电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 基于深度学习的智能问答系统研究

专业学位类别	工 程 硕 士
学　　号	201722040618
作 者 姓 名	李敬鑫
指 导 教 师	殷勇　教授

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

基于深度学习的智能问答系统研究

（题名和副题名）

李敬鑫

（作者姓名）

指导教师 殷勇 教 授

电子科技大学 成 都

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工 程 硕 士

工程领域名称 电子与通信工程

提交论文日期 2020.7 论文答辩日期 2020.7.15

学位授予单位和日期 电子科技大学 2020 年 8 月

答辩委员会主席 袁学松 教授

评阅人 袁学松 教授 张平 副教授

注 1：注明《国际十进分类法 UDC》的类号。

Research on Intelligent Question-Answering System Based on Deep Learning

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Discipline: **Master of Engineering**

Author: **Jingxin Li**

Supervisor: **Prof. Yong Yin**

School: **School of Electronic Science and Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：[signature]　　日期：2020年 7 月 16 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：[signature]　　导师签名：[signature]

日期：2020 年 7 月 16日

摘　要

近年来，随着人工智能领域尤其是深度学习的快速发展，自然语言处理在人类的学习工作与生活中正发挥着越来越重要的作用。智能问答的出现使得一大批应用或服务涌现出来，如百度的小度，阿里的天猫精灵，小米的小爱同学，苹果的 Siri。智能问答是知识的一种表现形式，知识是人类最宝贵的财富，所以研究智能问答对于人类知识的提取与运用有着重要意义，让机器明白的比人类更多，回答的更准确，这是智能问答研究者的共同目标。但是智能问答领域还不是特别成熟，很多人想当然的把科幻水平误认为是当今的发展水平，所以要提高智能问答系统的准确率，让机器更智能，回答的问题更接近于人类甚至超越人类的水平，让人类对机器的回答更满意。本文的研究就是基于此目的。

首先介绍了自然语言处理的基本模型，从独热编码，词嵌入到 Seq2seq(sequence to sequence)，然后介绍了由 Transformer 带火的预训练模型尤其是 BERT(Bidirectional Encoder Representation from Transformers)，标志着自然语言处理进入了黄金时期。自然语言处理的概念或基础为之后深入研究智能问答打下了坚实的基础。

然后在基本模型双向注意力流即 BiDAF(Bidirectional Attention Flow)模型的基础上，使用数据集 SQuAD 2.0(Stanford Question-Answering Dataset)作为训练数据，使用精确匹配 EM 值(Exact Match)和模糊匹配 F1 的值作为评价指标，EM 值代表模型预测的答案和标准答案是否完全一样，EM 值越大就表示模型预测的结果和标准答案越接近；F1 参数是根据模型给的答案与标准答案的重合度求出一个 0 到 1 之间的分数，这个分数就是精确率和召回率的调和平均。经过训练，得到 BiDAF 模型在 SQuAD 2.0 数据集上的表现是 EM 值为 58.60，F1 为 61.95。

最后，使用预训练模型 ALBERT(A Lite BERT)进行研究，共分为 5 个模型进行研究，模型一直接使用 ALBERT 加输出层，模型二在 ALBERT 的基础上增加了高速网络，模型三则增加了门控循环单元 GRU(Gate Recurrent Unit)和注意力层，模型四是 GRU 加高速网络，模型五是使用 ALBERT-xxlarge 版本。经过训练，在使用 ALBERT 基础版本的模型一、二、三、四中，模型一直接加输出层的效果最好，比基本 BiDAF 模型 EM 值提高了 17.68，F1 值提高了 16.73。5 个模型中 ALBERT-xxlarge 版本效果最好，较 BiDAF 模型的 EM 值提高了 24.35，F1 值提高了 23.35。

本论文的创新点是使用了现今最强大的 ALBERT 模型进行研究，而不是用前

一两年很火的 BERT 或 XLNet 进行研究。在 ALBERT 的基础上，提出了添加不同层来进行研究的方法，极大的提高了在 SQuAD 2.0 版本数据集上的准确率。其中参数最多的 ALBERT-xxlarge 模型效果最好，比基础模型的 EM 值和 F1 值分别提高了 41.55%和 37.69%，效果显著。

关键词：智能问答，自然语言处理，预训练模型，ALBERT，SQuAD 数据集

ABSTRACT

In recent years, natural language processing is playing an increasingly important role in human's life. The emergence of intelligent Question-Answering has led to a large number of applications or services, such as Baidu's Xiaodu, Ali's Tmall Genie, Xiaomi's Xiaoai classmate, and Apple's Siri. However, the field of intelligent question answering is substandard yet. Improving the accuracy of the intelligent question answering system makes the machine more intelligent and answer the questions closer to or even beyond the level of humans. The research in this article is based on this purpose.

Firstly, I introduced the basic model of natural language processing, from one-hot encoding, word embedding to sequence to sequence, and then I introduced the pre-trained model originated from Transformer, especially BERT which is Bidirectional Encoder Representation from Transformers, marking the natural language processing has entered a golden period. The concept and model of natural language processing has laid a solid foundation for further research on intelligent question answering.

Then, based on the bidirectional attention flow of the basic model, that is BiDAF, the Bidirectional Attention Flow model. I used the dataset SQuAD 2.0 which is Stanford Question-Answering Dataset as training data, and used the values of Exact Matching and F1 as evaluation indicators. EM stands for whether the predicted answers exactly the same as the groud-truth answers. The greater the EM value, the closer the model predicts to the standard answer; the F1 parameter which based on the coincidence of the answer given by the model and the standard answer is to find a score between 0 and 1. This score is a harmonized average of the accuracy rate and recall rate. After training, the performance of the BiDAF model on the SQuAD 2.0 dataset is that the EM value is 58.60 and the F1 is 61.95.

Finally, the pre-trained model ALBERT(A Lite BERT) is used for research. There are 5 models for it. The first model uses ALBERT directly plus the output layer, the second model adds a highway network on the basis of ALBERT, and the third model adds the Gated Recurrent Unit and Attention layer, the fourth model uses GRU(Gate Recurrent Unit) and highway network, the fifth model is to use ALBERT-xxlarge version. After training, in the model 1, 2, 3, and 4 which uses the ALBERT basic version, the first model which directly adding the output layer to model has the best effect, which is 17.68 higher

than the basic BiDAF model on EM value, and the F1 value is increased by 16.73. Among the five models, the ALBERT-xxlarge version has the best effect. The EM value is increased by 24.35 than the BiDAF model, and the F1 value is increased by 23.35.

The innovation of the paper is using the most powerful model nowadays, rather than BERT or XLNet. On the basis of ALBERT, a method of adding different layers for research is proposed, which improves the accuracy of SQuAD 2.0 greatly. The ALBERT-xxlarge model with the most parameters has the best effect, which is 41.55% and 37.69% higher than the EM value and F1 value of the basic model, respectively. The effect is remarkable.

Keywords: Intelligent Question-Answering, Natural Language Processing, pre-trained model, ALBERT, SQuAD dataset

目 录

第一章 绪 论

1.1 自然语言处理简介

语言或文字是一种抽象的不能计算的非结构化的数据，而自然语言处理可以将其转化为可以计算的结构化的数据。这样我们就可以做日常中见到的智能问答，机器翻译[1]，情感分析[2]，命名体识别[3]等任务。

1.1.1 独热编码

独热编码即 One-Hot 表示方法是自然语言处理中最直接也是过去很长一段时间很常用的向量表示方法[4]。独热编码是将一个词表示成一个长度为词表大小的向量，在这个长向量中，只有一位是 1，剩下的所有的位都用 0 表示。比如狗表示成[1,0,0,0]，猫表示成[0,1,0,0]，猴子表示成[0,0,1,0]，老虎表示成[0,0,0,1]。这样可以很直观清晰的得到词的表示，但是一种语言的词汇是上万甚至上十万计数的，如果把整个词表的大小都写进去，只有 1 位是 1，其他的几万位或者十几万位都是 0，这样就造成这个高维向量稀疏的缺点。

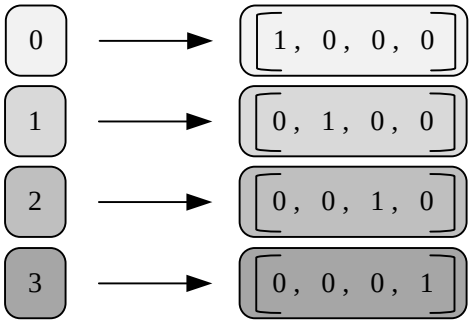


图 1-1 独热编码

独热编码的另一个特点是具有正交性，任意两个向量的内积都是 0，造成了任意两个词之间都是孤立的，猫和狗，猫和石头的距离都是 0，无法表示出在语义上的词与词之间的关系，这也是独热编码最大的缺点。

1.1.2 词嵌入简介

既然独热编码无法表示语义信息，那么如何能够表示语义信息呢？哈里斯提出了分布式假说，即上下文信息相同，那么词的语义也相同[5]。词嵌入表示就是分布式表示的一种[6]，Word Embedding 改进了独热编码高维稀疏的缺点，将向量的元素由整形变成浮点型，把原来稀疏的维度压缩，嵌入到了一个更小的维度空间，这样就把单词映射到语义空间或向量空间中，空间中的一个点就表示一个单词。

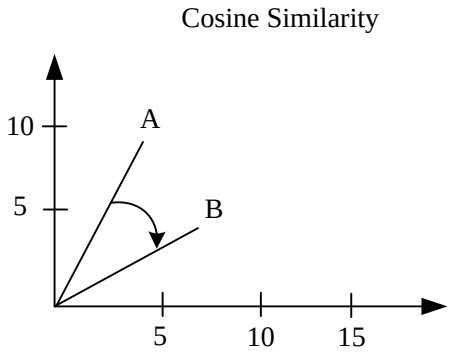


图 1-2 Cosine 函数计算相似度

如果两个词的语义相似，那么在语义空间中的点的距离也是相近的。相似度用 cos 函数来计算。

$$\cos(\mathrm{A,B})=\frac{A\bullet B}{\|A\|\|B\|}=\cos\theta \tag{1-1}$$

1.1.3 Word2vec 简介

Word2vec 是词嵌入的一种表示方法[7]，是 Mikolov 在 2013 年提出的。它包括连续词袋模型 CBOW 和 Skip-gram 模型。

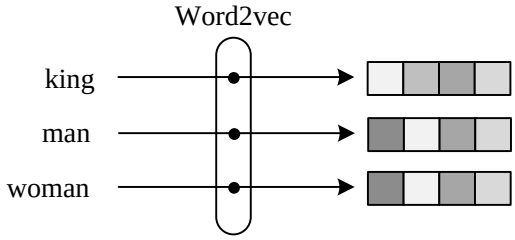


图 1-3 word2vec

1.1.3.1 CBOW 模型

连续词袋模型 CBOW[8]就是根据上下文来推测某个词的概率。相当于一句话挖空了一个词，预测挖空的这个词。

以“我爱自然语言处理”为例，假设要预测“自然”，输入是剩下的“我”“爱”“语言”“处理”4 个词的独热编码，每个独热编码的向量维度是 1×V，V 是词表的大小。每个词乘以 V×E 大小的权重矩阵得到 4 个 1×E 的隐藏层表示，E 是词嵌入的大小，是一个超参数。所有的 1×E 的向量会取平均，这样相当于只算一个隐藏层，这个过程也被称为线性激活函数。然后再跟 E×V 的矩阵相乘得到 1×V 的输出层，输出层中的数值代表每个词的概率。输出层跟预测词“自然”的独热编

码做比较计算损失 loss。

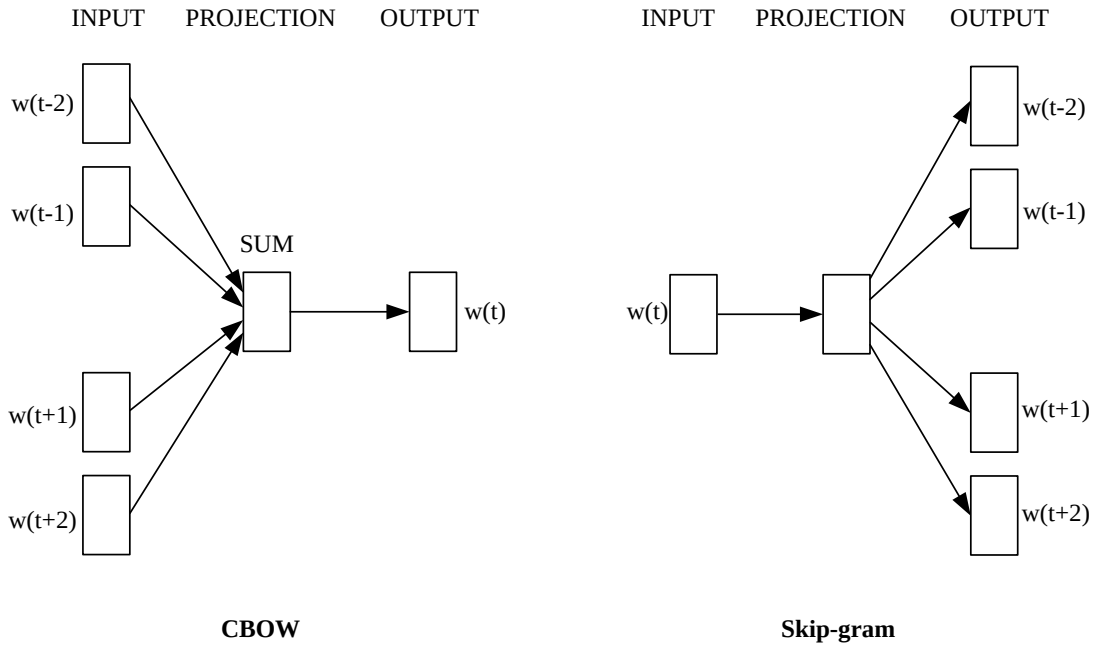


图 1-4 CBOW 模型和 Skip-gram 模型[9]

1.1.3.2 Skip-gram 模型

Skip-gram[9]模型与 CBOW 模型正好相反，是根据一个词来预测它的上下文。

还是以“我爱自然语言处理”为例，这次“自然”作为输入向量，维度是 1×V。隐藏层的维度是 V×E，隐藏层的输出是 1×E，然后将输出传入 softmax 层，得到 1×V 的向量，这个向量中的每个值是目标词在那个位置的概率分数。

Word2vec 的缺点也很明显，词与向量是一对一的关系，不能够解决多义词的问题。

1.1.4 语言模型

语言模型[10]是对一段文本是否合理的概率的估计，用来判断一句话从语法上是否通顺。一句话是由若干个单词组成，给定一个词序列$W = w_1, w_2, \ldots, w_n$，计算这些单词能够组成一句正确且语法通顺的话的联合概率是 P(W)：

$$P(W) = P(w_1, w_2, w_3, ...w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_m|w_1, w_2, ...w_m) \quad (1\text{-}2)$$

把句子的概率分解成句子中每个单词的条件概率的乘积。例如：他出生在中国，他能够说一口很流利的____。预测的这个词很大概率是“中文”而不是“英语”，“德语”等外语。这个例子就是常见的智能手机输入法预测下一个词的应用场景。

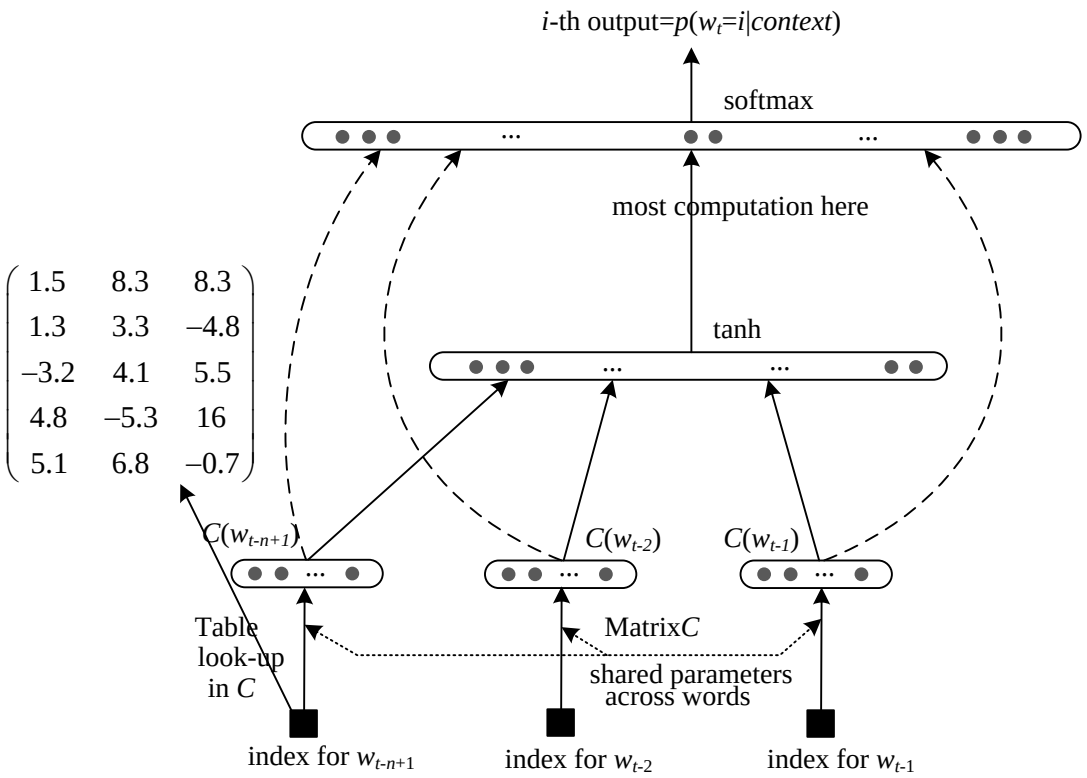


图 1-5 前馈神经网络模型[11]

但是会出现一个问题，如果句子比较长，那么最终的条件概率的计算将会很困难，这也是通常说的维数灾难。为了解决这个问题，约定此单词只和其前面的 N 个单词有关，再往前的单词就没有关系，所以计算每个词基于前面 N 个词的条件概率，这也就是 N-gram 语言模型[12]。

常用的语言模型主要有两类：统计语言模型和神经网络语言模型。N-gram 是统计语言模型，CBOW 和 Skip-gram 是神经语言模型。神经语言模型还包括现在最常用的预训练语言模型，这些将在第二章详细讲述。

其实（1.1.2）节的词嵌入表示是语言模型的副产品，把一个句子前面的单词转化成词向量，经过隐藏层与 softmax 函数来预测下一个单词，假如预测结果很好，那么就认为转换成的词向量能够很好的表示出单词的意思。这样就得到一个单词表，每个单词都能够在这个单词表中找到其对应的向量，我们称这个过程为词嵌入即 Word Embedding。

1.1.5 循环神经网络

词嵌入不能解决一词多义的问题，为了利用上下文信息，提出循环神经网络即 RNN[13]。假设从左到右读一个句子，读取的第一个词是 x_1，取出第一个词并输入到神经网络层，神经网络有一个隐藏层用来预测输出，RNN 做的是继续读取句子

中的第二个词，比如读取 x_2 的时候，不是仅仅考虑 x_2，还会把第一步的计算结果作为输入信息的一部分，特别是第一步的激活数值传递到第二步，在下一个步骤 RNN 输入第三个词，同时利用第二个的结果得到第三个词。直到预测最后一个值为止。

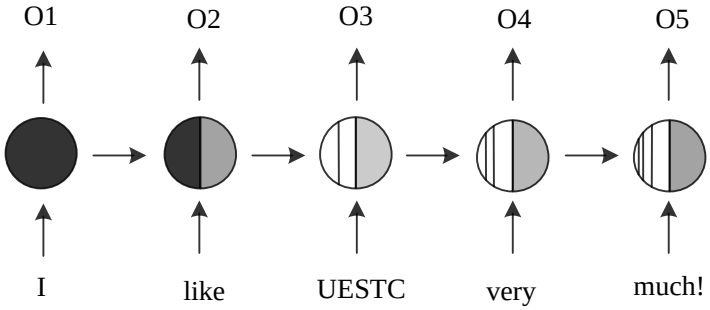


图 1-6 循环神经网络

在每一步中，循环神经网络都会传递激活值，传递到下一步供 RNN 继续使用，在第 0 步用自己设定的通常是零向量的激活值。循环神经网络的特点是从左到右读取数据，同时每步的参数是共享的。

循环神经网络的前向传播过程：

$$a^{\langle t\rangle} = g\left(\mathrm{W}_a\left[a^{(t-1)}, x^{(t)}\right] + b_a\right) \tag{1-3}$$

$$\hat{y}^{(t)} = g\left(\omega_y a^{(t)} + b_y\right) \tag{1-4}$$

W_a 表示将前一个时刻的权重与这个时刻输入的权重并列放置，且是用来计算 a 类型或者说激活值的。W_y 表示它计算的是 y 这种类型的权重矩阵。

循环神经网络的一个缺点是只用到了序列之前的信息，比如预测第三个词只用到了第一个和第二个词的信息，后面第四个第五个等词的信息都没有用到，也就是只利用了上文而忽视了下文。双向循环神经网络即 Bi-LSTM[14]可以解决这个问题。

另一个缺点是只能一个时刻接一个时刻的计算，不能并行计算，所以训练速度很慢。而且如果循环神经网络的路径很长，就会出现梯度消失，所以普通的循环神经网络很难学习到长距离的信息，为了学习长距离的信息，提出了 LSTM 长短期记忆和 GRU 门控循环单元。

