“莱斯杯”全国第一届“军事智能•机器阅读”挑战赛
xy 队技术总结报告

1. 任务描述

此次挑战赛聚焦于军事机器阅读理解任务，问题共涵盖如下 6 种类型：事实型问题、列表型问题、数值型问题、定义型问题、观点型问题，以及篇章型问题。数据示例如图 1 所示

question: 哪些国家被评为最和平的国家

title: 美媒发布全球和平指数：中国排名 80 高于美印俄

passage: 据美国《侨报》6 月 8 日晚报道，世界和平形势继续恶化，每年造成的经济损失超过 7 万亿美元。而中国的和平稳定局势虽有挑战，和平指数仍名列金砖四国之首。美国的和平指数持续好转，但仍位居中国之后。报道称，美国经济与和平研究所（IEP）8 日发布 2010 年度全球和平指数（GPI）时指出，全球经济衰退使得世界冲突和不稳定加剧，许多国家暴力犯罪显着增加。这是 IEP 连续第四年发布这一指数，全球和平指数连续第二年恶化。今年的全球和平指数是针对全球 149 个国家和地区进行的，选取了内政、外交、军事、治安等 23 个指标，进行定量定性综合评估。新西兰、冰岛、日本、奥地利、挪威、爱尔兰被评为最和平的国家；伊拉克、索马里、阿富汗、苏丹、巴基斯坦、以色列、俄罗斯被评为最不和平的国家。在金砖四国中，中国以总排名第 80 位排名第一，以下依次为巴西（第 83 位）、印度（第 128 位）、俄罗斯（第 143 位）。中国在与外部关系、军费和武器进出口、暴力犯罪率等方面得分较好。美国排名第 85 位，在主要发达国家中排名最低，但其得分持续好转，这主要得益于外部冲突的死亡数减少，以及政治稳定性的改善。影响美国得分的主要因素是内部和外部冲突、人权、恐怖袭击风险、武器出口、军力等。新华网

answer: 新西兰、冰岛、日本、奥地利、挪威、爱尔兰

图 1 数据示例

从直观上看，对于不同问题类型，理应采取不同的建模思路。建模思路可以概括为两种：生成式与抽取式。生成式主要基于 seq2sqe 架构，在解码阶段，逐词生成答案，优势在于可以生成未在原文中出现过的生词，所生成的答案在原文中也可以是非连续的。抽取式主要基于指针网络 pointer，从原文中对答案进行抽取，优势在于模型相对简单，计算效率较高。

通过对训练集数据进行分析可知，在所有数据中，约有 98%的答案是直接从原文（title + passage）中抽取的连续的片段。因此我们将该任务定义为“抽取式阅读理解”，另外不再对问题类型进行区分，统一建模。任务描述如下：

给定一个文档 c，以及与之相关的问题 q，对答案 a 进行预测，要求答案 a 是文档 c 的一个连续片段。具体来讲，就是对答案 a 在文档 c 的索引首尾位置进行预测，即

$$a = p[start : end]$$

优化目标为答案 a 的首尾索引联合条件概率最大，如下：

$$\max\ p(index_{start} \mid c, q) \times p(index_{end} \mid c, q)$$

本技术报告主要从数据预处理、模型构建、训练过程、集成策略、软硬件需求、软件运

行步骤等几个方面进行撰写。另外，需要说明的是，下面的分析过程，如未特殊说明，均指在初赛训练集上的结果。

2. 数据预处理

数据预处理在整个技术方案中非常重要，起到了举足轻重的作用。具体来讲，主要包含4个部分：数据转换、除燥、数据截断、答案构造。相比于英文数据，分词是中文数据预处理的一大特点，分词器的选择对数据预处理的效果具有很大影响，各分词器对比情况见2.5小节。

2.1 数据预处理

原始数据中含有大量噪声，主要包括繁简体、全半角、答案句首尾标点，具体转换方式如下：

（1） 将所有的繁体字全部转换为简体字
（2） 将所有的全角数字转换为半角数字
（3） 将所有的全角字母转换为半角字母
（4） 除去所有答案的首尾标点
（5） 将间隔符转换为空格，如‘\u8000’，‘\t’
（6） 除去标题、文档、问题、答案的句首尾空格

2.2 除燥

原始数据中有些数据是错误的，对于这些数据采用删除处理，错误数据类型如下：

（1） 标题和问题重复的数据
（2） 问题和答案重复的数据
（3） 问题、标题、原文、答案为空的数据

2.3 截断

我们使用（title + passage）作为我们的原文 p，原始数据中有许多原文 p 经分词后，长度过长，有些数据甚至过万，对于这些数据，计算机无法处理，我们对其进行截断，截断规则如下：

（1） 对原文进行分句，得到句子列表
（2） 按照下列顺序，从句子列表中，抽取句子，直到达到设定的最大长度
（3） 标题
（4） 核心句（与问题 rouge-L 匹配度最高的句子）
（5） 核心句的下一句
（6） 尾句
（7） 首句
（8） 蕴含句 上、中、下（问题的子序列）
（9） 核心句的下下句
（10） 核心句的上一句
（11） 核心句的下下下句
（12） 核心句的上上句

值得注意的是，原文中有许多句子是重复的，对于这些句子，我们保留第一次出现的情况，其他删除。如果答案出现在截断后的文本中，我们认为截断是准确的，我们仅使用截断准确的样本，作为我们的训练样本。截断长度对句子截断准确率的影响见表 1

表 1　截断长度对截断准确率的影响

Table 1　the performance of truncation length

	100	200	300	400	500	600	all
accuracy	91.96	95.80	96.54	96.74	96.86	96.87	97.88

综合模型精度和运行效率，我们设定截断长度为 500，截取准确率为 98.96%（96.86/97.88）

2.4 答案构造

模型通过对分词后的原文 p 的首尾索引位置进行预测，进而达到抽取答案的目的。经过分析发现，许多答案在原文中出现过不止一次，重复答案的比率在 20%左右，答案构造规则如下：

（1）　对原文 p 和答案 a 进行分词

（2）　检测答案 a 分词列表在原文 p 列表中出现的次数

（3）　当出现次数大于 1 时，候选答案前后扩展 5 个词，取与 question 的 rouge-L 匹配度最高的候选项作为标准答案，若有多个最高项，则按出现的先后顺序

（4）　当出现次数等于 1 时，直接返回答案索引

（5）　当出现次数小于 1 时，丢弃

由于分词错误问题，我们发现使用 jieba 细粒度分词，答案构造错误率较低，仅为 2.03%

2.5 分词器的选择

我们发现不同分词器对数据截断准确率和答案构建准确率具有巨大影响，分词准确度较高的分词器，在本任务中，并没有表现出较好的性能。分词错误样本数据如图 2 所示

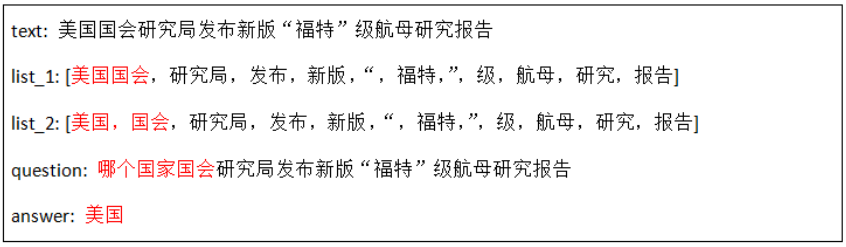

图 2 数据示例

其中 text 是从原文中抽取的一段话，list_1 表示分词列表，list_2 表示正确的分词列表。list_1 并没有把答案“美国”分离出来，这将会导致模型无法从 text 中抽取出正确的答案。

