东北大学

英文论文翻译

使用Match-LSTM与答案指示进行机器阅读理解

姓名：李佳政
班级：计算机1402班
学号：20143616

June 12, 2018

东北大学

英文论文原文

Machine Comprehension using Match-LSTM and Answer Pointer

姓名：李佳政
班级：计算机1402班
学号：20143616

June 12, 2018

使用Match-LSTM与答案指示进行机器阅读理解

Shuohang Wang,[1] Jing Jiang,[2]
*E-mail: shwang.2014@phdis.smu.edu.sg[1], jingjiang@smu.edu.sg[2].

摘要

文本机器阅读理解在自然语言处理中是一个重要的问题。SQuAD是最近开放的一个数据集，提供了大规模的由人类标注的真实问题和答案。它提供了一个在线的测试平台来检验机器阅读理解算法的性能，和以往不同，SQuAD中的答案不仅仅来自一个小规模候选答案，而是从文章中选取具有不定长度的答案。本文提出的架构基于Match-LSTM（之前提到的文本蕴含模型）及Pointer Net模型（由Vinyals [1]等于2015年提出的端到端的模型），为了从输入的文本序列中限制输出的词汇。在本次任务中，我们提出了两种使用Pointer Net的方式。我们的实验表明效果好与Rajurkar [2]于2016年使用逻辑回归的效果与手动提取特征。

1 引言

文本的机器理解是自然语言处理的最终目标之一。同时机器能够理解文本的能力以多种不同的方式来评估，近几年已经创建了几个基准数据集，将机器在问答的表现作为评估机器阅读理解的一种方式。在这种设置下，机器首先阅读一个新闻文章或者是一个故事，然后机器回答一个和文本相关的问题。

在大部分评测数据数据集中，一个问题的答案有多种选择得到，更多候选答案的问题更具挑战性。最近，斯坦福问答数据集，包含更具挑战性的问题，其正确答案可以是任何的单词序列。此外，与其他问答的数据集不同，完形填空类型的问题与答案是自动生成的，SQuAD问答数据集是通过众包创建的，这使得数据集更加真实。鉴于这些数据集的优势，在本文中，我们专注于这一新的数据集进行研究。一个样例文本与三个相关问题如表1所示。

Table 1: 维基百科的一段文字和三个相关问题及其答案，取自SQuAD数据集。段落中粗体的标记是我们预测的答案。问题旁边的段落是人标记的答案。

In 1870, Tesla moved to Karlovac, **to attend school at the Higher Real Gymnasium**, where he was profoundly influenced by a math teacher **Martin Sekulić**. The classes were held in **German**, as it wasa school within the Austro-Hungarian Military Frontier. Tesla was able to perform integral calculus in his head, which prompted his teachers to believe that he was cheating. He finished a four-year term inthree years, graduating in 1873.	
1. In what language were the classes given?	German
2. Who was Tesla's main influence in Karlovac?	Martin Sekulić
3. Why did Tesla go to Karlovac?	attend school at the Higher Real Gymnasium

这种问答任务的传统解决方案依赖于涉及语言分析和特征工程的多种NLP技术，包括句法解析、命名实体。机器对文本的理解。近年来，随着神经网络模型在NLP中的应用，对各种NLP任务的端到端神经结构进行了大量的研究，包括机器理解方面的几项工作。然而，先前机器理解数据集的特性，现有的端到端神经结构依赖于候选答案或假定答案是单个词语，这使得这些方法不适合SQuAD数据集。在本文中，我们提出了一种新的端到端神经网络架构，以解决在SQuAD数据集中定义的机器阅读理解问题。

具体地说，观察到在SQuAD数据集中，许多问题答案是从原始文本中的句子得到的释义，我们采用先前研究的先前开发的文本蕴含Match-LSTM模型，它允许我们从输入序列中预测令牌，而不是从一个更大的固定词汇表中选择得到答案，从而允许我们从原文中生成由多个单词组成的答案。我们提出了两种方法来应用Ptr-Net模型：序列模型和边界模型。我们还用搜索机制进一步扩展边界模型。在数据集上的实验表明，我们的两个模型都拥有最佳的表现性能。此外，使用几个模型的集成让我们在竞赛中取得非常有竞争力的表现。

