中山大学本科生

优秀毕业论文

（中文题目）基于维基百科检索和机器理解的问答系统

（英文题目）Question Answering System based on Wikipedia Retrieval and Machine Comprehension

学位申请人：刘　昕

导　　师：潘　嵘

专　　业：计算机科学与技术

学　　院：数据科学与计算机学院

2018 年 6 月

基于维基百科检索和机器理解的问答系统

[摘 要] 问答系统是人工智能的一个应用，最初限制在专业的领域，由人工编制的数据库、产生式规则集和控制策略三部分组成。但是由于系统的封闭以及推理的困难，导致问答系统发展陷入瓶颈。随着互联网的发展，网络上产生了越来越多的信息可以被人获取和利用，开放域的问答系统成为了人们研究的热点。2016 年斯坦福大学举办 SQuAD 机器理解比赛，各个科研团队尝试和探索不同的算法和模型，最终在精确匹配的指标上超过人类。本文设计了一个大规模开放域的问答系统，包括三个部分：基于 bigram hashing 和 TF-IDF 的信息检索模型 Document Retriever，基于多层循环神经网络、语义融合单元和注意机制的机器理解模型 Stochastic Mnemonic Reader，对机器理解模型的候选答案进行排名并且提高问答系统的准确性答案排名算法 Answer Ranking Algorithm。实验证明 Document Retriever 优于维基百科的搜索引擎，同时 Stochastic Mnemonic Reader 在精确匹配和 F1 指标上都超过了已公开发表的论文单模型 (截至到 2017 年 12 月)，Answer Ranking Algorithm 使得该系统对比其他开放域问答系统仍然具有竞争力。

[关键词] 自然语言处理；信息检索；机器理解；问答系统

Question Answering System based on Wikipedia Retrieval and Machine Comprehension

[Abstract] The Question Answering Systems are an application of Artificial Intelligence. They were limited to professional fields at first, with organized database, rules of productions and control strategies. But they had been fallen into bottleneck because of the closure themselves and difficulty in inferring. Since the development of the Internet, there has been more and more available information to utilize and open-domain Question Answering Systems have become one of research focuses. Stanford held a machine comprehension competition, SQuAD. Many universities and research teams have attempted various algorithms and models and some models exceed human beings in the evaluation of Exact Match. This paper designs a large-scale open-domain Question Answering System, including: the Document Retriever which is an Information Retrieval model based on bigram hashing and TF-IDF, the Stochastic Mnemonic Reader which is a Machine Comprehension model with a multi-layer Recurrent Neural Network model, Semantic Fusion Units and Attention Mechanism and the Answer Ranking Algorithm to rank candidate answers, successfully improving the accuracy. Experiments indicate that the Document Retriever outperforms Wikipedia Search Engine, and the Stochastic Mnemonic Reader defeats all public single models in the evaluation of Exact Match and F1 (until Dec 2017). What's more, the Answer Ranking Algorithm helps the system be highly competitive with existing open-domain systems.

[Keywords] Natural Language Processing; Information Retrieval; Machine Comprehension; Question Answering System

目录

1. 引言 1

2. 相关工作 3

3. 任务分析和方法设计 5

3.1 任务分析 5

3.2 信息检索模型 Document Retriever 6

3.3 机器理解模型 Stochastic Mnemonic Reader 6

3.3.1 编码器 6

3.3.2 对齐器 8

3.3.3 随机预测网络 10

3.3.4 目标函数 12

3.4 答案排名算法 Answer Ranking Algorithm 12

3.5 参数设置 13

4. 实验与分析 15

4.1 数据集 15

4.1.1 维基百科 15

4.1.2 SQuAD 15

4.1.3 WikiMovies 15

4.1.4 CuratedTrec 16

4.2 信息检索模型评估 16

4.3 机器理解模型评估 17

4.4 问答系统评估 19

4.4.1 远程监督 19

4.4.2 评估结果 20

5. 结语 23

参考文献 24

致谢 26

附录 27

1. 引言

问答系统 (Question Answering System, QA) 是人工智能的一个应用，最初限制在专业的领域，由人工编制的数据库、产生式规则集和控制策略三部分组成。但是由于系统的封闭以及推理的困难，导致问答系统发展陷入瓶颈。随着互联网的发展，网络上产生了越来越多的信息，尤其是文本信息，可以被人们获取和利用，开放域的问答系统成为了人们研究的热点。开放域的问答系统主要有三个方向，基于知识库 (KB-based)，基于文本 (document-based)，以及两者的融合。知识库是知识的结构化形式，常用的知识库有 Freebase[1], YAGO[2], DBPedia[3]。但是知识库不可避免存在知识不完全性和知识的固定性，所以知识库中知识是现实知识的很小的子集，这类问答系统覆盖范围往往很稀疏，同时高度依赖知识库。基于文本的问答系统信息来源更加广泛，维基百科、百度百科等百科是人类协同构建的文本集合，收录着传统百科全书所收录的信息，同时也能够十分迅速地更新与最近发生事件相关的信息，是基于文本的问答系统常用的数据集合。但是基于文本的问答系统存在着超大规模数据和机器理解的挑战。基于文本和知识库的混合问答系统是两个方向的平衡与折中。

2015 年以来，研究人员公开了越来越多的数据集，来不断挑战机器理解难题，包括 MS-MARCO[4], SQuAD[5]。其中在 2016 年斯坦福大学举办 SQuAD 比赛中，各个科研团队尝试和探索不同的算法和模型，最终在精确匹配 (EM) 的指标上超过人类。但是，目前的机器理解模型往往只适用基于一小段段落回答问题，所以这些模型并不能直接拿来构建基于文本的大规模问答系统，但是这些模型仍然对问答系统的发展甚至是整个自然语言处理的发展起了推动作用。

本文设计了一个大规模开放域的问答系统 WRMCQA (Wikipedia Retrieval and Machine Comprehension Question Answering)，充分利用了维基百科的丰富的知识和广泛的时事新闻，利用优秀的机器理解模型进行答案匹配，同时利用答案排名算法对候选答案进行排名，提高问答系统的准确性。WRMCQA 包括三部分

1. Chen D[6]的基于 bigram hashing[7]和 TF-IDF 的信息检索模型 Document Retriever，

2. 基于多层循环神经网络 (RNN)、语义融合单元 (Semantic Fusion Unit)[8] 和注意机制 (Attention Mechanism)[9] 的机器理解模型 Stochastic Mnemonic Reader，
3. 综合考虑 TF-IDF 分数、机器理解的候选答案次数和候选答案分数的答案排名算法 Answer Ranking Algorithm。

实验证明信息检索模型优于维基百科的搜索引擎，同时机器理解模型在精确匹配 (EM) 和 F1 指标上都超过了已公开发表的论文单模型 (截止 2017 年 12 月)，答案排名算法使得该系统对比其他开放域问答系统仍然具有竞争力。

文章正文分为五个部分，第一部分是引言，介绍文章和模型设计背景，模型概要；第二部分是相关工作论述，列举并分析相关工作的优缺点，主要包括问答系统和机器理解；第三部分包括任务分析、模型讲解和算法设计；第四部分分析数据集、模型以及相关模型的对比；第五部分总结本文工作以及展望未来。

2. 相关工作

早期的问答系统基于信息检索，设计为返回包含答案的文本片段，在响应时间和问题覆盖方面都有限制[10]。随着大规模的知识库的出现，基于语义分析和知识库的新型问答系统可以返回更加精确的答案[11]，也掀起了知识库和知识图谱的狂潮。但是由于知识库覆盖的稀疏性，仍然需要利用文本来补充知识库中缺少的信息[12]，例如开放性的信息提取 (OpenIE)。近年也有将文本和知识库方法进行混合的系统，例如 DeepQA[13]，YodaQA[14]。DeepQA 是一个非常复杂的系统，它依赖包括文本文档在内的非结构化信息以及结构化数据，在生成候选答案时进行投票。YodaQA 是一款以 DeepQA 为原型的开源问答系统，同样将网站、信息提取、数据库和维基百科相结合。利用多个信息来源会增加系统的准确度，但是同样会使得系统变得庞大而且复杂。所以目前基于文本的问答系统仍然具有研究价值。基于文本的问答系统面临的难题有两个，一个是文本数据的大规模，另一个是机器理解。维基百科作为文本来源可以保证质量和数量，已经有人进行尝试并且取得了进展。2004 年 Ahn D 等人[15]将维基百科作为文本资源利用信息检索构造了问答系统。2014 年 Ryu PM 等人[16]使用基于维基百科构建问答系统，他们将文章内容与百科中的信息框、目录等半结构化知识结合进行匹配答案。D-NMN[17]利用神经网络从知识库中寻找问题的答案，构造了一个问答系统，但更偏重于字符串的匹配而不是推理。DrQA[6]利用信息检索和机器理解模型实现了开放域的问答系统。目前一些问答系统的数据集也逐渐丰富起来，包括 CuratedTrec[18]，WebQuestions[19]，WikiMovies[20]。这些为问答系统的发展提供了数据的支持。

