第六届“泰迪杯”
数据挖掘挑战赛

优
秀
作
品

作品名称：基于双重注意力机制与Bi-LSTM 的智能阅读系统

荣获奖项：特等并获企业冠名奖

作品单位：山东大学

作品成员：弭晓月 毕冠群 黄成梓

指导老师：刘保东

基于双重注意力机制与 Bi-LSTM 的智能阅读系统

摘要

深度学习与自然语言处理技术发展日新月异，在生活中的应用也越来越广泛。本文构建了一个基于 Bi-LSTM 和注意力机制的智能文本阅读模型。对于用户输入的问题与文档，对于用户输入的问题与文档，模型可以定位并给出答案。

在数据预处理阶段，我们从数据源文件中抽取语句后进行分词和去停用词处理。通过 word2vec 获得初步的词向量表示。为了进一步挖掘答句内与答句间的语义信息，我们将初步词向量放入 Bi-LSTM 中，得到最终词向量表示。

为了凸显出答句整句的语义信息，我们利用词频对属于同一答句的词向量重要程度进行衡量，并进行加权平均，再通过除去数据中 PCA 的第一主成分，即可生成答句的句向量。即 sentence2vec。多次实验测算证明，转化为句向量后，模型的 F1-score 提升了 19.776%，ACC 提升了 19.061%，性能显著提高。

对于一个问句答句组，我们将问句词向量矩阵和答句句向量矩阵进行矩阵乘法，得到初步匹配矩阵。然后，与普通模型仅考虑应用简单的启发式方法（如求和或求平均）来将得出注意力不同，我们使用了一种创新的 attenton-over-attention 机制来衡量问题与候选答案的内容匹配程度。

我们先对矩阵的每一列做 $softmax$ 归一化，从而得到问题到答案的注意力矩阵；再计算反向注意力，即对匹配矩阵的每一行做 $softmax$ 归一化，获得答案到问题的注意力矩阵，并按列求平均，获得一个注意力向量。最后，我们计算正反注意力的点积，这样便得到注意力汇聚向量。利用这种方法，每个问句词组对答案的重要程度可以明确地获知。注意力汇聚向量的分量表达了其对应位置的答句与问句的匹配程度，将分量值作为匹配分数，通过多次实验确定最佳阈值，将匹配分数和阈值进行比较，从而得出结果标签。

实验证明，与应用简单的启发式方法得到注意力相比，我们的方法可以将 F1-socre 提高 22.351%。在实验过程中，我们分别对比了基于传统方法的模型与基于卷积神经网络的模型，我们的模型均优于上述模型。此外，为了测试模型的泛化能力，除题目所给数据集外，我们还在百度的 WebQA 数据集和机器学习保险行业问答开放数据集上进行了性能测试，及以及参数设置进行了详细的分析。在给出的测试集中，F1-score 为 0.854，识别正确率为 81.201%，验证了本文模型的有效性。

关键字：智能阅读　word2vec　Bi-LSTM　sentence2vec　attention-over-attention

Abstract

Deep learning and natural language processing technology are developing rapidly, and their application in life is more and more extensive. This paper builds an intelligent text reading model based on Bi-LSTM and attention mechanism. For questions and documents, the model can be located in the document to help us answer the question.

In the data preprocessing stage, we extract sentences from data source, divided sentences into words, and obtained the initial word vectors with word2vec.To make use of context information, we put initial word vectors into the Bi-LSTM to get the better word vectors. To highlight the sentence semantic information, we used word frequency to measure the importance of each word vector in a sentence, weighted, then remove the first principal component got by PCA ,we get the sentence vector from candidate answers. This method is also called sentence2vec.Measuring by many experiments, the modelś F1-score increase 19.776%, and ACC increase 19.061%.

With given a group of answers and query, we computed a matching score by their dot product and got a match matrix. Different from normal model only considers simple heuristic method, we adopted a creative attention-over-attention mechanism to measure how well the questions match the candidate answers. We applied a column-wise softmax function to get the attention matrix of query to answers as well as a row-wise softmax function to get that of answers to query. Calculating the average of row-wise attention matrix, we got an attention vector of answers to query. We calculate dot product of bi-direction attention and finally got a final attention vector.

We can know the importance of each word in query for choosing the answer exactly with this mechanism. Each component of the final vector represents the match of answer in corresponding position and the query. Regard the value as the matching score, we can determine the threshold an use it to get result labels. Experiments proved that compared with simple heuristic model, our method increase F1-score by 22.351%.

In the experiment, we analyzed how to set parameter in detail.Compared with the model based on the traditional method or convolution neural network. our model is better. In addition, in order to test the generalization ability of the model, we tested our model not only on given data but also on WebQA dataset and Insurance Library dataset. With the given test set,our F1-score is 0.854, and the accuracy is 81.201%, which prove the validity of our model.

key word: intelligent reading system word2vec Bi-LSTM sentence2vec attention-over-attention

目录

一、 简介

1.1 挖掘意义

关于阅读，相信大家都不陌生，在学习方面，我们接受的传统语文教育中阅读理解是非常常规的考试内容；在生活中，我们也常常会遇到阅读文件、论文、书籍等应用场景。除去欣赏优美的语言艺术，更多情况下，我们只是需要从文本中查找某一些片段来解决我们的实际问题。比如，通过查找法律文献中的一些段落来解决我们的法律疑惑，这时并不需要精读整个法律文献；就算对于小说，有时候我们也只是想知道其中一些特殊细节，并不想花时间去通读整个小说。但是，这对人类来说无疑是一个难题。浩如烟海的书籍作为古往今来人类智慧的结晶，其中内容往往很难在短时间内为人掌握，就算是略读也不容易，更不必说从浩繁卷帙中准确的定位答案。因此，我们希望智能阅读技术能够在这方面提供一些帮助。

为了构建智能文本挖掘模型，学界对于机器阅读理解的研究从未止步。机器阅读理解作为目前热门的自然语言处理任务，目标是使机器在能够理解原文的基础上，正确回答与原文相关的问题。提高机器对语言的理解能力。机器阅读理解技术的发展对信息检索、问答系统、机器翻译等自然语言处理研究任务有积极作用，同时也能够直接改善搜索引擎、智能助手等产品的用户体验，因此，以阅读理解、文本挖掘为契机研究机器理解语言的技术，具有重要的研究与应用价值。

1.2 挖掘目标

我们要构建一个智能的文本挖掘模型。模型可以起到辅助阅读的作用，帮助人们显著提高阅读的效率。具体到使用情景上，对于用户输入的问题与文档，模型可以定位到文档中能帮我们回答问题的所在行，或者直接给出明确的某个词作为答案输出。

