密级：_____

浙江大学

硕 士 学 位 论 文

论文题目 基于非结构化文档的开放域自动问答系统技术研究

作者姓名 徐灿

指导教师 王东辉 副教授

学科(专业) 计算机科学与技术

所在学院 计算机学院

提交日期 2017-01

A Dissertation Submitted to Zhejiang University for the Degree of Master of Engineering

TITLE: A Techniques Research on Open-domain Question Answering Systems Using Unstructured Documents

Author: Can Xu

Supervisor: A.P. Donghui Wang

Subject: Computer Science

College: College of Computer Science

Submitted Date: 01/2017

摘要

自动问答系统能够根据用户输入的自然语言问题，直接返回精确的答案。本文的研究方向是基于非结构化文档的开放域自动问答系统，其特点是背后的数据源是非结构化的文档库，面向的问题是通用问题，并不局限于某个领域。

典型的基于非结构化文档的开放域自动问答系统一般由问题处理模块、文档处理模块和答案处理模块三个部分组成，其主要存在两个问题，第一是文档处理模块返回的段落候选集规模过大导致答案处理模块准确率降低。第二是基于规则的答案抽取过于繁琐，灵活性差。针对第一个问题，本文使用句子筛选和句子排序模块将段落候选集缩减为单个的答案句子。针对第二个问题，本文使用了端到端的深度神经网络模型代替传统的基于规则的答案抽取算法。

针对句子筛选模块，本文改进了一种计算文档相似度的算法 Word Mover's Distance (WMD)，并提出了一种将 BM25 和 WMD 结合的混合模型。本文分别进行了文档分类和文本排序实验。实验结果表明，改进后的 WMD 算法和混合模型比其他基准算法有更好的效果。针对句子排序模块，本文设计了五种特征来衡量问句和候选答案句子之间的相关性，并以此相关性得分对候选答案句子进行排序。这些特征包含了不同的级别。该模型称为 Multiple Level Feature Rank (MLFR)模型。本文测试并对比了若干基于深度神经网络的句子排序模型。实验结果表明，MLFR 模型有更好的排序效果。最后，本文引入了一个端到端的深度神经网络模型用于答案抽取，并将此模型与前面的句子筛选和句子排序模块组合在一起，设计实验进行了整体的性能评估。

本文对典型的基于非结构化文档的开放域自动问答系统中存在的问题提出了相应的解决方案，改进了计算文档相似度的相关算法，提出了一种基于多级特征的句子排序模型（MLFR），同时引入了一种端到端深度神经网络来进行答案抽取。实验结果表明，本文的解决方案是行之有效的。

关键词： 非结构化文档，自动问答，句子筛选，句子排序，神经网络

Abstract

The Question Answering Systems are concerned with providing relevant answers in response to questions proposed in natural language. This paper is about Question Answering Systems based on open-domain and unstructured documents, whose dataset is unstructured documents and can deal with questions about nearly everything.

Classic Question Answering Systems based on open-domain and unstructured documents usually composed of three distinct modules. These three components are: question processing module, document processing module and answer processing module. There are two problems. Firstly, the scale of candidate paragraphs returned by documents processing module is too big. Secondly, heuristic measures to extract the answer candidates are too complex. To deal with the first problem, we use sentences filtering module and sentences ranking module to reduce candidate paragraphs to single sentence. To deal with the second problem, we use an end-to-end neural network to replace heuristic measures to extract the answer candidates.

We improve a distance function between text documents named Word Mover's Distance (WMD) and present a novel model compounded of BM25 and WMD in sentence processing module. Experiments on document classification and document ranking task demonstrate the superiority of our new model. In the process of sentence ranking, we design Multiple Level Feature Rank (MLFR) model consists of five features at different levels of granularity to measure the relevance between questions and candidate sentences. Experiments on sentences ranking task demonstrate the superiority of MLFR model. Finally, we use an end-to-end neural network to replace heuristic measures to extract the answer candidate. We combine sentence filtering module, sentence ranking module and answer extraction module. Experiments are designed to test system performance.

This paper present two resolutions to deal with the problems of classic Question Answering Systems based on open-domain and unstructured documents. We improve a distance function between text documents, design Multiple Level Feature Rank (MLFR)

model consists of five features at different levels of granularity to measure the relevance between questions and candidate sentences and use an end-to-end neural network to replace heuristic measures to extract the answer candidate. Experiments demonstrate the superiority of our system.

Keywords： Unstructured documents, Question answering, Sentences filtering, Sentences ranking, Neural network

目录

图目录

表目录

第1章 绪论

1.1 课题背景

随着互联网的发展与普及，互联网上的信息也在爆炸式的增长。传统的搜索引擎能够根据关键词快速地找到相关的网页，其主要的技术原理是利用网络爬虫以一定的规则搜集和爬取网页，并将这些网页进行解析、存储、索引，然后根据关键词匹配以及页面链接来进行搜索。传统搜索引擎的这种算法已经成熟稳定，并且在很长一段时间内，在很大程度上解决了用户对信息获取的需求。然而，信息爆炸时代给搜索引擎带来的挑战越来越大，人们对信息检索质量的要求也越来越高。人们需要搜索引擎给出的是精准的答案，而不是成千上万个网页链接。尽管答案可能在这些网页中，但是人们必须花费大量的时间和精力来阅读这些网页才能找到答案。根据英国莫里调查公司的关于用户对搜索引擎的满意程度的结果显示，只有 18%的用户表示对搜索结果比较满意，并认为搜索引擎能够帮助自己找到相关的信息，同时有 68%用户表示搜索引擎的结果很让人失望[36]。

学术界和工业界也很早意识到了传统搜索引擎中存在的问题。1999 年，第八届文本信息检索大会（TREC-8：Text Retrieval Conference）首次增加了一个自动问答系统的竞赛。自动问答系统的目标就是根据用户输入的问题直接给出相应的答案。自动问答系统不再单纯是一个检索和匹配问题，更多的是变成了一个自然语言处理的问题。在 2000 年的 ACL 国际计算语言学学术会议上，有一个专题的讨论会，题目就是“Open-Domain Question Answering”[2]。在学术界和工业界的共同努力下，同时也得益于自然语言处理领域的一些基本算法的成熟，比如词性标注（POS）、命名实体识别（NR）、句法分析和依存关系分析等，自动问答系统很快就取得了立竿见影的效果。2011 年，IBM 的自动问答系统 Watson 参加了一个智力问答方面的综艺节目并一举击败了人类记录的保持者，引起了很大轰动。Watson 的 4TB 磁盘内包含 200 万页结构化和非结构化的信息，其中就包括维基

百科的全文[2]。经过 Watson 事件后，自动问答在研究和商业领域获得了更多的关注，同时也成为了自然语言处理领域最热门的研究方向之一。

根据自动问答系统背后的数据源的类型可以将其分为两类，分别是基于知识图谱的自动问答系统和基于非结构化文档的自动问答系统。根据自动问答系统面向的任务也可以分为两类，分别是开放域（open-domain）自动问答系统和封闭域（closed-domain）自动问答系统。

在基于知识图谱的自动问答系统中，通常使用资源描述框架 RDF 来表示数据源，也就是知识图谱。知识图谱可以看做是一张图，现实中的实体和概念对应图中的节点，实体之间的关系对应图中的边。知识图谱往往需要图数据库进行存储，同时可以使用一种专门面向图数据库的查询语言 SPARQL（SPARQL Protocal And RDF Query Language）进行查询。基于知识图谱的自动问答系统接收用户自然语言形式的问句，并将非结构化的自然语言问句映射成结构化的 SPARQL 查询语句，然后根据查询语句查询知识图谱并获得精准答案。这种类型的自动问答系统的研究核心和难点在于如何将非结构化的自然语言问句映射成结构化的 SPARQL 查询语句[37]。这类自动问答系统需要完备的知识图谱来保证性能，但是目前公开的知识图谱，比如 Freebase，Wikidata 都很不完备。构建完备的知识图谱本身就是一个非常有挑战的研究问题。

在基于非结构化文档的自动问答系统中，数据源通常是规模庞大的文本文档的集合，比如万维网就是最大的数据源。这类自动问答系统接收用户自然语言形式的问句，并提取出关键词，然后传递给信息检索系统来搜索包含这些关键词的文档，并选择最相关文档组成段落候选集，最后从这些段落候选集中抽取出精准的答案。这类自动问答系统主要涉及信息检索和自然语言处理两个领域的技术，主要的难点在于如何从非结构化的文本中抽取出答案[38]。与基于知识图谱的自动问答系统相比，这类自动问答系统的一个巨大优势是非结构化文档的数据源易于获得。

根据自动问答系统面向的任务可将其分为开放域自动问答系统和封闭域自动问答系统。开放域自动问答系统需要回答的是通用的问题，并不局限于某个领

域。这类问答系统对数据源的要求比较高，数据源需要囊括各个方面的文档。通常可以直接使用万维网做数据源。此外，开放式问题和开放域文档的句式和语法结构变幻莫测，难以使用简单的模式和规则匹配来处理，因此，这类自动问答系统的技术难度往往比较大。封闭域自动问答系统需要回答的是某个领域的问题，技术难度相对于开放域自动问答系统要小很多。