机器理解是自然语言处理中无法绕过的一座大山，近年来机器理解取得了突破性的进步，其中很大程度上要归功于公开的大规模数据集。大规模的完形填空数据集 CNN/DailyMail[21]首先公开，为深度学习方法提供了数据的支持。但是完形填空类型的数据往往很少需要推理，机器理解的智能并没有真正实现。2016 年斯坦福大学公开 SQuAD 并举办了竞赛[5]。和以往的数据集不同，SQuAD 提供了大规模高质量的数据，答案包括 10 种不同的类型，并且每一个问题都是人工生成的。比赛用精确匹配 (EM) 和 F1 分数作为评价指标。同时

Rajpurkar 等人提供了人工提取特征和逻辑回归的基线法，同时也提供了人类的指标。比赛吸引了全球各大科研机构的参与，目前微软亚洲研究院和新加坡管理大学的 Reinforced Mnemonic Reader + A2D (ensemble model) 在 EM 指标上达到最高分，并且超过人类；哈尔滨工业大学和讯飞联合实验室的 Hybrid AoA Reader (ensemble model) 在 F1 分数上处于领先优势。在机器理解这项挑战中，深度学习模型被证明是强大有效的方法，各种各样的网络结构以及计算单元被提出和利用。Hermann KM 等人[21]将 Attention Mechanism 引入到机器理解中，Wang S 和 Jiang J[22]在模型中将 Match-LSTM 和 Pointer Network[23]结合，并且在 Answer Pointer layer 中提出 Sequence 和 Boundary 两种模型，其中 Boundary 模型被后来众多模型采用。Xiong C 等人[24]提出 Coattention Mechanism 和 Dynamic Pointer Network 使问题和段落进行关联，提升模型的效果。Wang W 等人[25]在他们的 R-Net 模型中使用了 Gated Attention-based RNN 以及段落的 Self-Matching Attention，大幅提高阅读理解的 EM 和 F1 分数。Chen D[6]在对段落进行进行编码时加入词性 (Part-of-Speech, POS),命名实体识别 (Named Entity Recognition, NER)和问题对齐权重 (Aligned question embedding)。Hu M[8]提出 Semantic Fusion Unit 来融合词的上下文到每个词中。Liu X 等人[26]在 SAN 模型中使用了 Contextualized Word Vectors (cove)[27]，大幅提高编码的效果，同时使用 Stochastic Dropout 提高模型的鲁棒性。BiDAF+SelfAttention+ELMo 模型使用了更大规模的上下文词向量 ELMo，目前在两种评价指标上均占有很大的优势。

3. 任务分析和方法设计

3.1 任务分析

对于基于文本的开放域的问答系统，输入为一个问题Q，任务为从文本集D中找到问题Q的答案A_Q。对于这个问题，本文将任务拆分成 3 个子任务，对应为模型的 3 个部分：

1. 根据问题Q，从文本集D中找到与问题相关的文本子集D_Q，并将D_Q拆分成段落集合P_Q，对应于信息检索模型 Document Retriever；
2. 根据问题Q，从段落集合P_Q中每一个段落P找到候选答案A，对应于机器理解模型 Stochastic Mnemonic Reader；
3. 根据所有段落的候选答案A_{QP}确定问题Q的最终答案A_Q，对应于答案排名算法 Answer Ranking Algorithm。

将任务拆分成 3 个子任务，会使模型的准确性降低，但是让模型变得低耦合，方便提升和扩展。

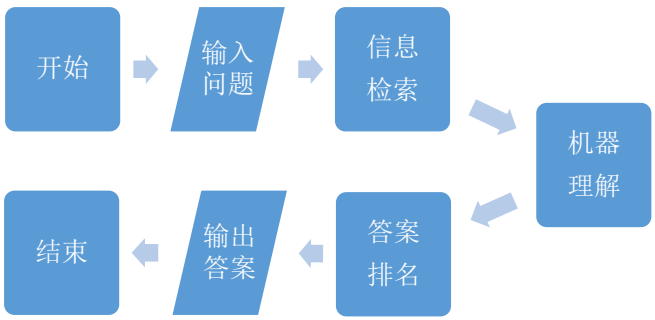

图 1 模型流程图

3.2 信息检索模型 Document Retriever

传统的问答系统利用信息检索的方法来缩小答案的搜索空间，DrQA[6]中的文本检索部分在多个数据集上被验证为是简单有效的检索模型，本文直接利用其模型完成信息检索的部分。和维基百科内置的搜索 API 相比，DrQA[6]中的 Document Retriever 增加了倒排索引和词向量。利用二元 (bigram) 词袋模型 (Bag-of-Words) 将文本和问题向量化，然后使用 TF-IDF 计算权重。为了计算和内存使用的效率，使用 unsigned murmur3[7]并设置哈希表的大小为 24bits。

每次对一个问题Q进行检索，默认返回 5 个相关度最高的文本子集D_Q，拆分成段落集合P_Q作为机器理解模型 Stochastic Mnemonic Reader 的输入。

3.3 机器理解模型 Stochastic Mnemonic Reader

机器理解模型是问答系统的核心，也是这个系统最复杂的地方。根据目的可以分为 3 个部分，分别为编码器，对齐器和随机预测网络。

3.3.1 编码器

在交互系统中，用户输入的问题Q和返回的答案A都是文本。在很长一段时间里，对文本的编码都是利用 one-hot 编码，使文本可以被计算机计算，但是高维向量并不能很好解决自然语言处理中的相似性问题，同时也影响内存和计算效率。Word2Vec[28]的出现使得词可以映射到低维向量空间，并且蕴含上下文信息，词与词之间的相似性和相关性也可以通过余弦相似性 (Cosine Similarity) 进行衡量。本系统和其他机器理解模型一样，使用 300 维的 Glove 词向量和字符向量[29]，并且保持其在训练过程中不进行更新。

给定一个问题$Q = \{w_t^Q\}_{t=1}^m$和一个段落$P = \{w_t^P\}_{t=1}^n$，利用如下特征对其进行编码：

1. 字符特征：为了解决词库外词 (Out-of-Vocab, OOV) 的问题，R-Net[25]使用了双向门控循环单元 (BiGRU) 对每个词的首字母进行编码。本模型使用双向长短期记忆网络 (BiLSTM) 对进行该编码过程。首先根据$\{w_t^Q\}_{t=1}^m$

和$\{w_t^P\}_{t=1}^n$得到字符序列$\{c_t^Q\}_{t=1}^m$和$\{c_t^P\}_{t=1}^n$，利用预训练的Glove字符向量，得到字符编码$\{\boldsymbol{e}_{\boldsymbol{c}_t}^{\boldsymbol{Q}}\}_{t=1}^m$和$\{\boldsymbol{e}_{\boldsymbol{c}_t}^{\boldsymbol{P}}\}_{t=1}^n$。将字符编码作为一个BiLSTM的输入，得到字符特征：

$$\boldsymbol{x}_{\boldsymbol{c}_t}^{\boldsymbol{Q}} = \text{BiLSTM}\left(\boldsymbol{x}_{\boldsymbol{c}_{t-1}}^{\boldsymbol{Q}}, \boldsymbol{e}_{\boldsymbol{c}_t}^{\boldsymbol{Q}}\right), t = 0,1,2,\dots m, \quad (1)$$

$$\boldsymbol{x}_{\boldsymbol{c}_t}^{\boldsymbol{P}} = \text{BiLSTM}\left(\boldsymbol{x}_{\boldsymbol{c}_{t-1}}^{\boldsymbol{P}}, \boldsymbol{e}_{\boldsymbol{c}_t}^{\boldsymbol{P}}\right), t = 0,1,2,\dots n. \quad (2)$$

2. 词向量特征：将每一个词的词编码$\{\boldsymbol{e}_{\boldsymbol{w}_t}^{\boldsymbol{Q}}\}_{t=1}^m$和$\{\boldsymbol{e}_{\boldsymbol{w}_t}^{\boldsymbol{P}}\}_{t=1}^n$作为另一个特征：

$$\boldsymbol{x}_{\boldsymbol{w}_t}^{\boldsymbol{Q}} = \boldsymbol{e}_{\boldsymbol{w}_t}^{\boldsymbol{Q}}, t = 0,1,2,\dots m, \quad (3)$$

$$\boldsymbol{x}_{\boldsymbol{w}_t}^{\boldsymbol{P}} = \boldsymbol{e}_{\boldsymbol{w}_t}^{\boldsymbol{P}}, t = 0,1,2,\dots n. \quad (4)$$

3. 词性 (POS) 特征：Chen D[6]使用POS作为特征，可以使其模型在F1分数上提高0.8个百分点。本系统一共使用了51个POS类型。对于每一个词，将其对应词性的向量数值设置为1，否则为0：

$$\boldsymbol{x}_{\mathbf{POS}_t}^{\boldsymbol{Q}} = \text{POS}\left(w_t^Q\right), t = 0,1,2,\dots m, \quad (5)$$

$$\boldsymbol{x}_{\mathbf{POS}_t}^{\boldsymbol{P}} = \text{POS}(w_t^P), t = 0,1,2,\dots n. \quad (6)$$

4. 命名实体识别 (Named Entity Recognition) 特征：使用NER作为特征来帮助系统更好地确定答案边界，针对对于事件、时间的问题效果显著。本系统一共使用了19个NER类型，包括时间、时间、日期等。对于每一个词，同样利用one-hot进行编码：

$$\boldsymbol{x}_{\mathbf{NER}_t}^{\boldsymbol{Q}} = \text{NER}\left(w_t^Q\right), t = 0,1,2,\dots m, \quad (7)$$

$$\boldsymbol{x}_{\mathbf{NER}_t}^{\boldsymbol{P}} = \text{NER}\left(w_t^Q\right), t = 0,1,2,\dots n. \quad (8)$$

