智能阅读模型的构建

摘要

近年来，人工智能在许多方面取得了突破性的成就，因此越来越受到人们的关注。智能阅读就是人工智能一个重要的领域，而自动问答就是一种用自然语言向机器提问，并且自主获取文本中的知识向用户直接返回所需答案的一种智能文本挖掘。

在数据预处理阶段，利用 Python 对所给的数据进行预处理，预处理主要从三个方面进行：分词、去除停用词和训练词向量。经过查阅相关文献和技术文档，采用中文支持良好的 jieba 库进行分词，利用常用的中文停用词库去除停用词，然后使用 genism 提供的 Word2vec 技术来训练词向量。通过数据预处理，可以得到回答文本的 50 维词向量数据。

在挖掘建模阶段，建立了基于 LSTM 的问答系统模型。首先，根据预处理的词向量，建立了一个索引词典，将问题和答案转换为索引表示。然后，借助基于 tensorflow 的 Keras 平台搭建出 LSTM 的双输入单输出的深度网络模型，使用预训练的词向量作为嵌入层参数，将问题和答案的索引列表分别输入到模型中进行训练。最后，模型训练得到的精度大约有 76%。

关键字：自然语言处理　Word2vec　Keras　双输入 LSTM

Abstract

In recent years, artificial intelligence has made breakthrough achievements in many aspects ,so people are paying more and more attention to it.The intelligent reading is an important field of artificial intelligence, and automatic answering question is a kind of intelligent text mining .It uses natural language to ask questions from machines and autonomously obtains the knowledge in texts to directly return the required answers to users.

In the data preprocessing stage, the given data is preprocessed using Python. The preprocessing mainly proceeds from three aspects: word segmentation, removal of stop words, and training word vectors. After consulting relevant documents and technical documents, use jieba library supporting Chinese well to word segmentation, and use commonly used Chinese stop word dictionary to remove stop words, and then use Word2vec technology provided by genism to train word vectors. Through data preprocessing, we can obtain the 50-dimensional word vectors of the answer text.

In the mining modeling stage, we build a LSTM-based question answering system model. Firstly, based on the pre-processed word vectors, an index dictionary is built to convert questions and answers into indexed representations. Then, use tensorflow-based Keras platform to build LSTM's two-input and single-output deep network model, and use pre-trained word vectors as embedded layer parameters,and then the model for training will input the question and answer index lists. Finally, the accuracy of model training is about 76%.

Key words: NLP LSTM Keras double-input LSTM

1．挖掘目标

1.1 挖掘背景

最近几年来，人工智能在许多方面取得了突破性的成就，因此越来越受到人们的关注。自动问答系统就是人工智能中的一个很重要的分支，也是自然语言处理领域中的一个值得长期研究的目标。

有数据表明，一个组织 80%的信息是以文本的形式存放的，包括 Web 页面、技术文档、电子邮件、小说等。人们迫切需要能从大量的文本中提取有效、新颖、有用、可理解的有价值知识，因此，智能阅读文本提取有价值的信息对人们来说将有很大的帮助。文本挖掘技术就是为了解决这一问题而产生的。

所谓文本挖掘,简单讲,就是从文本数据中挖掘有意义的信息的过程。从 1995 年 Feldman 提出这一概念到现在短短的二十几年左右时间里，围绕文本挖掘及相关技术的理论研究和应用实践如火如荼的开展起来。数据挖掘领域著名网站的调查显示，文本挖掘已经成为数据挖掘中一个日益流行而重要的研究分支。越来越多领域内的顶级国际会议都开始增设文本挖掘的专题，如 KDD、VLDB、SIGIR 等。

1.2 研究现状

在国外文本挖掘的研究开展的比较早，50 年代 H.Pluhn 在文本挖掘领域进行了开创性的研究，他提出了词频统计以及自动分类。随后众多学者在这个领域进一步做了很多深入细致的研究工作。

我国引入文本挖掘概念并展开中文的文本挖掘知识在最近几年才开始。现阶段，我国文本挖掘研究主要是对国外的相关理论和技术进行验证和探索，文本挖掘理论实际应用和适合中文的文本挖掘技术及算法研究都还处于初步阶段，目前还没有形成完整的或者成体系的适合针对中文信息处理的文本挖掘理论与技术框架。

2.分析方法

2.1 流程分析

2.1.1 总体流程

本文对文本挖掘的处理流程如图 1 所示。

图 1 总体流程

（1）数据预处理：将题目给出的原始数据通过预处理整合成格式规范的数据，方便后续处理，以形成训练所需的原始数据集。

（2） 问题检索：将测试问题与训练语料库中相同类别下的其它问题进行文本相似度计算，找出相似度较高的问题作为相似问题集合。

（3） 答案抽取：将相似问题集合中的所有答案进行排序，选出最佳答案反馈给用户。

2.2 数据预处理

因为计算机不能直接处理文本这样的非结构型数据，所以必须对数据中包含的问题、答案和标记进行预处理，转换成结构型数据。中文文本预处理包括分词和去停用词等步骤。根据统计，我们可得出数据中不同问题长度的数量如图 2 所示，不同答案长度的数量如图 3 所示以及答案的正反例数量如图 4 所示。清楚给定数据的数量关系，我们才能够更好的把握对数据预处理的方法。