1.1.6 长短期记忆 LSTM

长短期记忆 LSTM[15]是 RNN 的一种，主要区别是长短期记忆 LSTM 有 2 个传输状态，一个是 c^t（cell state）另一个是 h^t（hidden state）。循环神经网络中的 h^t 就

是长短期记忆中的c^t，这也是长短期记忆的关键。Cell 的状态贯穿整个 LSTM，只发生小的线性作用。长短期记忆利用门的结构来精确控制加入或移除信息到 cell 状态。

$$\tilde{c}^{<t>} = \tanh\left(W_c\left[a^{<t-1>}, x^{<t>}\right] + b_c\right) \tag{1-5}$$

门是由 sigmoid 函数和一个点乘组成，sigmoid 输出[0,1]之间的数，0 代表不通过，1 代表所有的都通过，其余的数代表通过的比例，0.5 代表通过一半，这样就组成了门。

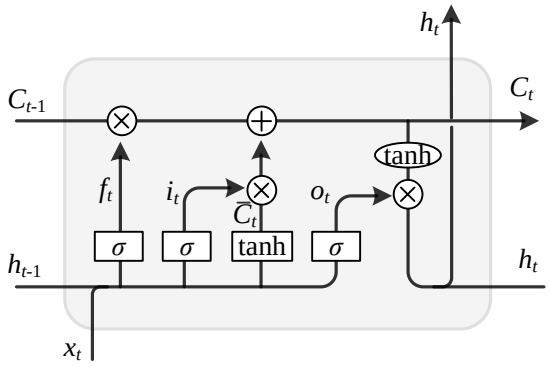


图 1-7 LSTM 结构[16]

长短期记忆有 3 个门：遗忘门，更新门，输出门。

（一）遗忘门Γ_f：

遗忘门决定从 cell 状态中舍弃信息。将之前隐藏层状态的信息和现在输入的信息传递到 sigmoid 函数，输出[0,1]之间的值，值越靠近 0 意味着越有可能舍弃信息，也就是遗忘；值越靠近 1 意味着越有可能保留信息。

$$\Gamma_f = \sigma\left(W_f\left[a^{<t-1>}, x^{<t>}\right] + b_f\right) \tag{1-6}$$

（二）更新门Γ_u：

更新门是为了更新 cell 的状态。将之前隐藏层状态和现在的输入传入 sigmoid 函数，产生的[0,1]之间的值将决定哪个值将会被更新。0 代表不重要，1 代表重要信息。同时把隐藏层状态的信息和现在的输入传入 tanh 函数，得到[-1,1]之间的值来帮助调节网络。然后用 sigmoid 的输出乘以 tanh 的输出。Sigmoid 的输出将决定 tanh 的输出的重要信息被保留多少。

$$\Gamma_u = \sigma\left(W_u\left[a^{<t-1>}, x^{<t>}\right] + b_u\right) \tag{1-7}$$

（三）输出门Γ_o：

输出门将决定下一个隐藏层状态。隐藏层状态不仅包含了之前输入的信息，同样也会被用于预测。首先要做的是把之前的隐藏层状态的信息和现在的输入传入 sigmoid 函数，然后将新更改的 cell 状态传入 tanh 函数，tanh 函数的输出与 sigmoid 函数相乘输出决定隐藏层状态应该保留什么信息。这个输出就是隐藏状态。然后新的 cell 状态和隐藏状态被传送到下一步。

$$\Gamma_o = \sigma\left(W_o\left[a^{<t-1>}, x^{<t>}\right] + b_o\right) \tag{1-8}$$

Cell state 通过更新门和遗忘门更新旧的 Cell state C_{t-1}：

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \tag{1-9}$$

最终通过输出门得到长短期记忆的输出：

$$a^{<t>} = \Gamma_o * \tanh c^{<t>} \tag{1-10}$$

1.1.7 门控循环单元 GRU

门控循环单元 GRU[17]是由 Kyunghyun Cho 等人于 2014 年提出的，是长短时记忆的一种简化结构，它将遗忘门和更新门合并成一个新的更新门，门控循环单元使用一个门就可以进行遗忘和选择记忆，长短期记忆则需要 2 个门控。所以 GRU 的模型更简单，也比长短期记忆更受欢迎。

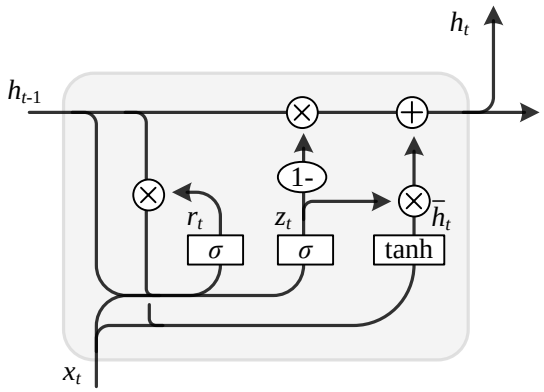


图 1-8 GRU 结构[16]

门控循环单元 GRU 有 2 个门，一个是重置门 r_t，另外一个是更新门 u_t。

通过上一个传输进来的 h^{t-1} 和当前输入 x^t 来获取两个门状态。同长短期记忆中的门计算相似，其中重置门为：

$$r_t = \sigma\left(W_r \cdot \left[h_{t-1}, x_t\right]\right) \tag{1-11}$$

更新门为：

$$\mathrm{u}_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right]\right) \tag{1-12}$$

Sigmoid 函数可以将数据转换成[0,1]范围内，充当门控信号。

首先通过重置门来获得重置后的数据：

$$h^{t-1'} = h^{t-1} * r \tag{1-13}$$

然后计算候选隐藏层状态 $\tilde{h}_t$，先将 $h^{t-1'}$ 与 x^{t} 拼接，再通过一个 tanh 激活函数得到：

$$\tilde{h}_t = \tanh\left(W \cdot \left[r_t * h_{t-1}, x_t\right]\right) \tag{1-14}$$

$\tilde{h}_t$ 主要由当前输入 $\mathbf{x_t}$ 构成，对于上个隐藏层状态 $\mathrm{h}^{\mathrm{t-1}}$ 有选择的添加。r_{t} 用来控制保存信息的多少，如果接近于 0，则几乎不保存之前的隐藏层状态信息，只保存当前输入的信息。

最后更新门 u_t 来控制上一个时刻隐藏层状态 h_{t-1} 中遗忘的信息的比例，当前时刻隐藏层状态 $\tilde{h}_t$ 的信息加入的比例，最后得到最终的隐藏层状态信息 h_t：

$$h_t = \left(1 - u_t\right) * h_{t-1} + u_t * \tilde{h}_t \tag{1-15}$$

在这里 $1\text{-}u_t$ 可以认为是长短期记忆中的遗忘门，忘记上一个隐藏层输出的一些信息。$u_t * \tilde{h}_t$ 表示对当前输入的某些信息进行选择记忆。更新门的操作就是忘记上一步传进来的某些信息并且加入现在输入的某些信息。

1.1.8 Seq2seq

Seq2seq[18]是 Sequence to Sequence 即序列到序列的模型。Seq2seq 包含编码器 Encoder 和解码器 Decoder，最常见的应用是机器翻译。

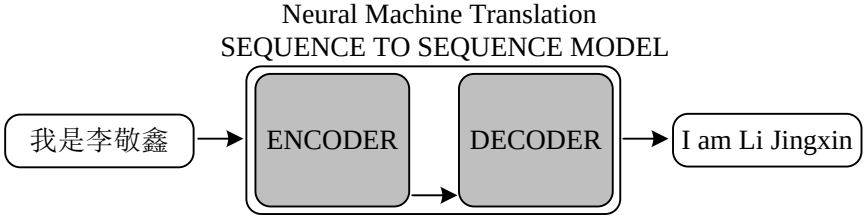


图 1-9 Seq2seq 应用在机器翻译

给定一句话如“我是李敬鑫”输入到编码器 Encoder，把所有的信息合并到最后一个 Encoder 状态，这个 Encoder 状态包含了所有的语义，然后对语义进行解码，解码用到解码器 Decoder。解码第一个的时候，输入是编码器的输出和特殊的<SOS>来预测第一个解码输出的词，然后把第一个输出作为第二个词的输入，再加上第一

个时刻的隐状态，得到第二个词，依次输出对应的英文翻译，直到<EOS>为止，或者超过设定的长度就截断。 <SOS>和<EOS>分别代表序列的开始和序列的结束。

编码器和解码器可以任意选择，可以选择循环神经网络 RNN，双向循环神经网络 BiRNN，卷积神经网络 CNN，长短期记忆 LSTM 或门控循环单元 GRU。

这种编码器-解码器的模型有一个问题，它需要用固定长度的上下文向量来编码源句子的输入的所有语义信息，也就是编码器状态很难用一个向量来编码所有的。这样就引入了注意力机制[19]。

1.1.9 注意力

注意力来自于计算机视觉领域，思路来源于人的视觉，人类在看世界的时候，会有注意力，注意力的范围有限，人不可能把整个世界都看全，所以人的注意力是有选择的。深度学习中的注意力机制就是借鉴的人类的注意力，关键就是在众多信息中能够选择关注更重要的信息。

以 Seq2seq 为例，编码器编码“深度学习”的语义信息，然后将所有的隐藏层状态信息全部传递给解码器。带有注意力的解码器在做解码输出之前要做以下几步：

1. 查看一系列接受的编码器隐藏层状态 h^1，h^2，h^3，h^4，每个编码器的隐藏层状态信息与输入序列中的某个词最相关。
2. 与每个隐藏层状态做内积，内积越大说明关注越多，内积需要用 softmax 函数将其变成[0,1]范围内的值，softmax 后的概率加权每个词的向量，求和得到上下文向量 C^0，C^1，C^2，C^3。
3. 将第 2 步的输出作为输入，传入解码器 Decoder 进行预测输出。

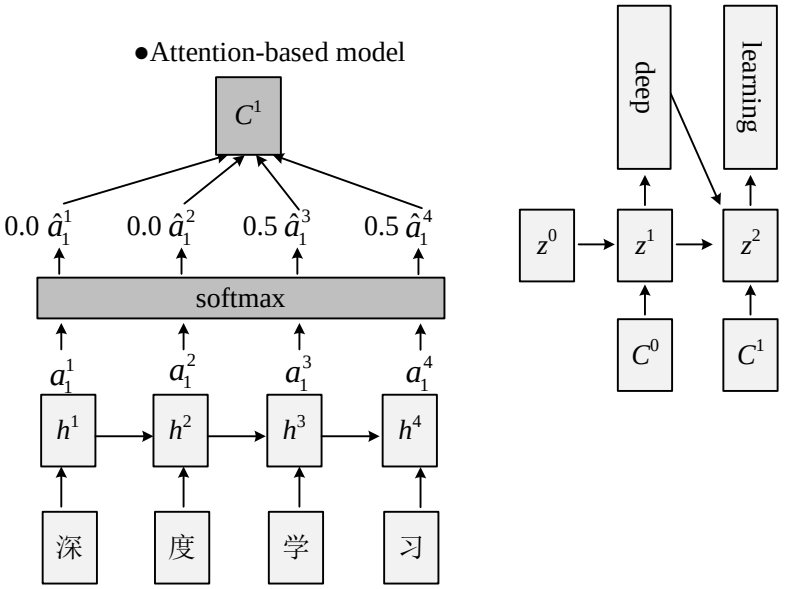


图 1-10 注意力模型

Attention 的问题是需要考虑整个句子，但是需要解码器，也就是需要两个句子，比如一个中文句子，一个英文句子。但是有的任务仅有一个句子，所以提出自注意力机制来解决这个问题。

1.2 Question-Answering 智能问答系统简介

1.2.1 智能问答系统概述

智能问答在当今工作、学习、生活中应用广泛。例如常见的苹果手机中的 Siri，谷歌的 Google Assistant[20]，亚马逊的 Alexa[21]，天猫精灵[22]，小度小度等应用或设备；京东智能客服[23]在线回答常问的简单问题，给企业大大节省了人力，提高了消费者购买的效率，减少了商家的成本；面向闲聊的聊天机器人，例如微软小冰[24]，用来打发时间；甚至通过对话可以进行中文、英语等语言的口语练习；以及面向如不同特定领域的智能问答[25]，节约专业人士的时间，提高了他们的效率。

1.2.2 智能问答系统分类

按照领域来分类，智能问答系统可以分成面向“特定领域”的智能问答[25]，例如面向教育、法律、体育、医学等专业领域的问答。面向“开放领域”的智能问答[26]，开放领域不限定范围，相当于对一整套百科全书的内容进行提问，上至天文地理，下至 Python、深度学习，都可以作为开放领域的提问范围。

按照答案的来源来分类，智能问答系统可以分为基于阅读理解的智能问答，如本文就是基于斯坦佛问题回答训练集 SQuAD[27]的回答，针对阅读理解，又继续细

分成抽取式的智能问答[28]和生成式的智能问答[29]。基于问答对的智能问答系统，比如常见问题集 FAQ 的问答[30]。

1.2.3 智能问答系统方法

智能问答系统最常用的方法就是信息检索[31]。首先用自然语言处理技术处理问题，对问题进行分析，确定问题的类别。得到搜索的关键词，通常是这个问题的主干，权重较高的部分。然后通过信息检索得到第一步的关键词，对文章中的关键词进行匹配，相似程度高的作为候选，得到几个候选后进行排序。最后是提取答案，答案的提取要先找到答案的段落，先找到段落再精确的进行查找答案。找到段落后，问答系统需要精确找到段落中的答案，通过对段落的词进行语义分析匹配到最有可能是答案的字段。

1.3 本论文的创新点及主要工作内容

本文是基于阅读理解的智能问答，使用的数据集是斯坦佛问题回答训练集 SQuAD 2.0[32]版本，相较于 SQuAD 1.0 版本，2.0 版在原来的基础上增加了 50000 个对抗性的问题使得人类难以回答，这也增加了一项任务即判断这个问题是否能够根据当前的文本来进行回答。SQuAD 数据集使得系统要在可以回答的时候进行正确回答，在不能回答的时候还要避免回答，这对自然语言处理的现有模型提出了一个挑战。

本文使用双向注意力流模型即 Bidirectional Attention Flow[33]作为基本模型，模型包括嵌入层，编码层，注意力层，建模层，输出层，最重要的是注意力层包括上下文对问题的注意力以及问题对上下文的注意力。

预训练语言模型今年来发展迅速，ELMo，BERT，XLNet 等模型相继出现，而本文在基本模型的基础上使用了现今最强大的 ALBERT 预训练模型来进行研究。

通过加入不同的层或进行替代来探索对 SQuAD 数据集的效果，如：

加入双向长短期记忆 BiLSTM 层

使用 GRU 替代 LSTM

加入额外的高速层

加入注意力层

加入自注意力层

加入自行设计的输出层

等方式进行研究探索，通过与 baseline 的对比，获取对 SQuAD 数据集准确率提升最大的模型或方法。

第二章 自然语言处理及预训练语言模型的理论研究

2.1 预训练模型 ELMo

2.1.1 ELMo 简介

ELMo 是 Embeddings from Language Models[34]的简称，即语言模型的嵌入表示。一个预训练的词不仅能够表征单词的语义和语法层面的特征，也能够随着上下文语境的变换而改变，能够对一词多义建模。但是传统的词向量是上下文无关的，所以使用语言模型来得到上下文语境信息相关联的预训练表示即 ELMo，并在 6 个自然语言处理的任务上获得了提升。

2.1.2 ELMo 结构

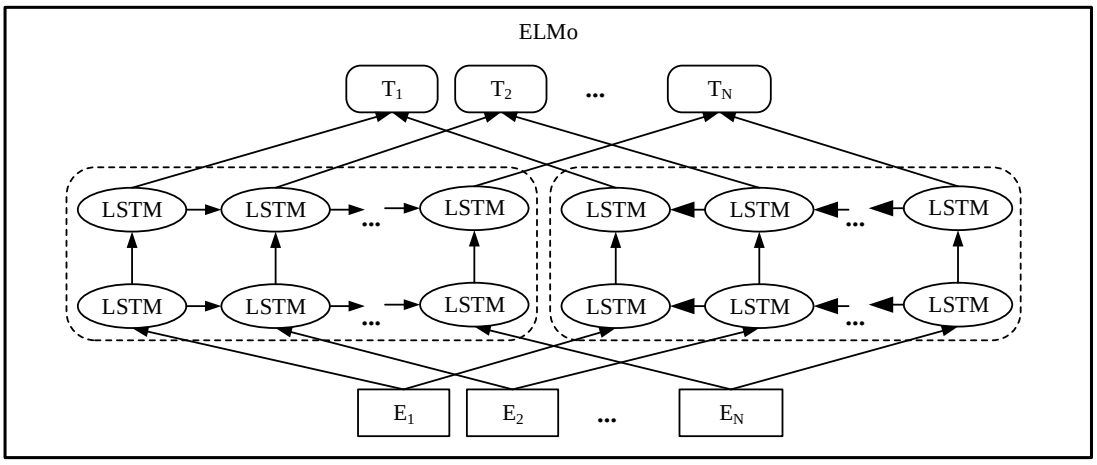


图 2-1 ELMo 的结构[35]

ELMo 使用的是一个双向的长短期记忆 LSTM 语言模型，有一个前向和后向语言模型构成。前向是指通过前面的词即上文来预测下文：

$$p(t_1,t_2,...t_N)=\prod_{k=1}^{N}p(t_k\mid t_1,t_2,...t_{k-1}) \tag{2-1}$$

后向指通过后面的词即下文来预测之前的词语：

$$p(t_1,t_2,...t_N)=\prod_{k=1}^{N}p(t_k\mid t_{k+1},t_{k+2},...t_N) \tag{2-2}$$

目标函数是取这两个方向语言模型的最大似然：

$$\sum_{k=1}^{N}\left(logp\left(t_k \mid t_1,t_2,...t_{k-1};Q,\vec{Q}_{LSTM},Q_s\right)+logp\left(t_k \mid t_{k+1},t_{k+2},...t_N;Q,\overleftarrow{Q}_{LSTM},Q_s\right)\right) \quad (2\text{-}3)$$

对于每一个词 E_i，一个具有 L 层的双向语言模型会计算出 2L+1 个表示。ELMo 本质上是一个任务导向的，双向语言模型内部的隐状态层的组合。ELMo 的通用表达式是：

$$ELMO_k^{task} = E\left(R_k;Q^{task}\right) = \gamma^{task}\sum_{j=0}^{L} s_j^{task}\, h_{k,j}^{LM} \quad (2\text{-}4)$$

其中γ是用来控制 ELMo 模型生成的向量大小，是一个缩放因子，相当于进行了层正则化（layer normalization）。该系数对于后续的模型优化有好处。如果没有这γ，只采用最后一层输出作为词向量的时候效果很差。另一个参数 s 是进行归一化处理。

ELMo 的缺点也很明显：

1. 虽然 ELMo 使用的是双向的语言模型，其实是 2 个单向语言模型的组合，只能单向的获取信息，不能同时获取全部的上下文的特征表示信息。
2. ELMo 使用的是长短期记忆 LSTM，不能够解决长距离依赖的问题。

2.2 Transformer

Transformer 来自在 2018 年发表的论文《Attention is all you need》[36]。需要注意的是 Transformer 并不是预训练模型，而是许多重要的预训练模型的核心或基础。因为 Transformer 十分重要，特单独列 2.2 小节进行过叙述。Transformer 号称可以完全舍弃卷积神经网络和循环神经网络。循环神经网络 RNN 一次只能接一个，不能进行并行计算，Transformer 可以并行的计算。

2.2.1 Transformer 结构

2.2.1.1 编码器与解码器

编码层是由 6 个编码器堆叠而成，编码器的结构完全相同，但是参数并不是共享的。解码层同样由 6 个解码器组成。

编码器由自注意力层和前馈网络层 Feed Forward Networks 组成。首先编码器的输入经过自注意力层，当进行编码时，自注意力层可以看见输入的其他单词，编码成考虑上下文的向量。然后接前馈网络层，一层一层往上传。

