哈爾濱工業大學

创新研修课程报告

专　　业　计算机科学与技术

学　　号　1160300901

姓　　名　孙月晴

小 组 成 员　柯文康　郭富博

目 录

1. 任务简介

1.1 课题背景及研究的目的意义

1.1.1 课题背景

人工智能在经历了几十年的寒潮后，随着深度神经网络的复苏与兴起，人工智能也迎来了新的生机。自然语言理解作为人工智能领域长久以来无人力及的明珠，也由于大规模数据集的出现和计算能力的急剧提升，达到了前所未有的研究热度，无论在学术界还是工业界，成为人工智能领域的必争之地，这背后自然有其重要意义。

自然语言理解作为一个终极的人工智能目标，其难度是不言而喻的，但从各个角度来分析，让机器理解语言的意义是所有人都无法视而不见的。从学术研究的角度来看，这是最前沿的学术进展，代表着机器在其所在时代具有的最高智能水平。从工业界的视角来看，对搜索引擎、聊天机器人、私人机器助手、翻译等重要产品都有直接的应用价值，进而产生巨大的商业价值。 但囿于目前的技术水平，用一个可评估、具有明确输入输出、定义清楚的任务来近似自然语言理解是一个实际可行的研究路线，机器阅读理解这一任务便是一个符合要求的近似任务。

阅读理解对于绝大多数人来说都不陌生，在中考、高考的语文和英语科目中，阅读理解都是很常见的一种题型，用来考察应试者对某一种类型文本的理解能力。想要正确的回答问题，需要理解原文、问题，综合原文和问题进行进一步的理解、推理从而得出正确答案，是一种简单、有效的评估人类理解文字的水平的方式。 机器阅读理解任务类似地包含原文、问题和答案，目标就是让机器阅读给定的原文和问题，得到与标准答案一样的答案，与面向人类的阅读理解要求是一样的。

出于方便评价的考虑，目前机器阅读理解的形式比较单一。从答案类型角度进行划分，可以分为完型填空型、选择型、原文片段型三种主要类型。根据答案类型的不同，数据集规模的不同，所选用的模型和方法会有很大的差异。因此，针对具有各自不同特点的数据集，研究如何应用相适应的模型与技术，经过探讨、分析总结出重要的结论，对于准确地评估机器在理解自然语言这一方面的智能具有重要意义，也为今后这一方向的研究提供具有参考性的经验。

1.1.2 研究的目的意义

近年来，随着机器学习技术的不断发展和计算能力的不断提高，在图像识别和语音识别等人工智能分支上，机器已经取得接近甚至超越人类的水平。2016 年，AlphaGo 以 4:1 的成绩击败围棋九段选手李世石后，人工智能成为了大众舆论的热点，今年改进版的 AlphaGo 又以 3:0 的比分击败柯洁，使人们又一次惊叹于人工智能的发展。

但如果我们冷静下来认真评估人工智能的发展水平时，并不会因为机器在众多领域上超过人类而恐慌，实际上，目前机器的智能仍然处于非常初级的阶段，因为机器目前无法比较好的理解人类的语言。

从某种程度上讲，语言是人类构建文明的基础，拥有复杂的语言是人类区别于其它低智能生物的重要特点。从而在评估机器拥有何种程度的智能时，机器对语言的理解程度是不可或缺的重要方面。同时，让机器具有理解语言的能力，也是目前人工智能的重要发展方向。

文字作为语言的重要部分，是传承人类文明和记录人类知识的主要载体，同时也是人们进行沟通的重要手段，因此，让机器具有理解我们的文字的能力具有重大的实际应用价值。比如能够极大的改变人与机器的交互方式，人们不必从计算机的角度出发，使用计算机能够理解的方式交互，转而很方便的使用自然语言进行交互。

从学术研究的角度看，自语言处理包括基本的分词、实体识别、句法分析等任务，还有比较高层次的问答、情感分析、文本摘要等任务。整体而言，这些任务之间相对独立，在开发机器理解语言的技术时，对于评估和综合各项自然语言处理任务是个很好的契机和指标。