更形式化地，我们要解决的智能阅读模型的构建问题可以被解释为一个三元组 $<D,Q,A>$。三元组由文档 D，问题 Q 和答案 A 的答案组成。作为最大粒度的输出要求，我们需要把答案 A 定位到文档中的某一句话，为了解决这个问题，需要同时利用问题 Q 中的信息与文档 D 中的全部上下文信息。

1.3 挖掘流程

如图1挖掘主要分为两大部分，预处理部分和候选答案评分部分。其中预处理包括分词，去停用词，将词组向量化（word2vec）。候选答案评分部分为核心步骤，为了进一步挖掘上下文信息，将预处理得到的词向量放入 Bi-LSTM 网络, 为了进一步凸显处答句整句的语义信息，对答句进行 sentence2vec, 最后使用注意力机制对 Bi-LSTM 的解码端进行信息整合。最后对已经有了评分的候选集，设置阈值，输出预测结果。

二、 预处理

2.1 分词

由于中文文本的特点是词与词之间没有明显的界限，从文本中提取词语时需要分词，本文采用 Python 开发的一个中文分词模块——jieba 分词[1]，对问题和回答中的每一句话进行分词进行中文分词。

jieba 分词用到的算法为最短路径匹配算法该算法首先利用词典找到字符串中所有可能的词条，然后构造一个有向无环图。其中，每个词条对应图中的一条有向边，并可利用统计的方法赋予对应的边长一个权值，然后找到从起点到终点的最短路径，该路径上所包含的词条就是该句子的切分结果。

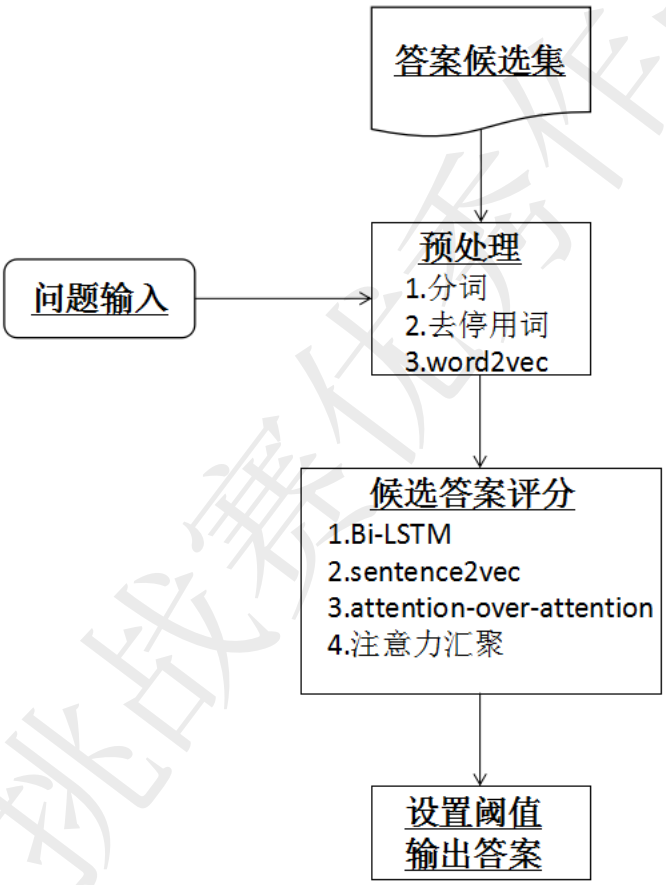

图 1: 挖掘流程

2.2 去停用词

在文本处理中，停用词是指那些功能极其普遍，与其他词相比没有什么实际含义的词，它们通常是一些单字，单字母以及高频的单词，比如中文中的“我、的、了、地、吗”等，英文中的“the、this、an、a、of”等。对于停用词一般在预处理阶段就将其删除，避免对文本，特别是短文本，造成负面影响。本文所用的停用词，取自四川大学机器智能实验室停用词表[2]。

2.3 word2vec

为了将语料输入神经网络进行训练，我们首先要将自然语言符号表示成计算机能够理解的数字形式。

一个自然的想法是把每个词表示为一个很长的向量。这个向量的维度是词表大小，其中绝大多数元素为 0，只有一个维度的值为 1，这个维度就代表了当前的词。这就是独热编码形式（One-Hot）。独热编码虽然方便易懂，但也有显而易见的不足：首先，

One-hot 编码的维数由词典长度而定，过于稀疏，存在降维难问题，给计算造成了很大不便；其次，One-hot 编码下任意两个词之间都是孤立的，丢失了语言中的词义关系。

Word2vec 是 Mikolov 在 2013 年提出的用于快速有效地训练词向量的模型[3]。作者的目标是要从海量的文档数据中学习高质量的词向量, 该词向量在语义和句法上都有很好地表现, 已经广泛应用于自然语言处理的各种任务中。Word2vec 包含了两种训练模型, 分别是 $CBOW$ 和 $Skip\_gram$ 模型, 如图2所示。中 $CBOW$ 模型利用上下文预测当前词, 而 $Skip\_gram$ 模型利用当前词预测其上下文。

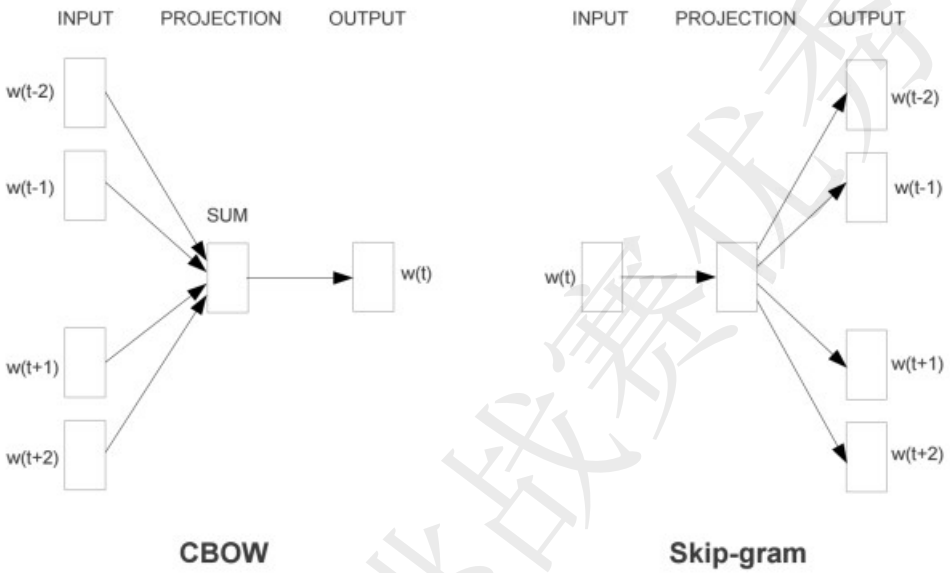

图 2: 两种 word2vec 算法网络示意图

我们使用 $Skip\_gram$ 模型构建智能阅读系统。Skip-gram 模型的训练目标就是使得下式的值最大:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c}\log p(w_{t+j}|w_t)$$