解码器多了一个编码器-解码器注意力层，因为要做普通的翻译，翻译也要考

虑编码器的输出，所以要有这个普通的注意力。

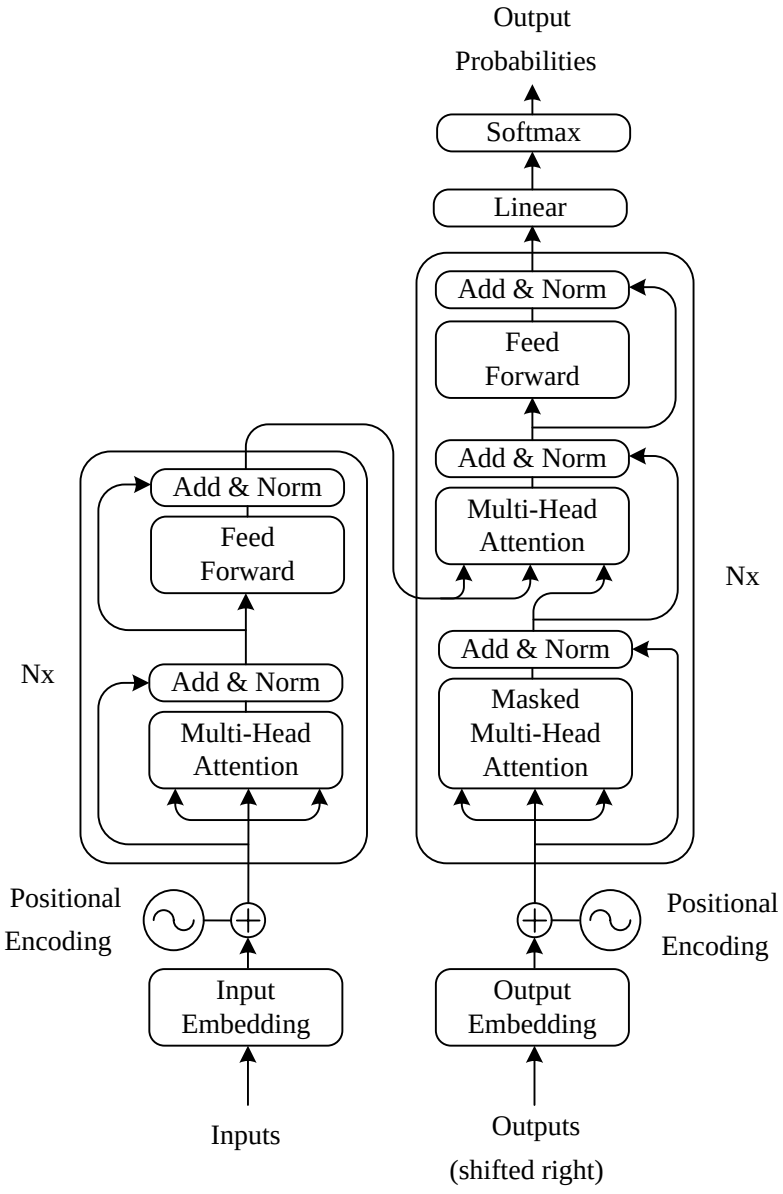


图 2-2 Transformer 结构[36]

2.2.1.2 自注意力层

对于自注意力来说，Q（Query），K（key）和 V（value）这三个矩阵来自同一个输入。首先 Q 和 K 进行点乘，点乘的结果可能会很大，所以对其进行限制，除以一个$\sqrt{d_k}$，d_k表示 Q 和 K 向量的维度。然后经过 softmax 层将结果进行归一化处理，得到它的概率分布。用得到的概率的值再乘以 V 的向量，这样就得到了带有权重的表示。这一步操作的公式为：

$$Attention(Q, K, V)=softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2-5}$$

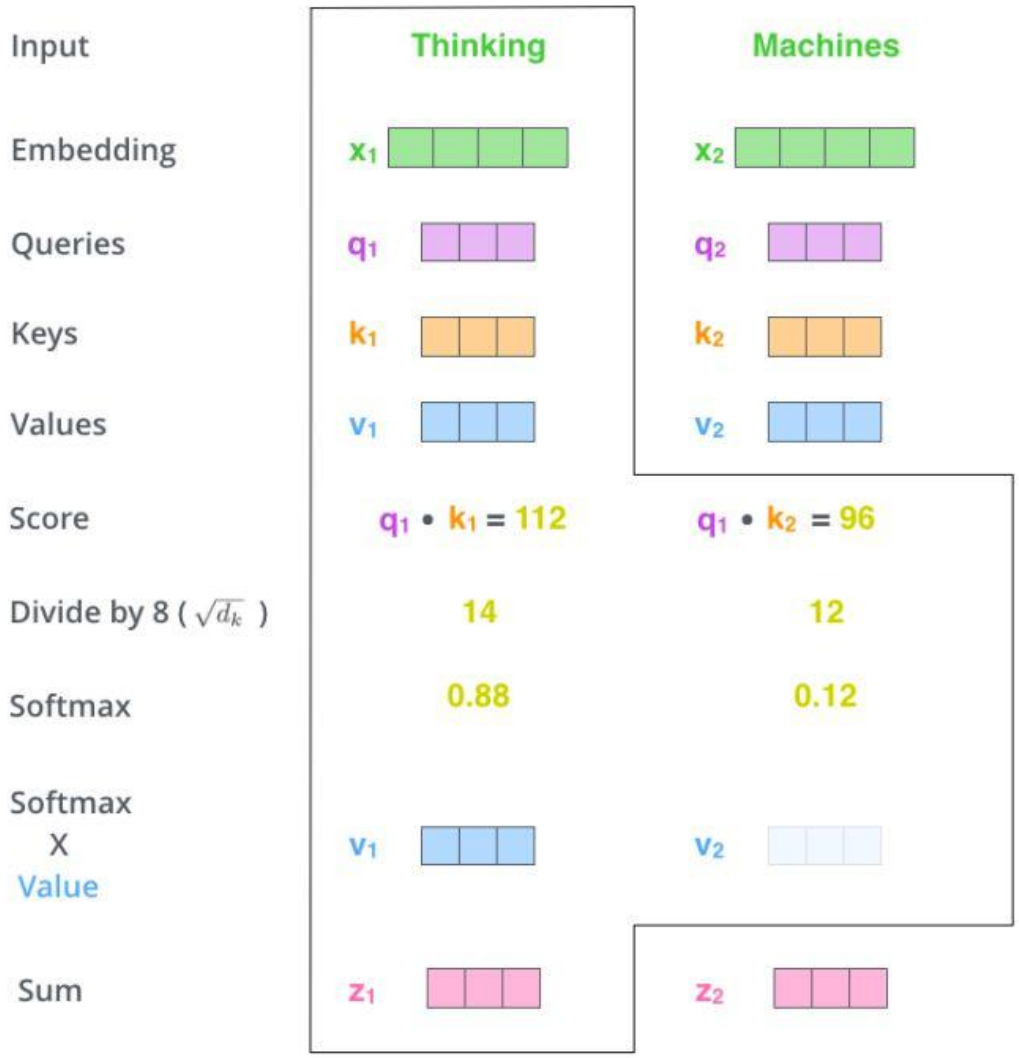


图 2-3 自注意力计算[37]

以 Thinking 和 Machine 作为输入为例：

1. 首先得到 Thinking 和 Machine 的词嵌入的向量表示
2. Thinking 的词嵌入向量与权重矩阵 W_q， W_k， W_v 相乘得到 q_1，k_1，v_1。权重矩阵是模型在进行训练的时候学习到的参数。
3. 为了计算句子中的其他词与 Thinking 的联系，用 Thinking 的查询向量 q_1 与其他单词以及自己的 key 向量 k_1 和 k_2 去做内积，得到相关度。内积越大，就越相似。用 softmax 得到概率，如 0.88 代表注意自己，0.12 代表注意另外的词 Machine。
4. 用 softmax 得到的概率值分别乘以 value 值 v_1 和 v_2，加权求和得到 z_1。

5. z_2 的计算类似，只不过是用 Machine 的查询向量 q_2 与 key 向量 k_1 和 k_2 做内积，然后 softmax 后加权求和得到结果。

总结一下，x 是词嵌入，编码了很多信息。q 是查询向量，用来查询别的词。k 是被查询的向量。$q_1 \times v_1$ 代表第一个词对于第一个词的注意力，$q_1 \times v_2$ 代表第一个词对于第二个词的注意力。v 代表真正的语义信息。

把输入向量合并成矩阵形式，那么 query 向量，key 向量，value 向量都可以合并成矩阵形式，一次性就把所有的计算完成，大大提高了效率。

2.2.1.3 多头注意力

所谓多头注意力就是把 Q，K，V 分成多个表示，最后将结果结合起来。类似于卷积神经网络中为了得到特征选择采用的多通道机制。不同的头可以学到不同的语义特征。

对于编码器来说多头注意力层的 query，key 和 value 都是来自前一层的输出，也就是一个编码器可以看到前一个编码器的所有信息。

对于解码器来说，第一个 query，key 和 value 是来自前一层解码器的输出，值得注意的是加入了掩码操作，也就是说解码的过程只能看到当前词前面的信息，后面的词是要预测的，不能提前看到，因为当前翻译的过程中还不知道下一个词是多少。第二个 query，key 和 value 构成了编码器-解码器注意力层，查询向量 query 是来自前面的解码器，而 key 和 value 向量是来自编码器的输出，这使得解码器能够看到所有输入序列的信息。

2.2.1.4 前馈网络 Feed Forward Network

前馈网络使多头注意力的结果映射到更大维度的空间，它的 ReLU 函数是主要的非线性变换单元。

2.2.1.5 位置编码

自注意力不考虑位置顺序的关系，例如“从济南到成都的机票”与“从成都到济南的机票”，“成都”编码的向量都是一样的，可是实际上“成都”第一个是作为目的地，第二个是作为出发地，语义是不一样的，所以加入位置编码来增强这个语义。

位置编码有两种：

（一）绝对位置编码：

每个位置编码都加上一个嵌入表示。还是用之前的例子，“从济南到成都的机票”与“今天，我要买从济南到成都的机票”。由于位置编码是绝对的，后一句加上了其他的字，“济南到成都的机票”的位置就后移了，语义就变了。所以绝对位置编码效果不好

（二）相对位置编码：

使用 sin 和 cos 来进行位置编码的建模：

$$PE_{(pos,2i)} = sin\left(pos / 10000^{2i/d_{model}}\right) \tag{2-6}$$

$$PE_{(pos,2i+1)} = cos\left(pos / 10000^{2i/d_{model}}\right) \tag{2-7}$$

使用 sin 和 cos 函数能够让模型学习到相对位置，因为位置 1 的 pos 与位置 2 的（pos+j）的位置编码是间距为固定值 j 的线性变化。可以证明的是：两个位置编码如果间隔为 j，那么它们的欧氏空间距离是恒等的，仅仅与 j 有关。

2.3 预训练模型 GPT

GPT 模型来自 Alec Radford 等人在 2018 年发表的文章[38]。

2.3.1 GPT 模型结构

GPT 的基本结构就是用的 Transformer 代替了传统的循环神经网络 RNN 来进行特征的提取，但是 Transformer 只用到了它的解码器部分，每一层是带有掩码 MASK 的自注意力及前馈网络。GPT 模型也是首次使用 Transformer 的预训练模型。

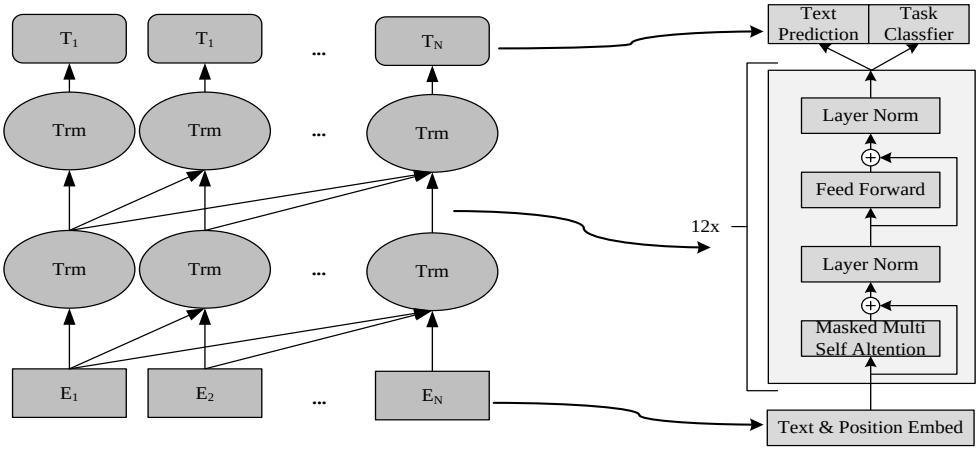


图 2-4 GPT 模型[35]

2.3.2 GPT 模型实现

对于非监督的预训练来说，给定一个词的 token，以及窗口 k，当预测下一个词 i 的时候，使用 i 前面 k 个词来进行预测。根据超参数对下一个词做出最有可能的预测。目标函数就是（1.1.4）节的语言模型最大化语句出现的概率。

$$L_1(u)=\sum_i logP\left(u_i|u_{i-k},...u_{i-1};\Theta\right) \tag{2-8}$$

$$P(u)=softmax\left(h_n W_e^T\right) \tag{2-9}$$

对于多层 Transformer 解码器，输入是词嵌入 U 乘词嵌入权重 W_e 加上位置参数 W_p。然后通过多层 Transformer 的隐藏层进行处理。

$$h_0 = UW_e + W_p \tag{2-10}$$

$$h_l = \text{transformer\_block}\left(h_{l-1}\right) \forall \text{i} \in [1,\text{n}] \tag{2-11}$$

对于有监督的微调 Fine-tuning 阶段来说，保留了预训练的 Transformer，用全连接层替换了最终的线性层，将 W_e 替换成了 W_y。W_y 是学习输入 x 和输出 y 关系的一个参数。

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P\left(y \mid x^1,\ldots,x^m\right) \tag{2-12}$$

$$P\left(y \mid x^1,\ldots,x^m\right) = \text{softmax}\left(h_l^m W_y\right) \tag{2-13}$$

兼顾式(2-8)的 L_1 和(2-12)的 L_2，加入一个权重参数 λ 控制 L_1 的比例，得到最终的 L_3，作为整个优化的依据。

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \tag{2-14}$$

2.3.3 GPT 模型缺点

GPT 使用的并不是双向的语言模型，依然是单向的，也就是说每次预测的时候仅仅参考了上文，不能够综合上下文的信息获取相关的特征表示。这个是语言模型的约束，并不是实际问题的约束，实际上文本给定了完整的句子，只是语言模型的要求使得 GPT 只能看上文的信息。

另一个缺点就是预训练和微调阶段的不匹配问题。预训练的时候是用的一个句子，下游任务进行微调的时候比如问题回答是多个句子，这样就造成了不匹配的问题。

2.4 预训练模型 BERT

BERT 是基于 Transformer 的深度双向预训练语言模型[35]，它的出现使得预训练模型微调优化在自然语言处理领域逐步走向成熟。

2.4.1 BERT 模型结构

BERT 模型由输入层、编码层和输出层三部分组成。

输入层是由 3 种特征嵌入表示相加得到。

编码层就是 Transformer 的编码器，不仅可以像循环神经网络可以保留长距离的信息，还能像卷积神经网络具有并行计算的能力。Transformer 编码器中的自注意力结构能够利用当前词的上下文来提取长距离的依赖关系，而且因为每个词的计算互不依赖，可以进行并行计算，提取所有词的特征。

输出层主要是有掩码的语言模型和下一句预测。

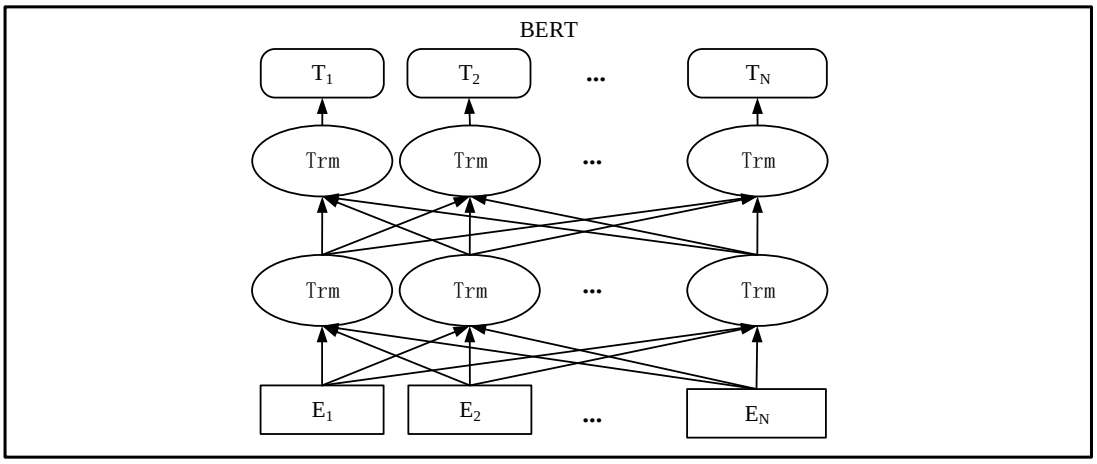


图 2-5 BERT 简易模型[35]

2.4.1.1 BERT 的输入表示

首先通过使用 Tokenizer 得到输入序列每个词的 Token 嵌入表示，然后加上段编码表示 Segment Embedding 与位置编码表示 Position Embedding。段编码和位置编码需要经过模型学习。[CLS]放在句首，是分类符，[SEP]放在两个句子之间，是分隔符。

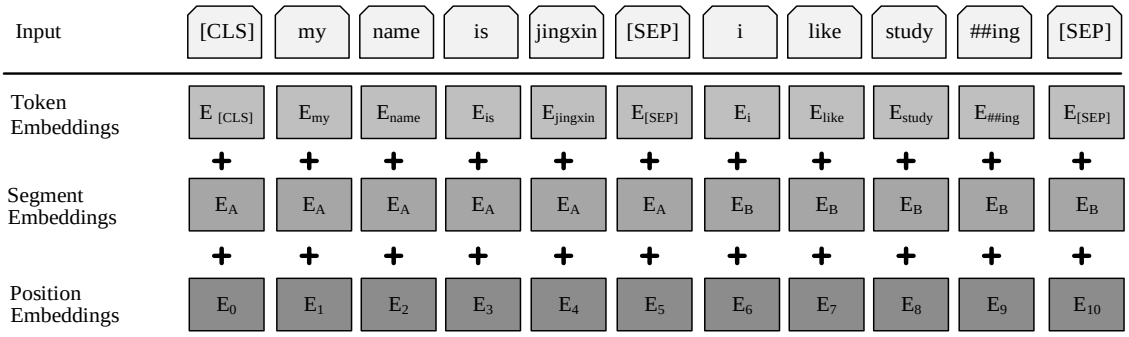


图 2-6 词嵌入表示[35]

2.4.1.2 BERT 输出层的掩码语言模型

ELMo 和 GPT 的缺点给了 BERT 启发，双向的 LSTM 会让每个词在多层上下文中间接的看到自己，影响了预测的效果，为了能够综合单词上下文的信息解决单向信息流的问题，BERT 使用了掩码 MASK。所谓掩码，就是在语言建模的任务中掩盖 15%的词，并让模型来预测被掩盖的词，这就是常见的完形填空。掩盖的这15%的词里面，有 80%是用的[mask]替代原来的词，10%是随机将某个词替换成另一个词，最后的 10%是不进行改变。然后让模型预测该位置正确的词。

掩码只用在预测的时候，因为预测的时候不能提前知道答案，否则就和 ELMo 一样。在训练的时候不用掩码，但是这样掩盖 15%再进行训练，导致训练速度变慢。

2.4.1.3 BERT 输出层的下一句预测

以图 2-6 为例，句子中的词都使用小写，第一句是“my name is jingxin”，第二句是“i like study ##ing”。第一句 Segment Embedding 用 A 表示，第二句用 B 表示。BERT 直接显示的编码出来，告诉 A 是第一个句子，B 是第二个句子。然后通过训练让 BERT 进行预测 B 是否是 A 的第一句。

因为在自然语言处理中，语言模型需要理解句子与句子间的关系，所以 BERT 的下一句预测就是为了这个目的。训练这个也很简单，训练样本选择 A 和 B，A 下一句有 50%的几率是 B，另外 50%的几率是随机抽取的一句也将就不是 A 的下一句，如果两个句子连续，那么预测输出 1，不连续就预测 0。这样模型经过训练得正确预测出 A 的下一句是否是 B。

最后整个预训练的目标函数就是掩码和预测下一个句子这两个任务的取和求最大似然。

2.4.2 BERT 与下游任务

针对不同的下游任务，BERT 会进行对应的微调。

以问题问答 Question Answering 做下游任务为例：

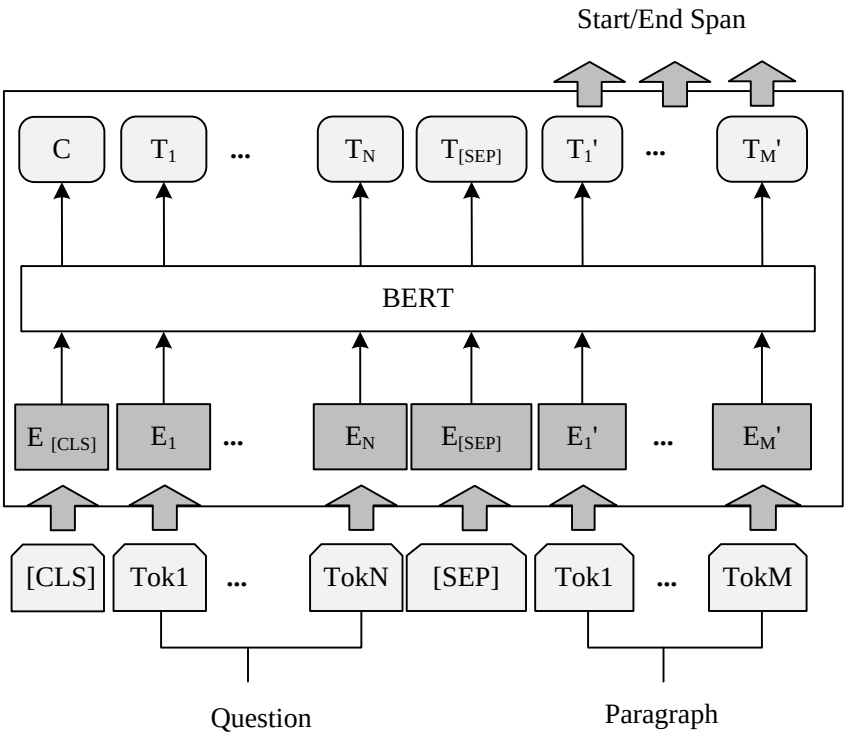


图 2-7 下游任务-问题回答[35]