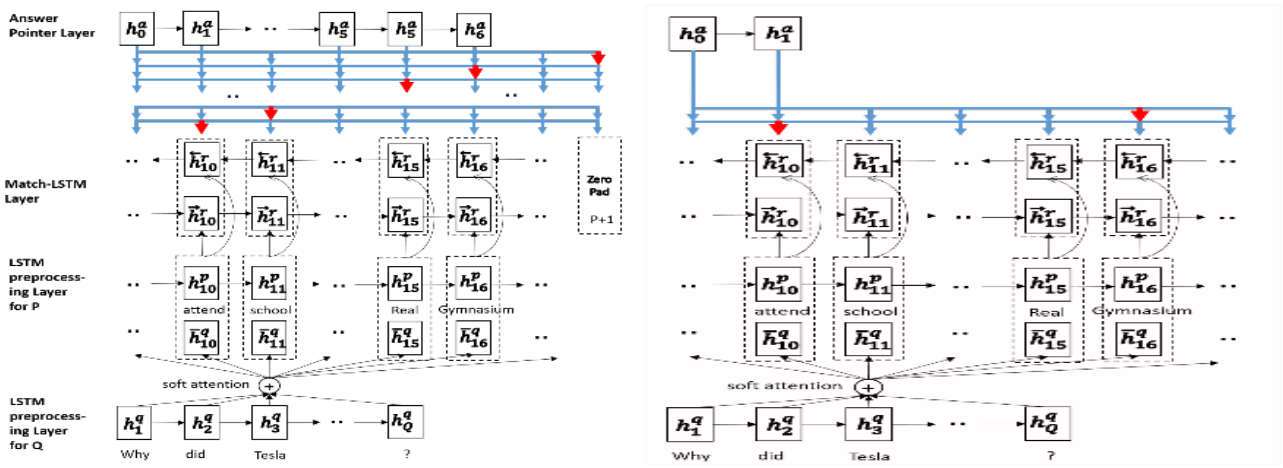
本文的贡献如下：（1）提出了两种新的机器理解端到端神经网络模型，将Match-LSTM和Ptr-Net相结合，以处理SQuAD数据集的特殊属性。(2）在未观察到的测试数据集上，我们获得了67.9%的精确匹配分数和77%的F1分数，这比特征工程解决方案好得多。我们的表现也接近SQuAD的最佳模型，即71.6%的准确匹配，80.4%的F1来自Salesforce的研究。（3）对模型的进一步分析，为进一步改进该方法提供了有益的启示。除此之外，我们的代码可以在线查看[1]。

2 方法

在这一节中，我们首先简单地回顾了Match-LSTM和Ptr-Net。这两个工作是我们研究方法的基础。然后，我们提出我们的端到端的机器理解神经架构。

2.1 Match-LSTM

在最近的自然语言推理的工作中，我们提出了一个Match-LSTM模型来预测文本蕴含。在文本蕴涵中，给定两个句子：其中一个是前提，另一个是假设，预测前提是否蕴含了假设，match-LSTM模型通过假设的词序来判断。在每个假设的位置，注意机制被用来获得加权向量表示的前提。然后将该加权前提与向量表示相结合。当前假设的词是送入LSTM，我们称之为Match-LSTM。Match-LSTM本质上顺序地聚集注意加权前提与每个词的匹配。对该假设进行表征，并利用聚合匹配结果进行最终预测。



(a) Sequence Model　　(b) Boundary Model

Figure 1: 两个模型的概述。两个模型都包括一个LSTM预处理层，Match-LSTM层和回答指针层。对于每个匹配的LSTM在特定的方向上，h_i^q定义为$H^q\alpha_i^T$，用相应方向的α计算，具体描述见公式（2）与（5）。