我们尝试了多种分类器，主要包括：jieba、jieba 细粒度分词、SnowNLP、thulac、pynlpir、CoreNLP、pyltp、pyhanlp。从数据截断准确率和答案构建准确率两个角度对分词器进行筛选，各分词器的表现见表 2

表 2　分词器的选择

Table 1　the performance of truncation length

	jieba	jieba 细粒度	SnowNLP	thulac	pynlpir	CoreNLP	pyltp	pyhanlp
截断准确率	94.32	**96.86**	93.1	94.53	92.44	94.62	95.21	96.79
答案构建准确率	95.3	**97.97**	95.4	96.1	95.91	92.43	91.62	97.61

我们发现使用 jieba 细粒度分词，在原文截断准确率和答案构建准确率两个指标，均取得了最优效果。

3 模型构建

我们的模型架构主要包含四个部分，如图 3 所示。主要包括表示层、编码层、匹配层以及抽取层。对于不同模型，采用相同的表示层与编码层，差别只存在于匹配层及抽取层。以下对模型架构的主要部分，分别展开介绍。

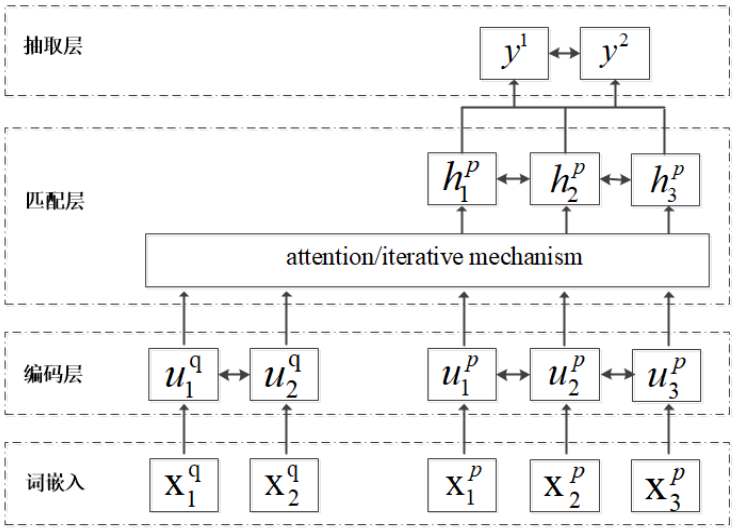

图 3 模型结构

3.1 表示层

该层主要负责对输入数据进行表示，即特征构建，我们采用三种表示方式，分别是词嵌入、词性标注、原文与问题词语的蕴含关系。其中，词嵌入是主要的特征表示方式。

当前，主流的词嵌入模型主要分为两种，静态模型（word2vec[1]、glove[2]）与动态模型（ELMo[3]、Bert[4]），在本任务中，我们尝试利用训练数据训练 glove 向量，但是发现该种方式泛化性能并不好，所以我们使用外部 word2vec 向量，该向量使用多种混合语料进行训练。经测试，通过引入外部词向量，在验证集上可以提高 0.5 个百分点。

另外，除了静态词向量以外，我们也尝试使用动态词向量对模型的泛化性能进行增强，但是实验结果显示，rouge-L 虽然略有增加，但是测试时间较长，显存占用较大，无法在决赛环节使用，所以我们并没有在最终方案中使用动态词向量。

除了词向量以外，在原文与问题的表示特征构建上，我们还引入了词性标注与蕴含关系特征，其中蕴含关系是指，原文中的词是否在问题中出现，反之亦然。

3.2 编码层

该层主要负责对文本表示进行编码，得到带有语义信息的向量表达，常用的编码模型为 LSTM、GRU。经测试，在隐层神经元数量相同的情况下，LSTM 的表达能力要优于 GRU，但是 LSTM 会占用较多的显存空间。因此我们针对显存占用较小的模型（Bi-daf、M-reader、M-reader-plus）使用 LSTM 编码方案，对于显存占用较多的模型（Match-lstm-plus、R-net）使用 GRU 编码方案。LSTM、GRU 计算方式如图 4、图 5 所示。

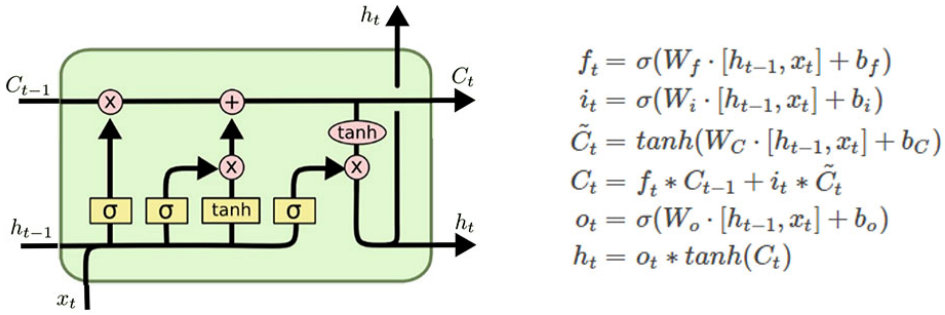

图 4　LSTM 计算流程图

在图 4 LSTM 计算流程图中，f_t，i_t，o_t 分别为遗忘门、输入门以及输出门，通过门的方式，达到对信息流的控制。

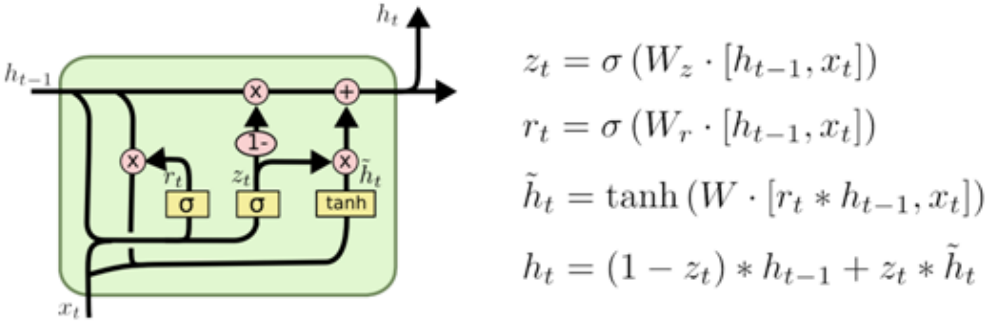

图 5　GRU 计算流程图

在图 5 GRU 计算流程图中，z_t，r_t 分别为更新门和重置门。相比于 LSTM，GRU 将遗忘门与输入门合成了一个单一的更新门，通过这种方式达到了简化计算的目的。与 LSTM 相比，GRU 参数空间较少，计算效率相对较高，但在相同隐藏层神经元数量的情况下，LSTM 的表达能力更强。