输入是两个 sentence，sentence1 是问题，sentence2 是一段上下文，目的是在这段上下文中找到问题的答案，预测答案开始的位置和结束的位置。

首先把问题和段落都输入 BERT 模型, 对于每一个 token 都会输出隐层向量 hidden vector。对于段落输出的隐层向量，用开始向量跟每一个段落输出的隐层向量做点乘, 用 softmax 去输出最大的 token。最大 token 的位置就是开始的位置。结束向量是同样的操作，用结束向量去点乘每个段落的隐层向量，然后经过 softmax 后得到结束位置。

2.4.3 BERT 的优缺点

优点：

1. BERT 最大的有点是实现了真正的双向特征表示，利用单词的上下文信息进行特征提取。
2. BERT 用到了自注意力可以进行并行计算，提取输入的每个词的特征。

3. 容易迁移学习。预训练好的 BERT 可以直接用，将其加载为当前任务的嵌入层，然后对当前任务构建结构即可。

缺点：

1. 在预训练中使用人为制造的掩码标记，这种掩码在实际的数据里是没有的，使得预训练阶段和微调优化阶段出现不一致的问题。
2. 在预训练阶段，句子中多个单词被掩码掉，BERT 假设这些被掩码的单词没有任何关系，是条件独立的假设。自编码模型使用没有掩码的词来预测掩码的词，如果 BERT 掩码两个有相互关联的单词，模型的预测效果就很差。
3. BERT 在生成任务上效果不好，因为 BERT 的预训练的过程和生成的过程不一样。
4. BERT 在文档级别的任务上效果不好，只适用于句子和段落这种级别的任务。

2.5 XLNet

自回归 Auto Regressive 语言模型的代表是 ELMo 和 GPT，自编码 Auto Encoding 语言模型的代表模式是 BERT，XLNet 的出发点是把自回归语言模型和自编码语言模型的优点结合起来，也就是说在自回归语言模型的角度，考虑如何引入和双向语言模型相等价的效果。如果在自编码的角度上，它本身是融入双向的语言模型的，这时要考虑如何去掉掩码 MASK 标记让预训练和微调优化保持一致。

2.5.1 排列语言模型

XLNet[39]做的是在预训练阶段引入了排列语言模型的训练目标，也就是把输入进行全排列，将所有可能的情况考虑进来。例如输入是“I really like UESTC”，有以下几种排列：

1. **like** really UESTC I
2. really UESTC **like** I
3. I UESTC really **like**
4. UESTC **like** I really
5. …

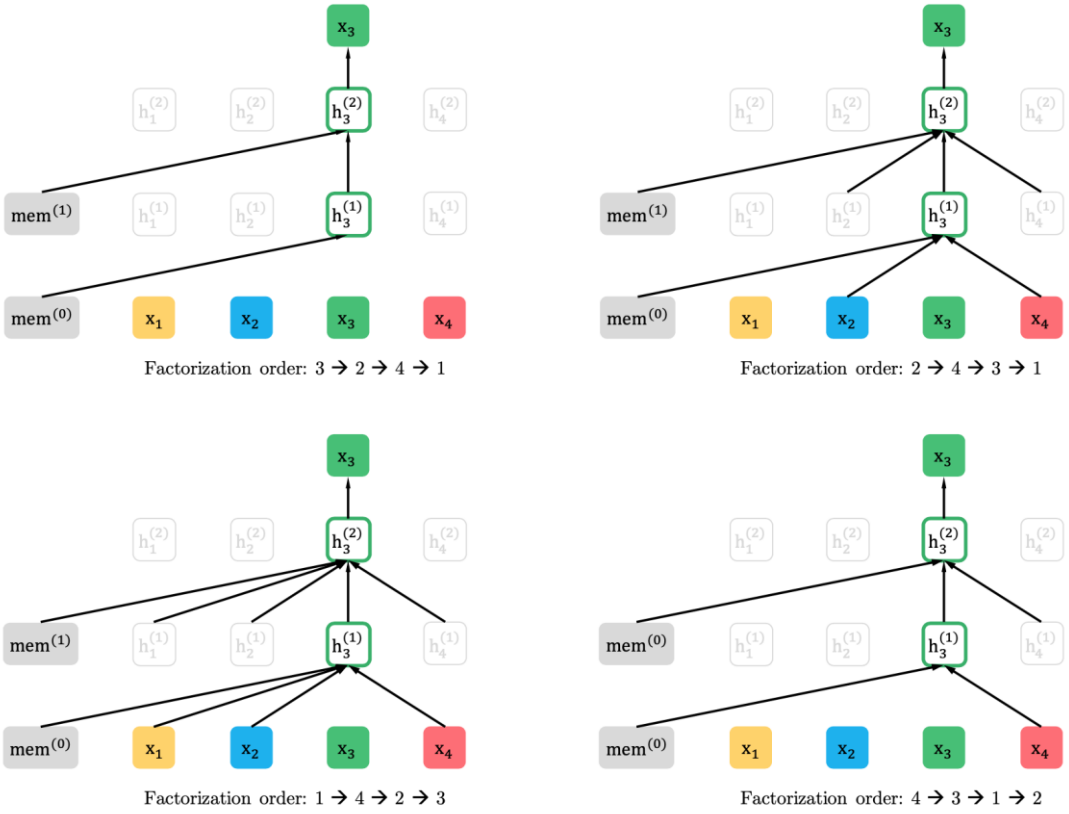


图 2-8 排列模型 XLNet[39]

假设预测第 3 个单词“like”,“like”的上文是“I”和“really”，下文是“UESTC”，希望看到下文的单词信息，XLNet 把“like”的位置固定，之后随机排列组合句子中的 4 个单词，在随机进行的排列组合里，选取其中的一部分输入到模型。比如之前的排列是“I really like UESTC”，分别用 1 2 3 4 代表。现在的排列有可能是[3 2 4 1]、[2 4 3 1]、[1 4 2 3]、[4 3 1 2]，在[3 2 4 1]中 3 的上文没有内容，所以 3 只与 mem 有关；在[2 4 3 1]中 3 只与 3 前面的 2 和 4 相关；在[1 4 2 3]中 3 与前面的 1 4 2 都相关，在[4 3 1 2]中 3 只与 4 有关，与图片一一对应。

2.5.2 Attention Mask

在 BERT 中掩码告诉模型要预测的单词的位置和上下文关系，在 XLNet 中用了 2 种流自注意力，一个是内容流，另一个是查询流。内容流负责查询上下文，并且可以看到自己。而查询流就是用来代替掩码，负责把内容流产生的表示拿来做预测，不能看到自己。同 BERT 的掩码一样，查询流只在预训练时预测单词会用到，到了优化的时候就不会用到了。

这种关系通过注意力掩码 Attention Mask 来实现：

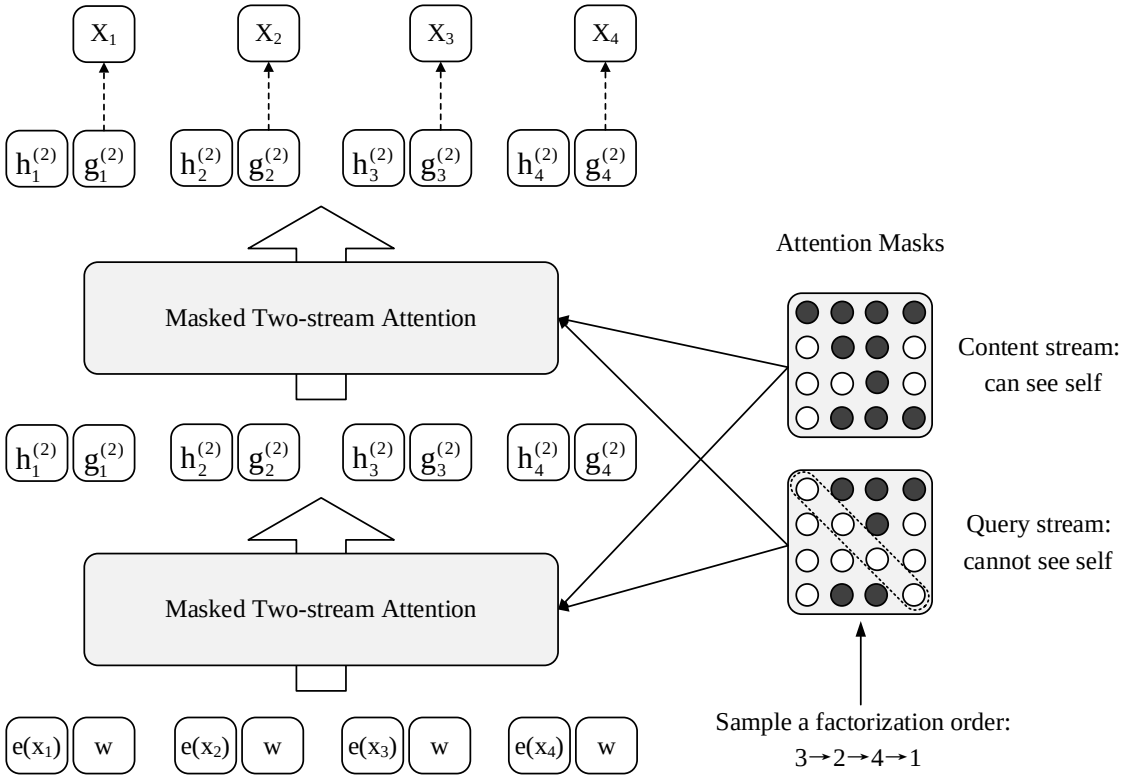


图 2-9 Attention Mask[39]

Mask 是一个矩阵，如图 2-9 所示： 空心部分表示信息不起作用，实心的表示信息起作用。

查询流的第一行代表 1 的信息，排列是[3 2 4 1] 所以 1 的信息与上文 3 2 4 有关，所以第一行的 2 3 4 是实心。

第二行代表的 2 的信息，2 的信息只与上文 3 有关，所以第二行只有第 3 个是实心。

第三行代表的是 3 的信息，3 的上文没有，所以第三行是全是空心。

第四行代表的是 4 的信息，4 的上文是 3 和 2，所以第四行的第 2 和第 3 个是实心。

以上就是 Attention Mask。

考虑所有的排列组合就得到了：

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim Z_T}\left[\sum_{t=1}^{T}\log p_\theta\left(x_{z_t}\mid \mathbf{x}_{\mathbf{z}<t}\right)\right] \tag{2-15}$$

其中 Z_T 代表的排列组合的所有可能。$z\sim Z_T$ 代表从集合中采样某个 z。训练的过程就是不断采样的过程。

XLNet 看起来依然是个从左到右的自回归语言模型，但是通过对句子里所有

的单词进行排列组合，下文的单词可以变成上文，这样就看到了全部信息，但是形式上还是从左到右地在预测后一个单词。

2.6 ALBERT

之前的GPT、BERT、XLNet等预训练模型的参数达到了百亿甚至千亿的级别，这无疑增加了训练的难度，还增加了训练的成本。为了使 BERT 模型“瘦身”，让它更轻，参数更少，于是蓝振忠等人提出了轻型的 BERT 即 ALBERT[40]。

BERT 的隐层大小 H 从 1024 变成 2048 的时候，这时候达到了 GPU 或 TPU 的内存限制，不能够继续变大模型了，所以 ALBERT 的提出就是为了减少模型的参数，同时不减少或者很少地减少模型的效果。

2.6.1 减少参数的方法

BERT 的输入是两个句子或段落，首先用 SentencePiece[41]做 tokenization，输入到网络里，经过 L 层的 Transformer 的编码器，之后网络要做 2 个自监督的损失，第一个是预测下一句话，也就是预测两个句子是否相邻，另一个是做掩码的语言模型，恢复输入时被掩码掉的 Token。

参数主要集中在 2 个模块：

（一）Token 嵌入模块：在这个模块的参数两个与词表的大小与词的嵌入大小相关，词表大小一般是 3 万左右，词嵌入大小一般与隐层大小相同。Token 嵌入模块的参数占总参数的 20%左右。

（二）注意力和前馈网络模块：这个模块的大小为（$12\times L\times H\times H$），这里有 H^2。这个模块的参数占总参数的 80%左右。

如果把隐层大小 H 变宽的话，在上述 2 个模块的参数都会增大很多，所以 ALBERT 通过尽可能的降低 H 的大小来达到缩减参数的目的。

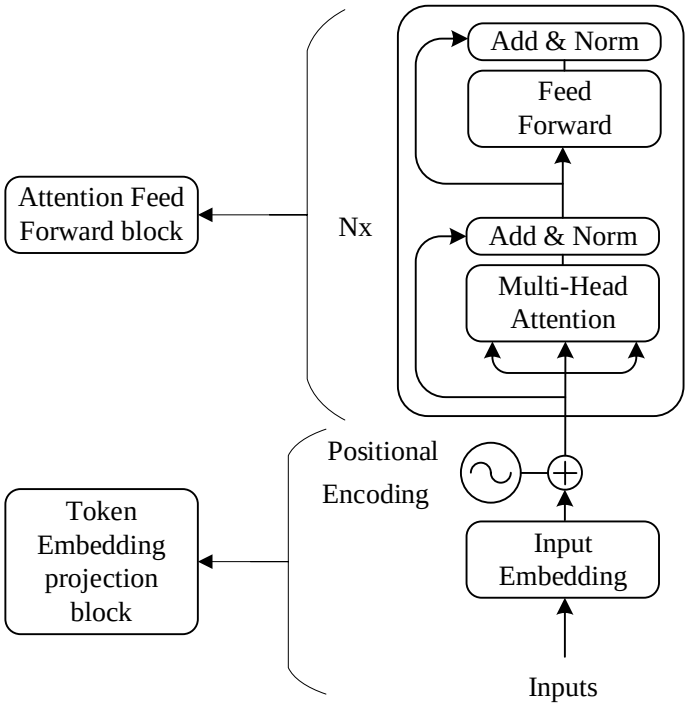


图 2-10 参数集中的 2 个模块

2.6.1.1 减少 Token 嵌入的参数

BERT 和之后产生的 XLNet 和 RoBERTa[42]使用的 WordPiece 的嵌入大小是 E，E 的大小和隐藏层的大小恒等。ALBERT 的输入是将稀疏的独热编码变成稠密的表示。在投影的过程中是一个全连接的网络，但是由于它是独热编码，每次更新的时候只更新独热编码为 1 的相关的参数，比如词表的大小是 3 万，每次更新 1 个，那么参数有大量参数未更新或没有有效利用。

第一次投影的时候词与词是没有进行交互的，没有交互就是独立的。一个词没有在句子中的意义，那么它表达的意义很简单，不需要高维的向量去表示它。只有在加上注意力操作的时候词与词才进行交互。

所以 ALBERT 在独热编码到 dense 表示的时候先降低到很低的维度，然后升高到需要的维度。这样有 2 个好处：第一是将上下文独立的表示和上下文相关的表示进行了解耦，之后就可以自由的将上下文相关的网络变宽，达到更好的结果。第二个好处是独热编码的参数很多，参数是词表的数量乘词嵌入的长度，如果词嵌入的长度变小，就可以把独热编码的参数减小。

简而言之就是把一个大的矩阵分解成两个小的矩阵的相乘。即：

$$O(V\times H)\rightarrow O(V\times E+E\times H) \tag{2-16}$$

2.6.1.2 跨层参数分享

对层数的权重进行可视化[43]，得到权重是呈现三角形的形状。注意到第一个Token [CLS]是一个全局的句子表达，还会注意到周围的词，如果参数是非常相似的话，那么各个层的参数可以共享，那么就可以把层数减小到 1，这样就减少参数 10 倍左右。相同的策略可以在 Dehghani 的论文[44]和 Bai[45]的论文中研究过。

2.6.2 句子顺序预测

BERT 中的预测下一句可以利用两个信息来解决，第一个信息是句子连贯性，另一个是句子之间的主题。如果句子来自不同的文章，那么可能就是不同的主题，所以模型可以通过学习句子的主题来预测下一句。这样 BERT 的预测下一句的存在意义也就不大。ALBERT 提出了一个改动，正样本还是正样本，负样本就把正样本反过来，原来的第一句话变成第二句话，这样就确保两句话都是来自同一个主题。

BERT 其实学习的并不是句子的连续，而是学习的主题的信息。ALBERT 用的句子顺序预测不但可以学习到句子的连续性，通过连续性信息可以区别不同的主题。

2.7 本章小结

本章主要介绍了预训练语言模型的发展。自回归模型 ELMo 使用了双向的长短期记忆，目标函数是对 2 个方向的语言模型取其最大似然。ELMo 不同于传统的每个词只能对应一个词向量，它可以根据输入从语言模型中获取上下文相关的词表示，上下文不同的词表示也是不相同的。但 ELMo 仅仅是简单的拼接，得到的仍然是单向的特征表示，无法同时获得上下文表示。

Transformer 结构，自注意力可以让 Transformer 模型能够同时拥有卷积神经网络并行计算和循环神经网络保留长距离信息的优点。

GPT 模型使用了单向的 Transformer 结构，所以它依然是个单向的语言模型，预测的时候只能参考上文的信息，不能利用整个上下文的信息。

BERT 是真正的双向 Transformer 模型，可以利用整个上下文信息进行特征提取。同时使用了自注意力，可以并行计算加快了训练速度。但是在预训练中使用的掩码标记使得 BERT 在微调优化阶段出现不一致的问题。BERT 在生成任务上表现也不好。

XLNet 模型为了解决 BERT 的问题，采用自回归模型的方式，为了获取下文，使用了排列语言模型，对输入进行了全排列。

ALBERT 模型使用 Transformer 的编码器和非线性函数 GELU[46]。通过减少输入的嵌入长度以及跨层参数分享，使得 BERT 模型的参数大大减少，并且效果几乎相同。ALBERT 使用了句子顺序预测代替 BERT 中的预测下一句，效果有了提升。虽然模型参数的减少但是并没有使得运算减少，所以 ALBERT 还有进步的空间。

第三章 基本模型 BiDAF 在 SQuAD 数据集的研究

基本模型是基于 BiDAF 的，即 Bidirectional Attention Flow 双向注意力流。用基本模型在 SQuAD 数据集上训练，得到基本模型的效果，然后再在基本模型双向注意流的基础上使用 ALBERT 进行研究，基本模型就是起到参考标准的作用，通过比较与基本模型的效果来看模型效果的提升情况。

3.1 SQuAD 数据集

SQuAD 1.0 数据集是问题回答或阅读理解最常用的数据集，由斯坦佛大学的 Rajpurkar 等人于 2016 年提出第一个版本[27]，在读取维基百科文章的基础上，它具有十万个问题回答对。给定一段话，给定这段话里的一个问题，每个问题都有 3 个人类回答的标准答案。

Rajpurkar 于 2018 年提出了 SQuAD 的第二个版本即 SQuAD 2.0[32]，在 1.0 版本的基础上，增加了 5 万个人类特别设计的很难回答的问题，有的问题并没有答案。所以对于模型提出了很高的要求，模型不仅要回答出有答案的问题，如果问题没有答案，模型还要得出“不能回答”的结论。本文使用的是 SQuAD 2.0 版本。

3.1.1 SQuAD 2.0 样例

给定一段文本：

Established originally by the Massachusetts legislature and soon thereafter named for John Harvard (its first benefactor), Harvard is the United States' oldest institution of higher learning, and the Harvard Corporation (formally, the President and Fellows of Harvard College) is its first chartered corporation. Although never formally affiliated with any denomination, the early College primarily trained Congregationalist and Unitarian clergy. Its curriculum and student body were gradually secularized during the 18th century, and by the 19th century Harvard had emerged as the central cultural establishment among Boston elites. Following the American Civil War, President Charles W. Eliot's long tenure (1869–1909) transformed the college and affiliated professional schools into a modern research university; Harvard was a founding member of the Association of American Universities in 1900. James Bryant Conant led the university through the Great Depression and World War II and began to reform the curriculum and liberalize admissions after the war. The undergraduate college became coeducational after its 1977 merger with Radcliffe College.