2.2 Pointer Net

Vinyals提出Pointer Network(Ptr-Net)模型用来解决输出的序列来自输入的特殊情况问题。与从固定的单词表中选取单词不同，Ptr-Net使用注意力机制作为一个输入指针来从输入序列中选择选择一个位置作为输出符号。指针机制在语言处理上已经取得了显著效果。我们采用Ptr-Net为了从输入单词序列中构造答案。

[1]https://github.com/shuohangwang/SeqMatchSeq

2.3 Our Method

形式上，我们试图解决的问题可以表述如下。我们得到了一段文本，我们称之为段落，以及与文章相关的问题。文章由矩阵$\mathrm{P} \in \mathbb{R}^{d \times P}$，其中$P$是文章的长度，$d$是词向量的维度。同样问题也由矩阵$\mathrm{Q} \in \mathbb{R}^{d \times Q}$表示。我们的目标是从文章中确定一个子序列来回答问题。

正如前面指出的，由于输出的词来自输入，所以我们采用了Ptr-Net模型来解决这个问题，使用Ptr-Net的一个方法是把答案作为文章中的一个单词序列，但忽略这些词在文章中是连续的事实，因为Ptr-Net不做连续性假设。特别地，我们把答案作为一个整数序列$a = (a_1, a_2, \ldots)$,其中每个a_i是一个介于1与P之间的整数，标识在文章中的位置。

或者，如果我们想要确保连续的词作为答案，我们可以使用Ptr-Net来预测开始和答案的结尾。在这种情况下，Ptr-Net只需要从输入通道中选择两个词，中间的部分都被当作答案。特别地，我们可以把答案的预测表示为两个整数，$a = (a_s, a_e)$,其中a_s和a_e都是介于1与P之间的整数。

我们把上述的第一种设置称为连续模型，第二种设置称作边界模型。对于每个模型来说，我们假定训练样例的集合由三元组$\{(\mathbf{P}_n, \mathbf{Q}_n, \mathbf{a}_n)\}_{n=1}^{N}$给定。

对两个神经网络模型的总结如图1所示。两个模型都由三层组成：（1）一个LSTM预处理层，使用LSTMs来预处理文章和问题。（2）一个match-LSTM层，将文章与问题进行匹配。（3）答案的指针层使用Ptr-Net来从文章中选择一组词汇作为答案。两个模型只有第三层不同。

LSTM预处理层
LSTM预处理层的目的是把问题的每个词及文章的上下文信息并入文章与问题的向量表示方式中。我们使用一个标准的单向LSTM[2]来分别处理问题与文章,如下所示

$$\mathrm{H^p} = \underset{LSTM}{\rightarrow}(\mathrm{P}), \mathrm{H^q} = \underset{LSTM}{\rightarrow}(\mathrm{Q}). \tag{1}$$

结果矩阵$\mathrm{H^p} \in \mathbb{R}^{l \times P}$和$\mathrm{H^q} \in \mathbb{R}^{l \times Q}$是文章和问题的隐藏表示，其中$l$是隐藏层向量的维度。对于其他单词，第$i$列向量$h_i^p$（或$h_i^q$）在$\mathrm{H^p}$（或$\mathrm{H^q}$）代表文章中第$i$个词，并且从左到右附带上下文。

Match-LSTM层 我们使用match-LSTM模型的目的是在机器阅读理解问题中，对待问题应当为前提，文章为一个假设，从中抽取文本蕴含。match-LSTM模型从文章序列地提出单词，在文章的第i个词，首先使用标准的词与词之间的注意力机制来获得注意力权重向量$\underset{\alpha_i}{\rightarrow} \in \mathbb{R}^Q$，公式如下：

$$\begin{aligned} \underset{\mathbf{G}}{\rightarrow} &= \tanh(\mathrm{W^q H^q} + \mathrm{W^r} \underset{\mathrm{h_{i-1}^r}}{\rightarrow} + (\mathrm{W^p h_i^p} + \mathrm{b^p}) \otimes \mathrm{e}_Q), \\ \underset{\alpha_i}{\rightarrow} &= \mathrm{softmax}(\mathrm{w}^{\mathrm{T}} \underset{\mathrm{G}_i}{\rightarrow} + \mathrm{b} \otimes \mathrm{e}_Q), \end{aligned} \tag{2}$$