另外除了 LSTM 与 GRU 外，我们也尝试使用 CNN 对原文与问题进行编码，但发现由于不同原文之间长度差别较大，需要添加大量的占位符<pad>，导致计算效率较低，同时计算精度也略有不足。

3.3 匹配层与指针层

除了 QA-net 以外，其他主流抽取式阅读理解模型的差异，主要体现于此。对于不同的模型，匹配层不同，相应的指针层也不同。我们在本小节，对匹配层和指针层，进行统一介绍。

在本任务中，我们采用了 5 种匹配方案，分别是 Match-lstm-plus、R-net、Bi-daf、M-reader 以及 M-reader-plus，以下对这 5 种方案，分别展开介绍。

3.3.1 Match-lstm-plus

Match-lstm[5]是最早用抽取式方式来解决阅读理解问题的模型之一，其主要结构如图 6 所示。图 6(a)和图 6(b)分别为 Match-lstm 的序列模式和边界模式，其中序列模式可以归属为生成式模型，而边界模式属于抽取式模型。在“任务描述”中，我们已经对比了生成式模型和抽取式模型的优劣，在此不在累述，在本任务中我们使用 Match-lstm 的边界模式。

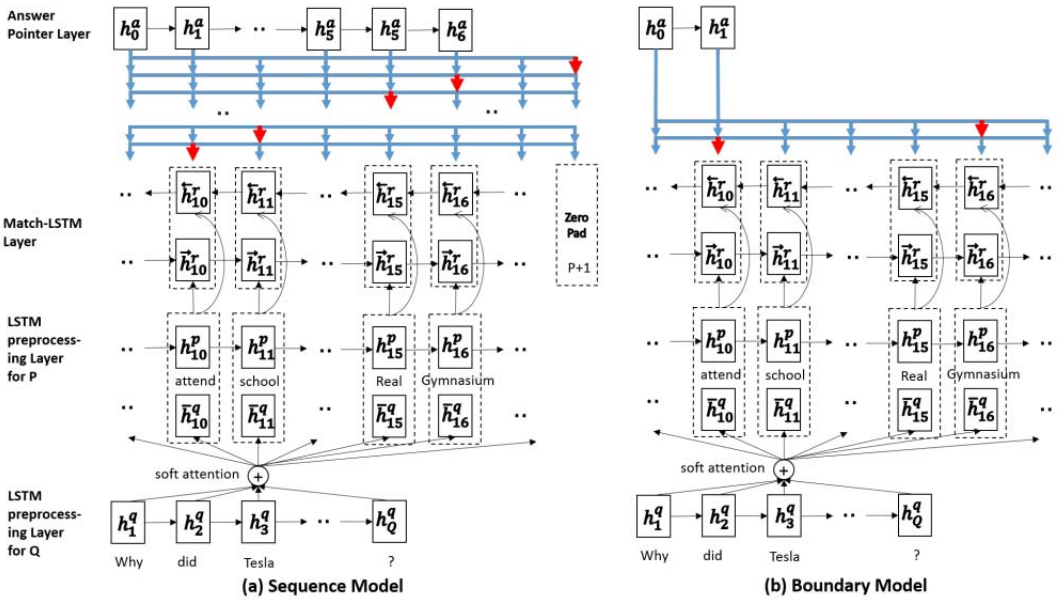

图 6　Match-lstm 模型结构

Match-lstm 的边界模型，详细描述如下

（1） 词嵌入、编码

首先通过 3.1 介绍的表示层，对原文 p 与问题 q 进行表示。然后使用双向 LSTM 模型分别对原文 p 与问题 q 进行编码，得到相应的编码向量

$$\mathrm{H}^{p} = BiLSTM(p)$$

$$\mathrm{H}^{q} = BiLSTM(q)$$

（2）匹配

该层主要通过注意力机制对原文 p 与问题 q 进行匹配。对原文 p 中的每一个词编码，计算其关于问题 q 的注意力分布 α，利用该注意力分布汇总问题的表示。串联原文 p 与汇总问题表示，并通过一个新的 LSTM 对其进行进一步的拟合，得到带有问题信息的原文表示，这种方式也被称为“question-aware attention passage representation”

$$G_i = \tanh(W^q H^q + (W^p h_i^p + W^r h_{i-1}^r + b^p) \otimes e_Q)$$

$$\alpha_i = \mathrm{soft\,max}(w^T G_i + b \otimes e_Q)$$

$$h_i^r = LSTM(z_i, h_{i-1}^r)$$

$$z_i = \left[h_i^p, H^q \alpha_i \right]$$

其中 α_i 表示原文 p 关于问题 q 的注意力分布，h_i^r 表示带有问题信息的原文表示，z_i 为原文 p 与汇总问题表示的串联向量。我们从正反两个方向，分别对原文 p 与问题 q 进行匹配，得到两个带有问题信息的原文表示 H_f^r，H_b^r，将其串联得到最终的带有问题信息的原文表示 $H^r = [H_f^r, H_b^r]$

（3） 答案抽取

边界模式仅对答案索引的首尾位置进行预测，计算方式与匹配模型较为相似，主要是通过 LSTM 对原文表示 H^r 进行迭代拟合，得到关于原文序列索引位置的概率分布，其中概率最大的值为我们所要预测的索引位置。

$$F_k = \tanh(VH^r + (W^a h_{k-1}^a + b^a) \otimes e_{(p+1)})$$

$$\beta_k = \text{soft}\max(v^T F_k + c \otimes e_{(p+1)})$$

$$h_k^a = LSTM(H^r \beta_k^T, h_{k-1}^a)$$

其中，β_k 为原文序列的索引概率分布。值得注意的是，在这里我们同样使用双向 LSTM，分别从 $index_{start} \to index_{end}$， $index_{end} \to index_{start}$ 两个方向对答案索引概率进行预测。

在实验的过程中，我们发现了一些有趣的现象：

1. LSTM 显存占用比 GRU 高
2. 在匹配模型中，隐层神经元数量越多，拟合能力越强
3. 仿照 R-net，在匹配模型中，使用门限单元对 z_i 做进一步处理，会得到更好的结果
4. 在匹配模型后，使用一个新的 LSTM 对原文表示 H^r 进行平滑，会得到更好的结果

基于上述现象，我们对 Match-lstm 进行改进，得到 Match-lstm-plus 模型，改进规则如下：

1. 将所有的 LSTM 换成 GRU，这种方式可以节省显存
2. 增大隐层神经元数量
3. 在匹配模型中，增加门限单元，对 z_i 中的信息做过滤
4. 在匹配模型后，增加一个 GRU，平滑 H^r