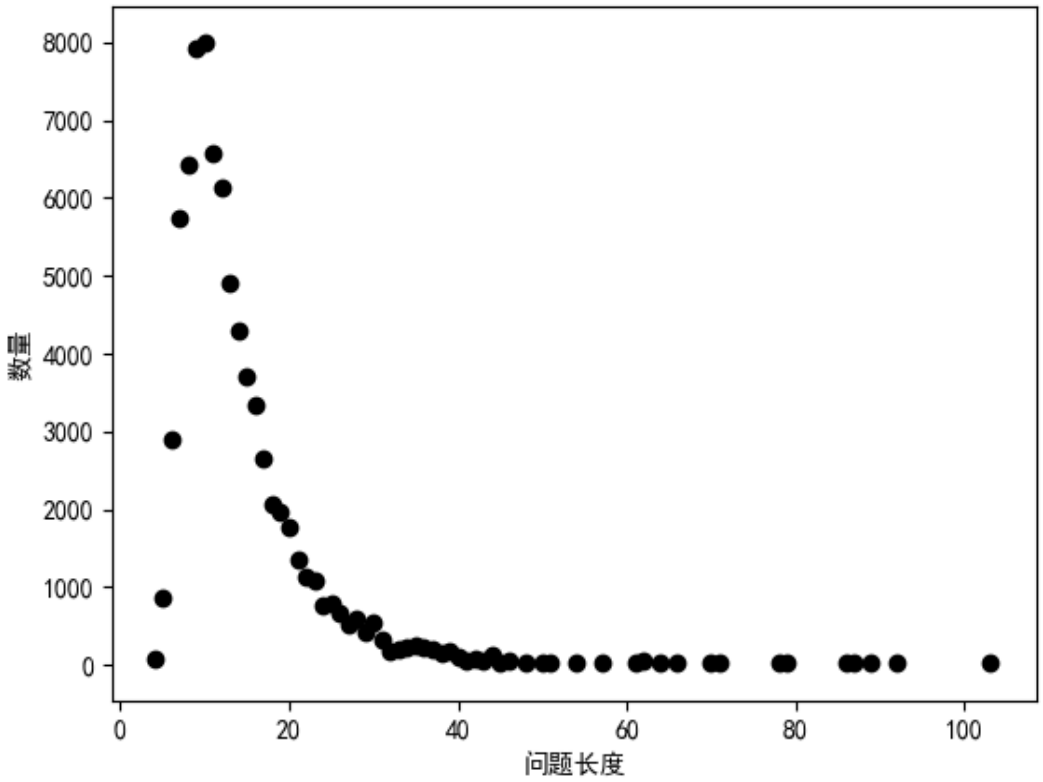

图 2 不同问题长度的数量

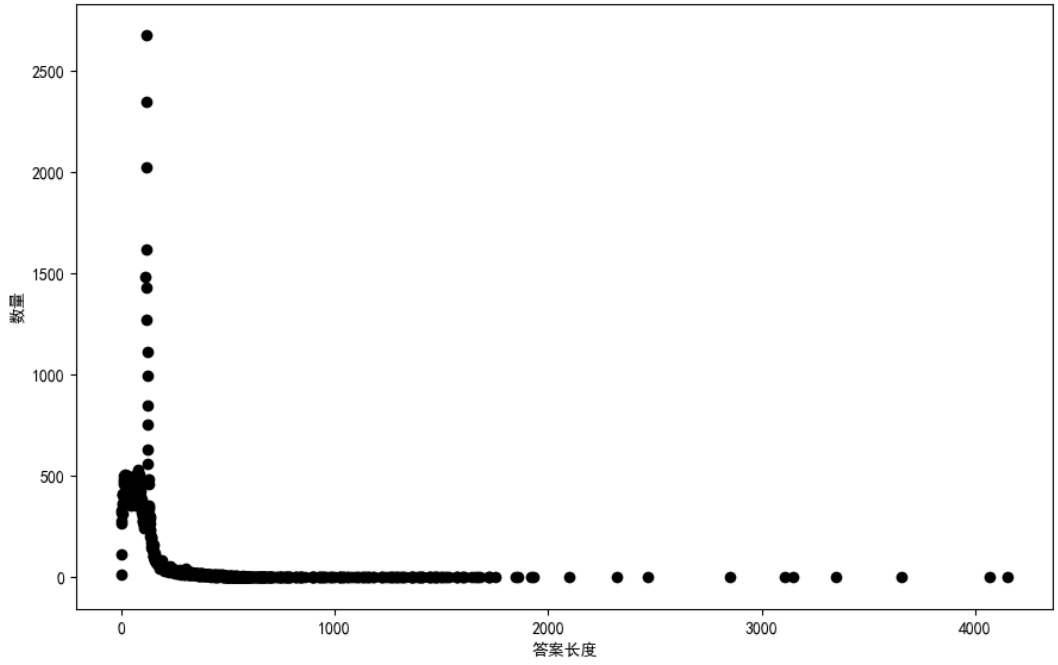

图 3 不同答案长度的数量

图 4 问题正反例数量

2.1.1 分词

一段中文文字中，词与词之间是连续的，每个词是最基本的语义单位。文本分类中的每一个特征都是一个词，因而文文本需要分词处理。中文分词的定义就是将连续的汉语字符串分割成一个个具有特定语义的词。

随着中文信息处理的发展，出现了许多分词方法[1]，主要有：基于字符匹配的分词方法、基于理解的分词方法、基于统计的分词方法等。基于不同的分词方法，出现了多种多样的分词工具，例如结合词典分词和文法分析算法的中文分词组件 *IKAnalyzer* 分词器，基于 *Trie* 树结构的词图扫描和动态规划寻找最大词频组合的 *jieba* 分词等。

IKAnalyzer 分词器[2] 需要一定的配置环境，且移植难度较大。相较而言，

jieba 分词器在 Python 中可移植性好，且其分词的准确性也比较好。基于这一考虑，本文采用 Python 中的中文分词包 *jieba* 来对 TXT 文档中的数据进行分词。此外，jieba 分词提供了分词、词性标注、支持用户字典等功能。

2.1.2 去除停用词

文本处理中，题目所给数据中包括许多的停用词，即出现频繁但无实意的虚词和类别色彩不强的中性词等。中文文本中停用词一般包括标点符号、数字、英文字符、单个字组成的词和没有什么实际意义的词（如“这些”、“不少”等）。去除停用词的方法[3] 有基于统计的方法和基于停用词库的方法，基于统计的方法是采用 TF（一个简单的评估函数）进行停用词过滤，若要专门设计一个适用于本文数据的停用词过滤方法需要较长的时间，因此，本文采用基于停用词库的方法来去除停用词。因此，本文采用基于停用词库的方法，利用常见的停用词库，对已经分词好的答案文本进行去除停用词。

2.1.3 基于 Word2vec 词向量