其中$\mathrm{W^q}, \mathrm{W^p}, \mathrm{W^b r} \in \mathbb{R}^{l \times l}, \mathrm{b^p}, \mathrm{w} \in \mathbb{R}^l$和$b \in \mathbb{R}^l$是学习得到的参数，$\underset{\mathrm{h}_{i_1}^{\mathrm{r}}}{\rightarrow} \in \mathbb{R}^l$是在位置$i-1$的单向match-LSTM（下文解释）的隐藏向量，它的输出乘积（$\cdot \otimes \mathrm{e}_Q$）得到一个矩阵或列向量，通过重复向量或从左向右$Q$倍标量。

重要的是，结果的注意力权重$\underset{\alpha_{i,j}}{\rightarrow}$表明了文章中第$i$个与问题中第$j$个词之间的匹配程度。下一步，我们使用注意力权重向量$\underset{\alpha_i}{\rightarrow}$来获得问题的权重，把它和当前文章的词汇组合后得到向量：

$$\underset{\mathrm{z_i}}{\rightarrow} = \begin{bmatrix} \mathbf{h}_i^{\mathrm{p}} \\ \mathbf{H}^{\mathrm{q}} \underset{\alpha_i^{\mathrm{T}}}{\rightarrow} \end{bmatrix} \tag{3}$$

这个向量$\mathbf{z}_i$馈入标准单向LSTM，所以我们称之为match-LSTM:

$$\underset{\mathbf{h}_i^{\mathrm{r}}}{\rightarrow} = \underset{LSTM}{\rightarrow}(\underset{\mathrm{z}_i}{\rightarrow}, \underset{\mathbf{h}_{\mathrm{i-1}}^{\mathrm{r}}}{\rightarrow}) \tag{4}$$

我们后来在相反的方向构建一个相似的match-LSTM模型。它的目的主要是对每个文章中的词上下文进行编码。为了构建反向match-LSTM，我们首先定义了

[2]作为预处理的输出门控，几乎不影响输出最终效果，在实验中移除。

$$\begin{aligned}\underset{\mathbf{G}_i}{\rightarrow} &= \tanh(\mathbf{W}^{\mathrm{q}}\mathbf{H}^{\mathrm{q}} + (\mathbf{W}^{\mathrm{p}}\mathrm{h}_{\mathrm{i}}^{\mathrm{p}} + \mathbf{W}^{\mathrm{r}} \underset{\mathrm{h}_{\mathrm{i+1}}^{\mathrm{r}}}{\leftarrow} + \mathbf{b}^{\mathrm{p}}) \otimes \mathbf{e}_Q),\\ \underset{\alpha_i}{\rightarrow} &= softmax(\mathrm{w}^{\mathrm{T}} \underset{\mathbf{G}_{\mathrm{i}}}{\rightarrow} + \mathrm{b} \otimes \mathrm{e}_Q),\end{aligned} \tag{5}$$

注意上述参数与公式2相同。我们定义z与h都是在位置i的隐藏层位置生成的反向LSTM。我们级联所有的隐藏层，定义$\mathrm{H}^{\mathrm{r}} \in \mathbb{R}^{2l\times P}$为两个级联后的结果:

$$\mathrm{H}^{\mathrm{r}} = \begin{bmatrix} \underset{\mathrm{H}^{\mathrm{r}}}{\rightarrow} \\ \underset{\mathrm{H}^{\mathrm{r}}}{\leftarrow} \end{bmatrix}. \tag{6}$$

答案指示层

在顶层是答案指示层（Ans-Ptr）层，是由Ptr-Net修改得到的。这层使用序列$\mathbf{H}^{\mathrm{r}}$作为输入，我们使用了两个不同的模型：序列模型输出一串单词，但是每个单词不一定在原始文章中连续。边界模型只输出起始和终止位置，在这两次词之间的所有词都被认为是答案。我们现在分别解释这两个模型。