本文的研究方向是基于非结构化文档的开放域自动问答系统，数据源是非结构化的文档库，面向的问题是通用问题。因此，本文后续内容只对这种类型的自动问答系统的研究进行综述，并针对该类自动问答系统中存在的问题提出相应的解决方案。

基于非结构化文档的开放域自动问答系统主要涉及信息检索和自然语言处理两个领域的技术。一个典型的基于非结构化文档的开放域自动问答系统通常由三个模块构成，分别是问题处理模块，文档处理模块和答案处理模块。每个模块又有一个核心子模块。其中，问题处理模型的核心是问题分类，文档处理模块的核心是信息检索，答案处理模块的核心是答案抽取。问题处理模块将问题进行分类，并从问题中提取关键词，方便后续进行信息检索。文档处理模块是根据输入的关键词从万维网或者其他数据源中检索出相关文档和网页，并将这些文档切分成段落，形成段落集合。答案处理模块是将段落集合处理成候选答案，然后根据一些规则从候选答案中选择出一个正确的答案返回给用户。问题处理模块和答案处理模块主要使用自然语言处理技术，比如词性标注、命名实体识别、句法分析和依存关系分析等。文档处理模块主要使用信息检索技术，比如文档相似性计算等。答案抽取模块往往使用人工设计的模式和规则来匹配答案。

这种典型的由三个模块组成的基于非结构化文档的开放域自动问答系统主要存在以下问题：

1. 文档处理模块返回的段落候选集规模过大。

为了保证答案包含在段落候选集当中，文档处理模块往往需要返回很多候选段落，然后将这些段落传递给答案抽取模块。然而，这些段落中大部分的内容都是非答案的文本，会严重降低答案抽取模块的准确率。

2. 基于规则的答案抽取过于繁琐。

目前成熟的基于非结构化文档的开放域自动问答系统基本上都是使用人工设计的模式和规则进行答案抽取。模式千变万化，规则千差万别，不仅计算复杂而且可扩展性差，不够灵活。

本文针对以上两个问题，提出了相应的解决方案。接下来的章节进行详细叙述。

1.2 本文的主要工作

典型的基于非结构化文档的开放域自动问答系统一般由问题处理模块，文档处理模块和答案处理模块三个部分组成，如图 1.1 左半部分所示。上一节提到，文档处理模块返回的段落候选集的规模过大，而且大部分内容都是非答案文本。过多的候选答案会严重降低答案抽取的准确率。此外，基于规则的答案抽取技术过于繁琐、可扩展性差、不够灵活。针对这两个问题，本文提出了如下的解决方案：

1. 缩减段落候选集规模。

本文在文档处理模块后使用句子筛选模块将候选段落集合缩减为候选句子的集合。然后使用句子排序模块对候选句子进行排序，选择最可能包含答案的句子。最后对选择的单个句子进行答案抽取。如图 1.1 右半部分。

2. 使用端到端的深度神经网络进行答案抽取。

本文放弃了传统答案处理模块中繁琐的人工设计规则和模板的抽取算法，引入了端到端的深度神经网络模型，该模型刚好能够适用于从短文本中抽取答案的任务。如图 1.1 右半部分。

进一步做句子筛选和排序其实很符合人类寻找答案的过程。人类如果需要在一篇包含多个段落的文章中寻找一个问题的答案的话，也就是我们平常说的阅读理解。一般的过程是先定位句子，即先找到答案所在的句子后，再通过分析推理出答案。本文中提出的自动问答系统是对传统的自动问答系统的一次改进。现将本文的主要工作归纳如下：

1. 本文引入了句子筛选和句子排序模块来缩减段落候选集规模，提高了答案抽取的准确率。

2. 针对句子筛选模块，本文改进了一种计算文档相似度的算法WordMover's Distance(WMD)，并将其用于问句和候选答案句子的相似度计算来筛选句子，并提出了一种将 BM25 和 WMD 结合的混合模型。本文分别进行了文档分类和文本排序实验，并对比了一系列计算文档相似度的算法。实验结果表明，改进后的WMD 算法和混合模型能够在计算文档相似度方面比原有的 WMD 以及其他算法有更好的效果。

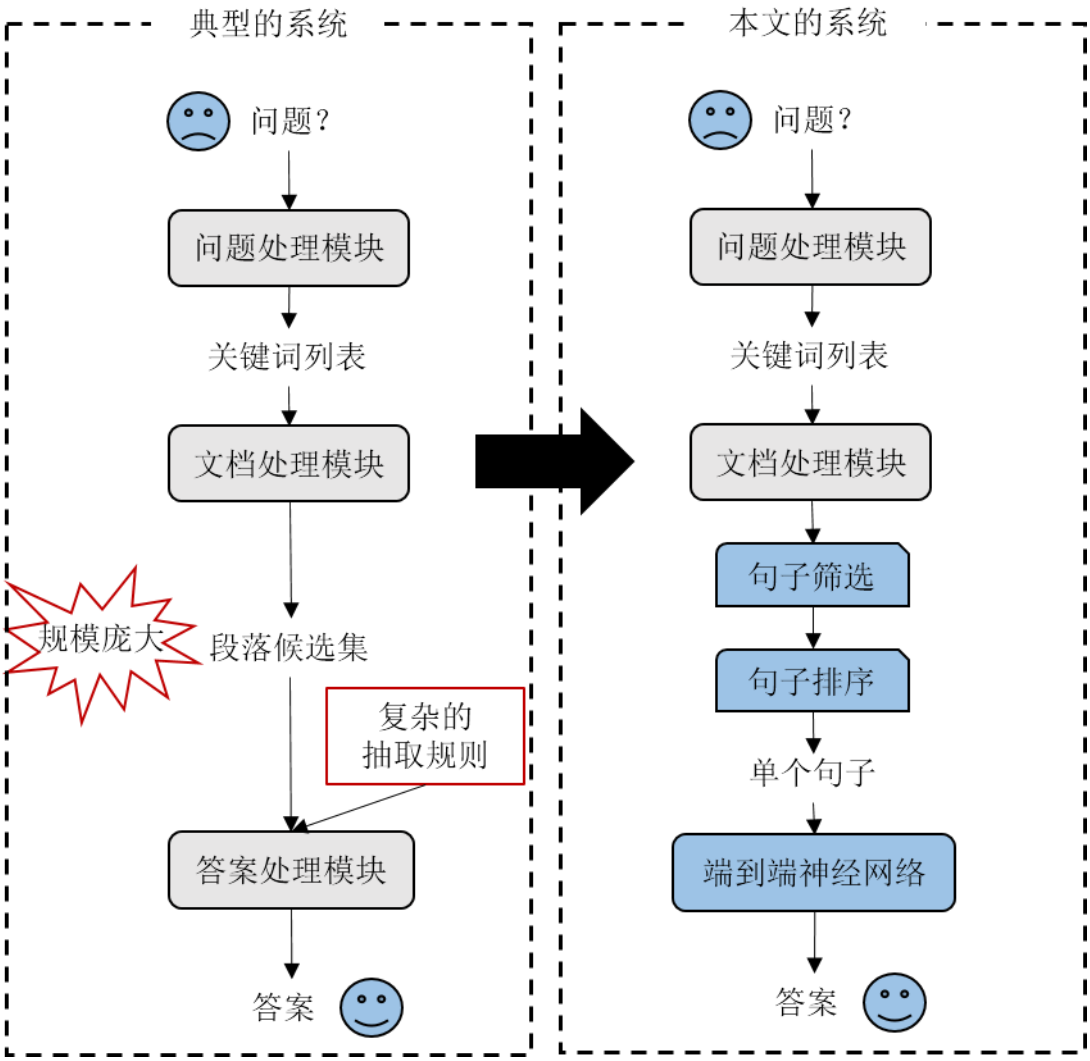

图 1.1 基于非结构化文档的开放域自动问答系统流程对比

3. 针对句子排序模块，本文设计了五种特征来衡量问句和候选答案句子之间的相关性，并以此相关性得分对候选答案句子进行排序。这些特征包含了不同的级别，分别是单词特征、短语特征、句子语义特征、句子结构特征、答案类型特征。我们将该模型称为 Multiple Level Feature Rank(MLFR)模型。基于 Stanford Question Answering Dataset（SQuAD）数据集，本文测试并对比了若干基于深度神经网络的句子排序模型。实验结果表明，MLFR 模型在各个评价指标上（MRR, p@1 和 p@3）都要比其他的基准模型有更好的排序效果。

4. 本文引入了一个端到端的深度神经网络模型用于答案抽取，并将此模型与前面的句子筛选和句子排序模块组合在一起，设计实验进行了整体的性能评估。实验结果表明，本文提出的以上改进方法能够提高答案抽取模块在两个指标（精确匹配，F1 分数）上的得分。

1.3 本文的组织结构

本文的组织结构如下：

第一章介绍了本课题的研究背景、研究意义以及本文所做的工作。简单介绍了传统搜索引擎存在的问题，并介绍了自动问答系统的发展。介绍了自动问答系统的分类。介绍了典型的基于非结构化文档的开放域自动问答系统存在的问题并针对这些问题，提出了相应的解决方案，并在本文所做工作中做了阐述。