其中，c 是窗口的大小，T 是训练文本的大小。基本的 $Skip\_gram$ 模型计算条件概率如下式:

$$p(w_O|w_I)=\frac{exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W}exp({v'_w}^T v_{w_I})}$$

其中 v_w 和 v'_w 是单词 w 的输入和输出向量，W 是词典的大小。

三、 对回答候选集评分

3.1 Bi-LSTM 层

为了尽可能保持语句中词组之间的上下文联系，我们决定将通过上一步 word2vec 得到的词向量放入循环神经网。同时为了获得尽可能多的上下文记忆信息，我们最终选

择了 Bi-LSTM 这一模型。

3.1.1 RNN 和 LSTM

循环神经网络（Recurrent Neural Network,RNN[5]）近年来由于其良好的性能代替深度神经网络（Deep Neural Network,DNN）成为主流自然语言处理建模方案，相对于 DNN，RNN 在隐层上增加了一个反馈，即 RNN 隐层的输入有一部分是前一级的隐层输出，这使 RNN 能够通过循环反馈看到当前时刻之前的信息，赋予了 RNN 记忆功能，能较好的表征上下文的语义。这些特点使得 RNN 非常适合用于对自然语言进行建模。如图3所示，所有的 RNN 都具有一种重复神经网络模块的链式形式，在标准 RNN 中，这个重复的模块只有一个非常简单的结构，例如一个 tanh 层。

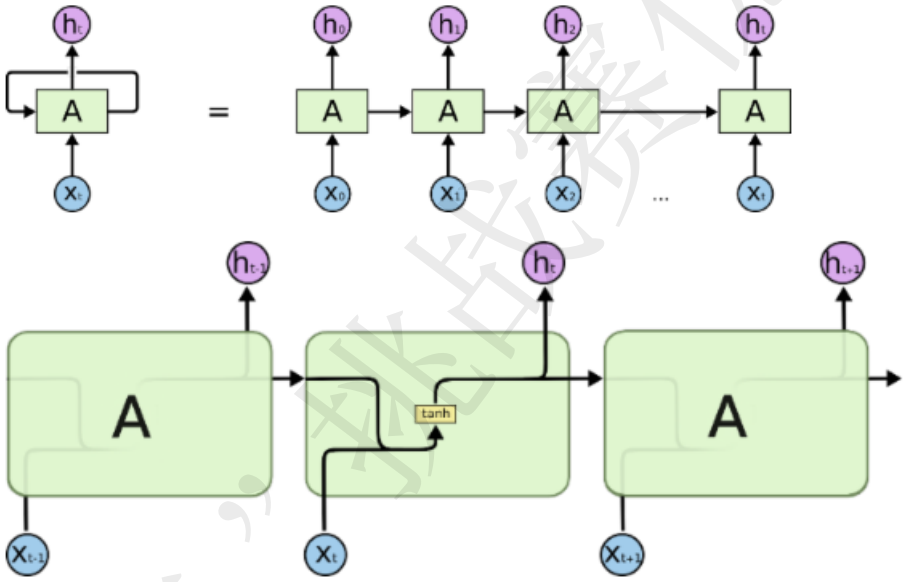

图 3: RNN 结构

RNN 虽然看似简单，却也有一些严重的缺点。首先是过拟合，在现在广泛应用的端到端 RNN 系统中，RNN 对上下文相关性的拟合较强，而这会导致 RNN 相比与 DNN 而言更易出现过拟合问题。其次是梯度消失和梯度爆炸，因为 RNN 比 DNN 更复杂，海量数据环境下的 RNN 模型训练难度较大，容易出现梯度消失和梯度爆炸问题，导致在构建大型系统方面表现较差。第三是 RNN 对较长的问句有严重语义遗忘问题，当相关信息和当前预测位置之间的间隔不断增大时，RNN 有限的记忆能力会因为信息不断增多而耗尽，因此当间隔增大，一部分初始的记忆信息就会被遗忘，这些长期依赖的遗忘和缺失最终会导致问句语义丢失。为了避免以上问题，我们选择用 LSTM[6] 这一 RNN 变种。

长短期记忆网络(Long Short-Term Memory, LSTM),是一种时间递归神经网络,适合于处理和预测时间序列中间隔和延迟相对较长的重要事件。与标准 RNN 相比, LSTM 在算法中加入了一个判断信息有用与否的“处理器”,这个处理器作用的结构被称为细胞。一个细胞当中被放置了三扇门,分别叫做输入门、遗忘门和输出门如图4所示,这些精心设计的“门”结构实现了 LSTM 遗忘或增加信息能力。一个信息进入 LSTM 的网络当中,可以根据规则来判断是否有用。只有符合算法认证的信息才会留下,不符的信息则通过遗忘门被遗忘。

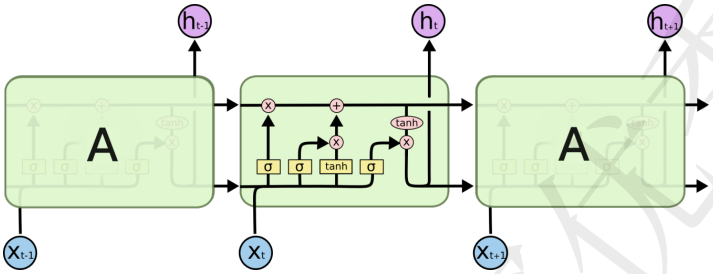

图 4: LSTM 细胞结构

某一时间步 t 的输入门 (i_t) 和遗忘门 (f_t) 都以输入变量(即我们的备选答案编码 x_t)、上一个时间步 $t-1$ 的输入向量 (h_t-1) 和偏置 (b) 作为输入,并通过激活函数得到响应值。

忘记门层:忘记门层决定了 LSTM 何时会从系统状态中丢弃信息。其公式为:

$$f_t = \delta(W_f \cdot [h_{t-1}, x_t] + b_f)$$

当我们在输入中看到新的主语时,可以通过使用忘记门层忘记旧的主语,提高语义准确度。

输入门层:输入门层确定了 LSTM 将要把什么样的信息保存在新的细胞状态中。其计算公式为:

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \bar{c_t} &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ c_t &= f_t * c_{t-1} + i_t * \bar{c_t} \end{aligned}$$

输出门层:最终 LSTM 使用输出门层确定需要输出的值,其计算公式为:

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(c_t) \end{aligned}$$

3.1.2 Bi-LSTM

由于自然语言的语义复杂性,语句的重要信息既可能出现在句首,也可能出现在句尾。普通的 LSTM 可以解决 RNN 的遗忘问题,但它只能做到根据前文理解后文,而

无法做到根据后文理解前文，这样在某些情况下，LSTM 仍然可能丢失部分句首或者句尾的重要语义，因此我们最终决定用双向长短时记忆模型（Bi-LSTM）。Bi-LSTM 可以将原来按序的输入转化成一正一逆的两股输入，通过两个 LSTM 单元组成一个新的双向 LSTM 单元，以尽量弥补纯问答对匹配方式对上下文信息考虑的不足，一定程度上解决 RNN 与单向 LSTM 的遗忘问题。采用了 Bi-LSTM 的模型经过实验检验，比普通的 LSTM、RNN 效果有明显提升，它可以更完整的对句子的深层语义进行编码，中间编码向量与既包含了句首的语义信息，也保留了句尾的语义信息，因此能更好的表征语句的深层语义。如图5所示：

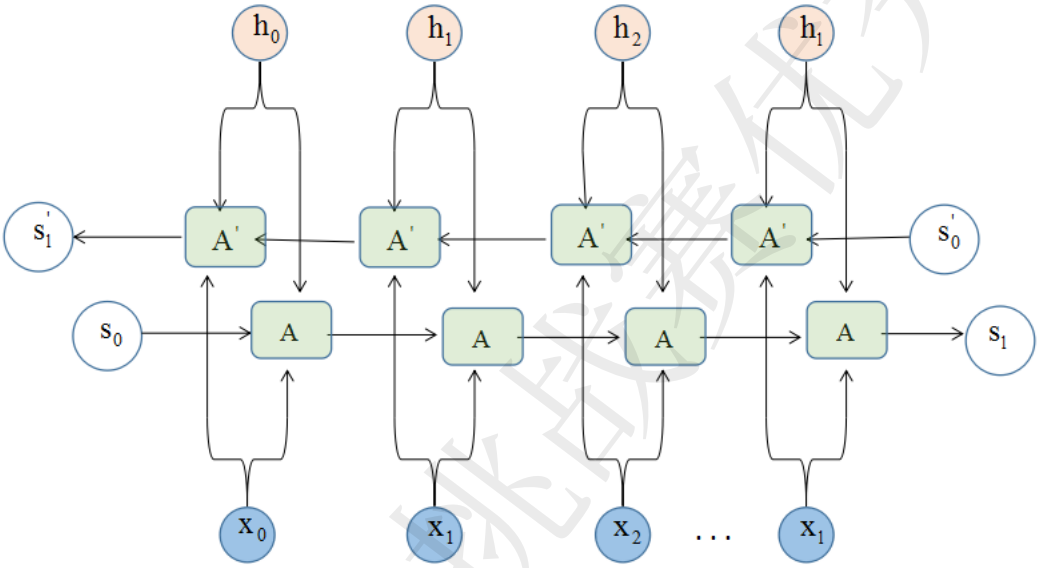

图 5: Bi-LSTM 网络结构

设第二步 word2vec 得到的向量为 $e(x)$,$x \in D, Q$, 则 Bi-LSTM 网络可以表示为：

$$\overrightarrow{h_s(s)} = \overrightarrow{LSTM(e(x))}$$
$$\overleftarrow{h_s(s)} = \overleftarrow{LSTM(e(x))}$$
$$h_s(s) = [\overrightarrow{h_s(s)}, \overleftarrow{h_s(s)}]$$

设 d 为单个 LSTM 的维度，则经过 Bi-LSTM 网络后的问句和答句中的词向量可表示为 $h_{doc} \in R^{|D|*2d}$ 和 $h_{query} \in R^{|Q|*2d}$。

3.2 句嵌入层

Arora 等人在 2017 提出了一种简单但非常有效的计算句向量的算法[4]。使用预估计的参数给句中的每个词向量赋予权重，然后使用 PCA 或者 SVD 方法移除句向量中

的无关部分。这种简单方法提高了文本相似性任务中的性能约 10%至 30%, 并击败了使用 RNN 和 LSTM 的复杂的监督方法。相关算法流程如表1所示:

表 1: 句向量表示算法

Algorithm 1 句向量表示算法

Input: Word embeddings $\{v_w : w \in V\}$,a set of sentence S,parameter a and estimation probabilities $\{p(w) : w \in V\}$of the words.

Output: Sentence embeddings $\{v_s : s \in S\}$

for all sentence s in S **do**

  $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$

end for

Form a martix X whose columns are $\{v_s : s \in S\}$,and let u be its first singular vector

for all sentence s in S **do**

  $v_s \leftarrow v_s - uu^T v_s$

end for

为了让注意力机制聚焦于语句，我们利用 Bi-LSTM 的输出 y 作为最终版本的词向量，应用 sentence2vec 算法，将答句的词向量转换成句向量，每条句向量成一行镶嵌成 h_{doc}, 再进行下一步处理。

3.3 注意力层

对于 Bi-LSTM, 其解码端的状态可以通过多种方式进行整合，较为简单的方法是直接采用最后一个时间片的状态，这种方法的有个比较明显的缺点是容易丢失句首的语义信息。另一种就是较为合理的方法就是将解码端的状态加权相加，也就是注意力 (Attention 机制)[7]。

将第二步得到问句矩阵 h_{query} 和由第三步得到句向量化的回答矩阵 h_{doc}，我们计算出一个成对匹配矩阵，它表明回答文档中的一句话和问句中的一个词的匹配程度，即将两矩阵做矩阵乘。

$$M(i,j) = h_{doc}(i)^T \cdot h_{query}(j)$$

在得到成对匹配矩阵 M 后，我们逐列进行 $softmax$ 操作，来获取每列的概率分布，其中对于问句中的单个词时每列是单个答句级别的注意力。即对于问句中的单个

词，对应答句的单个词的注意力为:

$$\alpha(t) = soft\max(M(1,t), ...M(|D|,t))$$
$$\alpha = [\alpha(1), \alpha(2), ..., \alpha(|Q|)]$$

其中 $\alpha(t) \in R^{|D|}$ 表示时间 t 时回答中每句关于问句中单个词的注意力。

3.4 ATT-over-ATT 层

与其他模型[8] 应用简单的启发式方法（如求和或求平均）来将这些个体注意力集中到最后的注意力上不同，我们用另一种注意力机制来自动确定每个个体注意力。首先我们计算反向注意力，即对于时间 t 的每一个答句，我们计算出问句中单个词的重要性，以指出给定单个答句时, 问句中哪个词更重要。