序列模型：在序列模型中，答案由在原文中的许多单词表示。Ans-Ptr层对这些整数在序列操作上建模。因为一个答案的长度是不固定的，所以为了在一个确定的点停止生成答案，我们把每个答案单词a_k视为一个1到$P+1$之间的整数，其中$P+1$是一个指示答案结束的特殊值。一旦a_k被设置为$P+1$，答案生成将停止。为了生成第k个答案词指示a_k，首先再次使用注意力机制来获得注意力权重向量，其中$\beta_{k,j}(1 \leqslant j \leqslant P+1)$是从文章中选择第$j$个词作为答案中第$k$个词的概率，并且$\beta_{k,(P+1)}$是答案停止在位置$k$的概率。

$$\mathrm{F}_k = \tanh(\mathrm{V}\widetilde{\mathrm{A}} + (\mathrm{W}^{\mathrm{a}}\mathrm{h}_{\mathrm{k-1}}^{\mathrm{a}} + \mathrm{b}^{\mathrm{a}}) \otimes \mathrm{e}_{P+1}) \tag{7}$$

$$\beta_{\mathrm{k}} = \mathrm{softmax}(\mathrm{v}^{\mathrm{T}} + c \otimes \mathrm{e}_{P+1}) \tag{8}$$

其中$\widetilde{\mathrm{H}}^{\mathrm{r}} \in \mathbb{R}^{2\mathrm{l}\times(\mathrm{P}+1)}$是$\mathrm{H}^{\mathrm{r}}$与零向量的级联，定义为$\widetilde{\mathrm{H}}^{\mathrm{r}} = [\mathrm{H}^{\mathrm{r}}; 0]$，$\mathrm{V} \in \mathbb{R}^{l\times 2\mathrm{l}}$，$\mathrm{W}^{\mathrm{a}} \in \mathbb{R}^{l\times 1}$，$\mathrm{v} \in \mathbb{R}^{\mathrm{l}}$与$c \in \mathbb{R}$都是需要学习得到的参数，$(\cdot \otimes \mathrm{e}_{P+1})$遵循之前的定义，并且$\mathrm{h}_{k-1}^{\mathrm{a}} \in \mathbb{R}^{l}$是在位置$k-1$的隐藏层向量如下定义:

$$\mathrm{h}_k^{\mathrm{a}} = \underset{\mathrm{LSTM}}{\rightarrow}(\widetilde{\mathrm{H}}^{\mathrm{r}}\beta_k^{\mathrm{T}}, \mathrm{h}_{k-1}^{\mathrm{a}}) \tag{9}$$

我们可以把建立概率模型来生成答案序列，

$$p(\mathrm{a}|\mathbf{H}^{\mathrm{r}}) = \prod_{\mathrm{k}} p(\mathrm{a_k}|\mathrm{a}_1, \mathrm{a}_2, \cdots, \mathrm{a}_{k-1}, \mathbf{H}^{\mathrm{r}}). \tag{10}$$

其中，

$$p(a_k = j|a_1, a_2, \cdots, a_{k-1}, \mathbf{H}^{\mathrm{r}}), \tag{11}$$

为了训练这个模型，我们基于训练样本最小化下列损失函数:

$$-\sum_{n=1}^{N} \log p(\mathbf{a}_n|\mathbf{P}_n, \mathbf{Q}_n). \tag{12}$$

边界模型：边界模型的工作方式与上述序列模型非常相似，而不是预测的指示位置$\mathrm{a}_1 \mathrm{fia}_2 \cdots$，我们只需要预测两个标记位置$\mathrm{a_s}$ 和$\mathrm{a_e}$。所以和序列模型的主要区别是，边界模型我们不需要添加额外的零向量，并且生成答案的简单预测模型如:

$$p(\mathrm{a}|\mathrm{H}^{\mathrm{r}}) = p(\mathrm{a_s}|\mathrm{H}^{\mathrm{r}})\mathrm{p}(\mathrm{a_e}|\mathrm{a_s}, \mathrm{H}^{\mathrm{r}}), \tag{13}$$