自然语言要让计算机“理解”，那么就需要把自然语言转换成计算机能够识别计算的形式，就需要将语言数学化，转换成深度学习算法能够识别的格式，其中一种有效的方法就是词向量。

Word2vec[2][4] 是 Google 在 2013 年开源的一款以深度学习为基础的学习工具包。Word2vec 的主要功能是利用神经网络为词寻找一个连续空间中的表示，它经词转换成向量，通过转换可以把对文本内容的处理简化为向量空间中的向量运算，然后计算向量空间上的相似度，来表示文本语义上的相似度。Word2vec 简单易用且在 Python 中能够很好的实现，GloVe 算法由于使用了全局信息，需要更多的存储容量，相对之下， Word2vec 在这方面节省较多的资源。由于设备的限制等因素，本文采用 Word2vec 方法来处理。由于没有 GPU 等设备的支持，我们需要尽可能的采用较为适合我们的数据大小来，因此，本文通过设置

Word2vec 的参数得到一个 50 维的词向量，以便于后面模型的训练。

3 模型构建与求解

3.1 基于 LSTM 的问题分类模型

LSTM 模型在处理序列数据方面有很多的应用，并且取得了相当好的实际应用效果。LSTM 可以有效地捕捉输入序列中的上下文信息，因此本文采用 LSTM 来做语言模型来计算每个候选的得分。在此先介绍 LSTM 的起源、基本原理以及它在问题分类中的应用。

3.1.1 LSTM 模型介绍

前馈神经网络有固定的的输入和输出的维数[5]，因此当需要处理不定长的序列数据时，前馈神经网络就不再适用了。但是自然语言中的文本大多数是不等长的，为了使得前馈神经网络能够处理变长的序列数据，其中的一种方法就是使用循环神经网络（Recurrent Neural Network，RNN），RNN 的结构能够有效解决该问题。

循环神经网络与前馈神经网络相比在相邻时刻状态之间增加了连接，而前馈神经网络的连接只存在于层与层之间。RNN 通过使用带自反馈的神经元，能够处理任意长度的序列数据，因而也比前馈神经网络更加符合生物神经网络的特征。为了更好地理解 RNN，将循环神经网络按时间顺序展开，如图 4 所示。