第二章对基于非结构化文档的开放域自动问答技术进行了相关综述。首先介绍了基于非结构化文档的开放域自动问答系统的各个组成模块和整体的框架。接着详细介绍了各个模块的功能并对相关研究工作进行了综述。

第三章介绍了基于文档相似度的句子筛选算法的设计与实现。首先介绍了传统的计算文档相似度的算法 BM25，然后介绍了最新的计算文档相似度的算法 WMD。经过分析两种算法的优缺点，我们提出了一种改进的 WMD 算法，同时提出了一种混合模型。最后分别在文本分类和文本排序任务上进行了对比实验。

第四章介绍了句子排序模型的设计与实现。首先介绍了句子排序的任务需求，然后提出了一种基于多级特征匹配的排序模型 MLFR，该模型分别从单词特征、

短语特征、句子语义特征、句子结构特征和答案类型特征五个级别来计算答案句子和问题句子之间的相关性得分。最后与若干基于深度神经网络的句子排序模型在文本排序任务上进行了对比实验。

第五章介绍了答案抽取模型的设计与实现。主要介绍了一种基于深度神经网络的端到端的模型，该模型可以用于从短文本中抽取答案的任务。并将此模型与前面的句子筛选和句子排序模块组合在一起，设计实验进行了整体的性能评估。

第六章对本文的工作进行了总结，并指出本文提出的自动问答系统可能存在的问题以及可以进一步优化的方向和对未来的展望。

1.4 本章小结

本章首先介绍了本课题的研究背景、研究意义以及本文所做的工作。简单介绍了传统搜索引擎中存在的问题，并介绍了自动问答系统的发展。介绍了自动问答系统的分类。介绍了典型的基于非结构化文档的开放域自动问答系统存在的问题并针对这些问题，提出了相应的解决方案，并在本文所做工作中做了阐述。最后介绍了本文的组织结构。

第2章 基于非结构化文档的开放域自动问答技术综述

2.1 自动问答系统的发展

自动问答系统是一种输入自然语言问句后能够直接返回精确答案的系统。同时，自动问答是一个结合了信息检索（IR）、信息抽取（IE）和自然语言处理（NLP）三个研究领域的一种技术。传统的信息检索系统或者搜索引擎仅仅是“文档检索”，比如用户输入查询关键词后，搜索引擎返回的仅仅是包含这些关键词的一系列排序后的文档。查找答案的工作要靠用户自己完成，这对用户来讲是一件很耗时间和精力的事。用户真正需要的是能够直接返回精确答案的系统。自动问答系统正是在这种需求下应运而生。

1999 年，第八届文本信息检索大会（TREC-8：Text Retrieval Conference）首次增加了一个自动问答系统的竞赛。这个竞赛的初衷就是为了测试自动问答系统回答自然语言问题的能力（比如：“How many calories are in a Big Mac”）[1]。这届的自动问答系统竞赛结果良好、意义重大，最好的系统能达到 60%多的正确率，也就是说每三个问题，系统就能从规模庞大的语料库中大海捞针般搜寻出两个正确答案。作为学术界在开放域自动问答系统的第一次尝试，这是个非常令人鼓舞和振奋的结果。新式的自动问答系统就这样走上了历史舞台。在 2000 年的 ACL 国际计算语言学学术会议上，有一个专题的讨论会，题目就是“Open-Domain Question Answering”。在 2002 年的 CLEF（Cross Language Evaluation Forum）和 NTCIR（NII Test Collection for IR Systems）会议上分别开展了多语言和跨语言自动问答的进程[2]。

在学术界和工业界的共同努力下，同时也得益于自然语言处理领域一些基本的算法的成熟，比如词性标注（POS）、命名实体识别（NR）、句法分析和依存关系分析等，自动问答系统很快就取得了立竿见影的效果。2011 年，IBM 自动问答系统 Watson 参加综艺节目危险边缘（Jeopardy）来测试它的能力，这是该节目有

史以来第一次人与机器对决。Watson 也奇迹般的击败了人类，成为了冠军。

近年来自动问答系统一直处于研究的前沿，得到了学术界和工业界的普遍关注[2]。2013 年 Tomas Mikolov 的两篇论文[39][27]提出了两种计算词向量的模型，分别是 Continuous Bag of Words（CBOW）模型和 Skip-grams（SG）模型。同年，Google 基于这两篇论文实现了 word vector 的工具包。2013 年 Socher 提出使用递归神经网络（Recursive Neural Network）对情感分类问题进行建模[40]。2014 年 Kalchbrenner 在 ACL 上的一篇论文第一次将 CNN 用到 NLP[41]。2014 年 Quoc Le 和 Mikolov 将 CBOW 和 Skip-grams 模型的思想应用到文档段落的向量化表达[42]。从 Recurrent Neural Network（RNN）到 Long Short Term Memory（LSTM）[30]再到带有 Attention 机制的 LSTM[43]，深度学习模型对句子语义的理解能力越来越强，句子匹配的精确度越来越高。深度学习的迅猛发展加快了自动问答系统的革新的步伐，自动问答系统的性能也在快速提高。

根据自动问答系统背后的数据源的类型可以将其分为两类，分别是基于知识图谱的自动问答系统和基于非结构化文档的自动问答系统。根据自动问答系统面向的任务也可以分为两类，分别是开放域（open-domain）自动问答系统和封闭域（closed-domain）自动问答系统。第一章绪论中已经介绍了各种类型的自动问答系统的区别，这里不再介绍。本文的研究方向是基于非结构化文档的开放域自动问答系统。基于非结构化文档的开放域自动问答系统中数据源是非结构化文档库，面向的问题是通用问题，不局限于某个领域。因此，本文后续内容只对这种类型的自动问答系统的研究进行综述。

2.2 基于非结构化文档的开放域自动问答系统的组成

2.2.1 整体架构

如图 2.1 所示。一个典型的基于非结构化文档的开放域自动问答系统包含三个独立的模块，每一个模块本身又包含了一个核心模块和若干个辅助模块。问题处理模块的核心是问题分类，文档处理模块的核心是信息检索，答案处理模块的

核心是答案抽取。

问题处理模块负责识别问题的关键词，将问题进行分类，同时推理出问题所需的答案类型，问题重述模块将原问题转化成一个查询关键词的列表，并对这个关键词列表进行查询扩展，这样能够提升信息检索模块的召回率（recall）。文档处理模块负责检索出包含关键词的文档，并将这些文档切分成段落，然后进行段落筛选，返回段落候选集。召回率是信息检索模块中非常重要的一个衡量指标。因为如果返回的结果中没有正确答案的话，后续答案处理工作也就没有任何的意义[3]。同时，信息检索系统的精确率（precision）和候选文档的排序也对问答系统的性能有着重要影响。答案处理模块是基于非结构化文档的开放域自动问答系统最后一个组成部分，也是自动问答系统区别于信息检索系统的重要特征。答案处理模块负责从段落候选集中抽取出答案，在自动问答系统中起着决定性的作用[4]。