5. 词频 (Term Frequency) 特征：利用TF来判断该词是否为该段落和问题的重点词，配合NER会提高进一步提高答案定位的准确性：

$$\boldsymbol{x}_{\mathbf{TF}_t}^{\boldsymbol{Q}} = \text{TF}\left(w_t^Q\right), t = 0,1,2,\dots m, \quad (9)$$

$$\boldsymbol{x}_{\mathbf{TF}_t}^{\boldsymbol{P}} = \text{TF}(w_t^P), t = 0,1,2,\dots n. \quad (10)$$

6. 精确匹配 (Exact Match) 特征：Chen D[6]使用精确匹配和对齐权重使模型F1分数提高了19.4%。由于模型后面有对齐的部分，只利用了精确匹配的思想。判断段落（或问题）的词是否出现在问题（或段落）中。可以利用该特征来直接定位简单的答案所在的上下文，例如段落连续若干个词都出现在问题中，那么这些词的上下文很可能包含着问题的答案：

$$\boldsymbol{x}^{\boldsymbol{Q}}_{\mathbf{EM}_t} = \mathbb{I}\{w^Q_t \in P\}, t = 0,1,2,\dots m\,, \tag{11}$$

$$\boldsymbol{x}^{\boldsymbol{P}}_{\mathbf{EM}_t} = \mathbb{I}\{w^Q_t \in P\}, t = 0,1,2,\dots n\,. \tag{12}$$

最后将得到的特征进行连结，得到问题特征向量和段落特征向量：

$$\boldsymbol{x}^{\boldsymbol{Q}}_t = \left[\boldsymbol{x}^{\boldsymbol{Q}}_{\boldsymbol{c}\,t}; \boldsymbol{x}^{\boldsymbol{Q}}_{\boldsymbol{w}_t}; \boldsymbol{x}^{\boldsymbol{Q}}_{\mathbf{POS}_t}; \boldsymbol{x}^{\boldsymbol{Q}}_{\mathbf{NER}_t}; \boldsymbol{x}^{\boldsymbol{Q}}_{\mathbf{TF}_t}\right], t = 0,1,2,\dots m\,, \tag{13}$$

$$\boldsymbol{x}^{\boldsymbol{P}}_t = \left[\boldsymbol{x}^{\boldsymbol{P}}_{\boldsymbol{c}\,t}; \boldsymbol{x}^{\boldsymbol{P}}_{\boldsymbol{w}_t}; \boldsymbol{x}^{\boldsymbol{P}}_{\mathbf{POS}_t}; \boldsymbol{x}^{\boldsymbol{P}}_{\mathbf{NER}_t}; \boldsymbol{x}^{\boldsymbol{P}}_{\mathbf{TF}_t}\right], t = 0,1,2,\dots n\,. \tag{14}$$

为了对序列更好的建模，同时将上下文的信息整合在一起，利用另一个 2 层 BiLSTM 去分别处理问题特征向量和段落的特征向量。利用一个 maxout 层[30]使得 2 层 BiLSTM 编码后的向量维度减为一半。得到最终编码后的向量：

$$\begin{aligned}
\boldsymbol{h}_{\mathbf{1}\boldsymbol{t}}^{\boldsymbol{Q}} &= \mathrm{BiLSTM}_1\left(\boldsymbol{h}_{\mathbf{1}\boldsymbol{t-1}}^{\boldsymbol{Q}}, \boldsymbol{x}_{\boldsymbol{t}}^{\boldsymbol{Q}}\right), t = 0,1,2,\dots m\,,\\
\boldsymbol{h}_{\mathbf{2}\boldsymbol{t}}^{\boldsymbol{Q}} &= \mathrm{BiLSTM}_2\left(\boldsymbol{h}_{\mathbf{2}\boldsymbol{t-1}}^{\boldsymbol{Q}}, \boldsymbol{h}_{\mathbf{1}\boldsymbol{t}}^{\boldsymbol{Q}}\right), t = 0,1,2,\dots m\,,\\
\boldsymbol{q}_{\boldsymbol{t}} &= \mathrm{Maxout}\left(\boldsymbol{h}_{\mathbf{1}\boldsymbol{t}}^{\boldsymbol{Q}}, \boldsymbol{h}_{\mathbf{2}\boldsymbol{t}}^{\boldsymbol{Q}}\right), t = 0,1,2,\dots m\,,
\end{aligned} \tag{15}$$

$$\begin{aligned}
\boldsymbol{h}_{\mathbf{1}\boldsymbol{t}}^{\boldsymbol{P}} &= \mathrm{BiLSTM}_1\left(\boldsymbol{h}_{\mathbf{1}\boldsymbol{t-1}}^{\boldsymbol{P}}, \boldsymbol{x}_{\boldsymbol{t}}^{\boldsymbol{P}}\right), t = 0,1,2,\dots n\,,\\
\boldsymbol{h}_{\mathbf{2}\boldsymbol{t}}^{\boldsymbol{P}} &= \mathrm{BiLSTM}_2\left(\boldsymbol{h}_{\mathbf{2}\boldsymbol{t-1}}^{\boldsymbol{P}}, \boldsymbol{h}_{\mathbf{1}\boldsymbol{t}}^{\boldsymbol{P}}\right), t = 0,1,2,\dots n\,,\\
\boldsymbol{p}_{\boldsymbol{t}} &= \mathrm{Maxout}\left(\boldsymbol{h}_{\mathbf{1}\boldsymbol{t}}^{\boldsymbol{P}}, \boldsymbol{h}_{\mathbf{2}\boldsymbol{t}}^{\boldsymbol{P}}\right), t = 0,1,2,\dots n\,.
\end{aligned} \tag{16}$$

编码后的向量$\boldsymbol{q_t} \in \mathbb{R}^{2h}, \boldsymbol{p_t} \in \mathbb{R}^{2h}$，序列矩阵$\boldsymbol{Q} \in \mathbb{R}^{m\times 2h}, \boldsymbol{P} \in \mathbb{R}^{n\times 2h}$，其中$h$为隐藏层大小。这些向量将作为对齐器的输入。

3.3.2 对齐器

机器理解模型中，注意机制 (Attention Mechanism) 或者对齐机制 (Alignment Mechanism) 的重要性不言而喻。它是连接问题和段落之间的桥梁。在 R-Net[25]中，使用门限 Gated-GRU 进行对齐；在 DrQA[6]、SAN[26]中，使用 Dot-Product Attention[9]进行对齐；在 Mnemonic Reader[8]中，使用相似矩阵和语义融合单元 (Semantic Fusion Unit) 进行对齐。本模型使用 Mnemonic Reader[8]的 SFU，这也是本模型叫做 Stochastic Mnemonic Reader 的原因。本模型中对齐分为 3 个部分，分别是交互对齐、自对齐和聚集，为了增强对齐的效果，模型采用多次进行对齐操作。在介绍这几个方法之前，先介绍语义融合单元。

3.3.2.1 语义融合单元

语义融合单元 (Semantic Fusion Unit, SFU) 的目的让经过对齐的信息融合到

原来的向量中，使之包含语义信息。SFU 输入为一个向量$\boldsymbol{i}$和k个融合向量$\{\boldsymbol{f}_i\}_{i=1}^{k}$，输出为和$\boldsymbol{r}$相同维度的向量$\boldsymbol{o}$。输出向量$\boldsymbol{o}$需要保证向量$\boldsymbol{i}$的信息仍然保留，并且引入了融合向量$\{\boldsymbol{f}_i\}_{i=1}^{k}$的信息。在这里使用一个全连接层和 tanh 激活函数对向量$\boldsymbol{i}$和k个融合向量$\{\boldsymbol{f}_i\}_{i=1}^{k}$进行融合：

$$\boldsymbol{h} = \tanh(\boldsymbol{W_1}[\boldsymbol{i}; \boldsymbol{f_1}; \boldsymbol{f_2}; \dots; \boldsymbol{f_k}] + \boldsymbol{b_1}) . \tag{17}$$

然后使用一个全连接层和 sigmoid 激活函数对向量$\boldsymbol{i}$和k个融合向量$\{\boldsymbol{f}_i\}_{i=1}^{k}$计算门限：

$$\boldsymbol{g} = \text{sigmoid}(\boldsymbol{W_2}[\boldsymbol{i}; \boldsymbol{f_1}; \boldsymbol{f_2}; \dots; \boldsymbol{f_k}] + \boldsymbol{b_2}) . \tag{18}$$

最终得到输出向量：

$$\mathbf{o} = \mathbf{g} \odot \mathbf{h} + (1 - \mathbf{g}) \odot \mathbf{i} , \tag{19}$$