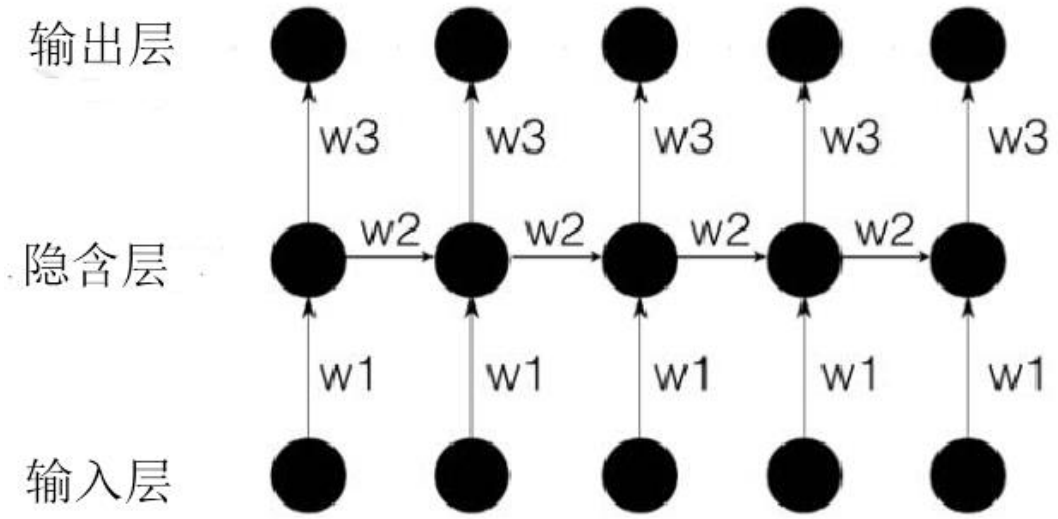

图 4 按时间顺序展开的循环神经网络

对于 RNN 来说，它每一时刻的隐层状态都保留了之前序列的信息，并会将新的输入编码进来，因此，RNN 对于之前输入的信息有记忆功能。但是在实际应用中，RNN 的记忆长度往往是有限的。随着时间间隔不断的增大，前面的序列的信息会在传递的过程中逐渐消失，最后的隐层状态往往只包含最后几个时刻的信息，同时在模型训练的过程中也会出现梯度爆炸和消失问题，因此只能学习到短周期的依赖。而长短期记忆网络（Long Short Term Memory，简称 LSTM）就能解决 RNN 模型梯度爆炸和消失以及长期依赖的问题。

LSTM[5]是循环神经网络的一种在结构上的改进模型，其主要目的是为了解决 RNNs 在实践过程中无法处理长距离依赖问题。LSTM 通过一种门结构来决定信息的流入流出，如只考虑第 i 个时间点的状态信息在后边时间点的留存情况，若后边的时间点控制之前状态的门一直保持开着的状态，则第 i 个时间点的状态信息可以一直保持下去。而实际上，LSTM 一共有三个门结构，分别是输入门、输出门和遗忘门。LSTM 单元的工作原理如图 5 所示。