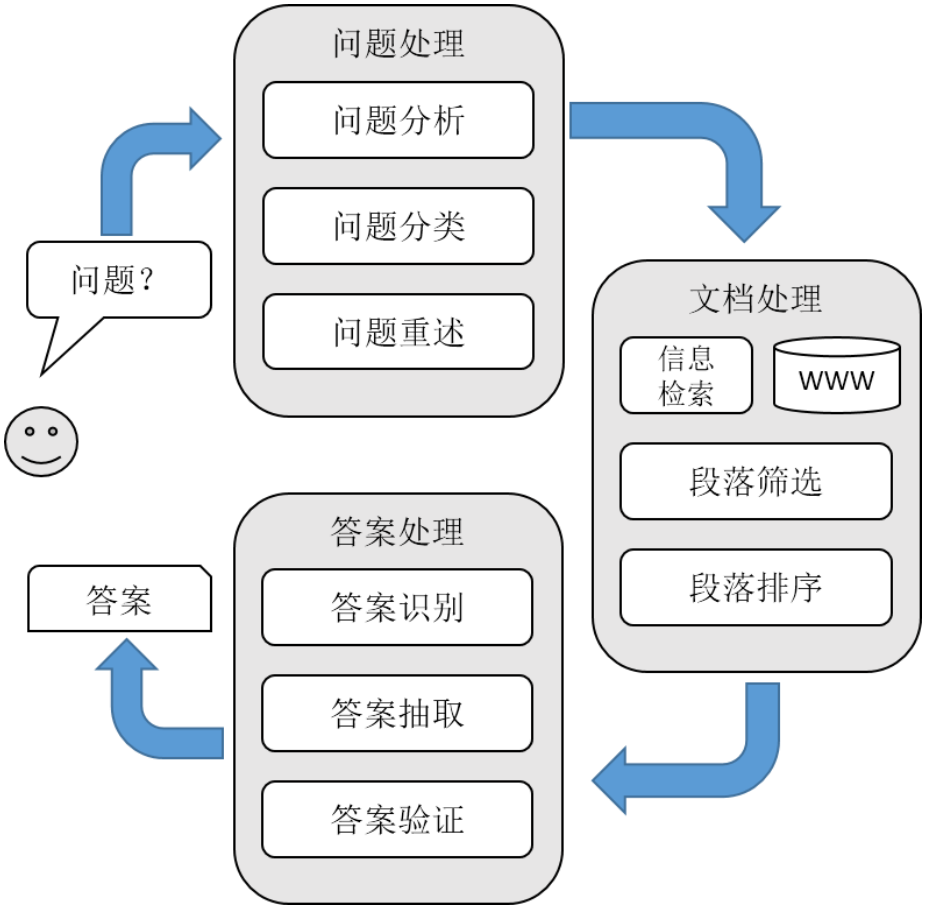

图 2.1 基于非结构化文档的开放域自动问答系统架构图

总的来讲，基于非结构化文档的开放域自动问答系统的整个流程如下：

1. 首先，用户输入一个自然语言问题。
2. 接着问题分析模块找出问题的关键词。
3. 问题分类模块将识别问题类型，以此来推理出所需答案的类型。
4. 问题重述模块将问题转化成一个关键词列表，并对这个列表进行查询扩展，将结果传递给信息检索模块。
5. 信息检索模块根据问题处理模块提供的关键词列表进行搜索，返回一些包含关键词的文档。
6. 检索出的文档被分割成段落，并通过筛选，只保留那些包含关键词的段落。
7. 接着筛选后的段落，经过排序，传递给接下来的答案处理模块。
8. 根据答案类型和其他一些识别方法，答案识别模块能够生成出候选答案集合。
9. 答案抽取模块定义一系列启发式的算法，只抽取与问题相关的单词或者短语。
10. 最终抽取出的答案经过答案验证模块验证正确性，然后返回给用户。

2.2.2 各模块功能综述

2.2.2.1 问题处理模块

给一个自然语言问题，问题处理模块的总体任务就是分析问题，并将问题重新描述为信息检索系统所需要的查询关键词列表。因此，问题处理模块需要具备以下功能：

1. 问题分析，找出问题表达的主要信息的关键词。
2. 问题分类，将问题进行归类，同时能够得到所对应的答案类型。
3. 问题重述，将问题转化成一个查询关键词的集合，并对这个集合进行同义词扩充，提供给后续的信息检索系统。

问题分析也可称为“问题关键词识别”。对于一个问题，如果我们仅仅知道问题的类型是不足以找出答案的，因为问题类型本身含有的信息量很少，比如“what”问题可能包含了各种各样、千奇百怪的问法和单词。因此，问题分析必

须能够识别出问题的关键词。一个问题的关键词被 Moldovan 定义为能够表达问题信息的一个单词或者多个单词的序列[5]。比如问题“What is the longest river in New South Wales?”问题的关键词就是“longest river”，如果问题类型（问题分类模块给出）和问题关键词都知道后，系统就能比较容易推断出所需答案的类型。识别问题关键词可以使用一些模式匹配的规则。

问题分类中常见的问题类型包括：what，why，who，how，when，where 等。比如问题“Who invented the first computer?”属于“Who”类问题。问题分类同时包括判断所需答案的类型。系统内部本身有个问题类型和答案类型的对应表，比如上面问题的答案类型很明显是个“人名”，因此我们在答案抽取时只关注被标记为人名的词或短语。值得一提的是，由于答案类型与问题类型并非严格一一对应，因此通常答案类型包括多个。

问题重述是在问题分析的基础上进行的，利用一些比较成熟的技术，包括命名实体识别，停止词检测以及词性标注等来抽取出查询关键词的集合，并利用 WordNet 的同义词集对查询关键词集合进行扩充。将关键词集合传递给后续的信息检索系统。图 2.2 显示了问题处理模块的整个流程。

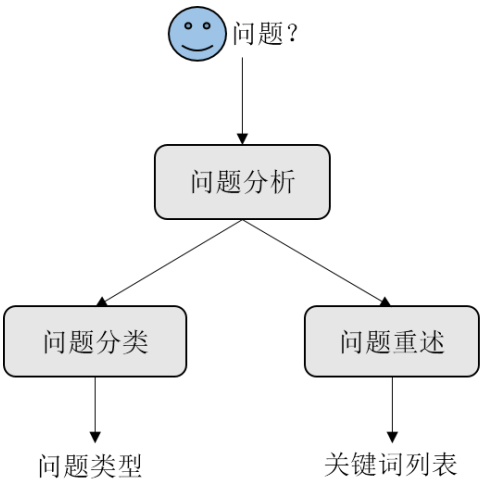

图 2.2 问题处理模块流程图

2.2.2.2 文档处理模块

文档处理模块也可以被称为段落索引模块。问题重述模块提供的问题关键词

列表被提交到信息检索系统中，系统从大量的语料库比如WWW（World Wide Web）来收集相关文档并对文档进行排序[6]。最终信息检索系统需要返回的是排序好的段落，因此文档处理模块需要具备以下功能：

1. 检索出与问题关键词相关的文档并排序。
2. 将文档切分成段落，并进行筛选生成候选的段落集合。之所以切分成段落，是为了加快处理速度，减少系统的响应时间。
3. 根据段落集合包含答案的合理程度进行重新排序。

信息检索系统是用来找出与问题相关的文档，并根据相关性进行排序，对于开放域的问题，信息检索可能要搜索整个万维网来搜集文档。信息检索系统通常使用两个评价指标，精确率和召回率。精确率代表返回的所有文档中相关文档的比例，召回率代表返回的相关文档占所有相关文档的比例。通常这两个目标是同时被优化的。

信息检索系统返回的文档数量会比较多，段落筛选能够用来过滤掉一些文档，以及文档中的某些段落，来缩小候选集的规模。段落筛选的依据是认为最相关的文档应该有连续的段落包含关键词，也就是说关键词分布在文档中连续的段落中，而不是散乱的分布在各个段落里。所以段落筛选的方式就是如果关键词出现在连续的段落里，那么这些段落将会被保存，如果某个文档中不存在这些段落，那么这个文档将会被删除。

段落排序通常是根据段落中相同单词序列出现的次数来对段落进行评分，比如问题“Who invented the first computer?”对应的查询关键词是“invented, first, computer”，而且这三个关键词是有顺序的。因此，我们制定一个滑动窗口，从头到尾遍历某个段落，计算这些关键词按照相同顺序在此段落中出现的次数，以此来作为段落的得分，并对所有段落进行打分，然后根据标准的基数排序对所有段落进行排序。图 2.3 显示了文档处理的整体流程。

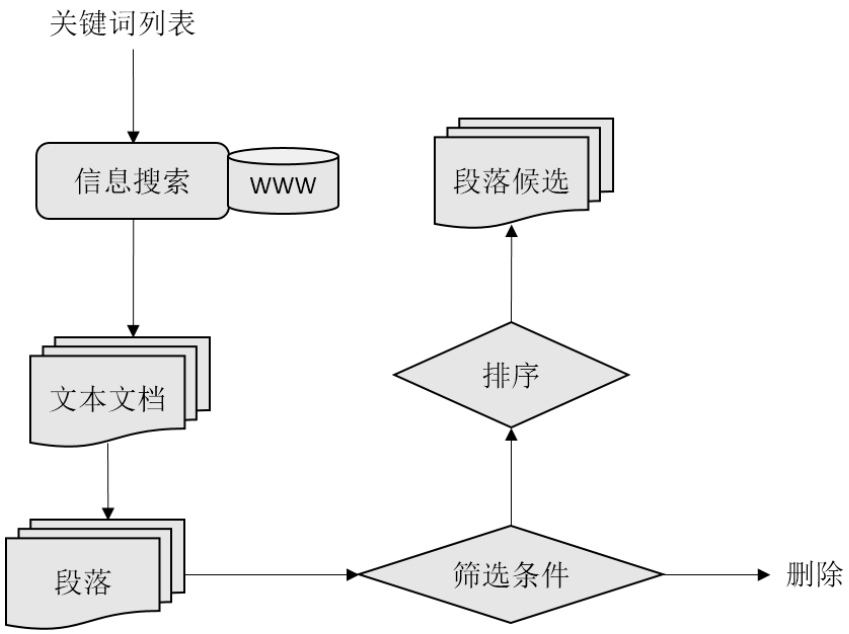

图 2.3 文档处理模块流程图

2.2.2.3 答案处理模块

答案处理模块是基于非结构化文档的开放域自动问答系统中最后一个模块，它的作用是识别抽取答案，经过验证后返回给用户。答案处理模块的输入是排序后的段落集合，输出是精确的答案，具备以下三个功能：

1. 解析候选段落，并从中识别出候选答案。
2. 利用一些启发式算法在候选答案集合中抽取能够回答问题的答案。
3. 验证抽取出的答案是否合理，若合理则返回给用户。

候选答案的识别主要依赖于问题分类模块推理出的答案类型。通常答案类型并不是很直接可以得到，需要借助于命名实体识别以及词性标注等技术来实现。从候选段落中找出与答案类型一致的词或者短语，形成候选答案集合。

答案抽取其实就是从候选答案集合中找出能够回答问题的答案。研究人员提出了一系列启发式的算法来筛选答案。比如根据候选答案到关键词的距离，关键词匹配个数等相似的启发式规则。如果答案抽取模块不能找出答案，系统则直接将排名第一的段落直接返回给用户。直接返回段落的这种方式目前在 TREC 竞赛中已经不能使用，因为原来的竞赛中允许返回多个句子或者答案，评价标准是真