其中⊙指向量的元素乘法。(19)第一项是融合的结果，第二项是保留向量$\boldsymbol{i}$的信息。

3.3.2.2 交互对齐

由于机器理解需要从段落中寻找问题的答案，模型需要将问题的信息传递给段落的向量中，得到问题感知 (Query-Aware) 的段落向量。在第t次交互对齐中，需要使用第$t-1$ 次对齐的结果$\check{\boldsymbol{P}}^{\boldsymbol{t-1}}$ (当$t=0$时为$\boldsymbol{P}$)。首先利用 Dot-Product Attention[9]计算 Attention Matrix:

$$\boldsymbol{A^t} = \text{softmax}\left(\check{\boldsymbol{P}}^{\boldsymbol{t-1}}\boldsymbol{Q}^{\mathrm{T}}\right) , \tag{20}$$

其中$\boldsymbol{A^t} \in \mathbb{R}^{\boldsymbol{n\times m}}$, $\boldsymbol{Q^{\mathrm{T}}}$是$\boldsymbol{Q^{\mathrm{T}}}$的转置。然后利用 Attention Matrix 计算出对应的权重：

$$\tilde{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{t}} = \boldsymbol{A}_{\boldsymbol{i},:}^{\boldsymbol{t}}\boldsymbol{Q}, i = 1,2, \dots n , \tag{21}$$

其中$\tilde{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{t}} \in \mathbb{R}^{\boldsymbol{n\times 2h}}$。然后将$\check{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t-1}}$作为输入，$\tilde{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{t}}, \check{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t-1}} \odot \tilde{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{t}}, \check{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t-1}} - \tilde{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{t}}$作为融合向量输入给一个 SFU：

$$\overline{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}} = \text{SFU}\left(\check{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t-1}}, \tilde{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{t}}, \check{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t-1}} \odot \tilde{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{t}}, \check{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t-1}} - \tilde{\boldsymbol{q}}_{\boldsymbol{i}}^{\boldsymbol{t}}\right), i = 1,2, \dots n , \tag{22}$$

得到问题感知的段落向量$\overline{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}}$和问题感知的段落矩阵$\overline{\boldsymbol{P}}^{\boldsymbol{t}}$，作为自对齐的输入。

3.3.2.3 自对齐

自对齐是为了让交互对齐的向量进一步综合上下文的信息。虽然 RNN 可以做到这一点，但是 RNN 更关注的是短距离的信息，所以需要使用自对齐来综合整个段落的信息，实现推理的功能。自对齐和交互对齐类似，不同的地方是在计算$\overline{\boldsymbol{P}}^{\boldsymbol{t}}$和$\overline{\boldsymbol{P}}^{\boldsymbol{t}}$的 Attention Matrix，并且将矩阵的对角线置为 0，更好地对齐到除去该词以外的文本信息。

$$\widetilde{\boldsymbol{A}}^{\boldsymbol{t}} = \text{softmax}\left(\text{mask\_diag}\left(\overline{\boldsymbol{P}}^{\boldsymbol{t}}\overline{\boldsymbol{P}}^{\boldsymbol{t}^{\mathbf{T}}}\right)\right), \tag{23}$$

$$\widetilde{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}} = \widetilde{\boldsymbol{A}}_{\boldsymbol{i},:}^{\boldsymbol{t}}\overline{\boldsymbol{P}}^{\boldsymbol{t}}, i = 1,2,\dots n, \tag{24}$$

$$\widehat{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}} = \text{SFU}(\overline{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}}, \widetilde{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}}, \overline{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}} \odot \widetilde{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}}, \overline{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}} - \widetilde{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}}), i = 1,2,\dots n, \tag{25}$$

得到自我感知的段落向量$\widehat{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}}$和自我感知的段落矩阵$\widehat{\boldsymbol{P}}^{\boldsymbol{t}}$，作为聚集的输入。

3.3.2.4 聚集

R-Net 在 2017 年 5 月份的论文[31]中提到，将自对齐的向量使用 Gated-GRU 能分别在 EM 和 F1 分数上提升 1 个点，再一次证明了 RNN 强大的学习能力。在本文中，使用 BiLSTM 再一次对向量进行处理，得到全感知的段落向量和矩阵：

$$\breve{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}} = \text{BiLSTM}(\breve{\boldsymbol{p}}_{\boldsymbol{i-1}}^{\boldsymbol{t}}, \widehat{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{t}}), i = 0,1,2,\dots n, \tag{26}$$

向量和矩阵会作为下一次对齐的输入，或者预测网络的输入。

3.3.3 随机预测网络

目前许多模型使用 Wang S 和 Jiang J[22]的答案边界指针 (Answer Boundary Pointer) ，解决了之前机器理解模型无法精确定位答案的困难。首先将预测网络前一层输出的最后一个位置的向量作为输入，利用带 ReLU 激活函数的前馈神经网络 (Feed Forward Network, FFN)，得到答案开始位置的预测分数，然后使用 softmax 激活函数进行归一化，得到开始位置的概率。然后利用 LSTM 对段落向量处理，得到最后的隐向量状态，作为另外一个带 ReLU 激活函数的 FFN 的输入，再使用 softmax 进行归一化，得到结束为止的概率。选择两个概率乘积最大的边界作为最后结果。SAN[26]通过更改预测开始位置的输入向量，重复得到多组

概率，使用随机丢弃计算平均概率，最终计算得到最后的答案。

Chen D[6]提出了另外一个计算答案边界概率的方法，简单高效。利用双线性项计算相似度，然后 softmax 得到开始位置的概率和结束位置的概率。

本文综合两种方法，并且利用随机丢弃增强模型的鲁棒性。

3.3.3.1 答案边界指针

为了更好预测答案，首先利用 FFN 得到初始化的开始位置内存变量$\boldsymbol{z_s^1}$：

$$\boldsymbol{B_i} = \mathrm{FFN}(\boldsymbol{q_i}) = \boldsymbol{W_4}\big(\mathrm{ReLU}(\boldsymbol{W_3 q_i} + \boldsymbol{b_3})\big) + \boldsymbol{b_4}, i = 0,1,2,\dots m\,, \tag{27}$$

$$\boldsymbol{Z_s^1} = \boldsymbol{B^{\mathrm{T}} Q}\,, \tag{28}$$

其中$\boldsymbol{A} \in \mathbb{R}^{m\times 1}$，是$\boldsymbol{Z_s^1} \in \mathbb{R}^{1\times 2h}$。第$l$次进行边界预测时，利用开始位置内存变量$\boldsymbol{z_s^l}$，另外一个 FFN 和 softmax，得到第$l$次的开始位置的概率：

$$s_i^l = \mathrm{FFN}\big([\breve{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{T}}; \boldsymbol{z_s^1}; \breve{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{T}} \odot \boldsymbol{z_s^1}]\big), i = 0,1,2,\dots n\,, \tag{29}$$

$$p_s^l = \mathrm{softmax}(s^l)\,, \tag{30}$$

其中$\boldsymbol{z_s^1} = \boldsymbol{Z_{s\,0:}^1} \in \mathbb{R}^{2h}$。

在3.3.2 中已经介绍了 SFU，使用 SFU 而不是 LSTM 来构造结束位置 FFN 所需要的结束位置内存变量：

$$\boldsymbol{u_s^l} = \boldsymbol{p_s^l \breve{P}^T}\,, \tag{31}$$

$$z_e^l = \mathrm{SFU}(z_s^l, u_s^l)\,, \tag{32}$$

其中$\boldsymbol{p_s^l}$是将p_s^l变为$\mathbb{R}^{n\times 1}$得到，u_s^l是$\boldsymbol{u_s^l}$降维得到。然后利用另外一个 FFN 和 softmax，得到第l次的结束位置的概率：

$$e_i^l = \mathrm{FFN}\big([\breve{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{T}}; \boldsymbol{z_e^l}; \breve{\boldsymbol{p}}_{\boldsymbol{i}}^{\boldsymbol{T}} \odot \boldsymbol{z_e^l}]\big), i = 0,1,2,\dots n\,, \tag{33}$$

$$p_e^l = \mathrm{softmax}(e^l)\,, \tag{34}$$

其中$\boldsymbol{z_{e_i}^l} \in \mathbb{R}^{1\times 2h}$是通过$z_{e_i}^l$重复$2h$次构造得到。为了可以多次循环这个算法，利用另外一个 SFU 构造第$l+1$次开始位置内存变量：

$$\boldsymbol{u_e^l} = \boldsymbol{p_e^l \breve{P}^T}\,, \tag{35}$$

$$z_s^{l+1} = \mathrm{SFU}(z_e^l, u_e^l)\,. \tag{36}$$

3.3.3.2 双线性项

使用双线性项来计算开始位置的概率和结束位置的概率就更加简单，只需要

满足：

$$p_s^0 = \text{softmax}\left(\tilde{\boldsymbol{P}}^{\boldsymbol{T}}\boldsymbol{W}_{\boldsymbol{s}}\boldsymbol{Q}\right), \tag{37}$$

$$p_e^0 = \text{softmax}\left(\tilde{\boldsymbol{P}}^{\boldsymbol{T}}\boldsymbol{W}_{\boldsymbol{e}}\boldsymbol{Q}\right). \tag{38}$$

为了3.3.3.3 的表述方便，使用p_s^0和p_e^0来表示开始位置的概率和结束位置的概率。

3.3.3.3 随机丢弃

SAN[26]采用对开始位置概率和结束位置概率求平均和随机丢弃的方法，使模型更快收敛，同时也增加了模型的鲁棒性：

$$p_s = \text{avg}(\text{dropout}(p_s^0), \text{dropout}(p_s^1), \ldots \text{dropout}(p_s^L)), \tag{39}$$

$$p_e = \text{avg}(\text{dropout}(p_e^0), \text{dropout}(p_e^1), \ldots \text{dropout}(p_e^L)), \tag{40}$$