我们将对第三步骤得到的 M 矩阵的每一行做 $softmax$ 归一化，即得到答句对问句中的单个词级别的注意力。设 $\beta(t) \in R^{|Q|}$ 为答句对问句中的单个词级别的注意力，即

$$\beta(t) = soft\max(M(t,1), ..., M(t,|Q|))$$

问题级别的注意力 β，可以看作是对 $\beta(t)$ 的加权平均，即

$$\beta = \frac{1}{n}\sum_{t=1}^{|D|}\beta(t)$$

3.5 注意力汇聚层

经过第四步和第五步骤，我们得到了回答到问题级别的注意力 α 和问题级别的注意力 β, 最后，我们计算 α 与 β 的点积来获得回答级注意力，即将两个级别的注意力进行汇聚。汇聚示例如图6所示, 其中 1 到 9 为答句候选集中语句标号，通过汇聚表明答句 5,6 和 9 的是有关答句的可能性较高。

$$s = \alpha^T\beta$$

该操作就是在时间 t 查看问句词组时，计算每个独立文档等级 $\alpha(t)$ 注意力的加权总和。通过这种方式，问句中的每个词的贡献考可以明确的了解到，并且最终的分数由每个查询词来“投票”得到。

最后通过设定相关阈值，来确定，回答文档集合中的每一句是否是回答了问句，低于该阈值的记为无关，高于该阈值的记为有关。

3.6 总结

本节中所提出的神经网络架构如图7所示: s 中每个元素的值，即其对应位置的答句与问句的匹配分数，也可以理解为该答句是正确结果的可能性 P。即

$$p(v|D,Q) = (s_i)^{y_i}(1-s_i)^{(1-y_i)}, v \in V$$

图 6: 汇聚过程示意图

其中语句空间为 V，每一个语句表示为 v。i 表示该语句所对应的句向量在其所属的句向量矩阵的第 i 行。我们将最大化正确答案的对数似然率作为训练目标。

$$\ell = \sum_{i} \log(p(x)), x \in A$$

则模型的损失函数为:

$$L(x) = -\sum_{i} \log(p(x))$$

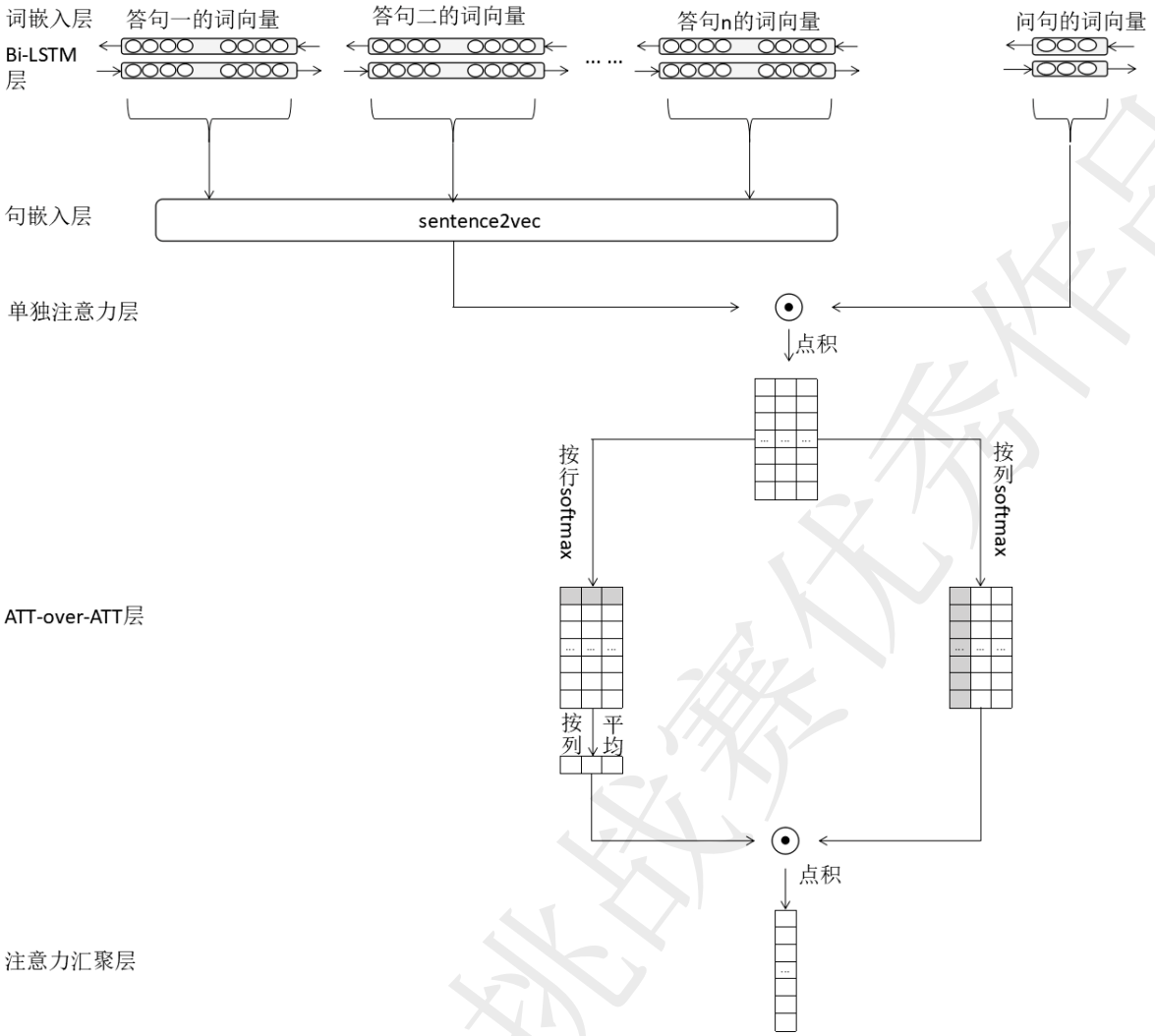

图 7: 神经网络架构

四、 实验评估

本节中我们将对上文中我们建立的模型进行实验验证，通过与传统方法以及其他深度学习方法进行比较分析，从而说明我们模型的合理性与有效性，同时介绍与分析模型的最优参数调整过程与各类参数对模型性能的具体影响。

4.1 实验平台

实验环境的软件硬件配置如表2所示：

使用到的 python 库有:

• tensorflow[9]: 由 Google 大脑小组的研究员和工程师们开发出来开源机器学习框架 tensorflow, 我们应用了其中 Bi-LSTM、Attention、Word2Vec,Softmax 这些深度学习算法。

表 2: 实验环境配置

CPU	Intel i7
显卡	GTX 1080
内存	16GB
操作系统	Ubuntu 16.04
python	3.5.2
CUDA	8.0
cuDNN	5.1
Tensorflow	1.2.1

