将一个句子的每个时间步与另一个句子的每个时间步计算相似度，只取最大值；(c)Attentive-Matching，每个时间步

的 cosine 相似度，再生成相关性矩阵，将一个句子的每个时间步与另一个句子加权求和计算相似度；(d)Max-Attentive-Matching，类似 Attentive-Matching，但不再加权求和，而是直接用 cosine 值最大的 embedding 作为 attentive vector。在我们的模型中，我们将 4 种匹配策略都采用了，并将最后的结果进行了拼接。

图 8 是 BIMPM 的网络概念图，图 9 是其 Loss & Accu 随 Epoch 变化的趋势图。

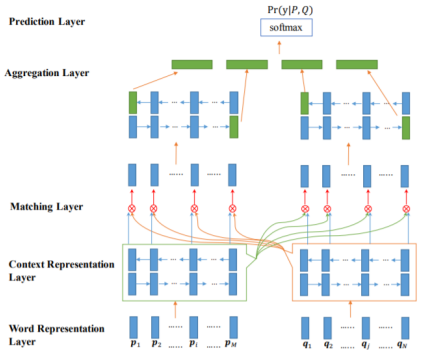


图 8: BIMPM

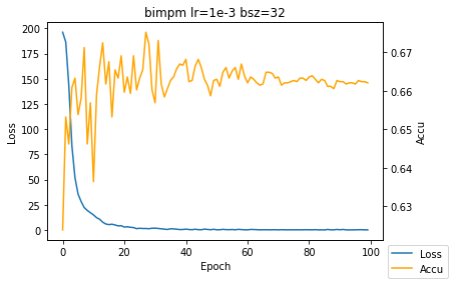


图 9: BIMPM Loss & Accu with Epoch

### 4.6 Vote 机制

通过对封闭式网络模型的结果分析，我们发现 ABCNN，LSTM-ATTM 与 BIMPM 这三个模型的表现较好，于是我们选取这三个模型，并引入了 vote 机制。首先我们分别将这三个网络进行独立训练，并独立地对 validation 数据集、test 数据集进行预测，就像是对 yes/no 两种类别分别进行投票。此时不同的网络可能会有不同的预测（投票）结果，我们选择票数最多的那个结果作为最终输出。比如对于一条数据，三个网络都输出 yes，则最终结果为 yes；若两个网络输出 no，一个网络输出 yes，则最终结果为 no.

实验结果证明，应用 vote 机制后，能使准确率有所提升（表 1）。

### 4.7 结果统计

见表 1：封闭式模型结果统计。

从结果上来看，几种模型的表现接近，其中最好的是 ABCNN 模型，最高可以在 valid 上面达到 0.676 的 accu。除此之外 BIMPM 的表现也非常的 comparable。这可能得益于二者的网络层次更深。几种模型都是在刚开始的一段波动之后最终达到稳定并呈现下降趋势，可能是因为略微的过拟合，然而该任务本身的拟合难度较高，所以过拟合现象并不明显。

## 5 开放式模型实验

开放式模型试验中，可以选用预训练模型进行训练，该实验中我们组分别用 Bert，Roberta 和 Roberta-large 三种模型进行实验。数据通路方面，封闭式模型中，我们分别对文本和问题进行 encode，而在 bert 及以下两个 roberta 的预训练模型中，我们用分隔符拼接文本和问题（如果使用到标题的话，直接将标题放在文本首句），然后将拼接后的整句作为网络的输入。因为 bert 和 roberta 都是相似的思想和模型，所以在这里我们统一介绍一下预训练模型的整体框架，后面只是对不同模型有个简单的界定，不再详细介绍。