经过上述改进，我们改良后的 Match-lstm-plus 模型，相比于原版模型，在 rouge-L 指标上，大约可以提升 0.5 个百分点。

3.3.2 R-net

R-net[6]是由微软亚洲研究院提出，相当于原版 Match-lstm 的升级版。在 Match-lstm 的基础上，增加了门限网络、self-attention、平滑层，模型架构如图 7 所示

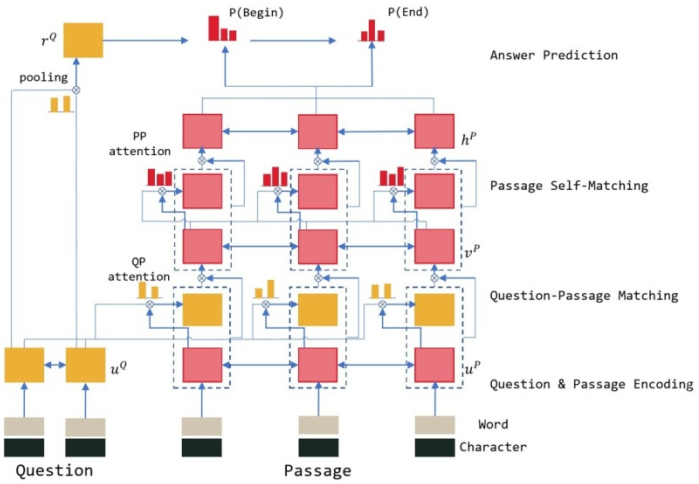

图 7　R-net 模型结构

（1） 词嵌入、编码

首先通过 3.1 介绍的表示层，对原文 p 与问题 q 进行表示。因为 R-net 模型占用显存较多，所以本方案使用双向 GRU 模型分别对原文 p 与问题 q 进行编码，得到相应的编码向量

$$\mathrm{H}^{p} = BiGRU(p)$$

$$\mathrm{H}^{q} = BiGRU(q)$$

（2）匹配

匹配层的基本架构与 Match-lstm 基本相同，不同之处在于 R-net 额外增加了门限网络、自注意力机制以及平滑层。

门限网络主要通过门控制的方式，对 p 与汇总问题表示 $H^{q}\alpha_i$ 的串联向量 $z_i = \left[h_i^{p}, H^{q}\alpha_i\right]$ 进行信息控制，可以有选择性地提取和文章有关的问题信息，具体计算方式如下：

$$g_t = sigmoid(W_g[h_i^{p}, H^{q}\alpha_i])$$

$$[h_i^{p}, H^{q}\alpha_i]^{*} = \mathrm{g}_t \otimes [h_i^{p}, H^{q}\alpha_i]$$

除了门限网络以外，模型还增加了自注意力机制，使得带有问题信息的原文表达 H^{r}，能得到进一步拟合，突出关键性信息，降低无效信息的干扰。该思想主要借鉴于“Attention is all you need”,具体计算方式如下：

$$h_t^{p} = BiGRU(h_{t-1}^{p}, [c_t, v_t^{p}]^{*})$$

$$s_j^{t} = v^{T} \tanh(W_v^{p} v_j^{p} + b)$$

$$a_i^{t} = \exp(s_i^{t}) / \sum_{j=1}^{n} \exp(s_j^{t})$$

$$c_t = \sum_{i=1}^{n} a_i^{t} v_i^{p}$$

其中，h_t^{p} 表示经过注意力机制处理后的带有问题信息的原文表示，v_t^{p} 表示上一层生成的带有问题信息的原文表示，c_t 为上一层生成的带有问题信息的原文 attention-pooling 向量，该向量可以认为是上一层的带有问题信息的原文综合信息的表达。另外，在自注意力机制中，该模型同样引入门限网络，即 $[c_t, v_t^{p}] \rightarrow [c_t, v_t^{p}]^{*}$

经过注意力机制编码后，得到带有问题信息的原文表示 H^{r}，模型使用另一个 GRU 作为平滑层，对原文表示 H^{r} 做进一步拟合。

（3） 答案抽取

该部分主要结构与 Match-lstm 基本相同，但在 RNN 的初始状态构造上略有不同。Match-lstm 在答案抽取层，RNN 的初始状态设定为空，而在 R-net 模型中，RNN 的初始状态被设定为问题 q 编码的 attention-pooling 向量，这种方式可以为答案抽取层提供更多的先验知识。

$$s_j = v^T \tanh(W_u^Q u_j^Q + b)$$

$$a_i = \exp(s_j) / \sum_{j=1}^{m} \exp(s_j)$$

$$r^Q = \sum_{i=1}^{n} a_i u_i^Q$$

其中 r^Q 是问题 q 编码的 attention-pooling 向量，u_j^Q 是问题 q 的编码向量。

R-net 占用显存极高，受限于计算资源，我们并没有改变其结构，代码是基于原始 paper 的复现。

3.3.3 Bi-daf

前两种模型所采用的注意力机制均是单向的，Bi-daf 模型[7]引入双向注意力机制，尝试从“原文→问题”与“问题→原文”两个方向建立 attention，以提高原文信息 p 的表达能力。该模型是五种模型中效果相对较差的一种，但由于所采用的结构与其他模型差异较大，所以我们在最后的集成方案中，也使用了该模型。Bi-daf 模型架构如图 8 所示

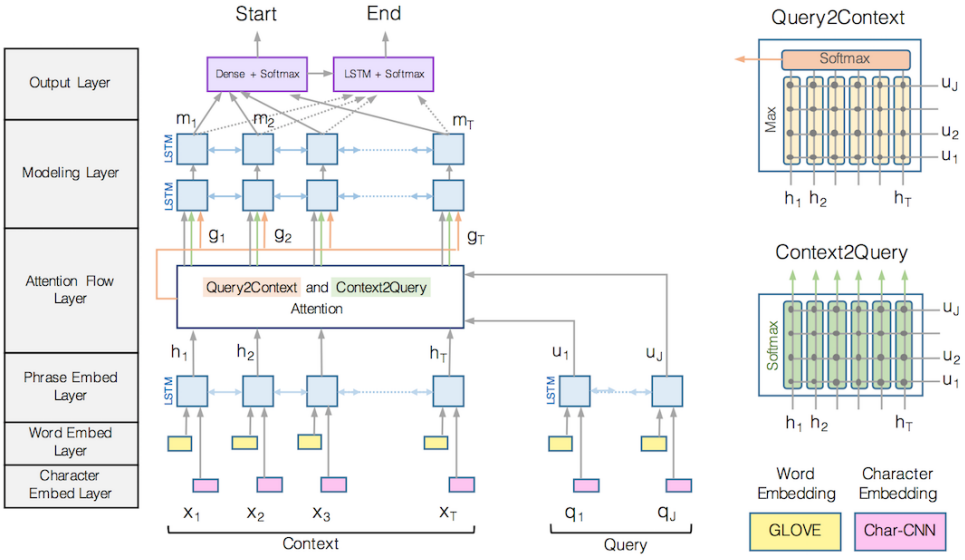

图 8 Bi-daf 模型结构

（1） 词嵌入、编码

首先通过 3.1 介绍的表示层，对原文 p 与问题 q 进行表示。然后使用双向 LSTM 模型分别对原文 p 与问题 q 进行编码，得到相应的编码向量

$$H^p = BiLSTM(p)$$

$$H^q = BiLSTM(q)$$

（2）匹配

Bi-daf 模型是一种相对早期的抽取式阅读理解模型，采用双向注意力机制，将问题信息融入原文。该模型首先基于某种相似性计算方式，构建一个相似性矩阵，然后基于该相似性矩阵，从两个方向“原文→问题”和“问题→原文”，分别得到相关的原文表达向量。针对这些向量，采用拼接、点乘的方式，生成新的带有问题信息的原文表达，最后通过一个双向 LSTM 对该原文表达做进一步的拟合。该模型认为这种注意力计算方式充分考虑了原文和问题之间的交互情况。相似性矩阵构建方式如下