我们通过搜索机制进一步扩展了边界模型。特别地，在预测中，我们尝试限制跨度长度，全局**搜索**跨度通过计算$p(\mathrm{a}_s) \times p(\mathrm{a_e})$。除此之外，作为边界有一串固定的值，**双向答案指针**层可以简单地调节正确的跨度。

3 实验

在本节，我们展示了我们的试验结果，进行分析模型以便更好的理解。

3.1 数据

我们使用了斯坦福问答系统数据集（SQuAD）v1.1来进行我们的实验。在SQuAD中的文章是来自维基百科的536篇文本，包含了广泛的主题内容。每个文章是一个维基百科文章的段落，并且每篇文章都有大概五个和它相关的问题。总计23，215个文章与107,785个问题。数据被分为训练集（有87,599个问答对），一个开发集（有10,570个问答对）和一个隐藏的数据集。

3.2 实验设置

我们首先把所有的段落、问题和答案都分词，结果词汇表中包含11万7千个不同的词。我们使用**GloVe**来初始化模型。在GloVe中没有找到的词使用零向量。词向量在模型训练过程中不更新。

隐藏层的维数l设置为150或300。我们使用ADAMAX模型，系数$\beta 1 = 0.9$，$\beta 2 = 0.999$来优化模型。每次更新都是通过30个样本的小批量计算。我们不使用L2正则化。

通过两个指标来衡量性能：精确匹配与地标记答案相似的百分比。词级别的F1得分：比较预测的答案中的词与人工标记的答案相似度。注意，在开发集和测试集中每个问题都有大约三个人工标注回答。使用最佳匹配答案的F1分数用于计算平均F1评分。

3.3 结果

我们的模型效果和基线程序的效果如表2所示。我们能够看到，两个模型都比简单的逻辑回归模型效果好，它们依赖于设计特征。此外，我们的边界模型优于序列模型，实现了精确匹配得分为61.1%，F1得分为71.2%。特别是在精确匹配分数方面，边界模型比序列模型具有明显的优势。我们的模型的比Logistic回归模型效果好表明，我们的端到端神经网络模型没有太多特征工程在这个任务和这个数据集上是非常有效的。考虑到边界模型的有效性，我们进一步探讨了该模型。观察到大多数答案是跨度相对较小的大小，我们只是限制最大预测跨度不超过15个单词，并进行实验与跨度搜索，这导致1.5%的改进F1的发展数据，并优于DCR模型，他们也引入了一些语言特征，如POS和NE到他们的模型中。此外，我们尝试增加模型中的记忆维度l与添加双向预处理LSTM或添加双向答案指针。使用前两种方法对开发数据的效果提升是相当小的。通过添加BiAN-PTR与双向预处理LSTM，我们可以得到1.2%的改进F1。最终，我们通过简单的计算边界概率来集成模型，通过对5个边界模型产出的答案计算他们的概率，然后搜索最可能的且不超过15个单词的答案作为输出。这个集成策略达到了表中所示的最佳评分。

3.4 深入分析

为了更好的理解我们的模型的优缺点，我们对下面的结果进行了深入的分析。首先，我们怀疑更长的答案更难预测。为了验证这一假设，我们分析了在精确匹配和F1得分方面关于开发集上的答案长度的性能。例如，对于答案包含超过9个词的问题，F1边界模型的得分下降到大约55%，精确匹配分数下降到只有大约30%，单个单词答案的问题F1得分和接近72%和67%的精确匹配分数。这支持了我们的假设。接下来，我们分析我们的模型在不同组问题上的表现。我们使用一种粗略的方式，根据我们定义的一组疑问词将问题分成不同的组，包括“什么”、“怎样”、“谁”、“何时”、“哪个”、“哪里”、“为什么”，这些不同的疑问词大致指的是不同类型答案的问题。例如，“当”问题寻找时间表达式作为答案，而“哪里”问题寻找位置作为答案。根据开发数据集的得分，我们的模型在“何时”问题上表现最好。这可能是因为在这个数据集中，时间表达式相对容易识别。其他问题的答案是名词短语的情况，如“什么”问题，“哪个”问题和“哪里”问题，也得到相对较好的结果。然而，“为什么”问题是最难回答的问题。这并不奇怪，因为对“为什么”问题的答案可以是非常多样化的，并且它们不局限于任何特定类型的短语。