图 3-1 SQuAD 样例的上下文[47]

根据这段文本提出问题，给定 3 个标准答案，让模型经过训练后进行预测：

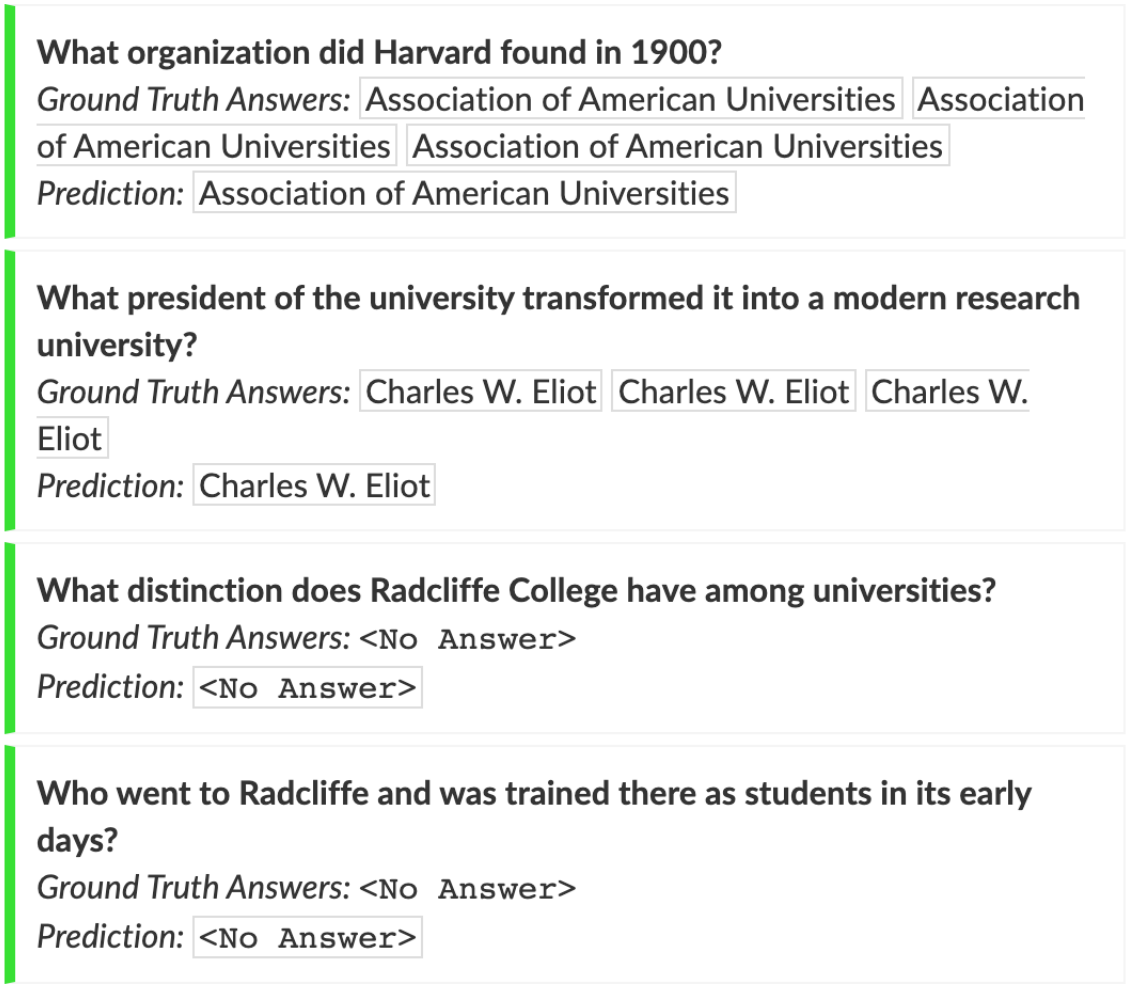


图 3-2 问题、标准答案和预测答案[47]

3.1.2 SQuAD 2.0 训练集

训练集有 129941 个训练样例，也就是 129941 个问题和相同数量的答案。有 442 个题目，每个题目对应 1 个段落。段落包括相同数量的“问题回答集合”和“上下文”，共有 19035 个“qas”即问题回答集合和“context”即上下文，每个问题回答集合包含多个问题回答对，共计 129941 个问题回答对。

3.1.3 SQuAD 2.0 验证集

验证集有 6078 个验证样例，有 16 个题目，每个题目对应 1 个段落。段落包括相同数量的“问题回答集合”和“上下文”，共有 646 个“qas”即问题回答集合和“context”即上下文，每个问题回答集合包含多个问题回答对，共计 6078 个问题回答对。

3.1.4 SQuAD 2.0 测试集

测试集有 5915 个测试样例，有 20 个题目，每个题目对应 1 个段落。段落包括相同数量的“问题回答集合”和“上下文”，共有 570 个“qas”即问题回答集合和“context”即上下文，每个问题回答集合包含多个问题回答对，共计 5915 个问题回答对。

训练集、验证集和测试集样例的数据结构相同，为：


```
{"version": "v2.0",
  "data": [
    {
    "title": "Beyonc\u00e9",
    "paragraphs":
      [
        {
          "qas":
            [
                {
                  "question": "When did Beyonce start becoming popular?",
                  "id": "56be85543aeaaa14008c9063",
                  "answers": [{"text": "in the late 1990s", "answer_start": 269}],
                  "is_impossible": false

                },

                {
                  "question": "What areas did Beyonce compete in when she was growing up?",
                  "id": "56be85543aeaaa14008c9065",
                  "answers": [{"text": "singing and dancing", "answer_start": 207}],
                  "is_impossible": false
                },

               {
                  "question": "What was Beyonc\u00e9's role in Destiny's Child?",
                 "id": "56d43ce42ccc5a1400d830b4",
                 "answers": [{"text": "lead singer", "answer_start": 290}],
                  "is_impossible": false
                },

                {
                  "question": "What was the name of Beyonc\u00e9's first solo album?",
                  "id": "56d43ce42ccc5a1400d830b5",
                  "answers": [{"text": "Dangerously in Love", "answer_start": 505}],
                  "is_impossible": false
                }

            ],
          "context":
            "Beyonc\u00e9 Giselle Knowles-Carter (/bi\u02d0\u02c8j\u0252nse\u026a/ bee-YON-say)
             (born September 4, 1981) is an American singer, songwriter, record producer and actress.
             Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child,
             and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father,
             Mathew Knowles, the group became one of the world's best-selling girl groups of all time.
             Their hiatus saw the release of Beyonc\u00e9's debut album, Dangerously in Love (2003),
             which established her as a solo artist worldwide, earned five Grammy Awards and featured the
             Billboard Hot 100 number-one singles \"Crazy in Love\" and \"Baby Boy\"."
        },
```


图 3-3 数据的结构

3.2 BiDAF 基本模型

基本模型 BiDDAF 的结构为：

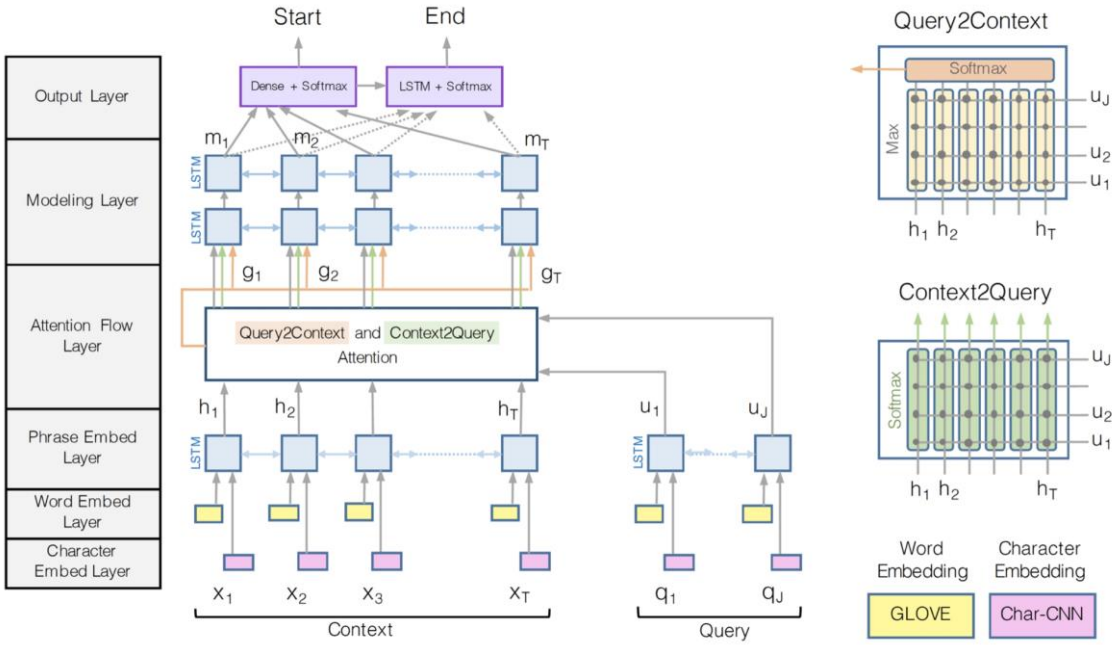


图 3-4 BiDAF 模型[33]

3.2.1 嵌入层

给定输入单词的索引 $w_1,w_2,...,w_k \in \mathbb{N}$，嵌入层将输入的索引进行嵌入词表的查找，将其转化成词嵌入 $v_1,v_2,...,v_k \in \mathbb{R}^D$。分别对段落和问题的词进行嵌入处理，查询得到段落上下文的词嵌入 $c_1,c_2,...,c_k \in \mathbb{R}^D$ 和问题的词嵌入 $q_1,q_2,...,q_k \in \mathbb{R}^D$。

在嵌入层，再进行下面两步的处理：

首先投影每个词嵌入使其变成 H 维，$W_{proj} \in \mathbb{R}^{H\times D}$ 是一个可学习的参数矩阵，每个词嵌入向量 v_i 被映射到 $h_i = W_{proj} v_i \in \mathbb{R}^H$

然后在嵌入表示层使用高速网络 Highway Network。给定一个输入向量 h_i，高速网络计算的是：

$$g = \sigma\left(W_g h_i + b_g\right) \in \mathbb{R}^H \tag{3-1}$$

$$t = \mathrm{ReLU}\left(W_t h_i + b_t\right) \in \mathbb{R}^H \tag{3-2}$$

$$h_i^{'} = g \odot t + (1-g) \odot h_i \in \mathbb{R}^H \tag{3-3}$$

其中 $W_g, W_t \in \mathbb{R}^{H\times H}$，$b_g, b_t \in \mathbb{R}^H$ 都是属于可以学习的参数，g 代表门 gate，t 代表变换 transform。高速网路是 2 层的，也就是说对每个隐藏向量 h_i 做 2 次变换。

3.2.2 编码层

编码层将嵌入层的输出作为本层的输入，然后使用了双向长短期记忆 Bi-

LSTM。编码层的输出是循环神经网络在每个位置的隐藏状态：

$$\mathbf{h}'_{i,fwd} = \mathrm{LSTM}\left(\mathbf{h}'_{i-1}, \mathbf{h}_i\right) \in \mathbb{R}^H \tag{3-4}$$

$$\mathbf{h}'_{i,rev} = \mathrm{LSTM}\left(\mathbf{h}'_{i+1}, \mathbf{h}_i\right) \in \mathbb{R}^H \tag{3-5}$$

$$\mathbf{h}'_i = \left[\mathbf{h}'_{i,fwd}; \mathbf{h}'_{i,rev}\right] \in \mathbb{R}^{2H} \tag{3-6}$$

$\mathbf{h}'_{i,fwd}$ 是前向长短期记忆的隐藏状态，$\mathbf{h}'_{i,rev}$ 是后向长短期记忆的隐藏状态，每个单向的维度是 H。将第 i 步的前向与后向隐藏层拼接得到双向的隐层状态 $\mathbf{h}'_i$，维度是 2H。上下文和问题分别得到其双向的隐层状态。

3.2.3 注意力层

注意力层是双向注意力流模型的核心层。双向注意力流包括从上下文到问题和从问题到上下文的注意力。上下文到问题类似看完文章后做问题，问题到上下文类似带着问题读文章。根据上下文的隐层状态 $c_1, c_2, ..., c_N \in \mathbb{R}^{2H}$ 和问题的隐层状态 $q_1, q_2, ..., q_M \in \mathbb{R}^{2H}$ 来计算相似矩阵 $S \in \mathbb{R}^{N \times M}$，相似矩阵 S 的任意一项 $S_{i,j}$ 代表第 i 个上下文隐状态和第 j 个问题的隐状态组成配对（c_i, q_j）：

$$\mathbf{S}_{ij} = \mathbf{w}_{\mathrm{sim}}^T \left[\mathbf{c}_i; \mathbf{q}_j; \mathbf{c}_i \circ \mathbf{q}_j\right] \in \mathbb{R} \tag{3-7}$$

$\mathbf{c}_i \circ \mathbf{q}_j$ 是按位相乘，$\mathbf{w}_{\mathrm{sim}}^T$ 是一个权重向量，它是 3 个向量的拼接，每个向量维度是 2H，所以权重向量的维度是 6H。

尽管相似矩阵 S 已经包含了对上下文和问题的信息，为了单独注意上下文或者单独注意问题，可以从行或列对相似矩阵进行归一化。

（一）首先获取从上下文到问题的注意力：对相似矩阵按行进行 softmax 得到注意力分布 $\bar{S}$，然后每一行的上下文的单词对所有问题单词的隐层状态加权求和，产生上下文到问题的注意力输出 a_i：

$$\bar{\mathbf{S}}_{i,:} = \mathrm{softmax}\left(\mathbf{S}_{i,:}\right) \in \mathbb{R}^M \quad \forall i \in \{1, ..., N\} \tag{3-8}$$

$$\mathbf{a}_i = \sum_{j=1}^{M} \bar{S}_{i,j} \mathbf{q}_j \in \mathbb{R}^{2H} \quad \forall i \in \{1, ..., N\} \tag{3-9}$$

其中 $q_1, q_2, ..., q_M \in \mathbb{R}^{2H}$。

（二）类似的可以得到从问题到上下文的注意力：对相似矩阵按列进行 softmax 得到注意力分布 $\bar{\bar{S}}$，每一列是问题中的单词对上下文所有单词的注意力。然后把两个注意力分布相乘，将得到的结果与上下文的隐层状态进行加权求和，得到问题对上下文的注意力输出 $\mathbf{b}_i$：

$$\bar{\bar{\mathbf{S}}}_{:,j} = \text{softmax}\left(\bar{\mathbf{S}}_{:,j}\right) \in \mathbb{R}^N \quad \forall j \in \{1,\ldots,M\} \tag{3-10}$$

$$\mathbf{S}' = \bar{\mathbf{S}}\bar{\bar{\mathbf{S}}}^T \in \mathbb{R}^{N\times N} \tag{3-11}$$

$$\mathbf{b}_i = \sum_{j=1}^{N} \mathbf{S}'_{i,j}\mathbf{c}_j \in \mathbb{R}^{2H} \quad \forall i \in \{1,\ldots,N\} \tag{3-12}$$

其中 $c_1, c_2, ..., c_N \in \mathbb{R}^{2H}$ 。

最后双向注意力流层把上下文的隐层状态 c_i，上下文到问题的注意力输出 a_i 和问题到上下文的注意力输出 b_i 拼接起来，得到最终的输出 g_i，它的维度是 8H：

$$\mathbf{g}_i = \left[\mathbf{c}_i; \mathbf{a}_i; \mathbf{c}_i \circ \mathbf{a}_i; \mathbf{c}_i \circ \mathbf{b}_i\right] \in \mathbb{R}^{8H} \quad \forall i \in \{1,\ldots,N\} \tag{3-13}$$

3.2.4 建模层

建模层就是在注意力层后面优化了序列向量，因为建模层在注意力层的后面，所以上下文的表示根据问题到达建模层的时间而作为条件，因此建模层将时间信息整合到了以问题为条件的上下文表示中。与编码层一样使用了双向的长短期记忆 Bi-LSTM，给定维度为 8H 的输入向量 g_i，建模层计算：

$$\mathbf{m}_{i,\,fwd} = LSTM\left(\mathbf{m}_{i-1}, \mathbf{g}_i\right) \in \mathbb{R}^H \tag{3-14}$$

$$\mathbf{m}_{i,rev} = LSTM\left(\mathbf{m}_{i+1}, \mathbf{g}_i\right) \in \mathbb{R}^H \tag{3-15}$$

$$\mathbf{m}_i = \left[\mathbf{m}_{i,\,fwd}; \mathbf{m}_{i,rev}\right] \in \mathbb{R}^{2H} \tag{3-16}$$

与编码层所不同的是，建模层使用了 2 层长短期记忆，而编码层使用了一层长短期记忆。

3.2.5 输出层

输出层是用的 BiDAF 输出，输出层对上下文中的每个位置都会产生一个概率向量，$p_{start}, p_{end} \in \mathbb{R}^N$，$p_{start}(i)$ 是输出层预测答案开始的位置在 i 点的概率，$p_{end}(i)$ 是输出层预测的答案结束的位置在 i 点的概率。输出层把注意力层的输出 $g_i \in \mathbb{R}^{8H}$ 与建模层的输出 m_i 当作输入，然后对建模层的输出使用双向长短期记忆网络，对每个 m_i 得到 m_i'：

$$\mathbf{m}'_{i,\,fwd} = LSTM\left(\mathbf{m}'_{i-1}, \mathbf{m}_i\right) \in \mathbb{R}^H \tag{3-17}$$

$$\mathbf{m}'_{i,rev} = LSTM\left(\mathbf{m}'_{i+1}, \mathbf{m}_i\right) \in \mathbb{R}^H \tag{3-18}$$

$$\mathbf{m}'_i = \left[\mathbf{m}'_{i,\,fwd}; \mathbf{m}'_{i,rev}\right] \in \mathbb{R}^{2H} \tag{3-19}$$

权重$G \in \mathbb{R}^{8H \times N}$是一个矩阵，矩阵的每一列是注意力层的输出$g_i$，共有 N 个。M 和$M^{'}$分别是建模层和输出层堆成的矩阵，分别有 N 个。输出层为了得到p_{start}, p_{end}，将 G 和 M, G 和$M^{'}$分别按行拼接，进行计算：

$$\mathbf{p}_{start} = \text{softmax}\left(\mathbf{W}_{start}[\mathbf{G};\mathbf{M}]\right) \tag{3-20}$$

$$\mathbf{p}_{end} = \text{softmax}\left(\mathbf{W}_{end}\left[\mathbf{G};\mathbf{M}^{'}\right]\right) \tag{3-21}$$

其中$\mathbf{W}_{start}$和$\mathbf{W}_{end} \in \mathbb{R}^{1\times 10H}$是可学习的参数。

3.3 损失函数与模型评估

损失函数是对答案开始点和结束点的负对数似然（交叉熵）的和。如果开始点$i \in \{1,2,...N\}$和结束点$j \in \{1,2,...N\}$，那么对于单个样例来说损失为：

$$loss = -\log p_{start}(i) - \log p_{end}(j) \tag{3-22}$$

在训练的时候对每一次批次求平均，并且使用 Adadelta 优化器来最小化损失。

评估主要是有两个参数：

1.第一个是 EM 参数，EM 代表完全匹配。也就是模型给的答案和标准答案完全一样。

2.第二个是 F1 参数，可以理解为模糊匹配。也就是根据模型给的答案与标准答案的重合度求出一个 0 到 1 之间的分数，这个分数就是精确率 precision 和召回率 recall 的调和平均。

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3-23}$$

精确率就是所有被预测为正确的样本实际上是正确的概率：

$$precision = \frac{TP}{TP+FP} \tag{3-24}$$

召回率就是所有正样本被正确识别为正样本的概率：

$$recall = \frac{TP}{TP+FN} \tag{3-25}$$

如果一个问题的答案是“Elon Musk”，如果模型预测的是“Musk”，那么 EM 的值就是 0，因为答案与标准答案不完全一致。但是这时候的精确率是 100%，召回率是 50%，那么 F1 的值为 66.7%。

3.4 推理与评估

（一）离散化：在测试阶段，离散化模型的软预测来获得开始于结束的索引位置。使用索引对（i,j）来最大化 $p_{start}(i)$ 和 $p_{end}(j)$，使其满足 $i \le j$ 且 $j\text{-}i\text{+}1 \le L_{\max}$。$L_{\max}$ 是一个超参数，它是设定的预测答案的最大长度。默认的 $L_{\max}$ 设置成功 15.