正的答案在输出答案列表中的排名。但是在2002年以后，竞赛规定只能输出一个答案[7]。

最后，要对筛选出的答案进行验证。验证方式有很多，比如可以使用词汇语义网络WordNet等来进一步验证答案的类型。也可以使用一些特定领域的知识库来进一步验证答案。如果特定领域的知识库已经被用来做信息检索，那么可以用整个互联网搜索进行答案验证。这种验证方式的理论依据是，答案和问题同时出现在文档中的次数越多，答案也就越可靠。基于以上理论，一些学者利用剩余的网页文档来验证答案取得了不错效果。图 2.4 显示了答案处理模块的整体流程。

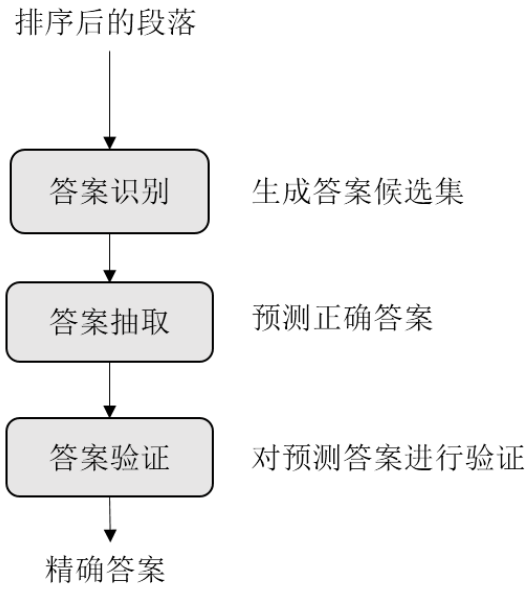

图 2.4 答案处理模块流程图

2.3 相关研究工作

自动问答系统中有三个核心模块，分别是问题分类，信息检索和答案抽取。因此我们分别对这三个模块的研究工作进行综述。

2.3.1 问题分类

问题分类通常是依赖于已经建立好的分类目录表，分类目录表有两种形式，

扁平式和层级式。扁平式分类目录中每个类没有子类，层级式分类目录中每个类包含了多级的子类。 Radev 的 NSIR 系统中使用的就是扁平式分类目录[8]，如表 2.1。Moldovan[5]使用了层级式的分类目录如表 2.2 所示。

问题类别有了之后，如何将问题映射到类别中去，一般有两种方法。基于规则的，比如以“Where”开始的句子对应的答案类型是“LOCATION”，另外一种方法是基于机器学习的。首先要为问句设计特征，比如单词、形态、语法、语义等特征。分类器可以使用支持向量机（SVM），最近邻（NN），决策树和朴素贝叶斯等，当然也可以使用近年来最火的深度学习，同时能够自动提取特征。

表 2.1 扁平式分类目录

PERSON	PLACE	DATE
NUMBER	DEFINITION	ORGANIZATION
DESCRIPTION	ABBREVIATION	KNOWNFOR
RATE	LENGTH	MONEY
REASON	DURATION	PURPOSE
NOMINAL	OTHER	

表 2.2 层级化分类目录

问题类型	问题子类	答案类型
WHAT	basic-what	Money/Number/ Definition/Title/NNP /Undefined
	what-who	
	what-when	
	what-where	
WHO		Person/Organization
HOW	basic-how	Manner
	How-many	Number
	How-long	Time/Distance
	How-much	Money/Price
	How-much<modifier>	Undefined

（续表 2.2）

问题类型	问题子类	答案类型
	How-far	Distance
	How-tall	Number
	How-rich	Undefined
	How-large	Number
WHERE		Location
WHEN		Date
WHICH	which-who	Person
	which-where	Location
	which-when	Date
	which-what	NNP/Organization
NAME	name-who	Person/Organization
	name-where	Location
	name-what	Title/NNP
WHY		Reason
WHOM		Person/Organization

2.3.2 信息检索

信息检索通常有两种方式，第一种是可以直接使用通用的搜索引擎，比如 Google，AltaVista 等，另外一种方式是自己设计关键词匹配算法来搜索本地语料库。两种方式各有优缺点，通常可以结合两种方式一起使用。直接使用现有的搜索引擎好处是方便，语料库涵盖范围非常广泛，可以搜索到大概 10 亿级规模的文档，而且速度快，不用自己建立索引等，缺点是通用的搜索引擎通过计算文档向量之间的余弦距离来排序文档，而问答系统的信息检索是希望基于完全的关键词匹配来排序文档，就是说，余弦距离小的文档，不一定代表关键词匹配程度高[6]。与此对应，自己设计匹配算法往往可以比较准确的匹配关键词，但是要求语料库规模比较小，适合于本身答案是从一个固定的语料库只抽取的情景，比如封闭域的自动问答系统。

对于使用 Google 来检索信息的时候，搜索引擎本身会检索一些专业的资源，比如亚马逊，IMDB 和 CIA World Fact Book 等，这些专业资源能够用于一些相关专业的问题，来提高系统的精确率和召回率。还有一些方法可以改善搜索结果，比如可以为每篇文档自动添加标签、主题、注释等。并将这些标签一并建立索引，当进行文档检索的时候，可以将标签与问题的答案类型进行匹配来提高文档检索的精确率和召回率。自动打标签的计算复杂度，以及为标签建立索引的比较高，因此适合于离线来做[6]。

我们希望检索出的文档是包含关键词的，但是单纯的单词匹配，比如使用 BM25 等算法，还是存在一些缺陷。研究学者们也意识到了简单的关键词匹配存在的问题，纷纷做了改进。比如 2008 年 Stoyanchev 等人的一篇 ACL 会议论文[3]提出了基于命名实体、名词短语、动词短语、介词短语作为精确匹配，能够大幅度提高信息检索的精确率和召回率。同年，Kangavari 等人[9]提出了一种新颖的、简单的方法。其论文中的自动问答系统是基于一个特殊的知识库，这个知识库本身就是由问句和答案组成的。因此，直接匹配新问题和知识库中的问题，然后将匹配成功的知识库中的问题对应的答案直接返回给用户。这种方式很简单，主要的技术体现在两个句子匹配上。后续有很多研究工作都是基于这种思想展开的，尤其是在深度学习发展起来之后，基于深度学习的两个自然语言句子匹配受到了越来越多的研究，比如 2015 年 IBM Watson 团队的 Minwei Feng 等人[10]提出了用单层 CNN 和多层 CNN 来进行句子匹配的深度学习框架。德国慕尼黑大学的 Wenpeng Yin 等人[11]考虑生成句子表达的时候使用带有关注机制的 CNN（Attention-based CNN），并将不同粒度的特征同时考虑。2015 年中国科学院大学的 Shengxian Wan 等人[12]使用带有关注机制的双向 LSTM 来匹配两个句子。

2.3.3 答案抽取

2002 年 Ravichandran 和 Hovy 的一篇 ACL 会议论文[13]根据文本的表面信息，手工构建一些表面模式来抽取答案，其他一些研究者，比如 2003 年 Xu 等人[14]使用一些自动获取文本模式的方法来抽取答案。2005 年剑桥大学的 Peng 等人[15]使

用语言结构提出了一些能够解决长期依赖性的文本模式。2005 年中国台湾中央研究院的 Lee 等人[16]使用了命名实体识别等方法来寻找答案。

除了模式匹配外，还有很多基于深度学习的自动寻找答案的研究，这些方法大多是从候选答案中选择一个答案，本质上是一个排序问题。使用一些深度学习模型来自动提取特征，计算问句和候选答案之间的相关性。2016 年 IBM Watson 团队的 Yang Yu 等人[17]就直接使用一个端到端的神经网络模型来进行答案抽取和对答案候选集的排序。2016 年 Shuohang Wang 等人[18]基于 LSTM 提出了一种 match-LSTM 来对候选答案进行排序和抽取。2016 年斯坦福大学的 Rajpurkar 等人[19]公布了一个规模庞大的数据集 SQuAD，SQuAD 既可以用于自动问答，也可以用于阅读理解，同时他在文章中设计了各种基于文本和语法依赖关系的特征，并使用了逻辑回归来对答案候选集进行排序。表 2.3 是对前面综述的研究工作与其对应的自动问答系统中的各个模块的总结。表 2.3 来源于 2012 年 Allam 等人在 IJRRIS 上发表的一篇综述自动问答系统的论文[38]，同时本文添加了一些新的研究成果。

表 2.3 研究工作与自动问答系统模块对应表

	问题处理			文档处理			答案处理		
	问题分析	问题分类	问题重述	信息检索	段落筛选	段落排序	答案识别	答案抽取	答案验证
Gaizauskas (QA-LaSIE) [20]		✓		✓			✓	✓	
Harabagiu (FALCON) [21]		✓	✓	✓	✓	✓	✓	✓	✓
Hermjakob [22]		✓					✓		
Kangavari [9]	✓	✓	✓	✓	✓	✓	✓	✓	✓
Lee (ASQA)[16]	✓	✓		✓			✓	✓	
Li & Roth [23]		✓					✓		