即在计算p_s和p_s时，随机丢掉若干个p_s^l和p_e^l。

经过分析之后，发现这样并非最佳的方法。原因是p_s^l和p_e^l是经过 softmax 之后的概率，概率都在[0,1]区间。但是如果其中一个概率出现较大误差，会对整个最终的概率产生很大的影响。所以在本模型中，随机丢弃只作用在$s^1, \ldots s^L$和$e^1, \ldots e^L$部分，保证s^0和e^0一定起作用，然后计算平均，利用 softmax 求出最后的概率：

$$p_s = \text{softmax}\left(\text{avg}\left(s^0, \text{dropout}(s^1), \ldots \text{dropout}(s^L)\right)\right), \tag{41}$$

$$p_e = \text{softmax}\left(\text{avg}\left(e^0, \text{dropout}(e^1), \ldots \text{dropout}(e^L)\right)\right). \tag{42}$$

最终根据$p_s \times p_e$选出概率最大的答案区间，得到答案A。

3.3.4 目标函数

和大部分机器理解模型相同，训练目标是最小化真正开始和结束位置的负对数概率的总和，即：

$$J_{\text{MLE}}(\theta) = -\sum_{i=1}^{N}\left[\log\left(p_s^L(y_i^s)\right) + \log\left(p_e^L(y_i^e)\right)\right], \tag{43}$$

其中y_i^s和y_i^e是第i个训练样本的真正开始位置和结束位置，N是训练数据集的大小。

3.4 答案排名算法 Answer Ranking Algorithm

对于开放域的问答系统而言，通过机器理解模型得到答案并不是真正的结束。

因为给定问题Q，通过信息检索会得到段落集合P_Q和段落对应答案集合A_{QP}。答案集合集合很难直接比较给出最终答案，主要有 2 个问题：

1. 答案在某个段落中的概率不能代表答案在段落集合的概率。答案的概率指的是通过 softmax 之后得到，这个概率只能体现其在该段落的可能性高低。
2. 答案答案集合A_P中出现的次数也不能完全代表其为问题正确答案。因为对于一些相似但具体的问题，例如Q_1 =“2008 年夏季奥运会在哪里举办”和Q_2 =“2012 年夏季奥运会在哪里举办”，问题只有时间不同，信息检索得到的P_{Q_1}和P_{Q_2}极为相似，最终得到的答案集合A_{Q_1P}和A_{Q_2P}大部分都相同，而且这些答案都是错误答案。

理想的解决办法是将信息检索模型、机器理解模型和答案排名模型一起训练，但是实际中并不可行，因为信息检索占据了大部分的时间，最终会使训练时间上千倍甚至万倍时间增长。所以本文利用信息检索返回的 TF-IDF 分数、机器理解模型得到的分数——对应于(41)和(42)中 softmax 的指数部分，和答案出现的次数，设计了答案排名算法。算法如下：

1. 计算每个答案的答案分数和维基百科文章 TF-IDF 分数的乘积的和：

$$overall\_score(A) = \sum[\text{answer\_score}(A) * \text{TF\_IDF}(D)] . \quad (43)$$

2. 每个答案最终得分既包括答案分数与所在文章 TF-IDF 的乘积，也包括该答案的累计分数和：

$$final\_score_A = \text{answer\_score}(A) * \text{TF\_IDF}(D) + overall\_score(A) . \quad (44)$$

在(43)式中使用TF_IDF(D)作为权重，使那些与问题相似的文章中的候选答案分数更高；使用机器理解分数作为权重，使那些更可能是答案的候选分数更高；计算答案的累积和，考虑了答案出现的次数，但是经过了加权处理，比单纯统计次数更加合理。

3.5 参数设置

在信息检索模型 Document Retriever 中，设置哈希表的大小为 24bits；每个问题返回 5 篇最相关的文章。在机器理解模型 Stochastic Mnemonic Reader 中使用 840B Web 预训练的 300 维的 Glove 词向量和 300 维的 Glove 字符向量，区分大

小写，并且保证其在训练的过程中固定不变；处理字符向量的 BiLSTM 的隐藏层大小为 50，其它所有隐藏层大小为 100；训练时的 batch 大小为 45；dropout 的概率为 20%；交互和预测的次数均为 2 次；使用 Adamax[32]进行网络的学习和优化。答案排名算法对每个问题只返回 1 个答案。

4. 实验与分析

由于本系统由信息检索、机器理解和答案排名算法三部分组成，所以实验共分为 3 个部分，分别去评估检索系统、理解系统和整体系统的优劣。在实验之前，首先介绍实验用到的数据集。

4.1 数据集

4.1.1 维基百科

由于英文的自然语言处理工具和资源更加丰富，同时研究工作也更加丰富。Document Retriever 使用 Chen D 的检索模型[6]，利用 2016 年 12 月 21 日的英文维基百科作为信息检索的文本[1]。由于 WRMCQA 是基于文本的问答系统，所以只将维基百科文章中的文本进行了抽取。经过消歧义、索引等操作，最终剩下 5075182 篇文章和 9008962 个词条。

4.1.2 SQuAD

2016 年斯坦福大学公开了基于众包的机器理解数据集 SQuAD[5]。该数据集包含了 10 万个从维基百科中构造的例子。每个例子包括一个从维基百科中抽取的段落、一个人工构造的与段落相关的问题和一个能从问题中找到的答案。该数据集公开后举办了机器阅读理解比赛，使用 EM 和 F1 分数去评估模型。由于 Stochastic Mnemonic Reader 使用该数据集训练，并且使用该数据集去评估机器理解模型。同时利用该数据集中的问题和答案去评估整个问答系统，将数据集中的问题当作系统的输入，经过检索、理解、排名得到答案，与数据集中提供的答案进行比较，使用 EM 来衡量模型的好坏。

4.1.3 WikiMovies

2016 年 Millier 等人[20]公开了含有约 10 万条问题对的关于电影的数据集 WikiMovies。该数据集根据 OMDb 和 MovieLens 两个数据库进行构造，可以利

[1] https://dumps.wikimedia.org/enwiki/latest

表 1 SQuAD、WikiMovies、CuratedTrec 中问题答案对的例子

数据集	例子
SQuAD	Q: How many halls are at Notre Dame that house students? A: 29
WikiMovies	Q: What year was the film The Godfather released? A: 1972
CuratedTrec	Q: Who played Frodo in Lord of The Rings? A: Wood

表 2 信息检索模型评估结果。答案出现在检索返回的 top5 的文章中的百分比。

数据集	维基百科搜索引擎	Document Retriever (unigram)	Document Retriever (bigram)
SQuAD	62.7	76.1	**77.8**
CuratedTrec	81.0	85.2	**86.0**
WikiMovies	61.7	54.4	**70.3**

用知识库中回答答案。作者利用 Key-Value Memory Network 构造了性能最优的基于知识库的问答系统。文本同样使用该数据集去评估 WRMCQA，与 Key-Value Memory Network 进行对比。

4.1.4 CuratedTrec

2015 年 Baudiš P 等人[18]从 TREC1999，2000，2001，2002 中抽取了 2180 个问题，创建了 TREC QA 任务。使用该数据集的目的一方面是测试 WRMCQA 在其他数据集的性能，另一方面也和基于文本和知识库的 YodaQA 进行对比。

4.2 信息检索模型评估

在基于文本的问答系统中，信息检索的质量至关重要，所以 Document Retriever 需要单独评估。Document Retriever 所使用的文本集是维基百科，对比

的检索模型同样要求能够检索维基百科。使用维基百科自带搜索引擎[2]作为对比模型。

通过输入问题，判断答案是否出现在检索返回的 top 5 的文章中。结果发现，Document Retriever 的结果远远优于维基百科自带的搜索引擎。同时在使用 TF-IDF 的信息检索中，使用二元 (bigram) 可以比一元 (unigram) 至少再提高 1 个百分点。详细数据列在

表 2 中，数据来源于 Chen D[6]的 Table 3。同时 Chen D 指出，他们利用 Okapi BM25 或者利用 bag-of-embeddings 的余弦相似性进行文章的检索，仍然不如 bigram hashing 和 TF-IDF 的组合。

4.3 机器理解模型评估

机器理解模型是本问答系统的核心，本部分将 Stochastic Mnemonic Reader 与 2017 年 12 月前的公开论文模型进行对比，包括微软亚洲研究院的 R-Net[25][31]，新加坡管理大学的 Mnemonic Reader[8]和 Mnemonic Reader + Reinforcement Learning[8]，Facebook 人工智能研究院的 Document Reader[6]以及微软商业人工智能解决方案小组的 SAN[26]。这 4 个模型都曾占据 SQuAD 榜单第一名，其中 SAN 使用了 Contextualized Word Vectors (cove)[27]，使模型结果大幅提升，和其他模型实验条件不同，不具有很强的可比较性。为了更好地分析模型，本人依照模型的论文以及其它相关论文复现了 R-Net、Mnemonic Reader，将复现的模型同开源的 Document Reader 以及本系统中的 Stochastic Mnemonic Reader 进行对比。