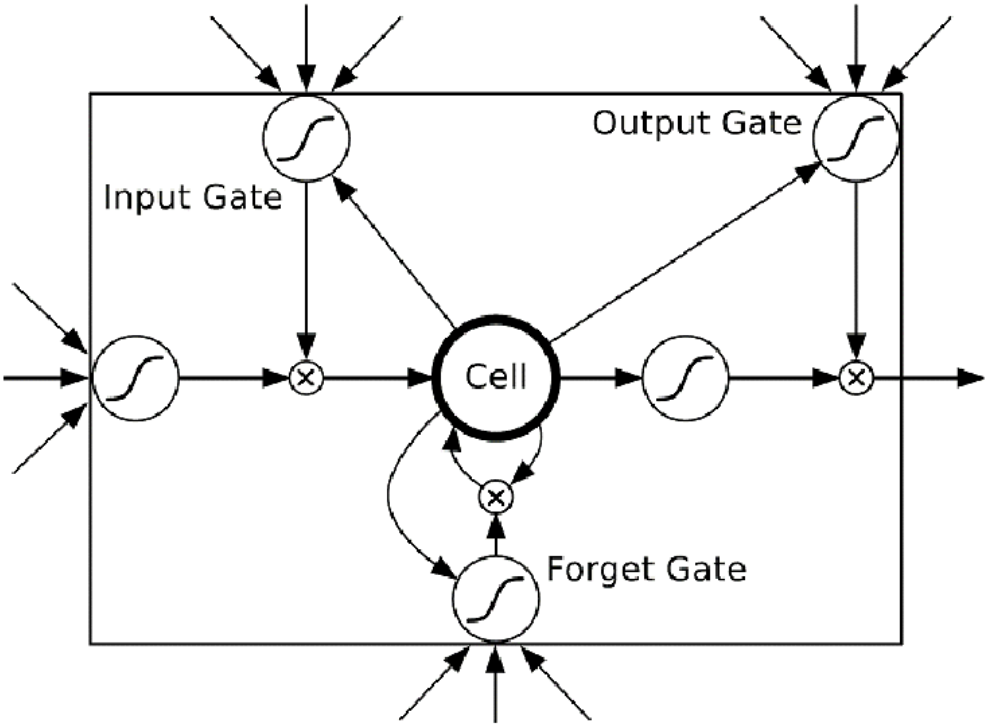

图 5 LSTM 单元内部工作原理

LSTM 模型的关键是引入了一组记忆单元，允许网络学习何时遗忘历史信息，以及何时用新信息更新记忆单元。在 t 时刻，记忆单元 c_t 记录了到当前时刻为止的所有历史信息，并受三个“门”所控制：输入门、输出门和遗忘门，三个门的元素的值在 [0，1] 之间。这样，LSTM 能够学习到更长周期的历史信息。

下面给出在时刻 t 时 LSTM 所对应的公式：

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{2}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{3}$$

$$\tilde{c} = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4}$$

$$c_t = f_t \odot c_t - 1 + i_t \odot \tilde{c} \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

其中 i_t 、o_t 、f_t 分别表示对应 t 时刻的输入门、输出门和遗忘门，他们的输入分别是当前时刻的输入 x_t 和上一时刻隐层的输出 h_{t-1} ,W 分别为相应的权值矩阵 b 分别为相应的偏置向量。$\tilde{c}$ 为当前时候的候选记忆单元值。c_t 为当前时刻记忆单元状态值，它是由当前时刻的输入门、遗忘门、候选记忆单元值以及上一时刻的记忆单元状态值共同决定的，⊙ 表示逐点乘积。h_t 是当前时刻的隐层输出值，通过当前时刻的输出门和记忆单元状态值计算得到。

3.1.2 基于 LSTM 的问题分类模型

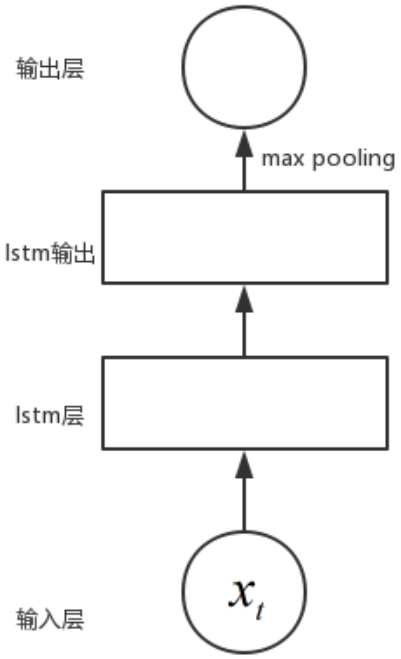

图 6　基于 LSTM 的问题分类模型

如图 6 所示是基于 LSTM 的问题分类模型，输入层是问题中词的词向量，经过 LSTM 之后得到每个输入所对应的隐层向量，然后对这些隐层输出按照时刻进行 max pooling 操作，最后经过 Softmax 层计算问题属于各个类别的概率，进而得到问题的类别

3.2 基于 LSTM 的答案选择模型

给定一个问题 q 和一些相关的候选答案$\{a_1, a_2, \cdots, a_s\}$（$s$是候选答案的数量），最终的目标是选择出其中的最佳候选答案 a_k ，$1 \le k \le s$ 。如果这个得到的候选答案 a_k是问题 q的其中一个答案（问题 q可能会包含多个答案），那么可以认为问题q被正确地回答了，否则的话问题q就是没有被正确地回答。