• numpy[10]:NumPy 系统是 Python 的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比 Python 自身的嵌套列表 (nested list structure) 结构要高效的多。是 TensorFlow 的前置依赖。

• jieba[1]: 在预处理部分用于中文的分词部分。

同时为了加速训练和测试的速度,使用了显卡厂商 NVIDIA 推出的运算平台 CUDA[11] 和 cuDNN[12]

4.2 实验数据来源

(1) 出题方

选取了出题方以 json 形式所给的 test_data_sample.json 和 test_data_complete.json 中大约 55 万个 <Q,A,T> 三元组用于我们模型的训练和测试。

(2) 百度 WebQA[13]

百度利用百度知道和其他资源，构建了 WebQA 数据集，目的是解决“如何从大片相关文本中，为给定问题提取正确的、简明的答案”问题，通过提供好的数据集进行训练，来发现好的算法。WebQA 中每条数据由 question,evidence,answer 三部分构成：question 是用户提出的问题。evidence 为对应的材料，即回答的文本段。answer 是一个答案列表（因为答案可能有多个），如果材料中并没有答案，那么答案是 [no_answer]。

(3) 机器学习保险行业问答开放数据集[14]

该语料库包含从网站 Insurance Library 收集的问题和答案。这是保险领域首个开放的 QA 语料库，语料库的内容由现实世界的用户提出，高质量的答案由具有深度领域知识的专业人士提供。

4.3 实验评价指标

对于本文中的模型，我们采用精确率（Precision）、召回率（Recall）、两者的调和平均数 F1-Score、准确率（accuracy）、MRR、MAP 来评价我们模型的表现效果。以上指标的详细定义如下: 为了放便后面符号的说明定义一个混淆矩阵，如表3所示:

表 3: 混淆矩阵

	相关	不相关
被检测到的	TP	FP
未被检测到的	FN	TN

- TP(True Positive): 正类项目被判定为正类
- FP(False Positive): 负类项目被判定为负类
- FN(False Negative): 正类项目被判断为负类
- TN(True Negative): 正类项目被判断为负类

(1) 精准率 (Precision)

是衡量某一检索系统的信号噪声比的一种指标，即检出的相关文献与检出的全部文献的百分比。

$$p = \frac{TP}{(TP + FP)}$$

(2) 召回率 (Recall)

召回率 (Recall) 是检索出的相关文档数和文档库中所有的相关文档数的比率，衡量的是检索系统的查全率。

$$R = \frac{TP}{(TP + FN)}$$

(3) 准确率 (Accuracy)

正确率（Accuracy）是指是指正确检索出的有关无关文档数，占文档库中所有文档数的比率。

$$A = \frac{TP + TN}{TP + FP + FN + TN}$$

(4)F1-Score

分类的 F1 值就是准确率和召回率的调和平均值，具体计算公式为:

$$F1 = \frac{2PR}{P+R}$$

(5)MRR

最后我们的系统会对回答的每一个句子打一个分，然后设置阈值返回含有多个句子的集合，同时将得分高的结果考前返回，用 MRR 评判这个系统系统好坏就是看第一个正确答案的位置，第一个正确答案越靠前，MRR 评分越高，即:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$

其中，Q 为样本 $query$ 集合，$|Q|$ 表示 Q 中 $query$ 个数，$rank_i$ 表示在第 i 个 $query$ 中，第一个正确答案的排名。

(6)MAP

定义单个问题检测出的相关回答的准确率为 AP

$$AP = \frac{\sum_{k}^{|R|}\frac{rank_k}{rank_i}}{|A|}$$

其中 A，R 分别为系统返回的相关回答和，其中正类项目的集合，$|A|$,$|R|$ 表示 A 和 R 集合的大小，$rank_i$ 和 $rank_k$ 表示该条回答在 A 和 R 中的排名。

MAP 则为对去一个查询集合中每条查询的 AP 值的平均，即

$$MAP = \frac{\sum_{k=1}^{|Q|}AP(K)}{|Q|}$$

其中，Q 为样本 $query$ 集合，$|Q|$ 表示 Q 中 $query$ 个数。

4.4 实验设置及结果分析

4.4.1 相关参数

我们实现了基于 Attention-over-Attention 机制下的双向 LSTM 智能阅读模型，系统训练时相关参数设置如表4所示:

4.4.2 阈值确定

模型经过最后的 attention-over-attention 机制得到的是对应位置答句与问句的匹配程度 P，我们需要设置一个阈值 δ, 当 $P > \delta$ 时，判断该答句为有关回答，即输出为 1; 反之，则判断该答句为无关回答，即输出为 0。同时实验来选取阈值 δ。

由图8可以看出，权衡精准率和召回率，我们取 $\delta = 0.5$ 作为最后的阈值。

表 4: 系统训练时相关参数

参数	参数值
学习速率	0.4
学习速率减缓因子	0.99
最大梯度范数	5.0
批次大小	8
层大小	128
batchSize	32
输入端词向量维数	50
输出端词向量维数	40