对比结果如表 3 所示，可以发现，除了使用了其它预训练的词向量的 SAN，Stochastic Mnemonic Reader 对比比其它的模型，无论在 EM 还是在 F1 分数上，都具有明显的优势。由于复现时并非完全按照论文，分数上会有损失或者增加，但是都没有刻意压低模型分数。

为了更好地比较和分析 Document Reader、R-Net、Mnemonic Reader 和 Stochastic Mnemonic Reader 四个机器理解模型，将其训练过程中 EM 和 F1 分数变化情况绘制成折线图，如图 2 所示。可以发现，R-Net 在训练刚开始的时候 EM 和 F1 分数均为最低，随着迭代次数的增加，EM 和 F1 分数很快上升，逐渐接近 Document Reader 并且最终超过。其中 R-Net 和 Document Reader 的编码部分都使用 3 层 BiLSTM，但是 R-Net 的交互部分、预测部分更加复杂，可能这是 R-

[2] https://www.mediawiki.org/wiki/API:Search

Net 初期处于劣势的原因，但随着训练次数的增加，模型具有更好的性能。Mnemonic Reader 和 Stochastic Mnemonic Reader 的趋势相似，但是大约 10Epochs 之后 Stochastic Mnemonic Reader 在 EM 和 F1 分数上均优于 Mnemonic Reader，最终差距大约有 0.7%，所以 SFU 可以推广到更多的模型当中进行对齐，并且 Stochastic Mnemonic Reader 的预测网络中的 dropout 并不会影响模型的预测性能，反而增强了模型的鲁棒性。

表 3 机器理解模型在 SQuAD 的测试结果

模型	验证集 (Dev)		测试集 (Test)	
	EM	F1	EM	F1
Document Reader (论文[6])	69.5	78.8	70.0	79.0
Document Reader (开源模型[3])	69.3	78.6	n/a	n/a
R-Net (论文[25])	71.1	79.5	71.3	79.7
R-Net (论文[31])	72.3	80.6	72.3	80.7
R-Net (复现模型)	70.2	79.2	n/a	n/a
Mnemonic Reader (论文[8])	71.8	81.2	n/a	n/a
Mnemonic Reader + RL (论文[8])	72.1	81.6	73.2	81.8
Mnemonic Reader (复现模型)	72.5	81.1	n/a	n/a
SAN (论文[26])	***76.2***	***84.1***	***76.8***	***84.4***
Stochastic Mnemonic Reader	**73.2**	**81.8**	n/a	n/a

[3] https://github.com/facebookresearch/DrQA/tree/master/drqa/reader

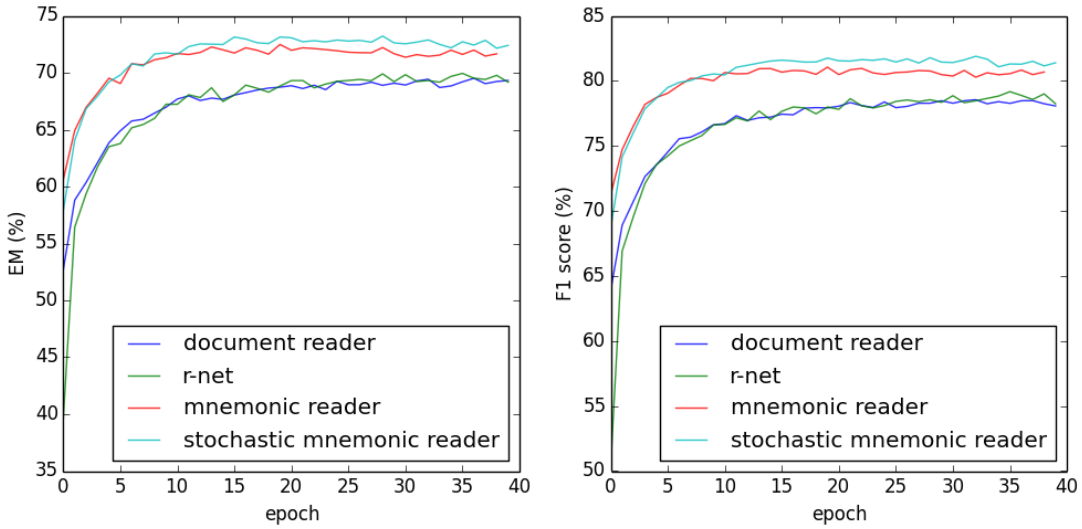

图 2 EM 和 F1 随时间变化

4.4 问答系统评估

最后本文分别利用 SQuAD、CuratedTrec、WikiMovies 对 WRMCQA 进行开放域问答的评估，这时不利用数据集中提供的段落，而是从整个维基百科中进行检索。为了更好评估问答系统，选择基于文本的 DrQA[6]、基于知识库的 Key-Value Memory Network[20]和基于文本和数据库的 YodaQA[14]进行对比。在进行整个问答系统的评估时，为了时间效率和模型鲁棒性上的考虑，不使用词性特征 (对应公式(5)(6))、命名实体特征 (对应公式(7)(8))、词频特征 (对应公式(9)(10))。此外，WikiMovies 提供了候选答案，所以 WRMCQA 在预测时只会选择候选答案之一。由于 CuratedTrec 和 WikiMovies 没有包括相关的段落供 Stochastic Mnemonic Reader 训练，所以需要使用一些其他方法来关联段落。4.4.1 会介绍远程监督 (Distant Supervision, DS)[33]的方法达到此目的。

4.4.1 远程监督

远程监督 (Distant Supervision, DS) 最初提出时被用于信息抽取 (IE)，用于在文本中进行关系抽取。DS 基于这样的思想：如果一个句子中包含一对知识库中的命名实体，那么很有可能这个句子表示着知识库中的与实体相关联的关系。那么问题就变成了多分类问题，通过句子中出现的一对实体和知识库找到可能的分类，利用文本信息以及提取的文本信息作为特征，使用逻辑回归确定分类情况。在本实验中，利用 DS 的思想，根据问题的答案和从 Document Retriever 检索的

表 4 DS 关联段落情况

数据集	DS 关联段落数目 / 训练集“问题-答案”对数目
SQuAD	80348 / 87599
CuratedTrec	3575 / 1486
WikiMovies	98527 / 96185

结果关联少量最可能的段落。具体操作如下：

1. 对训练集中的问题使用 Document Retriever 检索得到 top5 的文章，
2. 如果问题中的词和段落中的词没有一个匹配，筛选掉该段落，
3. 筛选掉短于 25 或者长于 1500 个词的段落，
4. 如果问题中包含命名实体，那么筛选掉未出现该实体的段落，
5. 依次按照匹配程度、文章 TF-IDF 分数选择最可能的 5 个 (或少于 5 个) 段落作为关联段落，
6. 将问题、关联段落、答案进行组合加入到数据集的训练集中。

最终通过 DS 关联段落的结果如表 4 所示。

有三种利用关联段落的方法，第一种是分别利用不同数据集去学习 3 个不同的模型，然后分别在 3 个数据集的测试集上评估；第二种是先利用 SQuAD 学习的到一个模型，然后利用 SQuAD、CuratedTrec 和 WikiMovies 的所有关联段落去微调；第三种是直接利用 SQuAD、CuratedTrec 和 WikiMovies 的所有关联段落一起学习，得到一个模型。由于 DS 的关联结果不能保证完全的正确，所以第一种方法在测试集上会导致准确度低，并且每个数据集都需要学习导致难以扩展；第 2 种方法在训练好的模型上进行微调，在保证机器理解部分准确度稳定的情况下扩展模型到其它的数据集；第 3 种方法可以保证训练的随机性，但是每次增加评估的数据集都需要重新训练模型。所以本文将分别使用第 2 种方法和第 3 种方法去评估模型，使结果更加全面。

4.4.2 评估结果

评估结果如表 5。可以发现，基于知识库的 Key-Value Memory Network 在 WikiMovies 上的结果远远好于其他模型在该数据集上的结果，说明了知识库在

问答系统中具有无可替代的优势。但是由于知识库自身的限制，导致该类问答系统很难扩展到其他问题类型上。基于文本和知识库的问答系统的出现保证了答案的准确性，同时扩展了问答系统，这也在 CuratedTrec 数据集上得到了验证，但是可以发现，YodaQA 并没有像 Key-Value Memory Network 效果那么拔群，说明了使用更大更全的知识库和文本可以解决基于知识库的问答系统的限制性，但是带来了性能上的损失。DrQA 和 WRMCQA 都是基于文本的问答系统，所以这两个系统的比较更能反映出 WRMCQA 的优势。首先可以看到，利用 DS 关联段落能够大幅提高系统的准确性，但是有趣的是，DrQA 在 DS 方法 3 效果好于+DS 方法 2，但是 WRMCQA 在+DS 方法 2 上在 SQuAD 和 CuratedTrec 上表现较好。其次，WRMCQA 模型在不使用 Answer Ranking Algorithm (ALA)的情况下，准确性和 DrQA 模型相比没有明显提升的趋势，这也证实引言的分析：不同段落中的答案分数不具有比较性，针对段落的机器理解模型准确性的提升并不能带来问答系统的准确性提升。使用 DS 关联的段落进行训练，WRMCQA 的提升明显大于 DrQA，说明了 WRMCQA 具有更好的学习和推理能力，但是需要更多的数据来学习。在引入 ALA 之后，WRMCQA 在 SQuAD 数据集上平均提升了 1.37%，在 CuratedTrec 数据集上平均提升了 3.89%，在 WikiMovies 数据集上平均提升了 1.22%，证明了 ALA 的有效性，ALA 让机器理解模型不局限于一个小的段落，为构造大规模机器理解模型提供了方向。最终 WRMCQA (DS 方法 2+ALA) 在 SQuAD 数据集上比 DrQA (DS 方法 3) 高出 1.57%，WRMCQA (DS 方法 2+ALA) 在 CuratedTrec 数据集上比 DrQA (DS 方法 3) 高出 2.17%，WRMCQA (DS 方法 3+ALA)在 WikiMovies 数据集上比 DrQA (DS 方法 3) 高出 0.24%。图 3 可以更加直观地得到上述结论。作者相信，WRMCQA 仍然有进一步提升的空间。