（二）没有答案的预测：只需要使用不在词表的标识 OOV，将其添加到每个上下文单词的前面，模型照样可以进行输出 $p_{start}(i)$ 和 $p_{end}(j)$，无需进行其他的改动。当对预测进行离散化获得索引的时候，如果 $p_{start}(0) \bullet p_{end}(0)$ 的值比任意预测的范围要大，那么模型就预测没有答案。否则，模型就正常预测最有可能的范围，同样使用负对数似然 NLL 作为损失函数。

（三）指数移动平均：在评估的时候，对模型的参数进行加权指数移动平均，衰减率设定为 0.999。

3.5 训练流程

3.5.1 数据预处理

数据预处理主要包括三部分：文件处理，获取嵌入表示，建立特征。

3.5.1.1 文件处理

以训练集为例，首先对于文章中的每一个段落，也就是给定的上下文的文本表示先进行引号的替换，然后把上下文的所有单词进行分词 Tokenize，单词中的每个字母也顺便进行 Tokenize，方便以后可能任务的使用。举个例子“I like studying”，对这句话的单词进行分词得到[‘I’，‘like’，‘studying’]，对字母的分词的到[[‘I’],[‘l’, ‘i’, ‘k’, ‘e’],[‘s’, ‘t’, ‘u’, ‘d’, ‘y’, ‘i’, ‘n’, ‘g’]]。

分词以后，通过查找单词在上下文中出现的位置来获取上下文中每个单词的范围，范围就是：[单词出现的位置, 单词出现的位置+单词的长度]。

对于上下文的每个单词进行计数，用计数器 counter 来保存，对每个单词加上本段落问题答案对“qas”的长度。对于每个字母也是加上本段落“问题答案对”的长度。然后对“问题答案对”中的“问题”进行计数，替换引号和分词操作，“问题标记”中的每个单词以及组成单词的字母也进行计数，出现一次就加上一次，这与上下文中的单词不同，上下文中的单词是出现一次加上问题答案对的长度。

```
Counter({',': 165, '-': 90, 'and': 90, 'in': 75, 'the': 75, '.': 60, 'of':
60, '"': 60, '(': 45, ')': 45, 'as': 45, "'s": 45, 'Beyoncé': 30,
'Knowles': 30, 'singer': 30, 'a': 30, 'girl': 30, 'group': 30, 'her': 30,
'one': 30, 'Love': 30, 'Giselle': 15, 'Carter': 15, '/biːˈjɒnseɪ/': 15,
'bee': 15, 'YON': 15, 'say': 15, 'born': 15, 'September': 15, '4': 15,
'1981': 15, 'is': 15, 'an': 15, 'American': 15, 'songwriter': 15,
'record': 15, 'producer': 15, 'actress': 15, 'Born': 15, 'raised': 15,
'Houston': 15, 'Texas': 15, 'she': 15, 'performed': 15, 'various': 15,
'singing': 15, 'dancing': 15, 'competitions': 15, 'child': 15, 'rose': 15,
'to': 15, 'fame': 15, 'late': 15, '1990s': 15, 'lead': 15, 'R&B': 15,
'Destiny': 15, 'Child': 15, 'Managed': 15, 'by': 15, 'father': 15,
'Mathew': 15, 'became': 15, 'world': 15, 'best': 15, 'selling': 15,
'groups': 15, 'all': 15, 'time': 15, 'Their': 15, 'hiatus': 15, 'saw': 15,
'release': 15, 'debut': 15, 'album': 15, 'Dangerously': 15, '2003': 15,
'which': 15, 'established': 15, 'solo': 15, 'artist': 15, 'worldwide': 15,
'earned': 15, 'five': 15, 'Grammy': 15, 'Awards': 15, 'featured': 15,
'Billboard': 15, 'Hot': 15, '100': 15, 'number': 15, 'singles': 15,
'Crazy': 15, 'Baby': 15, 'Boy': 15, 'When': 1, 'did': 1, 'Beyonce': 1,
'start': 1, 'becoming': 1, 'popular': 1, '?': 1})
```

图 3-5 word counter

```
Counter({'e': 994, 'a': 647, 'r': 617, 's': 616, 'n': 573, 'o': 543, 'i':
527, 't': 392, 'd': 362, 'l': 361, 'h': 256, 'g': 226, 'b': 196, 'u': 181,
'c': 167, 'm': 166, ',': 165, 'y': 151, 'w': 150, 'f': 135, 'p': 107,
'B': 106, '-': 90, '0': 75, '1': 60, '.': 60, 'v': 60, '"': 60, 'C': 45,
'(': 45, ')': 45, '9': 45, "'": 45, 'é': 30, 'G': 30, 'K': 30, '/': 30,
'A': 30, 'H': 30, 'T': 30, 'D': 30, 'M': 30, 'L': 30, ':': 15, '’': 15,
'j': 15, 'ɒ': 15, 'ɪ': 15, 'Y': 15, 'O': 15, 'N': 15, 'S': 15, '4': 15,
'8': 15, 'x': 15, 'R': 15, '&': 15, '2': 15, '3': 15, 'z': 15, 'W': 1,
'?': 1})
```

图 3-6 char counter

对于“问题答案”中的答案，得到答案的文本表示，答案的起始位置，答案的终止位置（起始位置+答案文本的长度），如果答案的范围在段落上下文的范围内，就把这个范围对应的索引加入答案的范围内，答案范围的第一个和最后一个成为答案的开始与结束位置。

最后获得一个训练样例，样例包括上下文单词标记，上下文单词的字母，问题单词标记，问题单词的字母，答案的开始位置，答案的结束位置，与问题答案对的个数。同时获得训练的评估样例，评估样本包括：上下文，问题，范围，答案文本，问题 ID。验证集和测试集同训练集的处理相同。

3.5.1.2 获取嵌入表示

使用 GloVe.840B.300d 模型[48]，840B 代表有 8400 亿个标记 token，300d 表示每个向量维度是 300 维。

这个模型的每一行有 301 个值，第一个值代表的是单词，其余的 300 个代表它的向量表示。如果单词在计数器 counter 里，并且出现的次数大于规定的最小值，那么就把这个词和它的向量表示加入到词嵌入的词典 embedding_dict 中。

没有的词标记为“--NULL--”加入到词嵌入词典的第 0 个位置，超出词表的词标记为“--OOV--”，加入词嵌入词典的第 1 个位置。NULL 和 OOV 的向量表示都是 300 维的 0 向量。原来词嵌入词典的第一个值现在成了第 3 个，后面的以此类推。创建 token 到 index 的词典即 token2idx_dict。

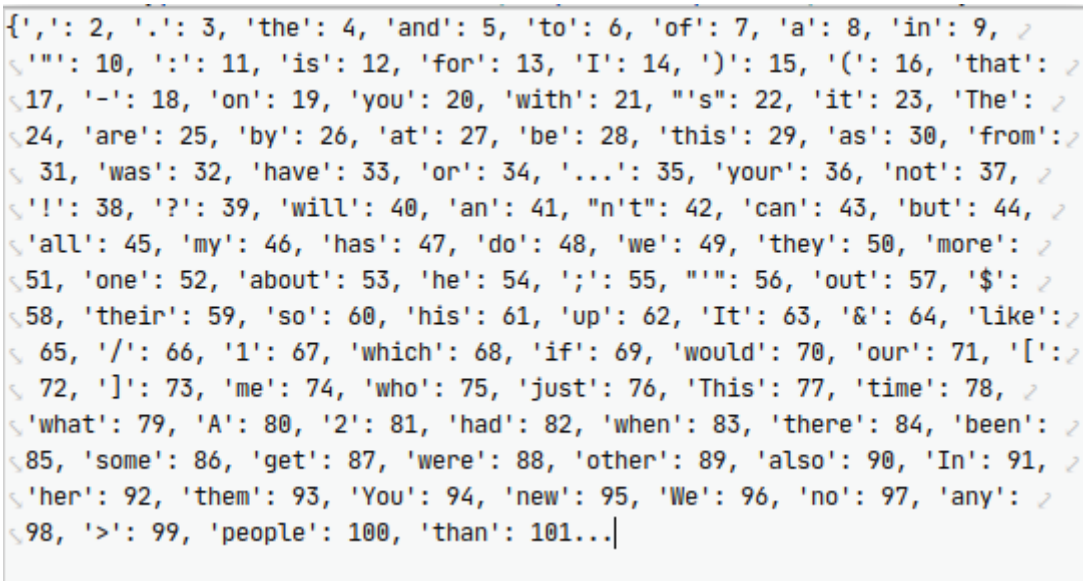


图 3-7 token2idx_dict

然后再创建一个索引 index 到词嵌入 embedding 的词典 idx2emb_dict：从 token 到 index 的词典 token2idx_dict 中找到索引和 token，这时索引 index 对应的是 token 在词嵌入词典的表示。

idx2emb_dict 中的词嵌入表示作为词嵌入矩阵 word_emb_mat, token2idx_dict 作为 word2idx_dict。

使用上述方法也可以获得 char_emb_mat 和 char2idx_dict。

3.5.1.3 建立特征

首先限定段落中最多有 400 个词，问题中最多有 50 个词，答案最多有 30 个词，一个单词的字母最多有 16 个。如果超过限定，就舍弃。

然后建立元素全为 0 的上下文字母索引数组 context_idx，它的大小为设定的段落最多的词 400；建立上下文字母级的数组 context_char_idx, 形状为（400,16），16 是单词中最多的字母数；建立元素全为 0 的问题单词索引数组 ques_idx，大小为设定的问题的最大词数 50；建立问题的字母级的数组 ques_char_idx，形状为（50,16）。

下一步对上下文中的单词进行查找，在词典 word2idx_dict 中找到单词的索引位置，然后加入上下文索引数组 context_idx。同样对问题的单词进行查找，得到的索引加入问题索引数组 ques_idx。对上下文的字母进行查找，在词典 char2idx_dict 中找到字母的索引位置，加入上下文字母的数组 context_char_idx。对问题的字母进行查找，得到字母的索引位置，加入问题字母的数组 ques_char_idx。

如果有答案的开始点和结束点，那么问题就是可以回答的，将开始点加入 y1s，将结束点加入 y2s。最后将上下文单词索引的数组，上下文字母索引的数组，问题单词索引的数组，问题字母索引的数组，开始位置索引的数组，结束位置索引的数组，以及 ID 保存为输出的文件。这样就构成了特征文件。

建立特征和处理文件的区别是，特征都是单词或者字母的索引，而处理文件的时候得到的是单词或者字母本身。

3.5.2 训练

3.5.2.1 注意事项

在编码层，编码一个序列的时候会遇到序列长度不一致的问题，通常是进行填充‘PAD’将短句子填充成长句子的长度。但是有的句子比较长，有的句子可能就一个单词，对于一个单词的句子来说，填充了很多无用的字符，这样造成的句子就会有误差。为了解决这个问题，就引出了 Pytorch 中处理变长输入的方法 pack_padded_sequence 和 pad_packed_sequence，分别是对填充过的变长序列进行压紧以及对压紧的序列进行恢复。压紧以后，填充的‘PAD’（一般初始化为 0）这个无意义的占位符就被删除了。

3.5.2.2 训练流程

1. 首先获取词嵌入表示，从生成的词嵌入文件 word_embed 中加载词嵌入表示。

2. 构造双向注意力流模型 BiDAF，开始训练前对模型的参数进行指数移动平均处理，使得测试指标提高并增加了模型的鲁棒性。指数移动平均就是分配给相近

的数据更高的权重的平均方法，也称权重移动平均。

$$v_t=\beta\cdot v_{t-1}+(1-\beta)\cdot\theta_t \tag{3-26}$$

BiDAF 模型的结构及参数为：

```
BiDAF(
  (emb): Embedding(
    (embed): Embedding(88714, 300)
    (proj): Linear(in_features=300, out_features=100, bias=False)
    (hwy): HighwayEncoder(
      (transforms): ModuleList(
        (0): Linear(in_features=100, out_features=100, bias=True)
        (1): Linear(in_features=100, out_features=100, bias=True)
      )
      (gates): ModuleList(
        (0): Linear(in_features=100, out_features=100, bias=True)
        (1): Linear(in_features=100, out_features=100, bias=True)
      )
    )
  )
  (enc): RNNEncoder(
    (rnn): LSTM(100, 100, batch_first=True, bidirectional=True)
  )
  (att): BiDAFAttention()
  (mod): RNNEncoder(
    (rnn): LSTM(800, 100, num_layers=2, batch_first=True, dropout=0.2,
 bidirectional=True)
  )
  (out): BiDAFOutput(
    (att_linear_1): Linear(in_features=800, out_features=1, bias=True)
    (mod_linear_1): Linear(in_features=200, out_features=1, bias=True)
    (rnn): RNNEncoder(
      (rnn): LSTM(200, 100, batch_first=True, bidirectional=True)
    )
    (att_linear_2): Linear(in_features=800, out_features=1, bias=True)
    (mod_linear_2): Linear(in_features=200, out_features=1, bias=True)
  )
)
```

图 3-8 BiDAF 模型的参数

3. 使用保存器，保存检查点 checkpoints 文件，用来存储所有的参数，包括权重，偏差，梯度等。设置最多保存 5 个检查点，超过五个后就删除最差的检查点。
4. 使用 Adadelta 优化器[49]，学习率为 0.5。Adadelta 优化器比 Adagrad 优化器[50]更鲁棒，因为 Adadelta 并不积累之前所有的梯度，它随着渐变更新的移动窗口来改变学习率，这样的话即使更新了许多，Adadelta 仍然在继续学习。

5. 用 SQuAD 类来构造 torch 能够识别的数据集，数据集中包括建立的特征：上下文单词索引，上下文字母索引，问题单词索引，问题字母索引，开始位置索引，结束位置索引。然后用 Data Loader 来迭代数据。Data Loader 可以方便的对模型输入数据进行操作，首先将初始数据加载到 DataLoader 里，需要随机打乱的话就对数据进行随机打乱操作。然后通过迭代器按照设置好的批次来迭代输出随机打乱后的数据。迭代器的优点是能够明显降低内存的使用，在需要的时候才把数据加载入内存。
6. 开始正式训练，设置 batch_size 为 64，dropout 为 0.2，epoch 为 30，也就是数据被训练的总轮数为 30。评估步数是 50000，每经过 50000 步，模型进行评估一次。首先进行前向传播，求得负对数似然。然后进行反向传播，对梯度进行裁剪，对参数使用 Adadelta 优化器进行优化以及进行指数移动平均操作。
7. 对模型进行评估：model.eval()表示模型进入评估模式，在该模式下，dropout 层[51]会让所有激活单元全部通过，batchnorm 层[52]会终止计算和更新均值和方差，直接使用在训练的时候学到的均值和方差。评估模式不会影响梯度的计算，与训练模型一样，只是不进行反向传播。with torch.no_grad 主要是停止反向传播的自动微分模块，可以加速和节省显存。实际上就是停止了梯度的计算，节省了 GPU 的算力和显存。评估的时候进行前向传播，求负数对数似然，得到 F1 和 EM 的值，输出到日志中。最后得到的结果包括负数对数似然“NLL”，模糊匹配“F1”和完全匹配“EM”。因为使用了 SQuAD 2.0，所以结果还有一个“AvNA”有无答案的指标。

3.6 训练结果

经过 30 轮 3852449 步的训练，验证集上最佳检查点 checkpoints 的结果是：

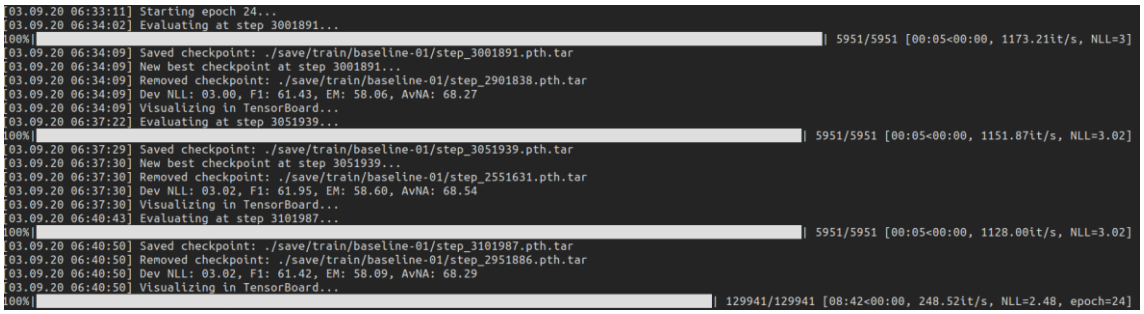


图 3-9 训练结果

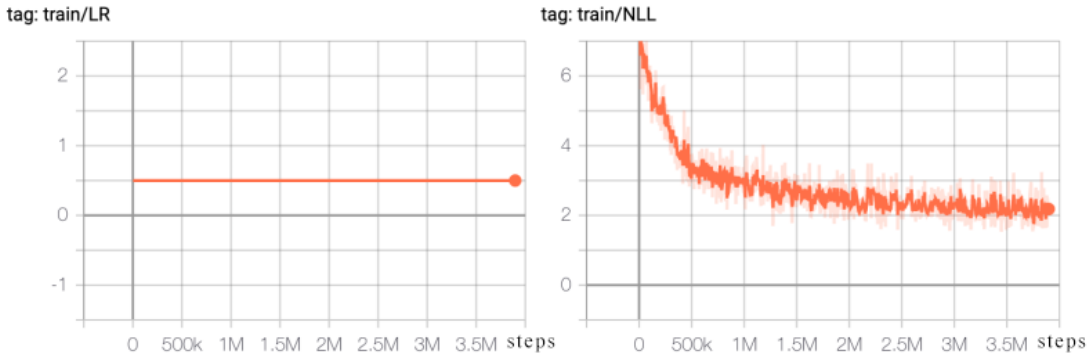
表 3-1 BiDAF 在验证集结果

评价指标	数值
NLL	03.02
F1	61.95
EM	58.60
AvNA	68.54

该结果在训练的第 3051939 步得到。这就是基本模型的效果，下一章 ALBERT 在基本模型的基础上进行优化，研究如何能提高现在的效果。

在 TensorBoard 中可以更直观的看到模型训练的结果。训练集结果为：

学习率 LR 保持 0.5 不变。负对数似然 NLL 在[2,7] 区间，经过训练步数的增加而逐渐降低。



(a) LR (b) NLL

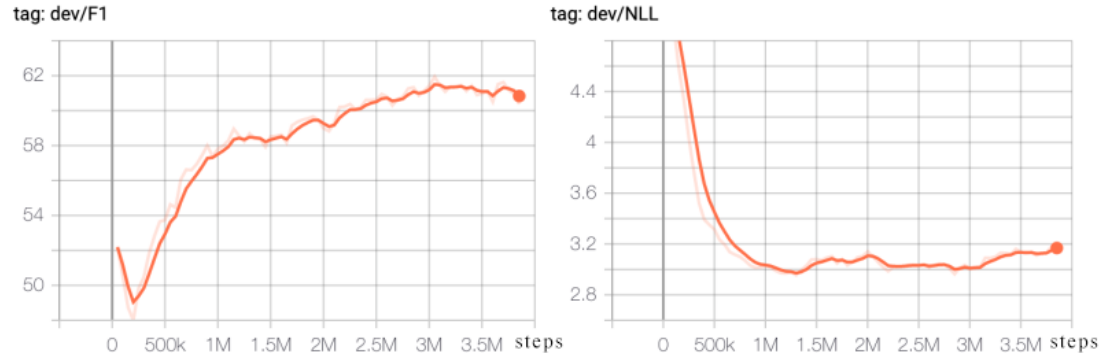
图 3-10 训练集结果

验证集结果：AvNA 表示有答案和无答案的比较，当模型仅考虑有答案和无答案的预测时，它是用来测量模型预测的准确性。同样这个指标用来 debug。最重要的 2 个指标完全匹配 EM 和模糊匹配 F1 分别达到了 58.60 和 61.95 的结果。

验证集的曲线为：

(a) AvNA (b) EM

图 3-11 验证集结果 1



(a) F1 (b) NLL

图 3-12 验证集结果 2

验证集的最佳结果：EM 值为 58.60， F1 的值为 61.95。这两个值将作为参考标准，与第 4 章模型结果进行比较。

3.7 本章小结

本章主要介绍了双向注意力流 BiDAF 的模型结构，完整的训练流程以及在双向注意力模型的基础上对 SQuAD2.0 数据集作训练预评估，得到了基本模型的参考指标 F1 为 61.95，EM 为 58.60。

BiDAF 基础模型作为参考的标准，对模型进行改进与优化比如增加层数、使用预训练模型等方式，观察这两个主要指标的变化，与基本模型作比较，得到模型优化或改进的程度。

第四章 ALBERT 在 SQuAD 数据集的研究

4.1 ALBERT 预训练模型概括

预训练模型在自然语言处理中发挥着越来越重要的作用，在多项自然语言处理的任务中效果有了很明显的提升。基于 Transformer 的双向编码器表示即 BERT 以及近几个月的轻型化的 BERT 即 ALBERT 效果更是很显著，所以本章在 ALBERT 基本模型第二版即 ALBERT-base-v2 的基础上，添加了不同的层进行了研究，通过观察两个重要指标 F1 和 EM 的值来推测模型的提升效果。

本章所用到的模型以及后面加的层为：

1. 模型一：ALBERT-basse-v2 + ALBERT-OUT
2. 模型二：ALBERT-base-v2 + Highway + ALBERT-OUT
3. 模型三：(ALABERT-base-v2+char_emb) + GRU-Enc + Attention + Self-Attention + GRU-Dec + BiDAF-OUT
4. 模型四：(ALABERT-base-v2+char_emb)+ GRU-Enc + Highway + GRU-Dec + BiDAF-OUT
5. 模型五：ALBERT-xxlarge-v2 + ALBERT-OUT

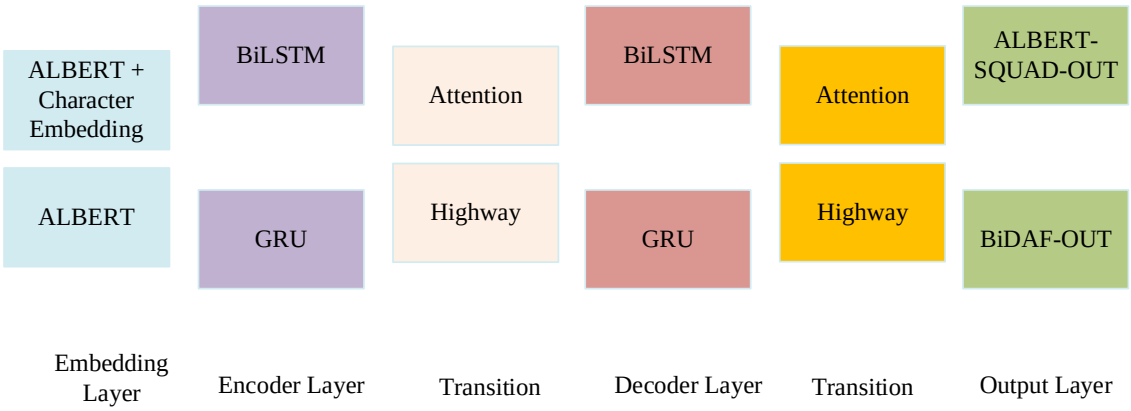


图 4-1 模型的构造

4.1.1 整体架构

4.1.1.1 嵌入层

在嵌入层使用 ALBERT 模型，嵌入层分为 2 种：一种只用到 ALBERT；另一种是 ALBERT 和字母嵌入 Character Embedding 的组合，用(ALBERT + char_emb)表示。为了生成组合，对原始的词嵌入使用加权平均的注意力，这被证明是一种有效的可以检测到不能回答的问题的方法。加权平均的注意力通过 softmax 函数来产生即 *softmax(EW)*，其中 E 代表词嵌入，W 代表可以学习的权重参数。对于预训练的字母嵌入，使用卷积神经网络 CNN 从原始的嵌入中来池化信息，然后加上字母嵌入的加权平均注意力。最后把得到的词嵌入和字母嵌入进行拼接。然后用两层高速网络编码器来优化拼接后的嵌入表示。加上了字母嵌入能够比原来只有词嵌入的模型提供更多的语义信息，尤其是输入中包含拼错的单词或新单词等情况。