同时，我们也应该意识到，往往越高层次的智能越难以准确地予以评价。在评价人类自身理解语言的能力时，往往会通过各种语言考试来评价，比如考察英语能力的 TOFLE、GRE 和我国的语文科目考试。

机器阅读理解任务作为研究和评价机器理解语言文字重要方式，具有重要的理论研究和实际应用价值。

1.2 相关工作

近几年，机器阅读理解任务的研究在国外上获得空前的瞩目，许多著名的研究机构，如斯坦福大学、卡内基梅隆大学、艾伦研究院等，工业界如 IBM、Google、Facebook 等巨头也纷纷加入到这一任务的研究中来。

微软研究院在 2013 年 EMNLP 会议上发布了 MCTest[1]这一机器阅读理解数据集。MCTest 作为机器阅读理解数据集，只包含 660 篇文本，共 2640 个选择题型的问题，其较小的数据规模导致大多数工作都是基于特征工程的工作。

Richardson[1]在发布 MCTest 数据集时提出两种朴素的非监督基线方法，这两种方法都只使用词法特征。第一种方法使用 BOW 模型，通过将问题与候选答案拼接得到相应 4 段文本，对于每一段拼接后的文本，使用与拼接得到文本相同长度的滑动窗口在阅读材料中进行滑动，并对窗口中含有与文本同样的词的个数进行计数，藉此来度量问题答案文本与阅读材料的语义匹配程度，进而选择答案。第二种方法同样先拼接文本，然后使用文本蕴含系统 BIUTEE 来判断假设文本与阅读材料的蕴含关系来选择答案。这两种方法虽然都比较简单，但是无论在 MC500 还是 MC160 上的效果都很好。Smith[1]对这两种方法进行了扩展，使用多个不同大小的滑动窗口；Wang[3]添加共指规则、否定检测等语言学特征和来提高系统的表现；Sachan[4]使用了修辞结构理论（RST）和事件实体共指方法对多对句子进行建模，获取和问题相关的部分，并进行有机的组合，来判断阅读文档和问题+答案文本之间的蕴含关系，也是将阅读理解任务形式化为文本蕴含来处理。

随后，Herman[5]在 2015 年发布填空型大规模英文机器阅读理解数据集 CNN&Dailymail，使深度学习方法应用在机器阅读理解任务上成为可能。Herman 提出三个深度学习模型，内部结构有所差别，但整体框架均为通过神经网络学习到问题和原文中每个词的表示，并基于这些表示进行打分，得分最高的词即为最后的答案。

对于 CNN&Dailymail 数据集，其答案一定为出现在原文中的一个词，根据这一特点，Kadlec[6]受 Pointer Network[7]启发提出 ASReader 模型，直接将原文中每个词的注意力相加作为其成为答案的概率，进而输出答案。这一模型非常简单，但取得了当时最好的结果，也启发了后续一系列模型。

受 ASReader 启发 Cui[8]提出了 AoA Reader，该工作的主要创新点是在注意力的基础上再计算一次针对注意力的注意力，借此来指示每个注意力的重要性，最后使用与 ASReader 相同的机制来计算每个词作为答案的概率。

同样受 ASReader 启发，Dhingra[9]提出 GAReader 来引入一种计算注意力的新机制。首先，GAReader 使原文和问题进行多层交互，在每一层之间，都通过 叫做 gated-attention 的机制使用 element-wise 的相乘来计算 attention，而不是采用以往计算注意力的点积方式计算，最后同样采用 ASReader 的求和机制来综合计算候选答案的概率。

Chen[11]提出基于人工构建的特征的传统分类模型和基于 Herman 提出的模型改进后的深度神经网络模型，主要是在计算注意力时使用 Bilinear 代替 Tanh，还对原模型进行简化，去掉最后的非线性层，实验表明 Bilinear 能够显著提升模型效果，而且最后的非线性层对结果的影响不大。更重要的是作者对数据进行了分析，通过将 100 个采样的问题进行分类分析，由于数据集本身的问