表 5 不同模型在 SQuAD 验证集、CuratedTrec 测试集、WikiMovies 测试集上的 EM

模型		数据集		
		SQuAD	CuratedTrec	WikiMovies
Key-Value Memory Network (论文[20])		n/a	n/a	***93.9***
YodaQA (Benchmarks[4])		n/a	***31.3***	n/a
DrQA	SQuAD 单模型 (开源模型[5])	27.05	20.17	23.28
	+DS 方法 2	28.82	22.19	35.26
	DS 方法 3	29.11	24.92	36.43
WRMCQA	SQuAD 单模型	19.56	15.99	19.54
	+ALA	22.02	19.88	21.00
	+DS 方法 2	30.15	22.77	34.54
	+DS 方法 2+ALA	**30.68**	**27.09**	35.78
	DS 方法 3	26.66	22.62	35.71
	DS 方法 3+ALA	27.79	26.08	**36.67**

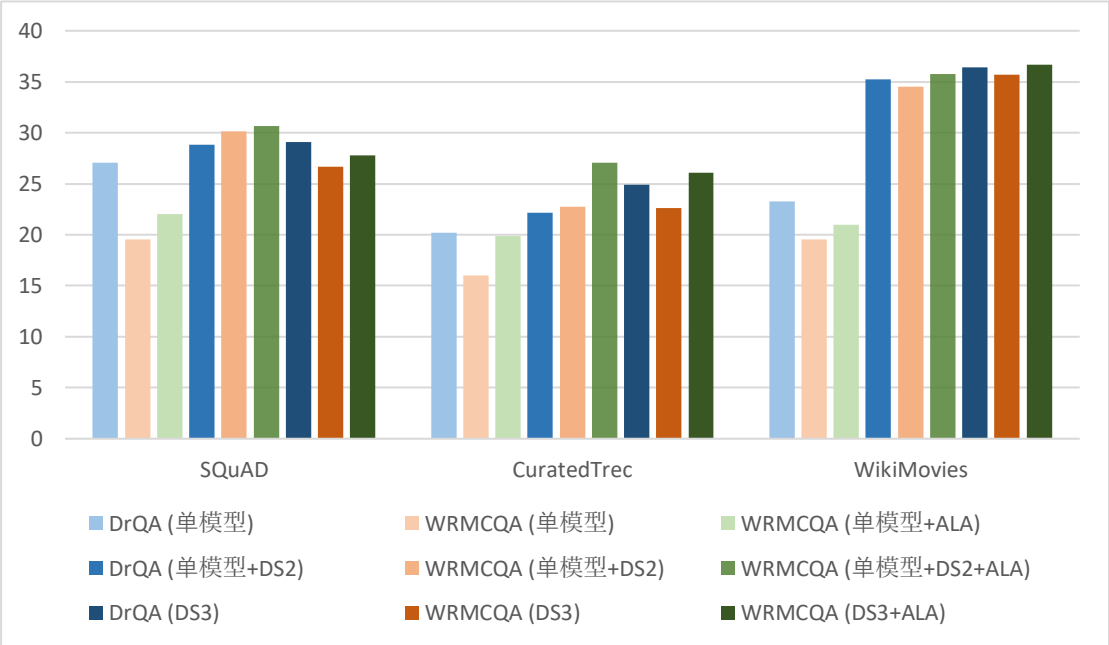

图 3 基于文本的问答系统 EM 的柱形图

[4] https://github.com/brmson/yodaqa/wiki/Benchmarks

[5] https://github.com/facebookresearch/DrQA

5. 结语

深度学习使机器理解模型有了质的飞跃，基于文本的问答系统随着机器理解模型的进步也迎来了小幅度的提升。本文使用 BiLSTM、Maxout、Dot-Product Attention、SFU、指针网络、双线性项和随机丢弃构造了机器理解模型 Stochastic Mnemonic Reader，并且在 SQuAD 数据集 EM 和 F1 分数上超过了已公开发表论文的单模型 (截止到 2017 年 12 月)。提出利用 TF-IDF 分数、机器理解的候选答案次数和候选答案分数的答案排名算法 Answer Ranking Algorithm，确定最终答案。将信息检索模型 Document Reader、机器理解模型 Stochastic Mnemonic Reader 和答案排名算法 Answer Ranking Algorithm 结合，构造了问答系统 WRMCQA，能够通过检索、理解、排名三个步骤返回问题的答案，既有基于文本问答系统的优势，又进一步提升系统的鲁棒性和准确性。

在同其他模型对比中发现，基于文本的问答系统仍然与基于知识库的问答系统和基于文本和知识库的问答系统有着不小的差距。虽然 Key-Value Memory Network 在 WikiMovies 数据集上表现拔群，但是该模型不能很好地扩展到其他领域的问题。基于文本和知识库的问答系统是两者的折中，相比基于文本的系统能进一步带来提升。同 DrQA 相比，WRMCQA 具有更好的学习能力和推理能力，但是代价是需要更多的数据。

在 WRMCQA 中，由于检索、理解和排名三步骤是分开训练的，所以导致在整个系统进行评估时并没有达到特别理想的准确性。从表 2 中可以看到，检索步骤可以达到 70%甚至 80%的命中率，但是最终准确率在 20%～30%之间，仍然有很大提升空间。一个方法是利用负采样使正确答案分数更高，另一个方法是将理解和排名算法联合训练，利用神经网络强大的学习能力去确定答案排名的参数。

未来也考虑引入大规模的知识库，例如 Freebase，来进一步提升系统在确定答案时的准确性。例如在机器理解部分将实体和关系一起进行编码，或者在答案排名时考虑实体和关系引入额外权重。

参考文献：

[1] Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J. Freebase: a collaboratively created graph database for structuring human knowledge[C]. SIGMOD, 2008: 1247–1250.

[2] Suchanek F M, Kasneci G, Weikum G. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia[C]. WWW, 2007: 697-706.

[3] Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives Z. DBpedia: A Nucleus for a Web of Open Data[C]. The Semantic Web, 2007: 722–735.

[4] Nguyen T, Rosenberg M, Song X, Gao J, Tiwary S, Majumder R, Deng L. MS MARCO: A human generated machine reading comprehension dataset[OL]. CoRR, abs/1611.09268, 2016.

[5] Rajpurkar P, Zhang J, Lopyrev K, Liang P. SQuAD: 100,000+ Questions for Machine Comprehension of Text[C]. EMNLP, 2016: 2383–2392.

[6] Chen D, Fisch A, Weston J, Bordes A. Reading Wikipedia to Answer Open-Domain Questions[C]. ACL, 2017: 1870–1879.

[7] Weinberger K, Dasgupta A, Langford J, Smola A, Attenberg J. Feature hashing for large scale multitask learning[C]. ICML, 2009: 1113–1120.

[8] Hu M, Peng Y, Qiu X. Reinforced Mnemonic Reader for Machine Comprehension[OL]. CoRR, abs/1705.02798, 2017.

[9] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention Is All You Need[C]. NIPS, 2017: 6000-6010.

[10] Banko M, Brill E, Dumais S, Lin J. Askmsr: Question answering using the worldwide web[C]. AAAI, 2002: 7-9.

[11] Berant J, Chou A, Frostig R, Liang P. Semantic parsing on freebase from question-answer pairs[C]. EMNLP, 2013: 1533-1544.

[12] Fader A, Zettlemoyer L, Etzioni O. Open question answering over curated and extracted knowledge bases[C]. SIGKDD, 2014: 1156-1165.

[13] Ferrucci D, Brown E, Chu-Carroll J, Fan J, Gondek D, Kalyanpur AA, Lally A, Murdock JW, Nyberg E, Prager J, Schlaefer N. Building Watson: An overview of the DeepQA project[J]. AI magazine, 2010 Jul 28, 31(3): 59-79.

[14] Baudiš P. YodaQA: a modular question answering system pipeline[C]. International Student Conference on Electrical Engineering, 2015 Sep 8: 1156-1165.

[15] Ahn D, Jijkoun V, Mishne G, Müller K, de Rijke M, Schlobach S, Voorhees M, Buckland L. Using Wikipedia at the TREC QA Track[OL]. In proceedings of TREC, 2004.