4.1.1.2 编码-解码层

在编码-解码层，主要用到了 2 种模型，与 BiDAF 模型使用双向长短期记忆不同，ALBERT 的模型使用的是门控循环单元 GRU。

4.1.1.3 自注意力层

自注意力在注意力层的后面，自注意力是一种注意力机制，使得每个位置的输出注意到其他所有位置，包括本身。

4.1.1.4 高速网络 Highway Network

高速网络使用同（3.2.1）。高速网络主要是随着层数的增加，Loss 函数可能不会下降，使用高速网络就可以解决这个问题。

4.1.1.5 输出层

输出层包括 2 个：BiDAF 模型的输出层 BiDAF-OUT 和 ALBERT 模型的输出层 ALBERT-OUT。

其中 BiDAF 的输出层 BiDAF-OUT 把注意力层和编码层输出的线性变化相加得到了开始点；为了得到结束点，将编码层的输出传入到循环神经网络 RNN 来得到结束点的表示，把注意力层的输出的线性变换和结束点的表示相加，得到输出点。最后对问题的单词做一个掩码，使得最终仅用上下文中的单词来进行回答。

ALBERT 的输出 ALBERT-OUT 建立了一个线性的输出层，使得输出序列的维度从（batch_size, seq_len, hidden_state）变成了（batch_size, seq_len, 2），将其分开

得到开始点和结束点，然后将不在原始上下文中的词进行掩码操作，确保答案都是来自上下文中。最后对开始点和结束点的位置向量做交叉熵损失的计算。

4.1.2 训练流程

4.1.2.1 数据预处理

在数据预处理阶段，使用 SentencePiece 进行分词，它非常适合神经机器翻译，SentencePiece 可以直接从原始句子训练子词模型，这样在对于问题回答的任务上可以实现更高的性能。ALBERT-base-v2 模型的句子最大长度设置为 384，ALBERT-xxlarge-v2 模型的句子最大长度设置为 280。滑窗法的步长 doc_stride 设置成 128，问题最大长度是 15。

使用 AlbertTokenizer 作为分词器，分词器词表大小为 30000，特殊词有以下 5 个：<unk>未知词，[mask]掩码，[CLS]分类符，[SEP]分隔符，<pad>补全符，对应的 ID 分别为 1，4，2，3，0。最大长度 512。

使用 SquadProcessor 获取 SQuAD 2.0 的数据：如果不进行评估的话就获取训练数据样例，进行评估就获取验证集数据样例。然后创建 SquadExample，包括：问题对的 ID，问题文本，上下文文本，答案文本，开始位置，文章题目，是否为不能解答的题目，以及总答案。

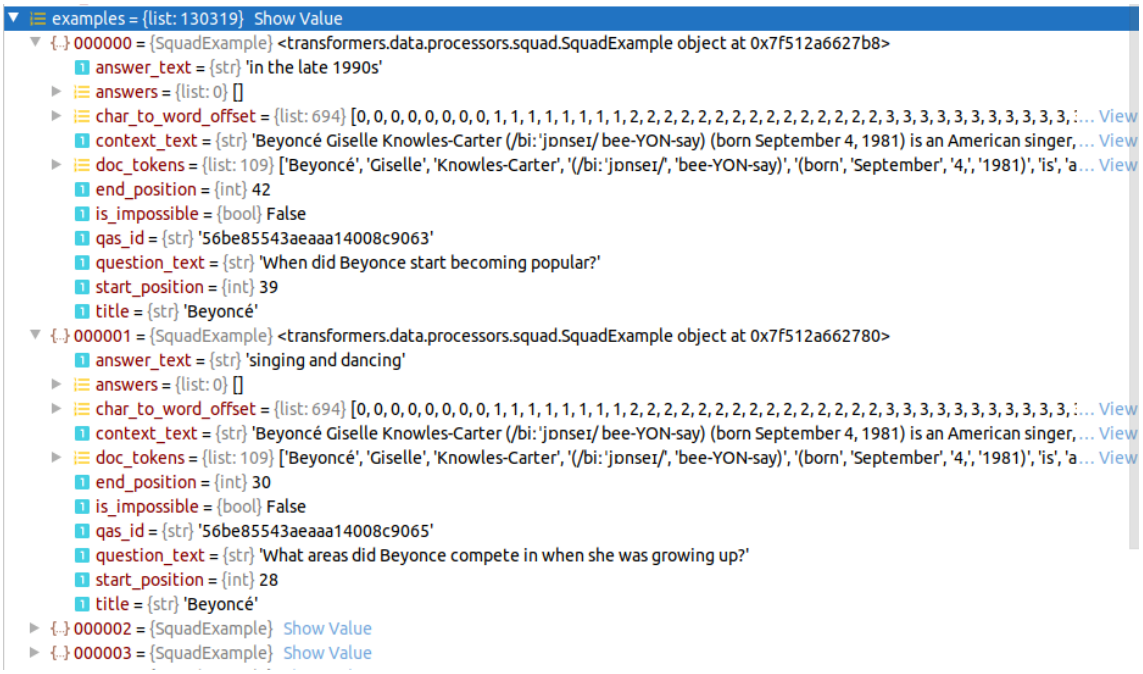


图 4-2 数据样例

最后将样例转换为特征，将输入 ID，注意力掩码，句子分句标记，分类标记，位置掩码，是否为不能解答的题目，所有开始点，所有结束点转换为 Tensor，然后

汇总成 TensorDataset，直接作为输入给到模型。

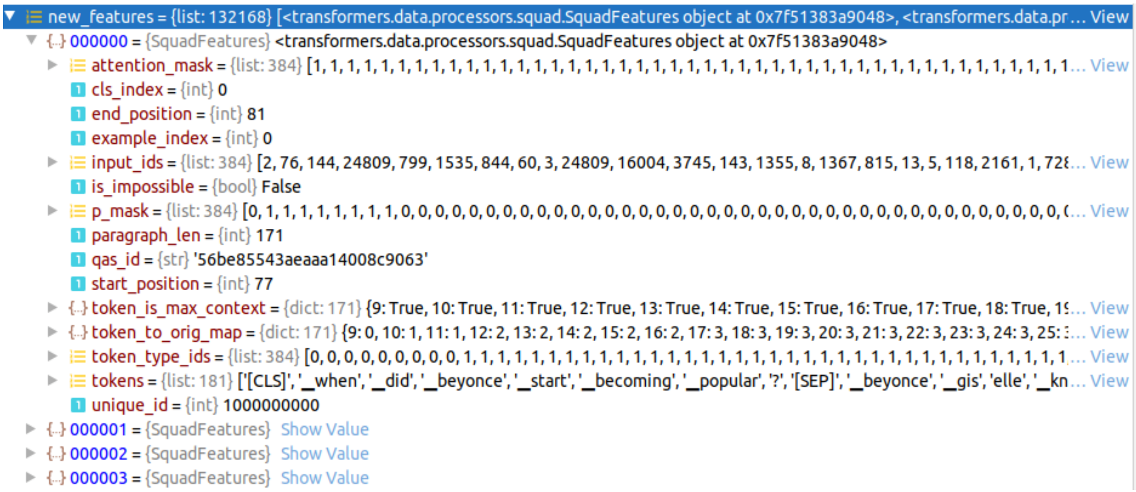


图 4-3 构造的特征

4.1.2.2 训练

训练步骤：

1.首先确立预训练模型“albert-base-v2”，它堆叠了 12 层，嵌入层的维度是 128，隐藏层的维度是 768，12 个头，1100 万个参数。如图 4-4 所示。根据模型要求也可以选择 “albert-xxlarge-v2”。它堆叠了 12 层，嵌入层维度也是 128，隐藏层维度大大增加，变成 4096，有 64 个头，有 22300 万个参数，相比较 base 模型，参数量提升巨大。

2. 获取模型。模型有多种选择：“AlbertForQuestionAnswering”，“AlbertLinear_highway”，“albert_bidaf” 和 “albert_bidaf2”。分别代表 ALBERT + ALBERT-OUT， ALBERT + Highway， (ALABERT-base-v2 + char_emb) + GRU-Enc + Attention + Self-Attention + GRU-Dec + BiDAF-OUT 和(ALABERT-base-v2 + char_emb) + GRU-Enc + Highway + GRU-Dec + BiDAF-OUT。图 4-4 表示 AlbertForQuestionAnswering 的模型结构，使用“albert-base-v2”：

```
AlbertForQuestionAnswering(
  (albert): AlbertModel(
    (embeddings): AlbertEmbeddings(
      (word_embeddings): Embedding(30000, 128, padding_idx=0)
      (position_embeddings): Embedding(512, 128)
      (token_type_embeddings): Embedding(2, 128)
      (LayerNorm): LayerNorm((128,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0, inplace=False)
    )
    (encoder): AlbertTransformer(
      (embedding_hidden_mapping_in): Linear(in_features=128,
 out_features=768, bias=True)
      (albert_layer_groups): ModuleList(
        (0): AlbertLayerGroup(
          (albert_layers): ModuleList(
            (0): AlbertLayer(
              (full_layer_layer_norm): LayerNorm((768,), eps=1e-12,
 elementwise_affine=True)
              (attention): AlbertAttention(
                (query): Linear(in_features=768, out_features=768,
 bias=True)
                (key): Linear(in_features=768, out_features=768, bias=True)
                (value): Linear(in_features=768, out_features=768,
 bias=True)
                (dropout): Dropout(p=0, inplace=False)
                (dense): Linear(in_features=768, out_features=768,
 bias=True)
                (LayerNorm): LayerNorm((768,), eps=1e-12,
 elementwise_affine=True)
              )
              (ffn): Linear(in_features=768, out_features=3072, bias=True)
              (ffn_output): Linear(in_features=3072, out_features=768,
 bias=True)
            )
          )
        )
      )
    )
    (pooler): Linear(in_features=768, out_features=768, bias=True)
    (pooler_activation): Tanh()
  )
  (qa_outputs): Linear(in_features=768, out_features=2, bias=True)
)
```

图 4-4 AlbertForQuestionAnswering 模型结构

3.训练的时候最多保留 5 个检查点，超过 5 个就删除前面最差的点。优化器还用 AdamW 优化器[53]，学习率为 3e-5。分词器使用预训练模型“albert-base-v2”的 AlbertTokenizer。

4.训练的时候用小批次的数据，批次大小设置为 7，一个批次包括六部分，第一部分是输入的 ID，第二部分是注意力掩码，第三部分是分隔符，第四部分是开始的位置，第五部分是结束的位置。如果使用双向注意力流模型，还会用到第六部分即字母的表示。

5.进行前向传播，获取输出：

a) 首先进入嵌入层，获取输入 ID，位置 ID，分隔符 ID，通过 ALBERT 的权重进行查找，得到输入嵌入，位置嵌入，分隔符嵌入。然后三者相加，经过归一化层 LayerNorm[54]，经过随机失活的 dropout 得到最终的嵌入输出。

b) 然后进入编码层，编码层要用到 AlbertTransformer，把嵌入层得到的输出转化为 Transformer 所需要的维度 768，隐藏层共 12 层，每一层要有一个隐层输出。计算每一层的隐层输出需要先计算注意力层的输出，由公式(2-5)可以得到注意力层的输出，对注意力的输出进行 softmax 得到注意力的概率表示，然后与 Value 相乘得到上下文层的表示，这样就得到了注意力的输出。将注意力的输出传入前馈网络，将维度从 768 变为 3072，经过激活函数 GELU，再经过前馈网络的输出层将维度从 3072 变回 768，最后经过全层的 LayerNorm 得到这一层最终的隐层输出。经过 12 层后得到最终的隐层输出，此输出为编码层的输出。编码层输出的第一个就是序列的输出。

c) 进入池化激活层，池化激活函数用 Tanh，得到池化层的输出。将序列的输出和池化层的输出相加，就是 ALBERT 的最终的输出。

d) 经过问题回答的输出层得到总的预测的 logits。将其分为两份，一份为开始点向量，一份为结束点向量。开始点向量和结束点向量作为输出。对开始点向量和开始位置求交叉熵损失，同样的，对结束点向量和结束位置做交叉熵损失。将二者的交叉熵损失求平均，得到最终的损失，将损失与上一步的输出组合，得到训练的输出。

6.然后进行后向传播，进行梯度裁剪，优化器优化，以及对参数进行指数移动平均等操作，这样就完成一个批次的训练流程。

7.经过 50000 步的训练，用验证集的数据 dev_dataset 进行一次前向传播，只不过开始位置并没有，所以得到的输出结果只有序列输出和池化输出。进行评估，参数主要是 EM 和 F1。计算方式如(3-23)。

4.2 模型结果

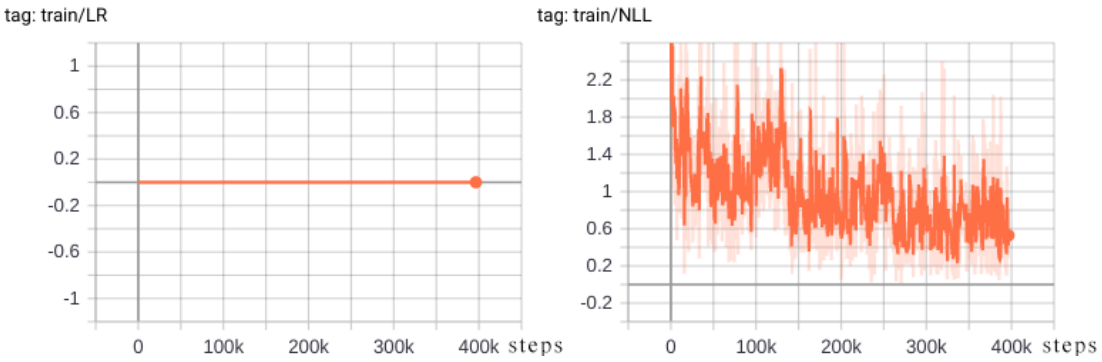
模型结果图中有 2 条线，虚线代表真实值，实线代表经过平滑处理的曲线，如果不经过平滑处理则曲线可能会显示不全，经过平滑处理更好的显示出曲线的趋势。所以平滑度统一设置成设为 0.6，使用 Tensorboard 输出图片。

（一）ALBERT + ALBERT-OUT

ALBERT 只加上 ALBERT-OUT 层，模型的目的是可以清楚地看到加上其它层对模型是否有改进。

训练集结果：

LR 保持常数 3×10^{-5} 不变，NLL 大约在 0.2 到 2.2 之间。

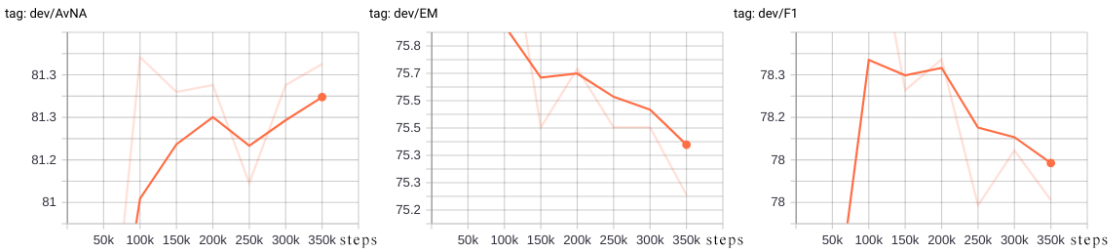


(a) LR (b) NLL

图 4-5 模型一训练集结果

验证集结果：

虚线为真实值，实线为经过平滑处理。最重要的两个指标为 EM 和 F1，其中 EM 在经过 200k 步的训练后达到 75.70，F1 达到 78.30。



(a) AvNA (b) EM (c) F1

图 4-6 模型一验证集结果

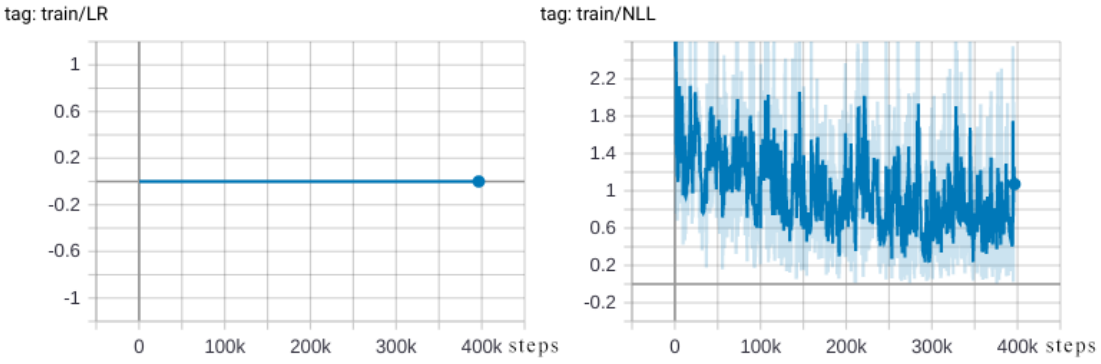
模型一在 200k 步之前是上升的趋势，200k 步以后 EM 的值和 F1 的值开始减少，说明过拟合了。所以在 200k 步的结果为该模型的最佳表现。

模型一在 EM 指标的值比 BiDAF 模型提高了 17.10，相当于比 BiDAF 模型提高了 30.17%，在 F1 指标的值比 BiDAF 模型提高了 16.35, 相当于比 BiDAF 模型提高了 27.01%。

（二）ALBERT-base-v2 + Highway + ALBERT-OUT

训练集结果：

LR 保持常数 3×10^{-5} 不变，NLL 大约在 0.4 到 2.0 之间。

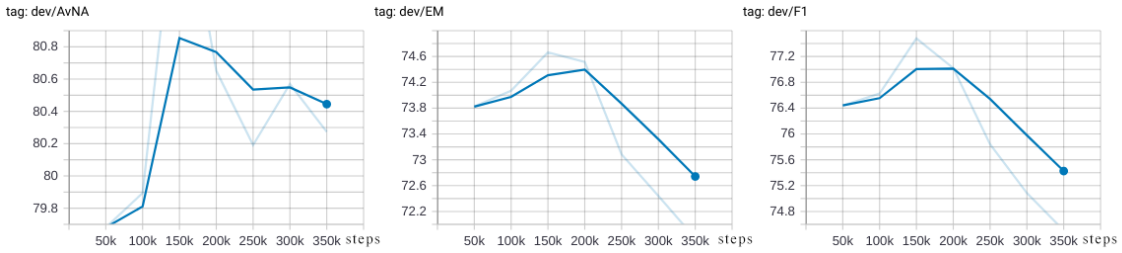


(a) LR (b) NLL

图 4-7 模型二训练集结果

验证集结果：

虚线为真实值，实线为经过平滑处理。其中 EM 在经过 150k 步的训练后达到 74.60，F1 达到 77.40。



(a) AvNA (b) EM (c) F1

图 4-8 模型二验证集结果

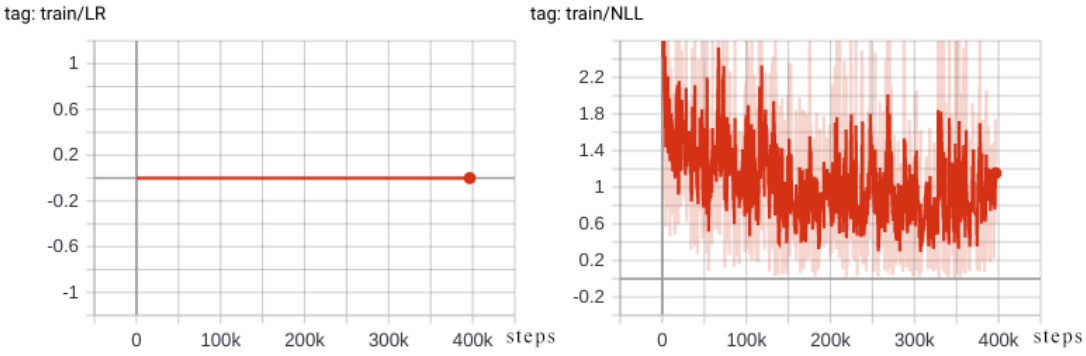
模型二的 EM 和 F1 真实值（虚线）在 150k 步之前逐步上升，在 150k 步以后 EM 的值和 F1 的值开始减少，说明过拟合了。所以模型二在 150k 步的结果为该模型的最佳表现。

模型二在 EM 指标的值比 BiDAF 模型提高了 16.00，相当于比 BiDAF 模型提高了 27.30%，在 F1 指标的值比 BiDAF 模型提高了 15.45, 相当于比 BiDAF 模型提高了 24.94%。

（三）(ALABERT-base-v2+char_emb) + GRU-Enc + Attention + Self-Attention + GRU-Dec + BiDAF-OUT