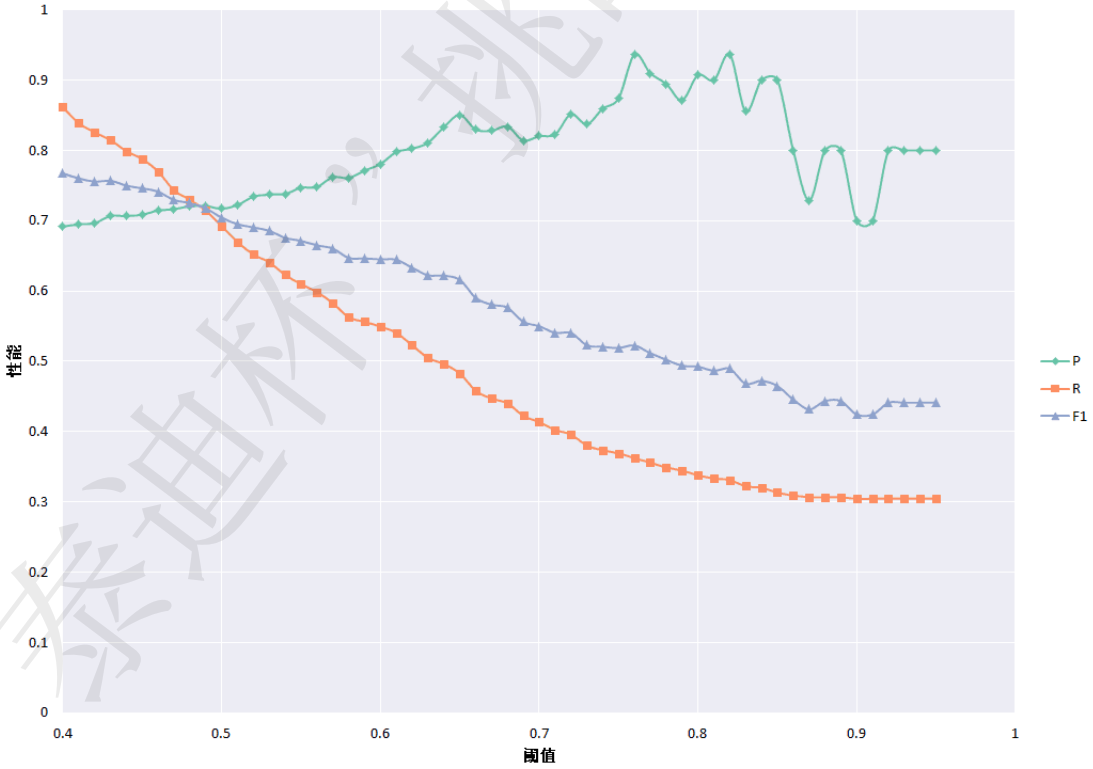

图 8: 阈值对性能的影响

4.4.3 相关结果

(1) 在不同数据集上验证模型的性能

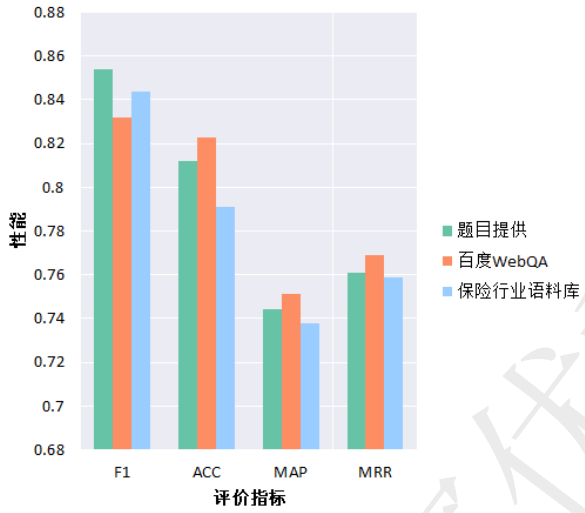

图 9: 在不同数据集上模型的性能

如图9所示，为模型在题目给定数据集，百度 WebQA 和保险行业语聊库上的相关性能。在三个数据集上，我们的模型均表现出了优异的性能，说明该模型可以成功捕捉到不同语料库中的问答关系，具有较强的泛化能力。

(2) 使用不同的词嵌入方法的相关性能

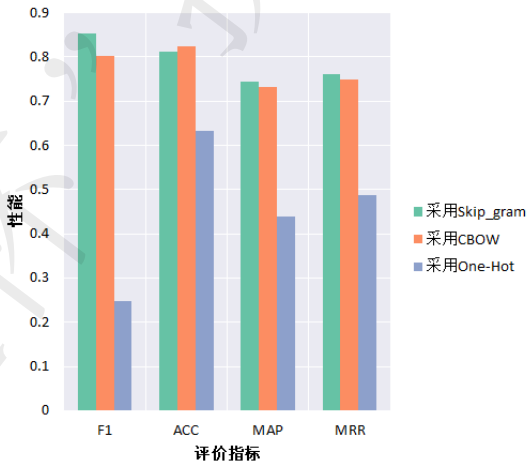

图 10: 采用不同的词嵌入方法的相关性能

在实验过程中，为了确定最佳方案，我们分别使用 Skip_gram 算法，CBOW 算法与传统的 One-hot 编码算法对语料进行词嵌入处理，并将测得性能进行了对比。实验结果，如图10所示。Skip_gram 与 CBOW 同属 W ord2vec 方法，而使用 Word2vec 获得的 F1-score 是只使用 One -Hot 方法的 3.5 倍，这说明 Word2vec 算法的性能要明显优于

One-hot 编码，而 Word2vec 中，Skip_gram 与 CBOW 性能相近，多次实验统计结果的平均水平表明，Skip_gram 的 F1-score 比 CBOW 高 6.484%，ACC 低 1.337%，MAP 高 1.639%，MRR 高 1.467%，Skip_gram 略好于 CBOW，因此我们最终采用 Skip_gram 算法进行词嵌入。

(3) 采用 sentence2vec、普通 RNN 生成句向量和不采用句向量下相关性能

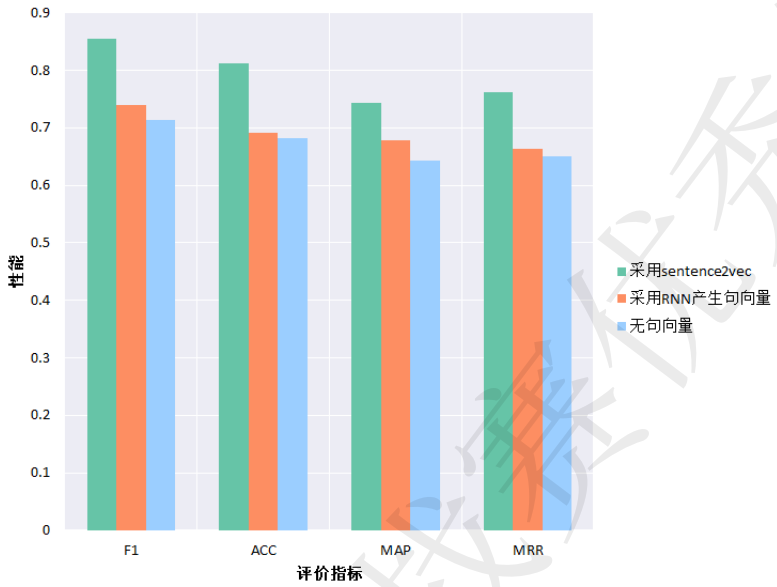

图 11: 采用 sentence2vec、普通 RNN 生成句向量和不采用句向量下相关性能

如图11所示，采用 sentence2vec，F1-score 比基于 RNN 生成句向量的模型高 15.405%，比不使用句向量的模型高 19.776%。经过分析，与基于 RNN 生成句向量的模型相比，sentence2vec 不会出现梯度消失问题，且训练速度明显加快；而 sentence2vec 嵌入的句向量兼顾了上下文提取句意，有助于最终将答案定位到具体的句子；与不使用句向量、仅用词向量的模型相比，性能提升很大。

(4) 采用 Attention-over-Attention 机制和与不采用机制下相关性能

如图12所示，采用 Attention-over-Attention 机制后，F1-score 比起单一注意力提高了 22.350%。Attention-over-Attention 机制捕捉了答案到问题、问题到答案的双重注意力，使模型可以利用问句和答句之间的交互信息，这样的机制比起只依赖单一注意力的普通模型，兼顾利用的信息更多，因此性能更优。