[16] Ryu PM, Jang MG, Kim HK. Open domain question answering using Wikipedia-based knowledge model[J].

Information Processing & Management, 2014 Sep 1, 50(5): 683-692.

[17] Andreas J, Rohrbach M, Darrell T, Klein D. Learning to compose neural networks for question answering[OL]. CoRR abs/1601.01705, 2016.

[18] Baudiš P, Šedivý J. Modeling of the question answering task in the yodaqa system[C]. CLEFm, 2015: 222-228.

[19] Berant J, Chou A, Frostig R, Liang P. Semantic parsing on freebase from question-answer pairs[C]. EMNLP, 2013: 1533–1544.

[20] Miller A, Fisch A, Dodge J, Karimi AH, Bordes A, Weston J. Key-Value Memory Networks for Directly Reading Documents[C]. EMNLP, 2016: 1400-1409.

[21] Hermann KM, Kocisky T, Grefenstette E, Espeholt L, Kay W, Suleyman M, Blunsom P. Teaching machines to read and comprehend[C]. NIPS, 2015: 1693-1701.

[22] Wang S, Jiang J. Machine comprehension using match-lstm and answer pointer[OL]. CoRR, abs/1608.07905, 2016.

[23] Vinyals O, Fortunato M, Jaitly N. Pointer networks[C]. NIPS, 2015: 2692-2700.

[24] Xiong C, Zhong V, Socher R. Dynamic coattention networks for question answering[OL]. CoRR abs/1611.01604, 2016.

[25] Wang W, Yang N, Wei F, Chang B, Zhou M. Gated self-matching networks for reading comprehension and question answering[C]. ACL, 2017: 189-198.

[26] Liu X, Shen Y, Duh K, Gao J. Stochastic Answer Networks for Machine Reading Comprehension[OL]. CoRR abs/1712.03556, 2017.

[27] McCann B, Bradbury J, Xiong C, Socher R. Learned in translation: Contextualized word vectors[C]. NIPS, 2017: 6297-6308.

[28] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space[OL]. CoRR abs/1301.3781, 2013.

[29] Pennington J, Socher R, Manning C. Glove: Global vectors for word representation[C]. EMNLP, 2014: 1532-1543.

[30] Goodfellow IJ, Warde-Farley D, Mirza M, Courville A, Bengio Y. Maxout networks[OL]. CoRR abs/1302.4389, 2013.

[31] Natural Language Computing Group, Microsoft Research Asia. R-Net: machine reading comprehension with self-matching networks[OL]. https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf , 2017.

[32] Kingma D, Ba J. Adam: A method for stochastic optimization[OL]. CoRR abs/1412.6980, 2014.

[33] Mintz M, Bills S, Snow R, Jurafsky D. Distant supervision for relation extraction without labeled data[C]. ACL, 2009: 1003-1011.

致谢

感谢从小到大父母对自己的照顾和帮助，让我养成了独立的习惯，同时也培养了合作的意识。

感谢从小学到本科期间所有老师无私传授知识，让我有幸进入中山大学计算机系学习，最后掌握了该学科中浅层但是却可以受益终身的知识和技能。

感谢中山大学的导师潘嵘老师，在加入实验室之后不断提供学习资源和路线规划，让我了解到信息检索、数据挖掘、自然语言处理的基础知识和最新动态，让我对自然语言处理充满了好奇和继续学习的动力，也帮助我在继续深造的道路上提供的帮助。

感谢自己在微软亚洲研究院实习的导师陈亮，他作为一名科研人员，用实际行动告诉我分析问题和思考问题的方法，以及同他人合作的重要性。前期沟通上的不愉快也让自己不断反思自己与他人沟通的方式。

感谢自己在香港科技大学实习的导师宋阳秋，老师博学的知识为我完成毕业设计以及今后学习提供了巨大的帮助，以后还请多多指教。

感谢自己的室友刘易林、刘渊、刘嘉祺、徐锦涛在至九 316 一起生活期间对自己的包容和帮助，让我这个北方人很快融入了包容的广州。祝愿大家前程似锦。

感谢那些对计算机科学、自然语言处理、信息检索、机器理解、问答系统有突出贡献的研究人员，感谢 pytorch、numpy、matplotlib、scikit-learn、spaCy、DrQA 等众多开源软件的贡献者，没有他们的努力，自己根本不可能设计出这样有趣的问答系统。

附录

附录 A1

WRMCQA 运行界面

```
* WRMCQA *

* Author: Xin Liu, undergraduate of SDCS, SYSU

* Implement based on Facebook's DrQA

>> process(question, candidates=None, top_n=1, n_docs=5)
>> usage()

>>>
>>>
```

图 A1 运行界面

问答展示

```
>>> process('What is Question Answering')
04/12/2018 09:48:31 PM: [ Processing 1 queries... ]
04/12/2018 09:48:31 PM: [ Retrieving top 5 docs... ]
04/12/2018 09:48:31 PM: [ Reading 106 paragraphs... ]
04/12/2018 09:48:31 PM: [ Processed 1 queries in 0.6367 (s) ]
Top Predictions:
+------+-------------------------------------------------------------------------------------
----------------------+------------------+--------------+--------------+-----------+
| Rank |                                                Answer
                      |        Doc       | Overall Score | Answer Score | Doc Score |
+------+-------------------------------------------------------------------------------------
----------------------+------------------+--------------+--------------+-----------+
|  1   | a computer science discipline within the fields of information retrieval and natu
ral language processing | Question answering |     6354.5    |    9.6899    |   327.89  |
+------+-------------------------------------------------------------------------------------
----------------------+------------------+--------------+--------------+-----------+

Contexts:
[ Doc = Question answering ]
Question Answering (QA) is a computer science discipline within the fields of information
retrieval and natural language processing (NLP), which is concerned with building systems
that automatically answer questions posed by humans in a natural language.

>>>
```

图 A2 问答展示 1

```
>>> process('Where is Sun Yat-sen University')
04/12/2018 09:49:02 PM: [ Processing 1 queries... ]
04/12/2018 09:49:02 PM: [ Retrieving top 5 docs... ]
04/12/2018 09:49:02 PM: [ Reading 307 paragraphs... ]
04/12/2018 09:49:03 PM: [ Processed 1 queries in 0.8172 (s) ]
Top Predictions:
+------+-----------+-----------------------+---------------+--------------+-----------+
| Rank |   Answer  |          Doc          | Overall Score | Answer Score | Doc Score |
+------+-----------+-----------------------+---------------+--------------+-----------+
|  1   | Guangzhou | Sun Yat-sen University |     59408     |    29.692    |   806.82  |
+------+-----------+-----------------------+---------------+--------------+-----------+

Contexts:
[ Doc = Sun Yat-sen University ]
In 1924, Dr. Sun Yat-sen founded National Canton University (国立广东大学) and inscribed i
n his own handwriting the school motto of "Study Extensively, Enquire Accurately, Reflect
Carefully, Discriminate Clearly, Practise Earnestly." After the death of Sun Yat-sen, the
national government that was set up during the first cooperation between the Communists an
d Nationalists formally decreed to change its name to Sun Yat-sen University on July 17, 1
926. In 1926, there were five National Sun Yat-sen (Zhongshan) Universities: National Firs
t Sun Yat-sen (Zhongshan) University in Guangzhou (the current Sun Yat-sen University), Na
tional Second Sun Yat-sen (Zhongshan) University in Wuhan (the current Wuhan University),
National Third Sun Yat-sen (Zhongshan) University in Zhejiang (the current Zhejiang Univer
sity), National Fourth Sun Yat-sen (Zhongshan) University in Nanjing (the current Nanjing
University, National Fifth Sun Yat-sen (Zhongshan) University in Zhengzhou (the current He
nan University). In the 1930s, there were seven schools in the university: the Schools of
Arts, Sciences, Law, Engineering, Agricultural Studies, Medicine and Education. In 1935, T
singhua University, Peking University and Sun Yat-sen University set up the first graduate
 schools in China and began to enroll graduate students. In the 1950s, colleges, schools a
nd departments were readjusted nationwide, and Sun Yat-sen University became a comprehensi
ve university with the liberal arts and sciences as its backbone disciplines.

>>>
```

图 A3 问答展示 2

```
>>> process('When was Sun Yat-sen University established')
04/12/2018 09:49:34 PM: [ Processing 1 queries... ]
04/12/2018 09:49:34 PM: [ Retrieving top 5 docs... ]
04/12/2018 09:49:34 PM: [ Reading 307 paragraphs... ]
04/12/2018 09:49:35 PM: [ Processed 1 queries in 0.9427 (s) ]
Top Predictions:
+------+--------+-----------------------+---------------+--------------+-----------+
| Rank | Answer |          Doc          | Overall Score | Answer Score | Doc Score |
+------+--------+-----------------------+---------------+--------------+-----------+
|  1   |  1924  | Sun Yat-sen University |     41957     |    14.688    |   872.91  |
+------+--------+-----------------------+---------------+--------------+-----------+

Contexts:
[ Doc = Sun Yat-sen University ]
Sun Yat-sen University, also known as Zhongshan University, is a public university in Guan
gdong, People's Republic of China. It was founded in 1924 by Sun Yat-sen, a revolutionary
and the founding father of the Republic of China.

>>>
```

图 A4 问答展示 3