（续表 2.3）

	问题处理			文档处理			答案处理		
	问题分析	问题分类	问题重述	信息检索	段落筛选	段落排序	答案识别	答案抽取	答案验证
Moldovan (LASSO) [5]	✓	✓	✓	✓	✓	✓	✓	✓	✓
Peng[15]		✓		✓			✓	✓	
Radev (NSIR) [8]		✓		✓			✓	✓	
Ravichandran & Hovy [13]		✓		✓	✓		✓	✓	
Stoyanchev (StoQA) [3]	✓	✓	✓	✓			✓	✓	
Xu [14]		✓		✓			✓	✓	
Zhang & Lee [24]		✓		✓			✓	✓	
Minwei Feng[10]	✓	✓	✓	✓	✓	✓	✓	✓	✓
Wenpeng Yin[11]	✓	✓	✓	✓	✓	✓	✓	✓	✓
Shengxian Wan[12]	✓	✓	✓	✓	✓	✓	✓	✓	✓
Yang Yu[17]							✓	✓	✓
Shuohang Wang[18]				✓	✓	✓	✓	✓	✓
Rajpurkar[19]							✓	✓	✓

2.3.4 评价标准

主要的评价标准有四个，分别是精确率、召回率、F 值、平均倒数排名（MRR）。其中信息检索系统中常用精确率、召回率、F 值作为评价标准。精确率代表返回的所有文档中相关文档的比例，召回率代表返回的相关文档占所有相关文档的比例。F 值是精确率和召回率的结合，可以保证精确率和召回率同时被优化。答案抽取的时候常用 MRR 作为评价标准，同时也会用到准确率和 F 值。

$$精确率 = \frac{正确的答案数}{返回的答案数} \qquad 公式（2.1）$$

$$召回率 = \frac{正确的答案数}{所有的答案数} \quad 公式（2.2）$$

$$F\ 值 = \frac{精确率 * 召回率 * 2}{精确率 + 召回率} \quad 公式（2.3）$$

$$MRR = \sum_{i=1}^{M} \frac{1}{r_i} \quad 公式（2.4）$$

M 代表样本个数，r_i代表第i个答案的排名。

2.4 本章小结

本章主要是对基于非结构化文档的开放域自动问答系统的发展和技术做了综述。首先介绍了自动问答系统的发展过程，主要是随着互联网的普及，人们的对搜索的需求也越来越高，希望能够通过自然语言问题直接获得精确的答案，也正是这种需求的驱动，同时得益于 TREC 的竞赛，自动问在近十几年得到了飞速的发展。

接着我们介绍了基于非结构化文档的开放域自动问答系统的组成，基于非结构化文档的开放域自动问答系统一般包括问题处理模块，文档处理模块和答案处理模块三个部分。其中，问题处理模块又包括问题分析，问题分类和问题重述，问题分类是问题处理模块的核心。文档处理模块包括信息检索，段落筛选和段落排序三个部分，其中信息检索是文档处理模块的核心。答案处理模块分为答案识别，答案抽取和答案验证三个部分，其中答案抽取时答案处理模块的核心。接着，我们分别介绍了各个模块的具体功能。然后，我们对基于非结构化文档的开放域自动问答系统中的三个核心模块的技术进行了综述，最后我们介绍了基于非结构化文档的开放域自动问答系统中的评价指标，分别是精确率，召回率，F 值和平均倒数排序。

1999-2007 这八年期间，自动问答系统的研究性工作层出不穷，主要要感谢 TREC 举办的自动问答的竞赛。2012 年至今，深度学习在计算机领域的各个方向都开始取得非常好的性能，因此近年来也涌现了非常多的，高品质的深度学习方

面的自动问答技术。

最后要说的是，本文作者能力有限，文章篇幅也有限，不能涵盖有关该领域所有的研究工作。本章综述的都是些已经在顶级会议或者期刊上发表的文章。自动问答系统也不局限于本章介绍的框架，深度学习的发展使得问答系统的框架越来越简单。

第3章 基于句子相似度的句子筛选模型

3.1 任务描述

在第二章中，我们对基于非结构化文档的开放域自动问答的组成和相关技术进行了综述。基于非结构化文档的开放域问答系统的一个核心就是信息检索模块。信息检索模块的输入是查询关键词列表，从丰富的语料库中检索出相关的文档，这一步的检索过程一般是基于现有的搜索引擎和一些标准的关键词匹配算法对规模庞大的文档库中的文档进行排序，并按照顺序返回。由于返回的文档规模过于庞大，因此为了加快自动问答系统的响应速度，通常会将文档切分成段落，并将段落进行筛选和排序，返回排序好的段落的集合。段落排序通常是根据段落中相同单词序列出现的次数来对段落进行评分。在得到排序后的段落后，很多自动问答系统就直接选择排名最高的段落，并将其传递给答案处理模块，答案处理模块一般是通过一些启发式的匹配模式来抽取答案，或者是将段落进行分词生成候选答案集，然后将答案抽取转换成一个排序问题，排名第一的就是答案。这里存在一些问题，首先只选择排序最高的段落并不能保证答案就在这个段落中，为了保证比较高的可能性，我们必须选择多个段落作为候选，这样又使得候选段落集比较大，无论是用模式匹配还是使用机器学习算法都不能达到很高的准确率。因此，我们在段落集合的基础上进一步做了句子筛选。

进一步做句子筛选其实很符合人类寻找答案的过程。人类如果需要在一篇包含了多个段落的文章中寻找一个问题的答案的话，也就是我们平常说的阅读理解，一般的过程是先定位句子，即先找到答案所在的句子后，然后通过分析推理出答案。句子筛选模块就是用来将冗长的段落处理成简短的句子的集合。图 3.1 描述了文档处理模块与句子筛选模块之间的关系。简单来说就是，文档处理模块检索出排序后的段落候选集。句子筛选模块从这些段落中检索出少量候选句子。筛选过程分成三步，首先将段落候选集切分成候选句子，然后计算问句与候选句子之

间的相似度，以此相似度得分对候选句子进行排序，从中筛选出排名最靠前的若干个句子。然后句子排序模块对这些少量的候选答案句子进行排序，选出答案所在的句子。最后答案抽取模块只在一个句子中抽取答案。

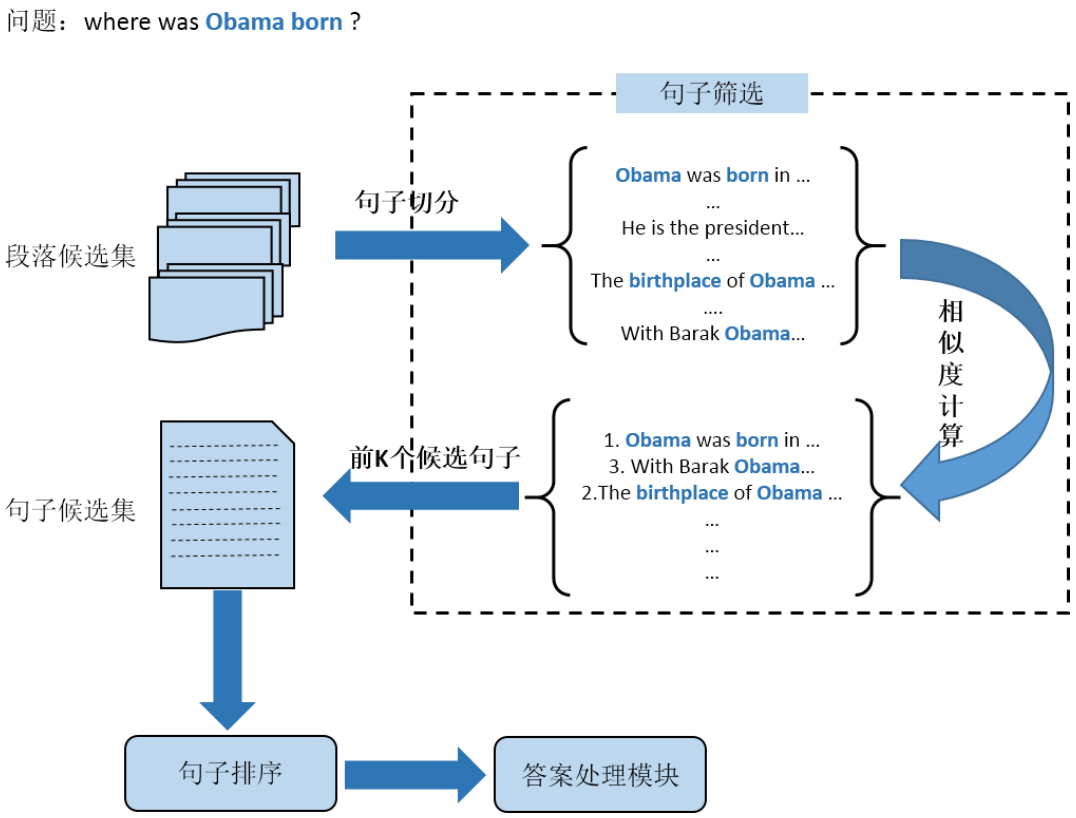

图 3.1 句子筛选功能图

3.2 基于句子相似度的句子筛选算法

华为诺亚方舟实验室的Ji等人[25]在论文中提到，虽然问题和答案句子通常包含相同的关键词这种说法并不绝对，但是一般来说与问题包含更多关键词的句子有更大的概率是正确的答案句子。因此，基于非结构化文档的开放域自动问答系统中的信息检索系统主要使用的算法就是关键词匹配。句子筛选模块也用到关键词匹配，同时针对单纯的关键词匹配算法中存在的问题，本文使用了新的相似度匹配算法。由图 3.1 可以看出，句子筛选模块最重要的部分是计算问题和候选句