为了找到这个最佳候选答案，我们需要一种用来衡量问答对匹配程度的度量方法，以便能够选择最高得分的问答对，从而以最大的概率回答正确该问题。

3.2.1 基于 LSTM 的答案选择模型

如图 7 所示是基于 LSTM 的答案匹配度计算模型，该模型是由输入层、LSTM 层、Pooling 层组成的。

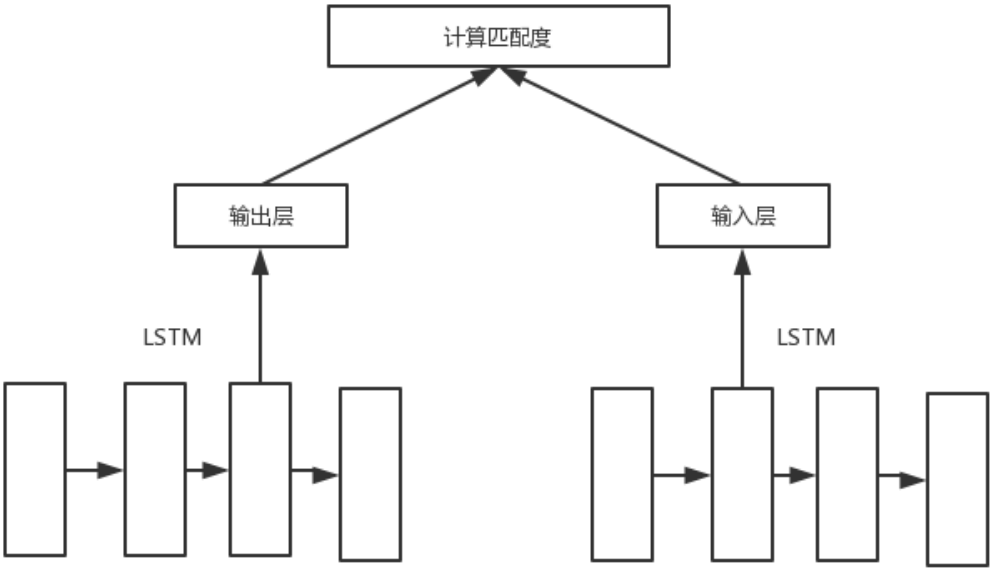

图 7 基于 LSTM 的计算答案匹配度框架

在输入层，问题和答案分别被转化成用向量表示的句子，作为 LSTM 层的输入，经过 LSTM 层的之后得到相应的向量语义表示，然后计算两者之间的匹配程度作为最终的匹配度计算结果。同理，两个 LSTM 网络之间的参数可以是

共享的也可以是分别计算的。

3.3 模型训练

在实际搭建模型时，我们通过查阅相关资料，并结合大量的实验，最终确定了以下流程，流程图如图 8 所示：

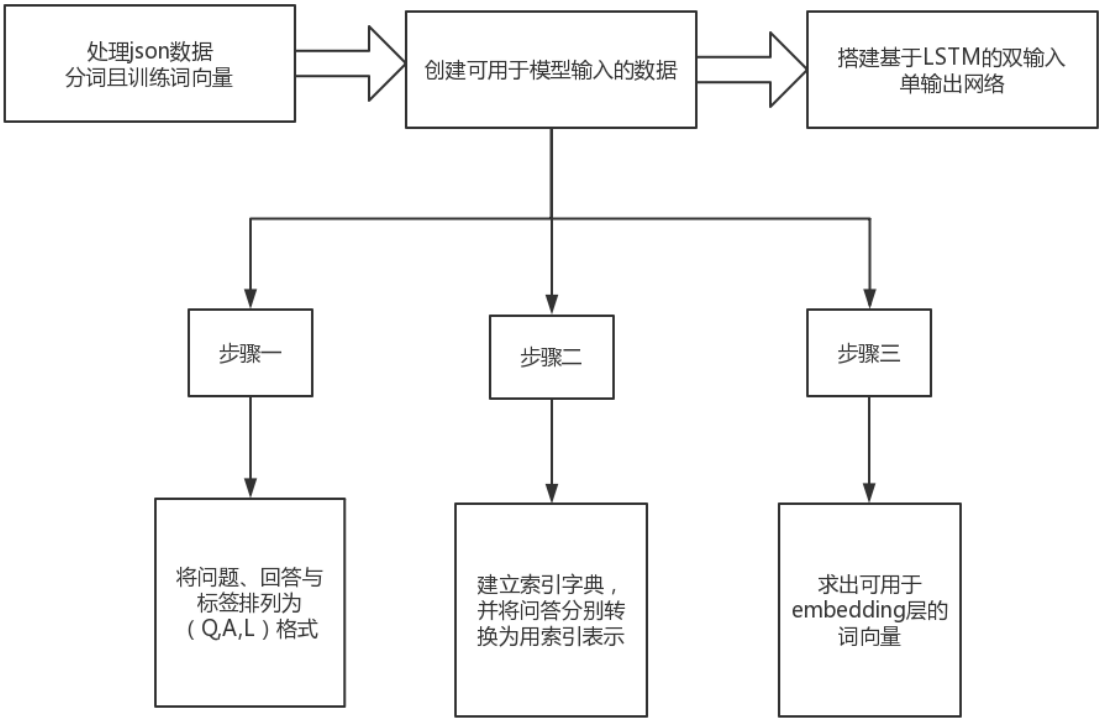

图 8 实验流程图

具体流程如下：

（一）处理 json 数据，分词且训练词向量

我们先使用 linux 的 cat 命令（文本输出命令），将文件中所有以“content”为标签的数据提取出来，得到“train_data_complete.txt”文件（见附件），这个文件中包含着所有的答案文本；然后使用 jieba 分词工具进行分词，输出为 txt 文本文档（这个文档中包含一行数据，以空格分割，包含所得的全部分词）；随后进行剔除停用词操作——我们结合相关资料和人工添加，建立了一个初步的停用词表（见附件“stopwords.txt”），然后使用遍历的方式逐一剔除停用词；最后使用 genism（基于 python 的）包所提供的 word2vec 工具训练词向量，输出后缀名为 vec 的非二进制文件，具体训练的参数设置如表 1 下。