题，得出当前模型已经达到了该数据集上的最佳效果，几乎没有进一步提升的空间。

随后，Rajpurkar[13]发布了 SQuAD 数据集，采用众包的方式构造包含 10,000 个问题的大规模数据集，可以算是机器阅读理解领域的 ImageNet，其测试集不可见，需要提交模型到开放平台后跑出测试集上的结果并公布。与 CNN&Dailymail 数据集不同，SQuAD 的答案不是文中的一个词，而是原文中的一段连续文本片段。SQuAD 发布后，大量深度神经网络模型如雨后春笋般出现，使机器阅读理解取得进一步发展。

从形式上看，SQuAD 并没有超出 CNN&Dailymail 数据集很多，答案都限定在原文之中，但在难度和数据质量上均要明显高于之前的任何数据集。在 SQuAD 数据集上的模型都具有非常复杂的结构，对原文和问题进行嵌入，之后再进行复杂的交互，最后预测答案。

Wang[14]结合 Match-LSTM[15]和 Pointer Network 进行实验，发现直接预测答案边界的模型要优于直接预测答案的模型。受 Match 启发，Wang 提出 R-Net[16]，并长期占据 SQuAD 排行榜榜首至今，R-Net 主要提出 Self-matching 的机制来克服循环神经网络 RNN 无法建模长距离依赖的问题，从实验结果看，其效果非常好。

未登录词对于机器来说是一个很大的阻碍，Yang[17]提出通过 Gating 机制来动态的组合字符级别的词向量和同构预训练得到的词级别的词向量，一定程度上缓解未登录词所带来的问题，现在同时使用字符级和词级词向量来共同作为词的表示已经成为机器阅读理解领域的通用做法。

1.3 主要研究内容

2. 模型

由于本实验对结果的准确率没有强制要求,我们采用了最容易实现也是最经典的最长公共子序列算法来解决阅读理解问题。基本思想是:先把段落通过标点符号分成句子，然后使用最长公共子序列算法匹配出和问题最相似的句子，然后根据疑问词在问题中的位置提取出答案。

下面主要介绍一下使用动态规划的思想来解决最大公共子序列问题。

首先考虑最大公共子序列问题是否满足动态规划问题的两个基本特性：

1. 最优子结构：

设输入序列是 X [0 .. m-1] 和 Y [0 .. n-1]，长度分别为 m 和 n。和设序列 L(X [0 .. m-1], Y[0 .. n-1]) 是这两个序列的 LCS 的长度，以下为 L(X [0 .. M-1], Y [0 .. N-1]) 的递归定义：

1）如果两个序列的最后一个元素匹配（即 X [M-1] == Y [N-1]）

则：L（X [0 .. M-1]，Y [0 .. N-1]）= 1 + L（X [0 .. M-2]，Y [0 .. N-1]）

2）如果两个序列的最后字符不匹配（即 X [M-1] != Y [N-1]）
则：L(X [0 .. M-1]，Y [0 .. N-1]) = MAX(L(X [0 .. M-2]，Y [0 .. N-1])，L(X [0 .. M-1]，Y [0 .. N-2]))

通过如下具体实例来更好地理解一下：

1）考虑输入子序列 <AGGTAB> 和 <GXTXAYB>。最后一个字符匹配的字符串。这样的 LCS 的长度可以写成：

L(<AGGTAB>, <GXTXAYB>) = 1 + L(<AGGTA>, <GXTXAY>)

2）考虑输入字符串“ABCDGH”和“AEDFHR。最后字符不为字符串相匹配。这样的 LCS 的长度可以写成：

L(<ABCDGH>, <AEDFHR>) = MAX (L(<ABCDG>, <AEDFHR>),
L(<ABCDGH>, <AEDFH>))

因此，LCS 问题有最优子结构性质。

2. 重叠子问题：

很明显，基于上述的分析，LCS 很多子问题也都共享子子问题，因此可以对其进行递归求解。具体的算法时间度为 O(m*n)，可以优化至 O(m+n)。

由此给出状态转移方程:

$$c[i,j] = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ c[i-1,j-1]+1 & i,j > 0 \text{ and } x_i = y_j \\ \max(c[i,j-1], c[i-1,j]) & i,j > 0 \text{ and } x_i \neq y_j \end{cases}$$

3. 实验

3.1 实验设计

实验流程图:

122 个停止词词表:

```
上面
下面
因为
许多
很多
之中
自己
于是
因此
.....
```

1. LCS 模块:

```
def lcs(string1_list, string2_list):
    n = len(string1_list)
    m = len(string2_list)
    if m == 0 or n == 0:
        return -1
    c = [[0 for _ in range(m + 1)] for _ in range(n + 1)]
    for i in range(1, n + 1):
        for j in range(1, m + 1):
            if string1_list[i - 1] == string2_list[j - 1]:
                c[i][j] = c[i - 1][j - 1] + 1
```

```
            else:
                c[i][j] = max(c[i][j - 1], c[i - 1][j])
    return c[-1][-1]
```

2. 解析数据集模块

将数据集分解为文本和问题:

```
for line in data:
    head, content = line.split(' ||| ')
    if 'qid' in head:
        one_question = content.split(" ")
        if one_question[-1] in ['? ', '?']:
            one_question = one_question[:-1]
        question.append(one_question)
        text.append(one_text)
        one_text = []
    else:
        one_text_line = content.split(" ")
        one_text.extend(one_text_line)
```

3. 匹配最长公共序列模块

利用标点符号将文本分割为句子,与相对应的问题进行匹配,取出文本中相似度最高的句子作为候选答案所在的句子。

```
most_lcs_sentence = []
most_lcs_length = 0
for sen in sentences:
    lcs_lenght = lcs(sen, question)
    if lcs_lenght > most_lcs_length:
        most_lcs_length = lcs_lenght
        most_lcs_sentence = sen
```

4. 抽取答案模块

这里给出了 7 个疑问词:

```
que_words = ['什么', '哪里', '谁', '哪', '哪儿', '什么样', '怎么']
```

这里可分为三种情况:

(1) que_words 在问题的开始

例如:什么 闯进 了 胆小 先生 的 房子

在候选句中遍历疑问词的后一个词,如果在候选句中,则取出候选句的开始到疑问词之前作为答案

```
for qw in que_words:
    if qw in question:
        que_len = len(question)
        qw_index = question.index(qw)
        qw_b_index = qw_index - 1
        qw_a_index = qw_index + 1
```

```
        if qw_b_index < 0:
            while qw_a_index < que_len-1 and question[qw_a_index] not in
sentence :
                qw_a_index += 1
            if question[qw_a_index] in sentence:
                answer = sentence[:sentence.index(question[qw_a_index])]
            else:
                answer = sentence[:qw_a_index]
```

(2) que_words 在问题的结尾

例如: 贾黄中 十五 岁 当 了 什么

从疑问词的前一个词开始逆序遍历候选句,取出疑问词下标到候选句结尾作为问题的答案

```
elif qw_a_index >= que_len:
    while qw_b_index > 0 and question[qw_b_index] not in sentence :
        qw_b_index -= 1
    if question[qw_b_index] in sentence:
        answer = sentence[sentence.index(question[qw_b_index])+1:]
    else:
        answer = sentence[qw_b_index:]
```

(3) que_words 在问题的中间

例如:慢羊羊 教授 开 着 什么 车 回 了 家

结合前两中情况的思路,从疑问词的前一个词逆序遍历候选句,从疑问词的后一个词顺序遍历候选句,抽取前后下标所在候选句中的字段作为候选答案。

```
else:
    while qw_a_index < que_len - 1 and question[qw_a_index] not in sentence :
        qw_a_index += 1
    while qw_b_index > 0 and question[qw_b_index] not in sentence:
        qw_b_index -= 1
    if question[qw_b_index] in sentence:
        start_index = sentence.index(question[qw_b_index])+1
        if start_index >= len(sentence) - 1:
            start_index -= 1
    else:
        start_index = qw_b_index
    if question[qw_a_index] in sentence:
        end_index = sentence.index(question[qw_a_index])
        if end_index <= 0 :
            end_index += 1
    else:
        end_index = qw_a_index
```

```
    answer = sentence[start_index:end_index]
```

3.2 实验结果及分析

1. 实验结果:

本实验采用的数据集主要面向”儿童读物”领域,篇章中包含较多拟人化的动植物,最后在测试集上达到 41.6 的 F1 值.