$$S_{tj} = \alpha(H_{:t}, U_{:j}) \in R$$

$$\alpha(h,u) = w_{(s)}^{T}[h;u;h \cdot u]$$

其中 h，u 分别表示原文和问题的某个词的编码向量，$\alpha(h,u)$ 表示相似度的计算方式，S 表示相似性矩阵。

$$a_t = \text{soft}\max(S_{t:}) \in R^J$$

$$\bar{u} = \sum_j a_{tj} U_{:j}$$

$$b = \text{soft}\max(\max_{col}(S)) \in R^T$$

$$\bar{h} = \sum_t b_t H_{:t} \in R^{2d}$$

$$G_{:t} = \beta(H_{:t}, \bar{U}_{:t}, \bar{\mathrm{H}}_{:t})$$

a_t 是 content-to-query（C2Q）的注意力分布，$\bar{u}$ 是基于 a_t 关于问题 q 的注意力表达

b 是 query-to-content（Q2C）的注意力分布，$\bar{h}$ 是基于 b 关于原文 p 的注意力表达

Bi-daf 采用点乘、拼接的方式对上述所生成的注意力表达进行处理，得到带有问题信息的原文表示向量 G，计算方式为 $\beta(h,\bar{u},\bar{h}) = [h;\bar{u};h \cdot \bar{u};h \cdot \bar{h}]$。最后通过一个两层双向 LSTM，对原文表示向量 β 做进一步拟合，得到最终的表示向量 M

（3） 答案抽取

相比于前两种模型，Bi-daf 的答案抽取方式较为简单，拼接 G 和 M，使用全连接层获得答案起始位置索引概率。然后通过一个双向 LSTM 对 M 做进一步转换得到 M_2，拼接 G 和 M_2，使用一个新的全连接层获得答案终止位置索引概率，计算方式如下。

$$p^{start} = \text{soft}\max(w_{p1}^{T}[G;M])$$

$$p^{end} = \text{soft}\max(w_{p2}^{T}[G;M_2])$$

由于 Bi-daf 模型效果相对一般，且占用显存相对较多，我们并未对其结构进行改变，代码是基于其原始 paper 的复现。

3.3.4 M-reader

M-reader[8]是国防科技大学胡明昊博士提出的一种新型抽取式阅读理解模型，该模型在attention 的基础上，仿照人思考的方式，引入了迭代机制。以人做阅读理解为例，对于某些需要逻辑推理的问题，人通常会带着问题反复阅读原文数次。另外，该模型除了引入迭代机制以外，还借鉴强化学习的思路，对 loss 优化函数做了改进，详情见第 3 节训练过程部分。M-reader 模型架构如图 9 所示。

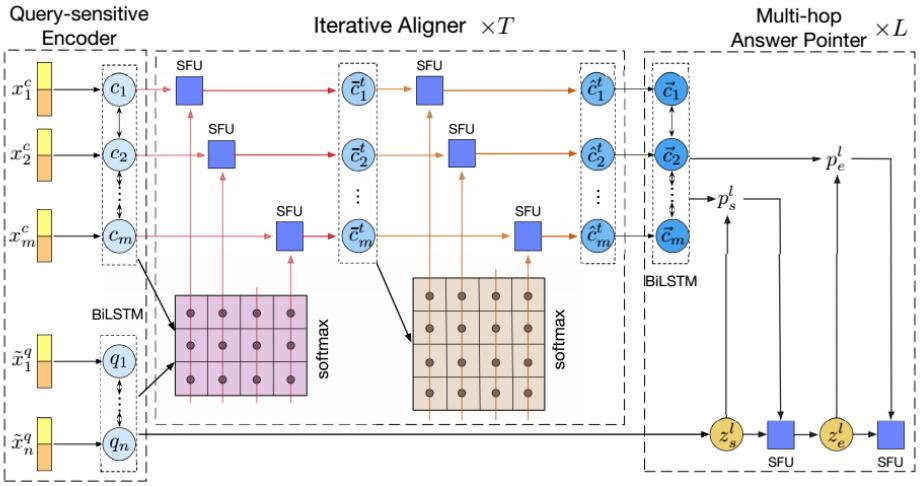

图 9　M-reader 模型结构

（1） 词嵌入、编码

首先通过 3.1 介绍的表示层，对原文 p 与问题 q 进行表示。然后使用双向 LSTM 模型分别对原文 p 与问题 q 进行编码，得到相应的编码向量

$$\mathrm{H}^{p} = BiLSTM(p)$$

$$\mathrm{H}^{q} = BiLSTM(q)$$

（2）匹配

M-reader 引入迭代机制，以提高模型的推理能力。在匹配层部分，每一个迭代块主要由三部分组成：迭代对齐(Interactive Aligning)、自对齐(Self Aligning)以及语义平滑（Aggregating），以下对这三部分分别展开介绍。

迭代对齐，该部分主要负责建立原文和问题之间的匹配关系，匹配关系由内积度量，通过匹配关系矩阵，可以得到一种新的问题表示向量，该向量在下一步计算原文表示向量时使用，具体计算过程如下

$$B_{ij}^{t} = q_{i}^{T} \cdot \overline{c_{j}^{t-1}}$$

$$\overline{c_{j}^{0}} = c_{j}$$

$$b_{j}^{t} = \mathrm{soft\,max}(B_{:j}^{t})$$

$$\overline{q_{j}^{t}} = \mathrm{Q} \cdot b_{j}^{t}$$

B 矩阵表示匹配关系矩阵，B_{ij}^{t} 代表问题中的第 i 个词与原文中的第 j 个词的相似度。b_{j}^{t}

表示问题 q 关于原文 p 的注意力分布，$\overline{q_j^t}$ 表示基于原文注意力的问题表示向量。基于原文表示 $\overline{c_j^{t-1}}$ 与新的问题表示向量 $\overline{q_j^t}$，通过语义融合单元，可以得到在迭代对齐部分带有问题信息的原文表示 $\overline{\overline{c_j^t}}$，计算过程如下

$$\overline{\overline{c_j^t}} = SFU(\overline{c_j^{t-1}}, \overline{q_j^t}, \overline{c_j^{t-1}} \cdot \overline{q_j^t}, \overline{c_j^{t-1}} - \overline{q_j^t})$$