表 1 训练参数设置表

参数名称	参数值	注释
sg	0	使用 CBOW 算法
size	50	词向量维度为 50 维
window	10	滑动窗口大小为 10
negative	5	使用 NEG 方法
hs	1	使用层级 softmax
iter	10	迭代 10 次

由于时间有限，无法测试所有的情况，可能会存在更好的分词和训练词向量方法，从而进一步提高模型的准确度。

（二）创建可用于模型输入的数据

目前的数据无法直接用于训练模型，所以还需进一步转换。我们的具体转换思路是：

Step 1：将问题、回答与标签进行排列，排列为（Q,A,L）格式

为了便于批量读取训练数据，我们利用 Python 对 json 格式文件的解析功能编写成为一个 python 脚本（见附件“read_write.py”），实现将问题、回答与标签排列为（Q,A,L）格式（Q:问题，A:回答，L:标签），最后输出为 txt 文本文件。其中，每一行是一个（Q,A,L）格式排列，对于每个问题，有多少个回答，就将会占据多少行。

Step 2：根据词向量，建立索引字典，并将问答分别转换为用索引表示

为了将训练数据表示为一种更直观的方式，我们想到使用词向量空间中的所有词建立一个字典，字典的索引为词，内容为编号（从 1 开始按顺序编号）。这样，对于每个句子（问题或回答），先将其分词，然后在索引字典里查找对应的

编号，返回一个值为整型编号的列表。在具体实现中，考虑到每个句子的长度不一，我们通过对句子的长度进行统计。最后决定取 100 为固定长度，若小于此长度，则补 0，若大于此长度，则截断。最终对于每个句子，都可得到长度为 100 的一维列表（具体代码实现见附件“dataproc.py”）。

Step 3：求出可用于 embedding 层的词向量

在搭建网络模型时，我们使用 Keras 来实现。我们分别定义了问题 embedding 层和回答 embedding 层，用于作为问题和回答的输入层。在这里，我们选择引入之前训练好的词向量嵌入到 embedding 层中，从而避免了对 embedding 层参数的训练。为了能够引入，我们将句子的索引列表（Step2 中所得的 100*1 的列表）通过索引映射到之前训练的 50 维词向量空间中，即每个句子将会转换为 100*50 的向量表示；最后得到 LEN(Q)*100*50 的三维列表（其中，LEN(Q)表示 step1 中读取的（Q,A,L）**总行数**），再对第二维上的向量求平均值，最终得到 LEN(Q)*50 的列表。这样，这个列表就可以作为 embedding 层的参数来使用（代码实现见附件 “mainLSTM.py”文件中的 get_train_data 函数）。

（一） 搭建基于 LSTM 的双输入——单输出网络

我们搭建了一个双输入——单输出网络，即：将问题和回答分别输入，分别连接两个不同的 LSTM 层，并且两个 LSTM 层还连接了两个全连接层，进一步缩小特征向量的维数，最后将两个全连接层的输出进行点乘融合，通过 softmax 层，转换为[0,1]的二值输出。具体的网络结构如下图 9 所示。