表 1: 封闭式模型结果统计

| Method | Title | Yes | | | No | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| | Use/Not use | P | R | F1 | P | R | F1 | |
| Majority-Baseline | – | – | – | – | – | – | – | 62.17% |
| LSTM-ATTN | Not use | 0.8057 | 0.7039 | 0.7514 | 0.4430 | 0.5811 | 0.5028 | 66.85% |
| LSTM-ATTN | Use | 0.7732 | 0.7113 | 0.7410 | 0.4842 | 0.5951 | 5215 | 66.39% |
| ABCNN | Not use | 0.8023 | 0.7107 | 0.7537 | 0.4632 | 0.5877 | 5181 | 67.40% |
| ABCNN | Use | **0.8618** | **0.6930** | **0.7683** | **0.3727** | **0.6213** | **0.4659** | **67.68%** |
| BIMPM | Not use | 0.8160 | 0.7075 | 0.7579 | 0.4454 | 0.5957 | 0.5097 | 67.58% |
| BIMPM | Use | 0.7718 | 0.7240 | 0.7471 | 05166. | 0.5793 | 0.5462 | 67.52% |
| ESIM | Not use | 0.7806 | 0.7094 | 0.7433 | 0.4745 | 0.5682 | 0.5172 | 66.48% |
| ESIM | Use | 0.7757 | 0.7097 | 0.7412 | 0.4786 | 0.5649 | 0.5182 | 66.33% |
| Vote | Not use | **0.8790** | **0.7047** | **0.7822** | **0.3945** | **0.64649** | **0.4952** | **69.57%** |
| Vote | Use | 0.8229 | 0.7119 | 0.7634 | 0.4527 | 0.6087 | 0.5192 | 68.29% |

## 5.1 预训练模型 Bert

通常情况 transformer 模型有很多参数需要训练。譬如 BERT BASE 模型: L=12, H=768, A=12, 需要训练的模型参数总数是 12 * 768 * 12 = 110M。这么多参数需要训练，自然需要海量的训练语料。如果全部用人力标注的办法, 来制作训练数据, 人力成本太大。受《A Neural Probabilistic Language Model》论文的启发，BERT 也用 unsupervised 的办法，来训练 transformer 模型，其核心思想是先用文章预训练通用模型，然后再根据具体应用，用 supervised 训练数据，精加工（fine tuning）模型，使之适用于具体应用。BERT 提出一种新的预训练目标：遮蔽语言模型（masked language model，MLM），来克服上文提到的单向性局限。MLM 的灵感来自 Cloze 任务（Taylor, 1953）。MLM 随机遮蔽模型输入中的一些 token，目标在于仅基于遮蔽词的语境来预测其原始词汇 id 与从左到右的语言模型预训练不同，MLM 目标允许表征融合左右两侧的语境，从而预训练一个深度双向 Transformer。

除了遮蔽语言模型之外，本文作者还引入了一个“下一句预测”（next sentence prediction）任务，可以和 MLM 共同预训练文本对的表示。许多重要的下游任务，如问答（QA）和自然语言推理（NLI）都是基于理解两个句子之间的关系，这并没有通过语言建模直接获得。在为了训练一个理解句子的模型关系，预先训练一个二进制化的下一句测任务，这一任务可以从任何单语语料库中生成。具体地说，当选择句子 A 和 B 作为预训练样本时，B 有 50%的可能是 A 的下一个句子，也有 50%的可能是来自语料库的随机句子。Bert 的模型如下图所示：

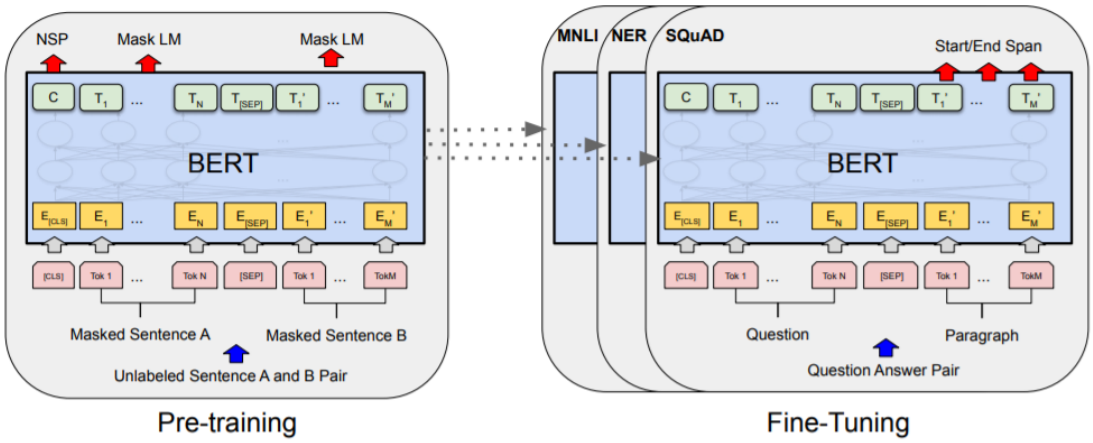


图 10: BERT Model

从模型图片里面我们可以很直观地看到，bertmodel 在 MLM 和 NSP 两个任务上进行预训练，得到的模型参数再在下游任务中进行 finu tuning 从而可以在下游任务中得到良好的表现，我们的实验也是基于此完成的。