其中 SFU 为语义融合单元，通过语义融合单元将问题信息融入原文表示中，可以得到带有问题信息的原文表示，SFU 计算方式如下

$$\bar{r} = \tanh(W_r([r;\ f_1;...;\ f_k]) + b_r)$$

$$g = \sigma(W_g([r;\ f_1;...;\ f_k]) + b_g)$$

$$o = g \cdot \bar{r} + (1-g) \cdot r$$

其中 r 为主要信息，$[f_1;...;\ f_k]$ 为融入信息。

自对齐，受限于循环神经网络在建模长序列依赖关系的不足，每个词只与周围的词有关系，而对整个长序列缺乏足够的认识。因此 M-reader 模型提出使用自对齐机制，对原文表示中的重要信息进行强化。自对齐与迭代对齐方式较为相似，差异主要体现在匹配关系矩阵的计算上，迭代对齐是计算原文 p 与问题 q 的匹配关系，而自对齐是计算原文表示与原文表示之间的匹配关系，计算方式如下：

$$\overline{\mathrm{B}}_{ij}^t = 1(i \neq j) \overline{\overline{c_i^t}}^T \cdot \overline{\overline{c_j^t}}$$

语义平滑，与前几种模型相类似，在得到带有问题信息的原文表示向量后，使用循环神经网络对原文表示做进一步拟合，得到拟合后的原文表示 $\overleftrightarrow{c}$

（3） 答案抽取

在答案抽取部分，M-reader 提出了一种带有记忆功能的答案指针网络，该网络维护了一个记忆向量 z_s^l，用于记录在预测答案片段时必要的阅读信息，记忆向量 z_s^l 的初始状态使用问题编码的终态。假定答案指针网络包含 L 次迭代，在 $l-th$ 迭代时，答案起始位置分布概率为：

$$s_i^l = FN(\overleftrightarrow{c_i}^T, z_s^l, \overleftrightarrow{c_i}^T \cdot z_s^l)$$

$$p_s^l(i) = \mathrm{soft\,max}(w_s^l \cdot s_i^l)$$

其中 $p_s^l(i)$ 表示答案起始位置的分布概率，FN 表示前馈神经网络，用于对输入信息进行非线性拟合。在得到答案起始位置的分布概率 $p_s^l(i)$ 后，可以得到相关的证据向量 $u_s^l = \overleftrightarrow{c}^T \cdot p_s^l$，利用该证据向量对记忆向量进行更新，更新方式为 $z_e^l = SFU(z_s^l, u_s^l)$，其中

SFU 如上所述，为语义融合单元。与答案起始位置分布概率的计算方式相似，答案终止位置分布概率的计算如下

$$e_j^l = FN(\overleftrightarrow{c_i}^T, z_e^l, \overleftrightarrow{c_i}^T \cdot z_e^l)$$

$$p_e^l(j) = \text{soft}\max(w_e^l e_j^l)$$

该模型相对复杂，所需训练参数较多，但显存开销极少。我们并未对模型结构进行更改，但相对于原始 paper，我们尝试将动态词向量 ELMo 引入其中，并取得了一定的效果，但是该方案测试时间所需较多，在决赛环节并不适用。

3.3.5 M-reader-plus

M-reader-plus[9]是国防科技大学胡明昊博士在 M-reader 的基础上，改进的一种抽取式阅读理解模型，与 M-reader 相比，该模型引入了重复注意力机制，对注意力取值范围进行控制，避免其过大或者过小。除此之外，该模型还改进了相似度的计算方式，优化了整体的迭代结构，模型整体架构如图 10 所示。

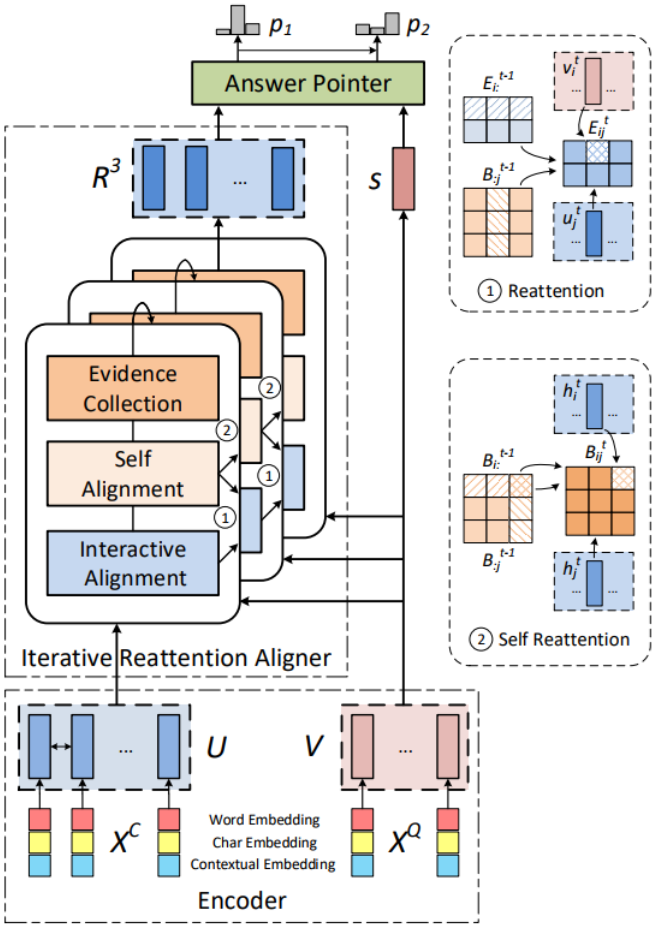

图 10　M-reader-plus 模型结构

（1） 词嵌入、编码

首先通过 3.1 介绍的表示层，对原文 p 与问题 q 进行表示。然后使用双向 LSTM 模型分别对原文 p 与问题 q 进行编码，得到相应的编码向量

$$\mathrm{H}^p = BiLSTM(p)$$

$$\mathrm{H}^q = BiLSTM(q)$$

（2）匹配

与 M-reader 模型相类似，M-reader-plus 模型在匹配部分，也分为三个主要部分：迭代对齐(Interactive Aligning)、自对齐(Self Aligning)以及语义平滑（Aggregating），在 M-reader-plus 的原始论文中，语义平滑也被称为证据收集（Evidence collection），每一个迭代组件如图 11 所示。