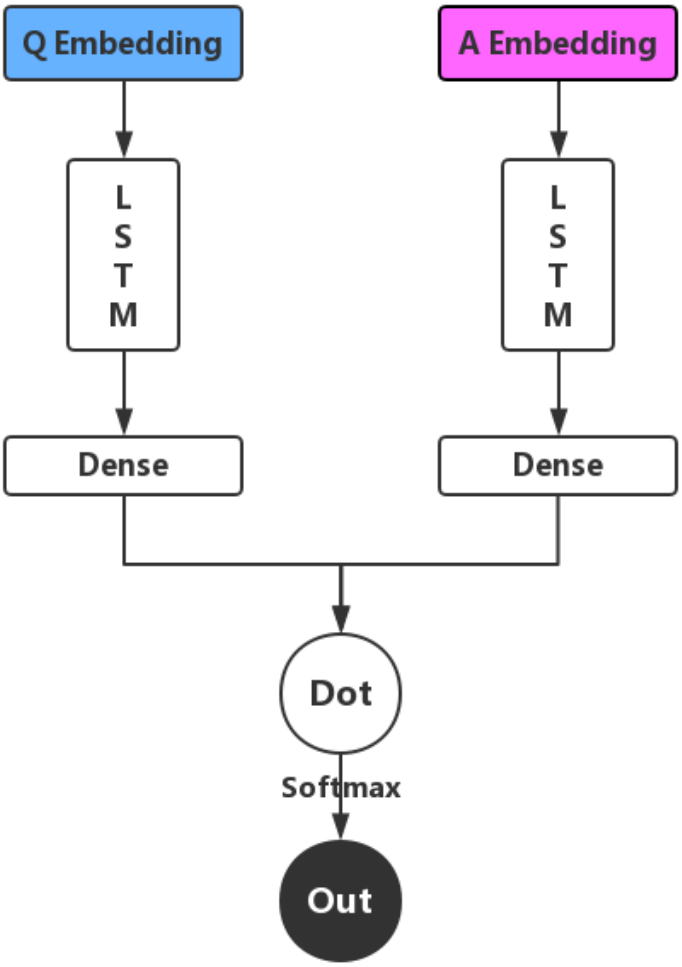

图 9 网络结构

在训练时，我们先使用 Kares 提供的 to_categorical 方法将标签数据转换为 one-hot 格式（在此处为 [0,1] 格式，属于二分类问题），然后选择 categorical_crossentropy 损失函数，可以取得较好的效果。除此之外，我们还按照一定比例划分数据集，分为训练数据、测试数据和对比检验数据；在训练中随机打乱数据集；使用 dropout 来防止过拟合等等。具体实现见附件“mainLSTM.py”文件。

4.模型的优缺点

4.1 模型的优点

我们的模型有如下的优点：

1、引入预训练的 Word2vec 模型，不仅可以防止过拟合,还能减少需要训练的参数个数，提高训练效率；

2 在限制长度为 100 的情况下，将单词在 Word2vec 中的词向量加和求平均获得整个句子的语义向量，从而减少了数据维度，且表现良好；

3 将全部回答单独映射到一个词向量空间，可以挖掘文档之间的内在关联。

4.2 模型的缺点

模型中的数据量过大，整个模型会较为复杂，模型训练时间较长。此外，我们对问题与文档之间的联系不能较好地把握。

5.模型的优化

在选择答案的过程中，计算答案匹配度[5] 的方法有很多，比如基于余弦相似度的匹配度算法、基于曼哈顿距离的匹配度计算方法、基于欧几里得距离的匹配度计算方法基于 Logistic 的匹配度计算方法、基于 Softmax 的匹配度计算方法、调和余弦相似度和欧几里得距离的置信度计算方法。模型主要使用两种类型的损失函数：合页损失函数（hinge loss function）和交叉熵损失函数。

有研究对比了 Euclidean、Cosine、Manhattan、CosEuclid 这些匹配度计算方法对性能的影响，实验结果表明，hinge2 损失函数的效果好于 hinge1 损失函数，CosEuclid 是其中的最佳匹配度计算方法。因此，经过多次实验与查阅相关文献，我们想到可以从以下方面进一步优化我们的模型：

1、 通过人工标注等方法，拓展停用词表，将更多无意义的词剔除；

2、 合并两个 LSTM 层输出时，引入欧氏距离的计算，从而综合考虑空间内的距离和角度因素；

3、 尝试增加一些感知器层；

4、 考虑 attention 机制，更好地处理较长输入。

6．结论

本文采用 Word2vec 模型生成的词向量来训练有利于防止过拟合问题，也可以减少需要训练的参数个数，相对于采用 Keras 的 embedding 层来处理会大大提高训练的效率。此外，我们采用 LSTM 对训练数据进行处理能够更好的解决长序列依赖的问题，为了充分发挥 LSTM 的优势，我们需要对文本数据进行截取，我们选择长度为 100 来对答案文本进行固定长度截取，以便于我们提高我们的训练效率。我们经过数次的尝试，终于我们的模型能够顺利的训练，最终，我们模型训练的准确度大约为 76%。

参考文献

[1] 张启宇，朱玲，张雅萍．中文分词算法研究综述[J]．情报探索，2008(11):53-56.
[2] 杨海丰，陈明亮，赵臻．常用中文分词软件在中医文本文献研究领域的适用性研究[J].世界科学技术-中医药现代化，2017，19(3):536-541.
[3] 陈磊.文本表示模型和特征选择算法研究[D].中国科学技术大学.2017
[4] 朱雪梅．基于 Word2Vec 主题提取的微博推荐[D]．北京理工大学，2014.
[5] 俞可.面向历史科目的问答技术研究[D].哈尔滨工业大学.2017.

附录

1. Data/final_lstm_model.h5

   Data/final_lstm_model.json

   //keras 训练模型文件

2. Data/Test_data_complete_done.txt

   //按照（Q,A,L）分割好的测试数据文件

3. Data/ans_train.vec

   //已训练好的词向量文件（非二进制文件）

4. Dataproc.py

   //数据预处理

5. MainLSTM.py

   //搭建和训练网络的主函数

6. Train_vec.py

   //分词以及训练词向量

7. Read_write.py

   //将数据文件分割成（Q,A,L）格式