最后，我们想检查在匹配LSTM层中使用的注意机制是否有效地帮助模型找到答案。我们展示了图2中的注意力权重α。在图中，颜色越深，权重越高。我们可以看到有些词基于注意力权重已经很好地对齐了。例如，段落中的“德语”一词与第一个问题中的“语言”一词很好地对齐，该模型成功地预测了“德语”作为问题的答案。对于第二个问题中的“谁”，“老师”一词实际上收到相对较高的注意权重，并且模型已经预测了“Martin Sekulic”这个短语作为答案，这是正确的。对于从“为什么”开始的最后一个问题，注意力权重分布更均匀，不清楚哪些单词已经对齐已经和“为什么”对齐了。

4 相关工作

近年来，机器对文本的理解越来越受到人们的关注，越来越多的研究者正在构建数据驱动、端到端的神经网络模型。我们将首先回顾最近发布的数据集，然后在这个任务上的一些端到端模型。

4.1 数据集

通过从原始语料库中的句子中删除单词，在完形填空风格中创建了许多用于学习机器理解的数据集，任务是预测丢失的单词。例如，赫尔曼等人从《美国有线电视新闻网》和《每日邮报》中着重介绍完形填空题。Hill等人建立了以儿童故事为基础的儿童书籍测试数据集。崔发布了两个相似的中文数据集，来自《人民日报》与《儿童童话》。

而不是在完形填空的问题，一些其他数据集依赖于人类标注。理查德森等人创建了著名的McTest数据集，TabasW创建了MeVIEQA数据集。在这些数据集中，提供候选答案。与这两个数据集类似，SQuAD数据集也由人类标注答案。然而，与前两个不同的是，SQuAD数据集不提供候选答案，因此从给定段落的所有可能的子序列都必须被认为是候选答案。

除了上面的数据集之外，还有一些为机器理解而创建的其他数据集。例如WikRead DataSet和Babi DataSet，但是它们与自然语言有很大区别。

与自然界中的数据集有很大的不同。

4.2 为机器阅读理解设计的端到端神经网络模型

已经有许多研究提出端到端神经网络模型的机器理解。一种常见的方法是使用递归神经网络（RNNs）来处理给定文本和问题，以便预测或生成答案。注意力机制也被广泛地应用在RNNs的顶部，以便将问题与给定的通道相匹配。鉴于答案通常来自给定的段落，Ptr-Net在一些研究中被采用，以复制词从给定的段落作为答案。与现有的工作相比，我们使用match-LSTM来匹配一个问题和给定的通道，并且我们以不同的方式使用Ptr-Net，使得我们能够生成包含给定通道中的多个词的答案。

记忆网络已经应用到机器理解中了，但其应用到大数据集仍然是一个问题。在这项工作中，我们不考虑记忆网络。

5 结论

在本文中，我们开发了两个模型的机器理解问题定义在斯坦福问答数据集，利用match-LSTM和Ptr-Net。实验数据表明，我们的第二个模型，边界模型，在测试数据集上可以达到67.6%的精确匹配分数和77%的F1分数，这比我们的序列模型和RajPurkar等人的特征工程模型。在未来，我们计划深入研究不同类型的问题，并将焦点放在那些目前表现较低的问题上，例如“为什么”问题。我们还计划测试我们的模型可以应用到其他机器理解数据集。

References

[1] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Proceedings of the Conference on Advances in Neural Information Processing Systems, 2015.

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016.