子之间的相似度。

本章节主要介绍四种基于句子相似度的句子筛选算法。首先是完全基于关键词匹配的 BM25 算法。然后是基于词向量转移的 WMD（Word Mover's Distance）算法，并针对 WMD 中存在的问题，提出了改进算法。BM25 和 WMD 两种算法各有优缺点，因此本文提出了一种线性模型，将两种算法进行混合。接着在两个数据集上将改进后的算法分别用于文本分类和文本排序任务，并对比了一些基准计算文档相似度的算法。实验结果表明混合模型的效果比现有模型的效果都要好。

3.2.1 BM25 算法

BM25 算法是非常常用的计算文档相似度的算法。算法的输入是问题句子和文档库，输出是排序后的文档。主要思想是，将问题句子进行分词，生成查询项，然后对于每篇文档计算每个查询项与该篇文档的相关性得分。然后将所有查询项与该篇文档的相关性进行加权求和，得到该篇文档的相关性分数。最后，对于语料库中的所有文档执行相同的过程，从而得到每篇文档的相关性分数。BM25 的一般性公式如下：

$$Score(Q,d)=\sum_{i}^{n} W_i \cdot R(q_i,d) \qquad \text{公式（3.1）}$$

其中Q代表问句，d代表一篇文档，q_i代表问句解析后的一个查询项，W_i代表查询项q_i对应的权重，$R(q_i,d)$代表查询项q_i与文档d的相关性得分。

我们首先来看查询项权重W_i如何定义。简单来说就是如何判断一个查询项的对于这个问题的重要程度。计算单词重要度的方法有很多，最为常用的是 IDF（Inverse Document Frequency），IDF 的公式如下：

$$IDF(q_i)=log\frac{N-n(q_i)+0.5}{n(q_i)+0.5} \qquad \text{公式（3.2）}$$

其中，N代表全部的文档数，$n(q_i)$代表了包含q_i的文档数，根据 IDF 的定义我们可以看出，对于给定的文档集合，包含q_i的文档数越多，q_i的权重就越低。主要思想是认为越频繁出现的单词越不重要，比如一些停止词。

接着我们介绍q_i与文档d的相关性得分$R(q_i,d)$。$R(q_i,d)$的一般形式是：

$$R(q_i,d)=\frac{f_i\cdot(k_1+1)}{f_i+K}\cdot\frac{qf_i\cdot(k_2+1)}{qf_i+k_2} \quad \text{公式（3.3）}$$

$$K=k_1\cdot(1-b+b\cdot\frac{dl}{avgdl}) \quad \text{公式（3.4）}$$

其中，k_1，k_2，b为调节因子，根据经验设置，通常$k_1=2$，$b=0.75$。f_i是q_i在文档d中出现的频率，qf_i为q_i在问句中出现的频率。dl代表文档的长度，$avgdl$代表所有文档的平均长度。由于在大部分情况下，q_i在问句中只出现一次，即$qf_i=1$，因此公式可以简化为：

$$R(q_i,d)=\frac{f_i\cdot(k_1+1)}{f_i+K} \quad \text{公式（3.5）}$$

从K的定义来看，b的作用是调节文档长度的影响，当b越大，文档长度对相关性得分的影响就越大。当文档长度越大时，相关性也越大。

综合上面的公式，我们可以得到 BM25 算法的总的计算公式：

$$Score(Q,d)=\sum_{i}^{n}IDF(q_i)\cdot\frac{f_i\cdot(k_1+1)}{f_i+k_1\cdot(1-b+b\cdot\frac{dl}{avgdl})} \quad \text{公式（3.6）}$$

3.2.2 改进的 WMD 算法

单纯的基于关键词匹配的算法存在一个明显的缺陷，就是有时候两个句子很相似，但是却很少有共同单词。比如，“Obama speaks to the media in Illinois ”和“The president greets the press in Chicago”，这两个句子没有共同的单词，但是却表达的是相同的内容。而且，（Obama, president），（media, press），（speaks, greets），（Illinois, Chicago）都是同义的单词。事实上这样的例子很多，我们在一些阅读理解中也能经常看到，问题通常是文中某一句话的同义转述。这时候如果单纯的依赖于 BM25 等基于关键词匹配的方法，就得不到真正的答案句子。

针对上述问题，我们利用了基于词向量转移的 WMD（Word Mover’s Distance）算法[26]。首先需要一种方法来计算两个单词之间的相似度，比如 Obama 和 president 相似度就比较高，Obama 和 Chicago 相似度比较低。Mikolov[27]基于大量

的语料库，设计了 word2vec 模型来生成词向量。由于一个词的向量是有上下文单词决定的，因此上下文相似的两个单词所对应的词向量之间的余弦距离必定很小，两个词也是可以相互替换的同义词。Mikolov 也用实验证明了词向量能够很好的保持了源语言中的语义关系。比如，vec(Berlin) – vec(Germany) + vec(France) 的结果跟 vec(Paris) 距离比较小。vec(Japan) – vec(sushi) + vec(Germany) 跟 vec(bratwurst)距离比较小。

有了衡量两个单词距离的方法后，接着作者将两个句子相似度建模成从一个句子转化成另外一个句子，所需要的代价。更具体的来说就是，从一个词向量列表转移到另外一个词向量列表所需移动的距离。作者把这个问题类比成 Earth Mover's Distance（EMD）问题。EMD 和欧式距离一样，也是一种距离度量的定义，可以用来测量某两个分布之间的距离。EMD 问题本身已经有成熟的算法来解决。一种启发式的解释是，两种分布，一种是很多堆土，另外一种分布式很多个坑，如何花费最小的工作量来移动这些土将坑填满。这个问题的约束条件是每个土堆都有体积大小，每个坑也都有容量大小。对应到 WMD 问题的话，第一个句子里的词代表土堆，第二个句子里的词代表坑，从一个词转换到另一个词就是“填坑”的过程。这里其定义为 WMD 单词的移动距离。图 3.2 描述了单词转移过程。

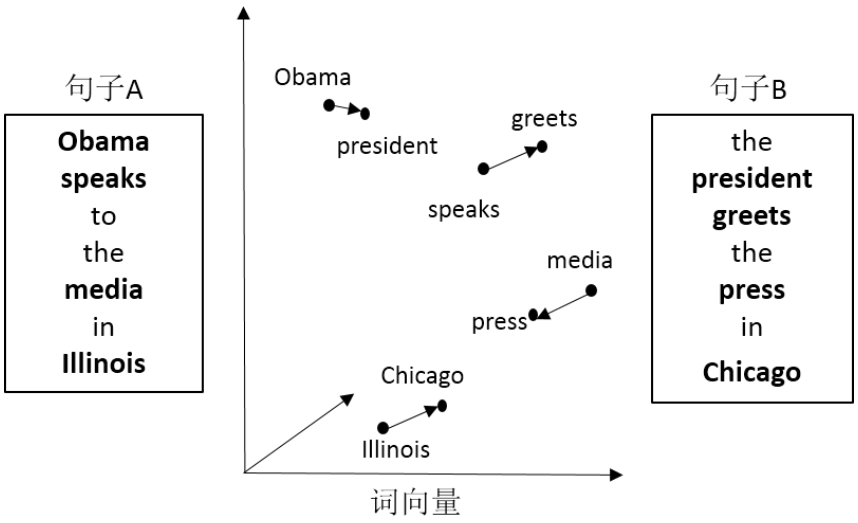

图 3.2 词向量转移示意图

作者接着用数学公式来建模词向量转移问题。图 3.3 对 WMD 进行了数学描述。图 3.3 是一个二分图，左边的节点代表问题中的关键词，右边的节点代表候选句子中的关键词。n代表词典的大小，d_i代表词典中第i个单词在问题句子中的权重，类比于土堆的体积。 d'_j代表词典中第j个单词在答案句子中的权重，类比于坑的容量。$T \in \mathcal{R}^{n\times n}$代表转移矩阵，$T_{ij} \geq 0$代表从单词$i$到单词$j$的转移量，类比于将第$i$个土堆中的多少土转移到第$j$个坑里。当然由于每个土堆离每个坑的距离不同，因此转移时的花费也不同，离得近的土堆和坑转移的代价低，同理离得近的两个词向量，转移的代价低。因此，我们需要一个衡量各个单词相互之间转移代价的矩阵。这里直接使用词向量之间的余弦距离来表示这一代价。有以下公式：

$$c(i,j) = \left\| x_i - x_j \right\|_2 \qquad \text{公式（3.7）}$$

其中，x_i，x_j代表连个单词的词向量，$c(i,j)$代表两个单词转移的代价。

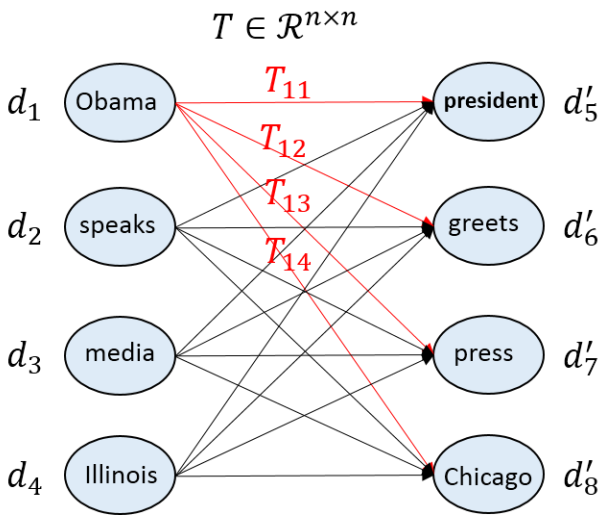

图 3.3 WMD 数学描述

单词转移的过程是有约束的，比如$\sum_j T_{ij} = d_i$，$\sum_i T_{ij} = d'_j$，也就是说每个土堆无论用于填多少坑，转移的体积等于该土堆的总体积，每个坑无论被多少土堆填，土的容量等于坑的总容量。综上，我们可以将词向量转移问题描述为线性规