## 5.2 预训练模型 Roberta

Roberta 与 Bert 的核心思想相同，区别在于 roberta

1. 去掉下一句预测 (NSP) 任务

2. 动态掩码。BERT 依赖随机掩码和预测 token。原版的 BERT 实现在数据预处理期间执行一次掩码，得到一个静态掩码。而 RoBERTa 使用了动态掩码：每次向模型输入一个序列时都会生成新的掩码模式。这样，在大量数据不断输入的过程中，模型会逐渐适应不同的掩码策略，学习不同的语言表征。

3. 文本编码。Byte-Pair Encoding（BPE）是字符级和词级别表征的混合，支持处理自然语言语料库中的众多常见词汇。原版的 BERT 实现使用字符级别的 BPE 词汇，大小为 30K，是在利用启发式分词规则对输入进行预处理之后学得的。Facebook 研究者没有采用这种方式，而是考虑用更大的 byte 级别 BPE 词汇表来训练 BERT，这一词汇表包含 50K 的 subword 单元，且没有对输入作任何额外的预处理或分词。

## 5.3 实验环境

- 计算资源：主要是 1080Ti * 1
- 实验工具：pytorch 1.4.0, NLTK, GloVe, transformers

## 5.4 BERT-BASE

网络参数：L（网络层数）=12, H（隐藏层维度）=768, A（Attention 多头个数）=12, vocab_size = 30522

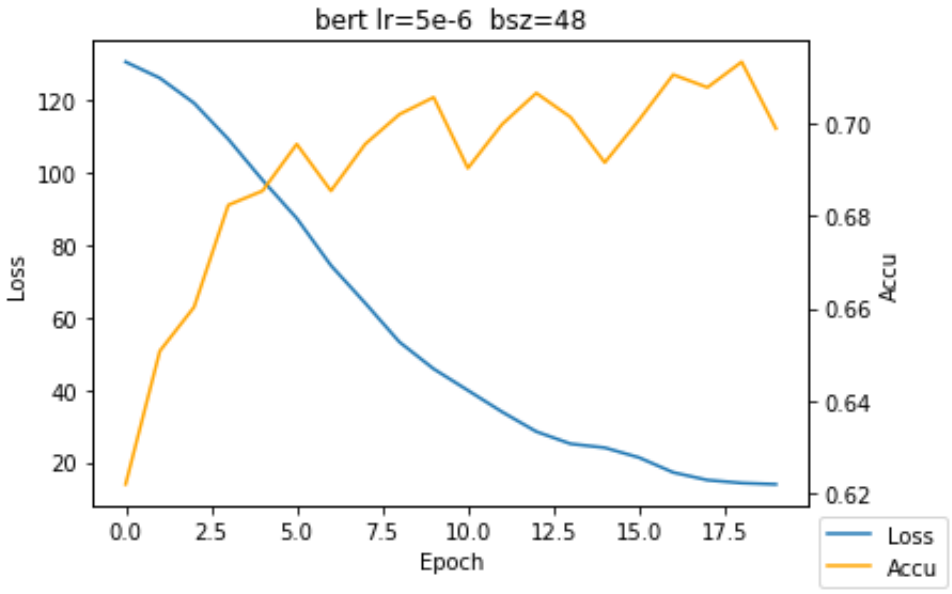