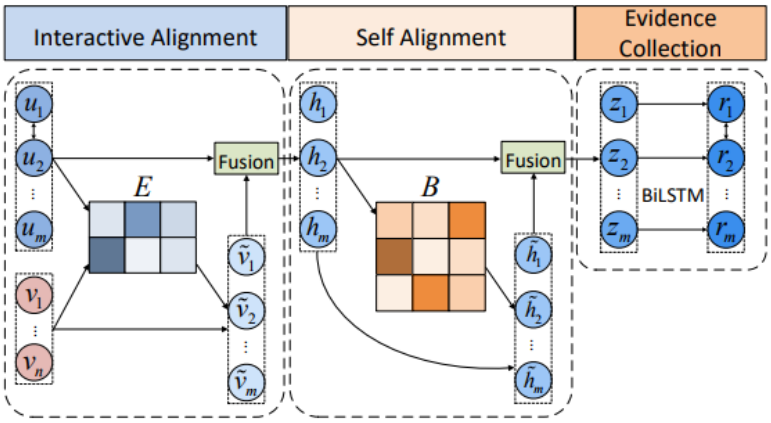

图 11　M-reader-plus 单轮迭代结构

迭代对齐，该部分主要负责建立原文和问题之间的匹配关系，计算方式与 M-reader 的相应部分相类似，但其额外引入了重复注意力机制，并改良了相似度的计算方式，具体细节如下。

$$\overline{E}_{ij}^{t} = \mathrm{soft\,max}(E_{i:}^{t-1}) \cdot \mathrm{soft\,max}(B_{:j}^{t-1})$$

$$E_{ij}^{t} = f(v_i^t, u_j^t) + \gamma \overline{E}_{ij}^{t}$$

其中，E 与 B 分别表示迭代对齐的匹配矩阵和自对齐的匹配矩阵，$f(v_i^t, u_j^t)$ 表示原文 v_i^t 与问题 u_j^t 的相似度计算，具体为 $f(u,v) = relu(W_u u)^T relu(W_v v)$。通过匹配矩阵 E 与问题表示 V，可以得到问题注意力向量 $\overline{v}_j = V \cdot \mathrm{soft\,max}(E_{:j})$。

然后模型使用语义融合单元，将问题注意力向量所携带的语义信息融合进原文表示中，得到带有问题信息的原文表示 $h_j = fusion(u_j, \overline{v}_j)$，语义融合单元 $fusion$ 计算方式如下：

$$\overline{x} = relu(W_r[x; y; x \cdot y; x - y])$$

$$g = \sigma(W_g[x; y; x \cdot y; x - y])$$

$$o = g \cdot \overline{x} + (1 - g) \cdot x$$

自对齐，与迭代对齐相似，在自注意力匹配矩阵的计算上，M-reader-plus 也引入了重复注意力机制，具体细节如下。

$$\overline{B}_{ij}^{t} = \text{soft}\max(B_{i:}^{t-1}) \cdot \text{soft}\max(B_{:j}^{t-1})$$

$$B_{ij}^{t} = 1_{(i \neq j)}(f(h_i^t, h_j^t) + \gamma \overline{B}_{ij}^{t})$$

在得到自注意匹配矩阵后，通过自注意力矩阵 B 与上一层的原文表示 $H = [h_1, ..., h_m]$，可以得到基于原文表示的自注意向量 $\overline{h}_j = H \cdot \text{soft}\max(B_{:j})$，使用语义融合单元 $fusion$ 将自注意力向量 $\overline{H} = [\overline{h}_1, ..., \overline{h}_m]$ 融合进原文表示 H 中，$z_j = fusion(h_j, \overline{h}_j)$，得到最后带有问题信息的原文表示 $Z = [z_1, ..., z_m]$

语义平滑，通过双向 LSTM 对原文表示 Z 做进一步拟合，得到原文表示 R

（3） 答案抽取

M-net-plus 将迭代次数设定为 3，每层之间使用匹配矩阵 E、B 作为连接，值得注意的是在最后对答案索引首尾位置进行预测时，使用三次迭代输出的平滑结果，具体计算方式如下：

$$R^1, Z^1, E^1, B^1 = align^1(\mathrm{U}, \mathrm{V})$$

$$R^2, Z^2, E^2, B^2 = align^2(R^1, V, E^1, B^1)$$

$$R^3, Z^3, E^3, B^3 = align^3(R^2, V, E^2, B^2, \mathrm{Z}^1, Z^2)$$

$$R^3 = BiLSTM([Z^1, Z^2, Z^3])$$

其中，U 和 V 分别表示原文和问题的编码向量，各个迭代单元使用上一层的匹配矩阵 E、B 作为连接。在前两次迭代中，R 是由相对应的 Z 经过 BiLSTM 平滑得到，而在最后一次迭代中，R^3 是由过去三次的中间输出结果 Z_1、Z_2、Z_3 经平滑得到。最后模型使用全连接网络对答案首尾索引位置进行预测。

$$p_1(i) \propto \exp(w_1^T \tanh(W_1[r_i^3; s; r_i^3 \cdot s; r_i^3 - s])$$

$$\overline{s} = fusion(s, R^3 \cdot p_1)$$

$$p_2(j \mid i) \propto \exp(w_2^T \tanh(W_2[r_j^3; \overline{s}; r_j^3 \cdot \overline{s}; r_j^3 - \overline{s}])$$

其中 s 是证据向量，由问题编码的 attention- pooling 进行初始化，$s = \sum_{i=1}^{n} \alpha_i v_i$，其中 α_i 是 attention-pooling 的注意力分布。

该模型相对复杂，显存消耗一般，训练参数巨多，我们并未改变该模型的结构，代码是基于原始 paper 的复现。

3 训练过程

我们针对不同的模型，采用相同的训练过程，主要分为两个阶段，首先利用标准交叉熵函数训练模型，直至收敛，对于不同模型，迭代至收敛所需要的 epoch 数量相差较大，R-net 需要 15 epoch+，而对于 Match-lstm-plus 模型，收敛仅需要 6 个 epoch。 我们统一设定迭代轮数 epoch=20，采用 early stopping 策略，取在验证集上 loss 最低的模型为输出模型。

$$J_{MLE}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(\log y^1 + \log y^2)$$

模型训练的第二阶段是采用强化学习策略，直接以 Rouge-L 为优化目标，对训练好的模型进行微调，这种方式也被称为最小风险训练。我们统一设定迭代轮数 epoch=2，取在验证集上 loss 最低的模型为输出模型。另外值得一提的是，为了避免模型过分追求 Rouge-L，导致模型泛化性能的降低，我们在最小风险 loss 中额外加入了交叉熵 loss。

$$J_{MLE}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(\log y^1 + \log y^2)$$

$$J(\theta) = \lambda J_{MLE}(\theta) + (1-\lambda) J_{MRT}(\theta)$$

我们将调节参数 λ 设定为 0.01，不同单模型在验证集上的表现，见表 2

表 2　单模型效果

Table 2　the performance of single model

Model	Rouge-L		Bleu	
	MLE	MRT	MLE	MRT
Match-lstm-plus	**91.32**	91.23	76.16	**81.19**
R-net	91.29	**91.35**	**79.53**	78.22
Bi-daf	**91.08**	91.07	**77.91**	75.21
M-reader	**91.1**	91.06	**79.67**	77.29
M-reader-plus	91.13	**91.32**	78.83	**81.11**