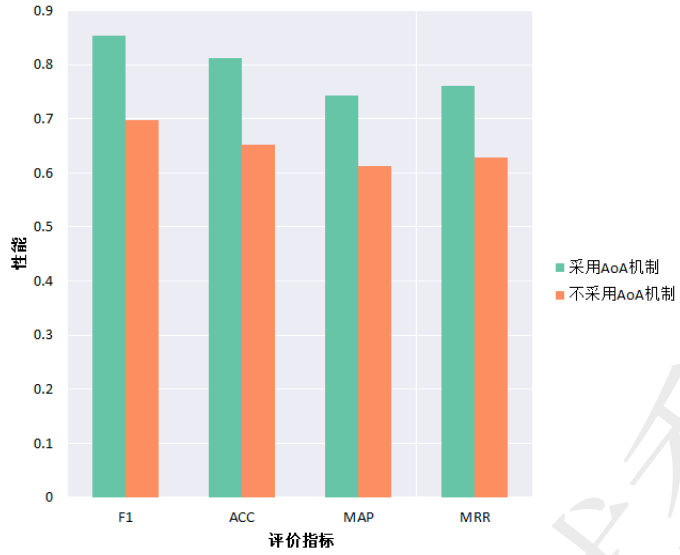

图 12: 采用 Attention-over-Attention 机制和与不采用机制下相关性能

(5) 本文模型与其他模型相比较

如图13所示，我们在出题方所给的数据库上，我们分别对比了基于传统方法进行语义分析的模型与基于卷积神经网络的模型，我们的模型均优于上述模型。

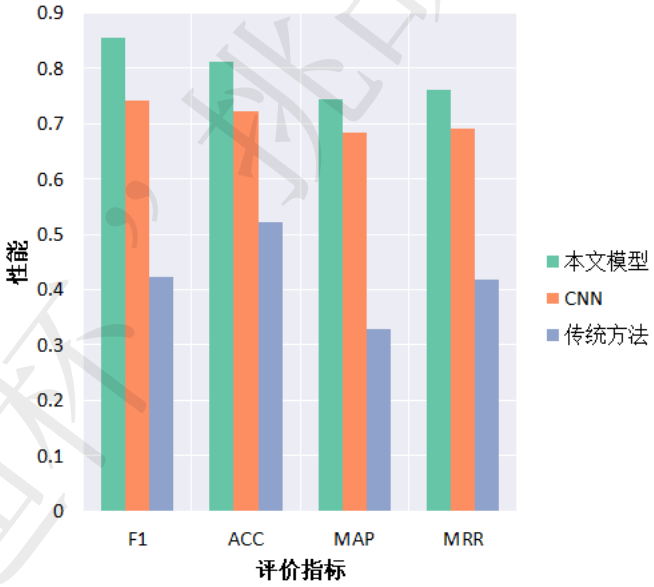

图 13: 本文模型与其他模型相比较

五、 模型优化

5.1 对问句语义提取

可以将知识库理解为一个三元组组成的集合实体, 实体关系, 实体。智能的阅读系统就算是可以看作给定一个实体和一个联系，查看候选答句中是否包另一个实体。

因此可以首先基于语义分析的方法通过将实体和联系进行识别和标注将自然语言形式的问句转换为 lambda 表达式或依存组合语义树等逻辑表达形式。通过对这种逻辑表达形式，进行向量化表示，从而进一步加强，对语义的聚焦。

5.2 文本蕴含

文本蕴含 (textual entailment) 是指文本对之间的指向关系，用符号 T 表示被蕴含的文本，用 H 表示被蕴含的文本，也就是假设。如果 H 意思能够从 T 中推导出来，那么就认为 T 蕴含 H。

我们也可以将文本蕴含的相关思想运用到智能阅读系统中，对候选答案进行重排序。因为可以假设在一个已经存在的阅读系统中返回的最佳的候选答案，虽然最佳候选答案可能不是正确答案，但是在大多数情况下正确答案位于返回的候选答案集合中。

因此可能创建一个蕴含对的集合，其中将系统返回的候选答案合并作为蕴含对的文本，将转换的问题作为蕴含对的假设。文本蕴含系统接下来依次应用在蕴含对上，那些能蕴含转换文本的候选蕴含对移到列表的顶端，非蕴含的则移动到底部。Harabagiu 和 Hickl[15] 在问答系统上对文本蕴含计算进行了测试，系统的准确率得到提高。

5.3 未来改进

由以上论述可知，对于输入的问题，我们的模型可以出色地完成将答案定位到所在句子的任务。下一步，我们希望继续改进模型，实现更细粒度的答案提取。如果需要将答案定位到词语，我们可以去掉嵌入句向量的步骤，用 Bi-LSTM 得到的蕴含上下文信息的词向量进行 Attention-over-Attention 的计算。按列计算的注意力代表了问题中每个词选择答案的重要性，按行计算的注意力代表了文档中的每个词响应问题的重要性。正反注意力经过点乘，最终获取的注意力向量每个分量值含义为文档中每个词与问题的匹配分数。统计每个词对应分量值之和，即以该词作为答案时与问题的匹配分数之和，获得所有和值并排序，分数越高则该词是正确答案的可能性越大。

六、 参考文献

[1] https://github.com/fxsjy/jieba

[2] http://www.scu.edu.cn/cs/xsky/gjhy/webinfo/2014/12/1416879433948314.htm

[3] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient Estimation of Word Representations in Vector Space[J]. arXiv preprint arXiv, 2013.

[4] A simple but tough-to-beat baseline for sentence embeddings."Arora, Sanjeev, Yingyu Liang, and Tengyu Ma.2017.

[5] MikolovT, Karafiat M,Burget L,et al.Recurrent neural network based language model[C].Interspeech.2010,2:3.

[6] Hochreiter S,Schmidhuber J.Longshort-term memory[J].Neural computation,1997,9(8):1735-1780.

[7] Yiming Cui, Zhipeng Chen, Attention-over-Attention Neural Networks for Reading Comprehension, 2016.

[8] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

[9] www.tensorflow.org/

[10] Van Der Walt S,Colbert S C,Varoquaux G.The NumPy array:a structure for efficient numerical computation[J].computing in Science & Engineering,2011,13(2):22-30

[11] http://www.nvidia.cn/object/cudazone-cn.html

[12] https://developer.nvidia.com/cudnn

[13] http://idl.baidu.com/WebQA.html

[14] https://github.com/Samurais/insuranceqa-corpus-zh

[15] A.Esuli and F.Sebastiani,and T.Fukushima,"Collecting evaluative expressions for opinion extraction" in Proceedings of International Joint Conference on Natural Language Processing and Computational Linguistics,2006.