2. 结果分析:

由于使用的是最直接的最长公共子序列和关键词抽取,F1 值的得分确实不高,主要有以下几点:

(1) 标准答案在抽取结果中,但冗余词太多,这种结果占 64%,也是导致评分低的关键原因。主要是因为最长序列匹配结果太长,应该缩小关键词的提取范围。

如: 一个会走路铅笔盒

(2) 关键词抽取错误,答案不在抽取结果中,这种结果占 25%左右。这里涉及语义理解的问题,而不是单纯的完形填空能解决的了。

也许该

件事

4. 结论及收获

总结来看,最长子序列匹配只能解决部分”完形填空”类型的阅读理解,性能并不能达到要求,要提高精确度必须转变模型,以后可以尝试使用深度神经网络模型,对原文和问题进行嵌入,进行复杂的交互来预测答案。不过这次使用经典算法解决机器阅读理解问题让我深受启发。

5. 组内分工

姓名	分工
调研,报告	孙月晴,郭富博
程序	柯文康

参考文献

[1] Richardson, Matthew, Christopher JC Burges, and Erin Renshaw. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text[C].// Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 2013:193-203.

[2] Smith E, Greco N, Bosnjak M, et al. A Strong Lexical Matching Method for the

Machine Comprehension Test[C]// Conference on Empirical Methods in Natural Language Processing. 2015:1693-1698.

[3] Wang H, Bansal M, Gimpel K, et al. Machine Comprehension with Syntax, Frames, and Semantics[C]// Meeting of the Association for Computational Linguistics and the, International Joint Conference on Natural Language Processing. 2015:700-706.

[4] Sachan M, Dubey K, Xing E, et al. Learning Answer-Entailing Structures for Machine Comprehension[C]// Meeting of the Association for Computational Linguistics and the, International Joint Conference on Natural Language Processing. 2015:239-249.

[5] Hermann K M, Tomáš Kočiský, Grefenstette E, et al. Teaching Machines to Read and Comprehend[C]// Proceedings of the 28th International Conference on Neural Information Processing Systems. 2015:1693-1701.

[6] Kadlec R, Schmid M, Bajgar O, et al. Text Understanding with the Attention Sum Reader Network[C]// Meeting of the Association for Computational Linguistics. 2016:908-918.

[7] Vinyals O, Fortunato M, Jaitly N. Pointer Networks[J]. Advances in Neural Information Processing Systems 28, 2015: 2692-2700.

[8] Cui Y, Chen Z, Wei S, et al. Attention-over-Attention Neural Networks for Reading Comprehension[C]// Meeting of the Association for Computational Linguistics and the, International Joint Conference on Natural Language Processing. 2017 Accepted.

[9] Dhingra B, Liu H, Yang Z, et al. Gated-Attention Readers for Text Comprehension[C] // Meeting of the Association for Computational Linguistics and the, International Joint Conference on Natural Language Processing. 2017:Accepted.

[10] Yu Y, Zhang W, Hasan K, et al. End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension[CL/OL].[2016-11-03].https://arxiv.org/pdf/1610.09996.

[11] Chen D, Bolton J, Manning C D. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task[C]// Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. 2016:2358-2367.

[12] Rajpurkar P, Zhang J, Lopyrev K, et al. SQuAD: 100,000+ Questions for Machine Comprehension of Text[C] //Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016:2383-2392.

[13] Wang S, Jiang J. Machine Comprehension Using Match-LSTM and Answer Pointer[C]// International Conference on Learning Representations, 2017.
[14] Wang S, Jiang J. Learning Natural Language Inference with LSTM[C]// Conference of the North American Chapter of the Association for Computational Linguistics. 2016:1442-1451.
[15] Wang, Wenhui and Yang, Nan and Wei, Furu, et al. Gated Self-Matching Networks for Reading Comprehension and Question Answering[C] // Meeting of the Association for Computational Linguistics and the, International Joint Conference on Natural Language Processing. 2017: Accepted.
[16] Yang Z, Dhingra B, Yuan Y, et al. Words or Characters? Fine-grained Gating for Reading Comprehension[C]// International Conference on Learning Representations, 2017.