从表 2 中可以发现，最小风险并不一直都是有效的，有些模型 Rouge-L 精度反而下降，因此，在最后的集成方案里，我们只取 MLE 和 MRT 表现较好的模型。不同模型训练曲线大致相同，见图 12

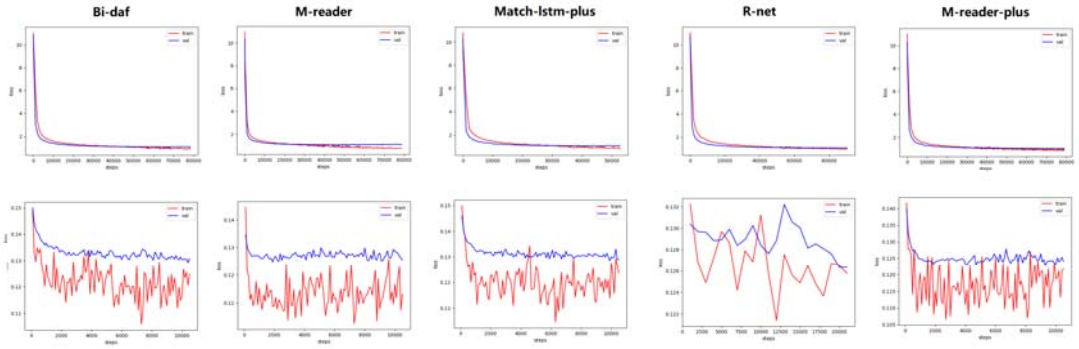

图 12 模型训练曲线

由于 R-net 训练时间非常长，所以我们调整了它验证的频率，导致曲线看起来相对稀疏。

4 集成策略

在初赛阶段，我们针对五种模型：Match-lstm-plus、R-net、Bi-daf、M-reader 以及 M-reader-plus，调整 dropout 参数（0.2,0.25,0.3），两种 Loss 训练方式（交叉熵、最小风险），共计训练了 30 个模型，取在两种 Loss 上表现较好的模型作为集成备选模型，最后我们的初赛集成方案是由 15 个模型，采用加权平均的方式构成，初赛结果如表 3 所示。

表 3　初赛测试结果

Table 3　test result in preliminary

	Rouge-L	△	Bleu	△
单模型	88.13		75.02	
集成模型	88.91	+0.78	75.81	+0.79

在集成策略上，我们尝试了三种方案，分别是**加权平均**，权重由模型在验证集上的表现决定；**概率最大值**，多个模型中 $p_{start} \cdot p_{end}$ 最大的作为输出结果；**加权投票**，权重由模型在验证集上的表现决定，票数最多的索引位置作为输出结果。三种集成策略在(训练集+初赛测试集)上的对比情况见表 4

表 4　集成方案效果对比

Table 4　the performance of ensemble method

	平均-5	**平均**-10	**平均**-15	**最大**-5	**最大**-10	**最大**-15	**投票**-5	**投票**-10	**投票**-15
Rouge-L	0.9099	0.9108	**0.9113**	0.9074	0.9074	0.9079	0.9087	0.9102	0.9100
Bleu	0.8086	0.8087	**0.8143**	**0.8143**	0.8132	**0.8143**	0.7961	0.7995	0.8022

5 软硬件需求

我们系统的软硬件的需求如下

（1）　系统：ubuntu 16

（2）　IDE: pycharm

（3）　编程语言：python3（通过 Anaconda3 安装）

（4）　工具库：jieba(0.39)、gensim(3.4.0)、pytorch(0.4.1)、rouge(0.3.0)、matplotlib(2.2.2)、nltk

（5）　硬件配置：GPU1080ti、内存 16G+、硬盘 100G+

6 软件运行步骤

系统主要由以下几个部分组成

（1）　train.py，训练接口

（2）　test.py，测试接口

（3）　preprocess_data.py，数据预处理接口

（4）　loader.py，数据加载接口

（5） data_pre 文件夹，内含一些数据预处理方法
（6） config 文件夹，内含模型配置文件
（7） modules 文件夹，主要的模型代码
（8） my_metrics 文件夹，内含测评方法

训练过程： 取消 train.py 中的 config 注释，运行即可，config 注释如图 13 所示

```
# config = config_match_lstm.config
# config = config_match_lstm_plus.config
# config = config_r_net.config
# config = config_bi_daf.config
config = config_bi_daf_plus.config
# config = config_qa_net.config
# config = config_m_reader.config
# config = config_m_reader_plus.config
```

图 13 train.py 中的 config 注释

测试过程：设定 test.py 中的 is_ensemble 参数，直接运行即可，当 is_emsemble=True 时，进行集成测试，反之进行单模型的测试，is_emsemble 参数在主方法下，如图 14 所示

```
if __name__ == '__main__':
    is_ensemble = False
```

图 14 test.py 中的 is_ensemble 参数

6 总结

首先，非常感谢主办方所提供的这次难得机会，以及为了筹备挑战赛所做出的辛苦付出。其次，感谢实验室、导师在参赛期间，所给与的大力支持和指导。通过这次挑战赛，我们团队所有人均得到了高强度的训练，无论是理论知识，还是实际编码能力，均得到了提高。以下对我们团队的技术方案做下简单总结。

我们将本次挑战赛定位为“抽取式阅读理解”任务。在数据预处理上，主要包括数据转换、除燥、数据截断、答案构造等四部分，并详细分析了分词器所带来的影响。模型结构主要包括四个部分：表示、编码、匹配以及答案抽取，在匹配及答案抽取部分，我们改进了 Match-lstm 模型，使用了 R-net、Bi-daf、M-reader 及 M-reader-plus 模型。在模型训练部分，我们分阶段使用两种 Loss 函数对模型进行训练，分别是交叉熵函数与最小风险函数。在模型集成上，我们采用加权平均的方式，对多模型进行集成。

最后，再一次感谢主办方所提供的宝贵机会，并真诚的祝愿主办方在未来举办的比赛中，越来越好！

参考文献：

[1] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[2] Pennington J, Socher R, Manning C. Glove: Global vectors for word representation[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.

[3] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[J]. arXiv preprint arXiv:1802.05365, 2018.

[4] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers

for Language Understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[5] Wang S, Jiang J. Machine comprehension using match-lstm and answer pointer[J]. arXiv preprint arXiv:1608.07905, 2016.

[6] Wang W, Yang N, Wei F, et al. Gated self-matching networks for reading comprehension and question answering[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017, 1: 189-198.

[7] Seo M, Kembhavi A, Farhadi A, et al. Bidirectional attention flow for machine comprehension[J]. arXiv preprint arXiv:1611.01603, 2016.

[8] Hu M, Peng Y, Huang Z, et al. Reinforced Mnemonic Reader for Machine Reading Comprehension[J]. arXiv preprint arXiv:1705.02798, 2017.

[9] Hu M, Peng Y, Huang Z, et al. Reinforced Mnemonic Reader for Machine Reading Comprehension[J]. arXiv preprint arXiv:1705.02798, 2018.