图 11: BERT Loss & Accu with Epoch

### 5.5 ROBERTA-BASE

网络参数: L（网络层数）=12, H（隐藏层维度）=768, A（Attention 多头个数）=12,vocab_size = 50265

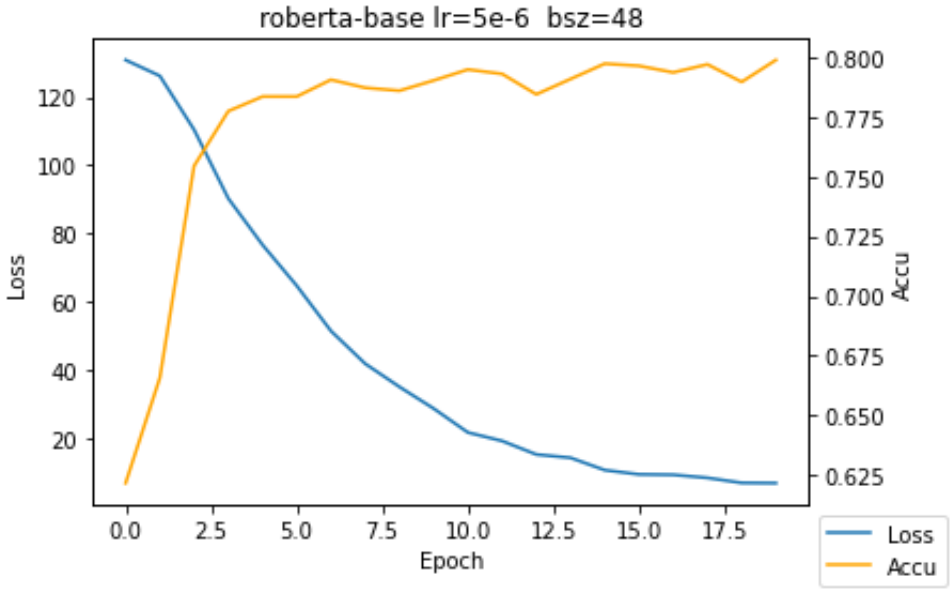


图 12: ROBERTA-BASE Loss & Accu with Epoch

### 5.6 ROBERTA-LARGE

网络参数: L（网络层数）=24, H（隐藏层维度）=1024, A（Attention 多头个数）=16,vocab_size = 50265

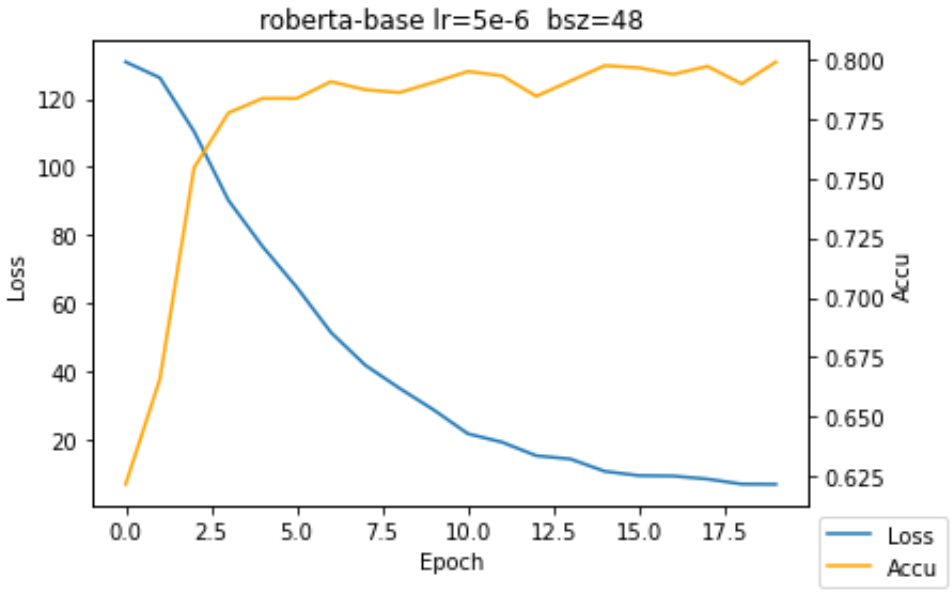


图 13: ROBERTA-LARGE Loss & Accu with Epoch

### 5.7 结果统计

开放式模型的详细结果见表 2: 开放式模型结果统计。我们的模型中，效果最好的是不使用 title 的 Roberta-Large 模型，在初始 lr=5e-6，batchsize=16 的情况下 finetune 20 个 Epoch 后，准确率达到 85.75%，其它 P、R、F1 值也大都超过其它模型。值得注意的是，原论文中给出的最好的模型（bert+MultiNLI）的准确率为 80.4%，所以我们的模型在性能上超过了原论文。另一方面，当前该数据集上的 SOTA 已经超过 90%，这也是原论文中人工标注员的准确率。鉴于资源有限，我们没有对 T5-11B 进行尝试。