训练集结果：

LR 保持常数 3×10^{-5} 不变，NLL 大约在 0.4 到 2.2 之间。

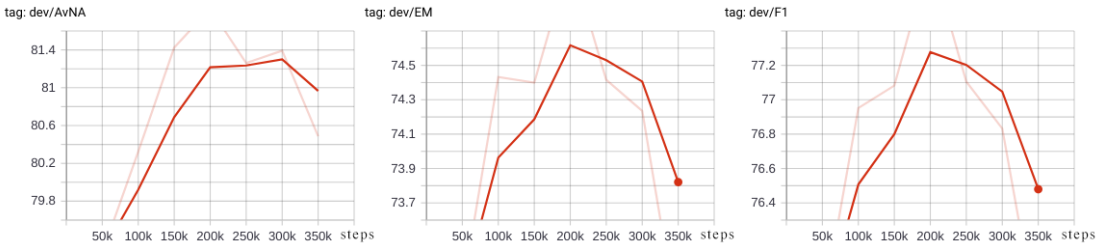


(a) LR (b) NLL

图 4-9 模型三训练集结果

验证集结果：

虚线为真实值，实线为经过平滑处理。其中 EM 在经过 200k 步的训练后达到 74.60，F1 达到 77.30。



(a) AvNA (b) EM (c) F1

图 4-10 模型三验证集结果

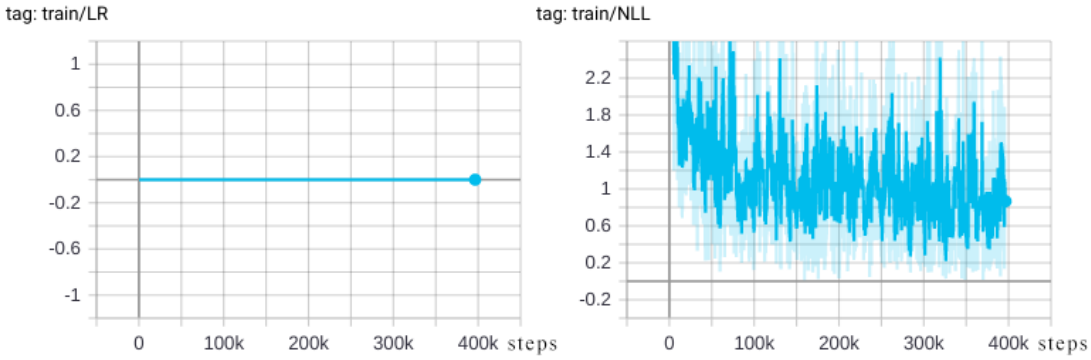
模型三在 200k 步之前 EM 和 F1 的值逐步上升，在 200k 步以后 EM 的值和 F1 的值开始减少，说明在 200k 步时过拟合了。所以模型三在 200k 步的结果为该模型的最佳表现。

模型三在 EM 指标的值比 BiDAF 模型提高了 16.00，相当于比 BiDAF 模型提高了 27.30%，在 F1 指标的值比 BiDAF 模型提高了 15.35, 相当于比 BiDAF 模型提高了 24.78%。

（四）(ALABERT-base-v2+char_emb)+ GRU-Enc + Highway + GRU-Dec + BiDAF-OUT

训练集结果：

LR 保持常数3×10^{-5}不变，NLL 大约在 0.4 到 2.2 之间。

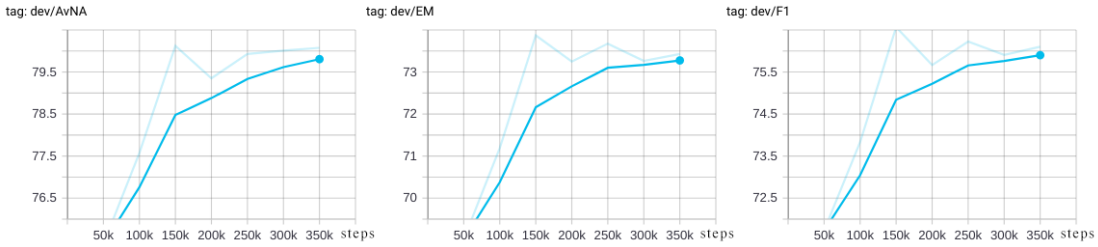


(a) LR (b) NLL

图 4-11 模型四训练集结果

验证集结果：

虚线为真实值，实线为经过平滑处理。其中 EM 在经过 150k 步的训练后达到 75.00，F1 达到 76.50。



(a) AvNA (b) EM (c) F1

图 4-12 模型四验证集结果

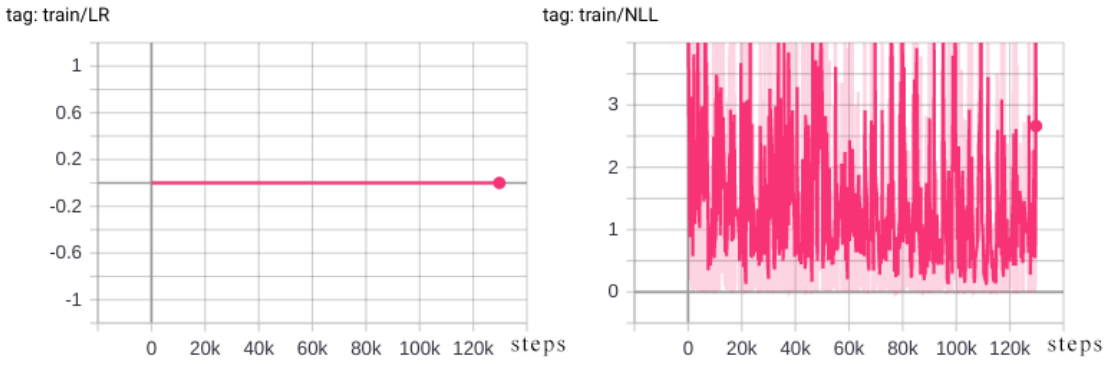
模型四在 150k 步之前 EM 和 F1 的值逐步上升，再 150k 步以后 EM 的值和 F1 的值开始减少，说明模型四在 150k 步过拟合了。所以模型四在 150k 步的结果为该模型的最佳表现。

模型四在 EM 指标的值比 BiDAF 模型提高了 16.40，相当于比 BiDAF 模型提高了 27.99%，在 F1 指标的值比 BiDAF 模型提高了 14.55, 相当于比 BiDAF 模型提高了 23.49%。

（五）：ALBERT-xxlarge-v2 + ALBERT-OUT

训练集结果：

LR 保持常数 7×10^{-6} 不变，NLL 大约在 0 到 4.0 之间。

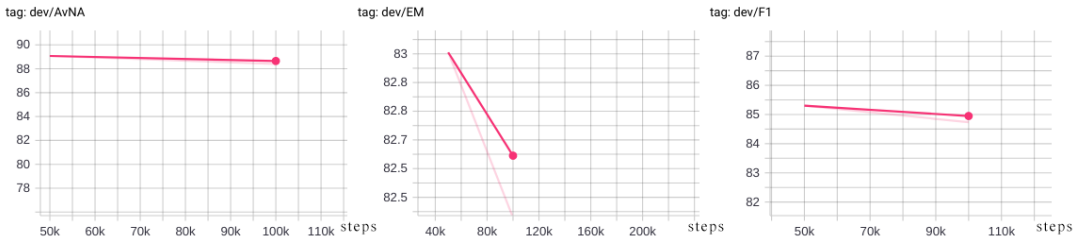


(a) LR (b) NLL

图 4-13 模型五训练集结果

验证集结果：

虚线为真实值，实线为经过平滑处理。最重要的两个指标为 EM 和 F1。在 50k 到 100k 之间模型效果开始下降，在 50k 步的时候模型效果最好。其中 EM 为 82.95，F1 的值为 85.30。



(a) AvNA (b) EM (c) F1

图 4-14 模型五验证集结果

模型五的效果是最好的，因为 ALBBERT-xxlarge 的参数确实太大了，在 50k 的效果就已经比前几个模型要好。

模型五在 EM 指标的值比 BiDAF 模型提高了 24.35，相当于比 BiDAF 模型提高了 41.55%，在 F1 指标的值比 BiDAF 模型提高了 23.35，相当于比 BiDAF 模型提高了 37.69%。

表 4-1 5 种模型在测试集中的表现

模型在测试集的表现	EM	F1
模型一	76.28	78.68
模型二	74.51	77.48
模型三	75.12	77.84
模型四	73.87	76.57
模型五	82.06	84.35

4.3 本章小结

本章主要在 ALBERT 模型的基础上，添加了不同的层，研究了对 SQuAD 第二版数据集的效果。经过试验，模型五 ALBERT-xxlarge-v2 是效果最好的模型，当然这得益于参数比前四个模型要大很多。在前四个模型中使用的都是 ALBERT-base-v2 模型，但是模型一直接加输出层的效果比加上高速网络、使用 GRU 和 attention 层的效果要好。

本文研究的意义：智能问答应用广泛，如 Siri，天猫精灵等服务广泛使用了智能问答。但是智能问答的发展还处于基础阶段，要想成熟的应用还有很长的路要走。人类在 SQuAD 第二版数据集的水平是 EM 为 86.831，F1 为 89.452。通过本文的研究，提高了机器在智能问答数据集上的表现水平，也发现经过训练后机器的水平正在无限接近甚至是超越人类的水平，这在智能问答领域是一个重要的节点，当机器超越人类的水平，意味着智能问答系统在不断成熟，促进了智能问答系统的发展，为之后的工业界的落地奠定了基础。

在国际上，Yuwen Zhang 等人在 BERT 的基础上通过增加不同的层来研究在 SQuAD 2.0 数据集的表现[55]，本文在其基础上，使用 ALBERT 进行了研究。相较于 BERT 研究的结果，本文最佳的结果较其所有结果都要更高，表现也更好。

第五章 总结与展望

自然语言处理是深度学习的一个重要的领域，而问题回答是自然语言处理的一个重要分支。在日常生活中，智能问答十分常见，如 Siri，天猫精灵，小爱同学，小度小度等应用或设备。如何能让机器更好的理解语言的信息并进行准确回答，是人类一直努力实现的目标。

本文首先介绍了自然语言处理的发展历程，从独热编码到词嵌入表示，从 word2vec 的两个模型 CBOW 和 Skip-gram 到语言模型，然后进入自然语言处理的经典循环神经网络 RNN，长短期记忆 LSTM 和门控循环单元 GRU。机器翻译 2 种或多种语言需要用得到序列到序列的模型即 Seq2seq，注意力的出现使得 Seq2seq 的效果又提升不少。

然后介绍了智能问答的方法，智能问答的分类以及本文对于智能问答所用到的预训练模型或方法，也就是主要的研究内容，由此引出了预训练模型。

预训练模型在自然语言处理中正发挥越来越重要的作用，ELMo 模型的出现对一词多义的问题有了解决的方法，但是缺点也明显，只能单向获取信息，不能同时获取全部上下文信息。Transformer 的出现使得自然语言处理进入了黄金时期，尤其自注意力的出现使得模型不仅具有 RNN 的优点，还能够并行计算，大大提高了模型的效果及运行效率。GPT 第一次使用了 Transformer 结构，不过只是单向的，而 BERT 使用了双向 Transformer，实现了真正的双向特征表示。XLNet 对 BERT 掩码的缺陷进行了改进，使用排列模型能够看到全部的信息且形式上还是从左到右的进行预测。ALBERT 模型的出现使得模型的参数大大减少，效果不减少或减少有限。

本文在 ALBERT 的基础上对智能问答做了研究，与双向注意力流模型 BiDAF 做了对比，ALBERT 的基本模型 ALBERT-base-v2 在 EM 指标上比 BiDAF 模型提高了 30.17%，在 F1 指标上比 BiDAF 模型提高了 27.01%。在 ALBERT 模型中添加高速网络，使用 GRU 和注意力层，使用 GRU 和高速网络，对模型的提高不大，甚至有的效果还不如仅使用基本模型。使用参数大的 ALBERT-xlarge 和 ALBERT-xxlarge 对模型的提升效果明显。

但是 ALBERT 虽然在参数上减少了，但是运算量和运算时间并没有减少，所以未来希望不仅能够大大减少参数，同时运算时间也同样大大缩减，这样可以提高人们的效率，节约更多的时间和资源。

致　谢

时光飞逝，三年的硕士生涯转眼间也即将结束，在这三年里，有过人生中的迷茫，有过对梦想的渴望，有过挫折，有过失望，但更多的是对未来的希望以及自身的成长。这三年，是我无悔的三年。我做出了人生中的重大决定，得到了我的导师，我的父母，我的朋友的支持。在这里，我要向他们表示感谢。

首先感谢我的导师殷勇老师，三年前能够接收我入组，成为团队的一员，没有殷老师也就没有我的今天。感谢殷老师尊重且支持我的决定，并在精神上给与我鼓励，这是宝贵的财富，我一直铭记在心。虽然只与您相处了三年，但这三年您不断教育我要做一名优秀努力上进的学生，虽然我现在谈不上优秀，但是努力上进我做到了，殷老师是我一辈子的导师，谢谢您殷老师。然后感谢团队的李海龙老师，王彬老师，袁学松老师的关怀与帮助。

其次，感谢我的父母。父母是我坚强的后盾，总是在我背后默默支持着我，给予我鼓励与帮助。在我遭遇挫折时，父母也是不断鼓励我，激励我让我奋起直追，迎头赶上。疫情期间，父母更是将后勤做到极致，保证我充足的学习时间。父母在变老，我也在长大，以后我要做父母的后盾，报答父母的养育之恩。

研究生三年，还要感谢李阳明老师对我的关怀与帮助，李老师更像一位兄长，我们无话不说，友谊深厚。感谢金重九老师对我模型问题的解答，没有金老师的耐心讲解与帮助，我的训练可能不会顺利进行。祝福老师们工作顺心，步步高升。

感谢瞿铖辉，简丽蓉，曾健等诸多好朋友对我的帮助与关心，感谢我的师兄师弟的友好相处，研会的小伙伴们的陪伴，祝福你们未来一切顺利，学业有成！

感谢崔凯，本科到硕士，走过的这七年，见证者彼此的成长与进步，在我最需要支持的时候坚定地站在了我这一边，支持并帮助我，希望未来我们继续携手并进，互相支持。

最后感谢评审专家，老师们，感谢您们百忙之中抽出时间给我的论文提出宝贵意见。

谢谢！

参考文献

[1] D. Bahdanau, K. Cho and Y. J. a. p. a. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate[J]. 2014.

[2] 赵妍妍,秦兵,刘挺.文本情感分析[J].软件学报. 2010, 21(8): 1834-1848.

[3] 向晓雯,史晓东,曾华琳.一个统计与规则相结合的中文命名实体识别系统[J].计算机应用. 2005, 25(10): 2404-2406.

[4] 严红.词向量发展综述[J].现代计算机. 2019, (8): 11.

[5] X. Lu and B. J. a. p. a. Ni. Low-Dimensional Semantic Space: From Text to Word Embedding[J]. 2019.

[6] 吴禀雅,魏苗.从深度学习回顾自然语言处理词嵌入方法[J].电脑知识与技术. 2016,12(36): 184.

[7] Y. Goldberg and O. J. a. p. a. Levy. Word2vec Explained: Deriving Mikolov Et Al.'S Negative-Sampling Word-Embedding Method[J]. 2014.

[8] T. Mikolov, K. Chen, G. Corrado, et al. Efficient Estimation of Word Representations in Vector Space[J]. 2013.

[9] T. Mikolov, I. Sutskever, K. Chen, et al. Distributed Representations of Words and Phrases and Their Compositionality[C]. Advances in neural information processing systems, 2013, 3111-3119.

[10] 张剑,屈丹,李真.基于词向量特征的循环神经网络语言模型[J]. 模式识别与人工智能.2015, 28(4): 299-305.

[11] Y. Bengio, R. Ducharme, P. Vincent, et al. A Neural Probabilistic Language Model[J]. 2003, 3(Feb): 1137-1155.

[12] W. B. Cavnar and J. M. Trenkle. N-Gram-Based Text Categorization[C]. Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval, 1994.

[13] A. J. P. D. N. P. Sherstinsky. Fundamentals of Recurrent Neural Network (Rnn) and Long Short-Term Memory (Lstm) Network[J]. 2020, 404: 132306.

[14] P. Zhou, W. Shi, J. Tian, et al. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification[C]. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2016, 207-212.

[15] F. A. Gers and E. J. I. T. o. N. N. Schmidhuber. Lstm Recurrent Networks Learn Simple Context-Free and Context-Sensitive Languages[J]. 2001, 12(6): 1333-1340.

[16] C.Olah. Understanding LSTM Networks[OL]. https://colah.github.io/posts/2015-08-Understa

nding-LSTMs/.

[17] K. Cho, B. Van Merriënboer, C. Gulcehre, et al. Learning Phrase Representations Using Rnn Encoder-Decoder for Statistical Machine Translation[J]. 2014.

[18] I. Sutskever, O. Vinyals and Q. V. Le. Sequence to Sequence Learning with Neural Networks[C]. Advances in neural information processing systems, 2014, 3104-3112.

[19] 王红,史金钏,张志伟.基于注意力机制的 lstm 的语义关系抽取[J].计算机应用研究.2018, v.35;No.319(5): 143-146+166.

[20] C. Picard, K. E. Smith, K. Picard, et al. Can Alexa, Cortana, Google Assistant and Siri Save Your Life? A Mixed-Methods Analysis of Virtual Digital Assistants and Their Responses to First Aid and Basic Life Support Queries[J]. 2020: bmjinnov-2018-000326.

[21] 李雨蒙. Alexa 语音识别交互开启万物互联新生态[J].中国民商.2017, 000(2): P.76-77.

[22] 张婷婷,许聪,孟晓波. 基于文本挖掘的智能音箱在线评论分析[C]. 2019 年(第六届)全国大学生统计建模大赛优秀论文集, 2019.

[23] 李方方,马昊宇.基于聊天机器人的智能导购系统[J].福建电脑.2018, 034(2): 27-28.

[24] 肖杰.从"微软小冰"探讨人工智能的前景与未来[J]. 科技创新与应用.2018, (7): 10-11.

[25] 杨理想. 面向特定领域的问答系统及其在 NAO 机器人平台上的实现[D]. 南京大学, 2015.

[26] 邢超. 智能问答系统的设计与实现[D]. 北京交通大学, 2015.

[27] P. Rajpurkar, J. Zhang, K. Lopyrev, et al. Squad: 100,000+ Questions for Machine Comprehension of Text[J]. 2016.

[28] 艾丽斯,唐卫红,傅云斌,等.抽取式自动文本生成算法[J].2018, No.200(4): 75-84.

[29] 桑志杰.生成式问答系统技术研究与实现[D].北京邮电大学, 2019.

[30] 张琳,胡杰. FAQ 问答系统句子相似度计算[J]. 郑州大学学报.2010, (1): 62-66.

[31] 曹勇刚,曹羽中,金茂忠,等. 面向信息检索的自适应中文分词系统[J].2006, (3): 14-21.

[32] P. Rajpurkar, R. Jia and P. J. a. p. a. Liang. Know What You Don't Know: Unanswerable Questions for Squad[J]. 2018.

[33] M. Seo, A. Kembhavi, A. Farhadi, et al. Bidirectional Attention Flow for Machine Comprehension[J]. 2016.

[34] M. E. Peters, M. Neumann, M. Iyyer, et al. Deep Contextualized Word Representations[J]. 2018.

[35] J. Devlin, M.-W. Chang, K. Lee, et al. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding[J]. 2018.

[36] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention Is All You Need[C]. Advances in neural information processing systems, 2017, 5998-6008.

[37] J. Alammar. The Illustrated Transformer[OL]. http://jalammar.github.io/illustrated-transformer/.

[38] A. Radford, K. Narasimhan, T. Salimans, et al. Improving Language Understanding by Generative Pre-Training[J]. 2018.

[39] Z. Yang, Z. Dai, Y. Yang, et al. Xlnet: Generalized Autoregressive Pretraining for Language Understanding[C]. Advances in neural information processing systems, 2019, 5753-5763.

[40] Z. Lan, M. Chen, S. Goodman, et al. Albert: A Lite Bert for Self-Supervised Learning of Language Representations[J]. 2019.

[41] T. Kudo and J. J. a. p. a. Richardson. Sentencepiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing[J]. 2018.

[42] Y. Liu, M. Ott, N. Goyal, et al. Roberta: A Robustly Optimized Bert Pretraining Approach[J]. 2019.

[43] L. Gong, D. He, Z. Li, et al. Efficient Training of Bert by Progressively Stacking[C]. International Conference on Machine Learning, 2019, 2337-2346.

[44] M. Dehghani, S. Gouws, O. Vinyals, et al. Universal Transformers[J]. 2018.

[45] S. Bai, J. Z. Kolter and V. Koltun. Deep Equilibrium Models[C]. Advances in Neural Information Processing Systems, 2019, 690-701.

[46] D. Hendrycks and K. J. a. p. a. Gimpel. Gaussian Error Linear Units (Gelus)[J]. 2016.

[47] R. J. Pranav Rajpurkar, Percy Liang. Squad 2.0[OL]. https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/Harvard_University.html?model=nlnet%20(single%20model)%20(Microsoft%20Research%20Asia)&version=v2.0.

[48] J. Pennington, R. Socher and C. D. Manning. Glove: Global Vectors for Word Representation[C]. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, 1532-1543.

[49] M. D. J. a. p. a. Zeiler. Adadelta: An Adaptive Learning Rate Method[J]. 2012.

[50] J. Duchi, E. Hazan and Y. J. J. o. m. l. r. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization[J]. 2011, 12(7).

[51] P. Baldi and P. J. Sadowski. Understanding Dropout[C]. Advances in neural information processing systems, 2013, 2814-2822.

[52] S. Ioffe and C. J. a. p. a. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. 2015.

[53] I. Loshchilov and F. Hutter. Fixing Weight Decay Regularization in Adam[J]. 2018.

[54] J. L. Ba, J. R. Kiros and G. E. J. a. p. a. Hinton. Layer Normalization[J]. 2016.

[55] Yuwen Zhang, Zhaozhuo Xu. Bert for Question Answering on SQuAD 2.0[J]. 2019.

攻读硕士学位期间取得的成果

[1] 李敬鑫.自然语言处理的预训练模型综述[J]. 电子科技大学研究生学报,2020(已录用)