表 2: 开放式模型结果统计

| Method | Title | Yes | | | No | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| | Use/Not use | P | R | F1 | P | R | F1 | |
| boolqPaper-BestResult | – | – | – | – | – | – | – | **80.4%** |
| Human-Annotator | – | – | – | – | – | – | – | **90%** |
| SOTA(T5-11B) | – | – | – | – | – | – | – | **91%** |
| Bert | Not use | 0.8022 | 0.7527 | 0.7767 | 0.5667 | 0.6355 | 0.5991 | 71.31% |
| Bert | Use | 0.8249 | 0.7453 | 0.7831 | 0.5368 | 0.6510 | 0.5884 | 71.59% |
| Roberta | Not use | 0.8662 | 0.8202 | 0.8426 | 0.6880 | 0.7578 | 0.7212 | 79.88% |
| Roberta | Use | 0.8392 | 0.8396 | 0.8394 | 0.7365 | 0.7359 | 0.7362 | 80.03% |
| Roberta-Large | Not use | **0.8864** | **0.8846** | **0.8855** | **0.8100** | **0.8127** | **0.8113** | **85.75%** |
| Roberta-Large | Use | 0.8839 | 0.8830 | 0.8835 | 0.8076 | 0.8089 | 0.8083 | 85.50% |

# 6 消融实验

## 6.1 关于 title 对于本任务的帮助

本实验的详细结果已呈现在表 1 和表 2 中。从表中数据来看，title 的使用对于性能没有显著的提升。当然，这也与我们对于 title 过于粗糙的使用有关。如果能更好的利用 title 里蕴含的信息，也许能对性能有所提升。

## 6.2 关于预训练模型在本任务中的效果

预训练模型在本实验中的效果是显著的，这体现在以下几个方面：

1. 预训练模型优于非预训练模型。
2. 更充分的预训练模型能有更好的效果。一方面，roberta 是在 bert 的基础上做了更充分预训练，可以看到它的性能远高于 bert；另一方面，roberta-large 的数据规模高于 roberta-base，它的性能比起 roberta-base 也有可观的提升。

# 7 总结与展望

总结部分：

此次实验探究过程中，我们小组分别从两种模型类型分别多次比较实验得到了最终结果，加深了对模型认识的同时，也对 NLI（natural language inference）和 QA 任务有了更深层次的简介，同时还很好的完成了小组合作，受益良多。

展望部分：

本次实验还可以在以下几个方面做进一步的工作：

1. 数据处理方面：可以尝试 bpe 编码，自行训练词向量，这样得到的词表更精炼，在应对词表外的词语时也能有更好的性能；关于 title 信息，也可以探索更好的方法来使用它；另外，还可以做一些基于规则的数据增强，来为封闭式模型的端到端训练提供更充分的数据。
2. 模型使用方面：封闭式模型这边可以尝试基于现有工具的 pipeline 模型；可以尝试基于 MultiNLI 等文本蕴含数据集训练的 bert 或 roberta 模型，预期经过这种 domain-specific 的训练后，模型能在本任务上有更好的性能；可以尝试 T5 等更庞大的预训练模型；可以尝试集成学习（Ensemble），比如整合几个封闭式模型，预期会达到更好的效果；最后，受时间与资源限制，我们没能很好的调整模型的各种超参数，这也是可以进一步挖掘的地方。
3. 建模策略方面：本实验的封闭式部分主要是借鉴文本蕴含任务的处理方式，开放式部分主要是借鉴文本分类任务的处理方式。而事实上，也可以尝试利用问答系统、阅读理解中其它成熟的方法。

# References

[1] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

[2] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Bert: Pre-training of deep bidirectional transformers for language understanding. 2019.

[3] Zhenhua Ling Si Wei Hui Jiang Diana Inkpen Qian Chen, Xiaodan Zhu. Enhanced lstm for natural language inference. *ACL*, 2017.

[4] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Computer Science*, 2015.

[5] Naman Goyal Jingfei Du Mandar Joshi Danqi Chen Omer Levy Mike Lewis Luke Zettlemoyer Veselin Stoyanov Yinhan Liu, Myle Ott. Roberta: A robustly optimized bert pretraining approach. *ICLR*, 2019.

[6] Radu Florian Zhiguo Wang, Wael Hamza. Bilateral multi-perspective matching for natural language sentences. *IJCAI*, 2017.