划问题：

$$\min_{\mathbf{T}\geq 0}\sum_{i,j=1}^{n}T_{ij}c(i,j) \qquad \text{公式（3.8）}$$

约束条件：

$$\sum_{j}T_{ij}=d_i \qquad \forall i\in\{1,\ldots,n\}$$

$$\sum_{i}T_{ij}=d'_j \qquad \forall j\in\{1,\ldots,n\}$$

上面的线性规划问题是 EMD 问题的一种特例，存在成熟的解法，因此 WMD 本身就是一个没有超参数的模型。我们需要计算每个问题和候选答案对，寻找每一对对应的最小转移代价作为相似度。选择转移代价最小的候选句子。图 3.4 描述了两个句子D_1，D_2与句子D_0之间的相似度的计算。

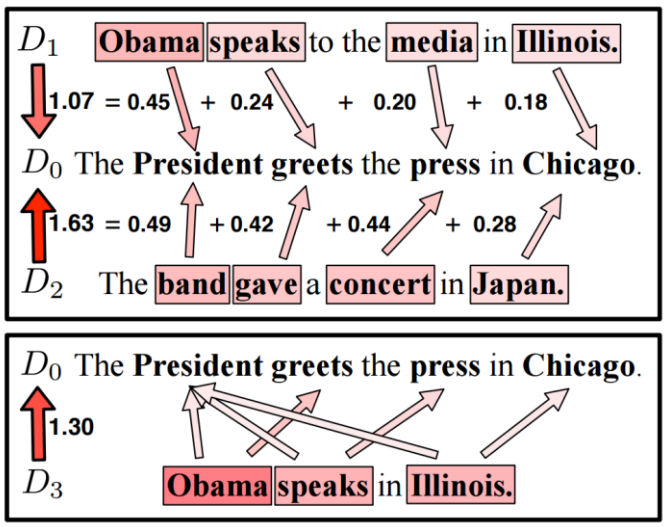

图 3.4 WMD 词向量转移相似度计算示例图[26]

在上面的例子中，我们计算D_1，D_2与句子D_0之间的相似度。D_1，D_2中每个单词的权重都是 0.25，也就每个单词的权重相同。箭头上的权重表明单词转移的代价，图中的转移代表代价最优的转移方式。我们可以看到从D_1转移到D_0所需要的最小代价是 1.07，从D_2转移到D_0所需要的最小代价是 1.63，因此D_1更接近于D_0。

当然，当句子长度不一致的时候，最优的转移方式可能是多对多的，比如D_3转移到D_0转移的最小代价是 1.30。

由此可以发现 WMD 是不带有超参数，只需要解一个线性规划的问题。然而这并不容易，因为当词典长度很大时，计算将会非常的耗时。针对计算复杂度太高的问题，作者提出了两个原优化目标的下界，分别称为 WCD（Word Centroid Distance）和 RWMD(Relaxed word moving distance)，其中 RWMD 是比 WCD 更强的下界。这两个下界计算都很容易，速度很快。这里不做阐述。根据这两个下界，作者提出了一种找出与问题最相关的 k 个文档的方法：

1. 计算所有文档的 WCD 得分，并排序。
2. 精确计算上步前 k 个排序文档的 WMD 得分。
3. 对于 k+1 之后的文档，首先计算 RWMD，如果得分大于前 k 个文档的 WMD 得分，则舍弃，否则计算精确 WMD 得分，如果当前 k 个文档的 WMD 得分小，则更新，否则舍弃。

利用上述方法，作者在八个数据集上用 KNN 算法进行分类，结果表明 WMD 的错误率比其他一些方法要低很多

整个 WMD 方法存在一个问题。原来的 WMD 模型将一句话中所有的单词的权重都置为相等，这很明显是不合理的，对于关键的词应该赋予更高的权重。基于这个想法本文用 IDF 作为单词权重用到 WMD 算法中，不妨称之为 WMD_IDF 算法。实验结果表明，用了 IDF 作为单词权重的 WMD 比原来的 WMD 在文本分类和文本排序方面有更好的性能。本章的 3.2.4 中做了相关的对比实验。

3.2.3 一种混合型算法

WMD 本质上是一种基于词向量的余弦距离的算法，能够解决近义词匹配的问题，把一个句子映射成一个高维空间中的一个分布，通过计算两个分布之间的距离来衡量两个句子之间的语义相似度。因此，WMD 并不能保证与问题共有更多相同词的句子的相似度一定更高。与此同时，BM25 是完全基于关键词匹配的算法。WMD 和 BM25 各有优缺点，我们希望融合这两个模型。因此，我们提出

了用简单的线性模型来对 WMD 和 BM25 的得分进行加权求和来综合这两个算法。不妨直接命名为 WMD_BM25 算法。公式如下：

$$score(WMD\_BM25) = \lambda \cdot score(WMD) + (1-\lambda) \cdot score(BM25)$$ 公式（3.8）

WMD 和 BM25 都没有超参数，因此融合后的线性模型只有一个权重参数。这个权重参数的选取可以直接通过枚举得到。

3.3 实验分析

我们通过两个实验来验证 WMD_IDF 和 WMD_BM25 的性能。第一个实验是延续论文[26]中使用 KNN 做文本分类的实验，第二个实验是在 Stanford Question Answering Dataset（SQuAD）数据集上进行文本排序的实验。

1. 文本分类任务

这里的文本分类任务使用的是 KNN 算法，简单来说就是对于每一个测试集中的样本，首先使用相似度算法在训练集中找到 k 个跟该文档最相近的文档，然后根据这 k 个文档的类别进行等权投票，取多数文档的类别来作为预测结果。评价标准沿用论文[26]中测试集上的错误率。

数据集：

BBCSPORT 是 2004-2005 年的 BBC 运动新闻，该数据集包含 737 篇文档，共分为 5 类（athletics，cricket，football，rugby，tennis）。其中 athletics 类有 101 篇，cricket 类有 124 篇，football 类有 265 篇，rugby 类有 147 篇，tennis 类有 100 篇[1]。我们将每一类的文章 80%划分为训练集，20%划分为测试集。

REUTERS R8 是路透社新闻数据集，是从 1986 年的路透社新闻中收集到的，每篇文档是根据新闻的主题打的标签，共八个不同的主题[28]。该数据集事先已经分好了训练集和测试集。类别信息和划分比例见表 3.1

本文将训练集按照 8:2 的比例划分为训练集和验证集。BM25，WMD，WMD_IDF 本身没有参数，KNN 中的k是唯一的一个需要优化的变量，在每个数据集上枚举$k \in \{1,5,8,10,13,17,19,25,30\}$，并通过该$k$值在验证集上的错误率来选

[1] 数据来源 http://mlg.ucd.ie/datasets/bbc.html

出最好的k。同时验证集也用来选择 WMD_BM25 算法中最好的λ。

表 3.1 REUTERS R8 数据集类别分布

类别	训练文档数	测试文档数	总文档数
acq	1596	696	2292
crude	253	121	374
earn	2840	1083	3923
grain	41	10	51
interest	190	81	271
money-fx	206	87	293
ship	108	36	144
trade	251	75	326
total	5485	2189	7674

基准实验：

本次实验的基准实验是 BM25 和原始的 WMD 算法。

实验步骤与配置：

1. 在进行 WMD，WMD_IDF 实验时，所有的数据集在预处理的时候过滤掉停止词（SMART stop word list）。BM25 实验不做处理。
2. 词向量选择免费公开的 word2vec 词向量，基于谷歌新闻训练，包含了 300 万个单词和词组，词向量维度是 300 维[2]。数据集中没有出现在该词向量中的单词将会被删除。
3. 分词工具选择斯坦福的 CoreNLP[3]。
4. 我们只做了四种算法的对比实验，分别是 BM25，WMD，WMD_IDF 和 WMD_BM25。WMD 和 WMD_IDF 均是利用原论文中公开的代码[4]，源代码

[2] 数据来源 https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[3] 数据来源 http://stanfordnlp.github.io/CoreNLP/
[4] 数据来源 https://github.com/mkusner/wmd

中用优化函数下界来提高了计算速度。

5. 对于 BM25，WMD，WMD_IDF 三个算法，首先需要确定各自的k值。首先枚举$k \in \{1,5,8,10,13,17,19,25,30\}$，分别计算验证集上的错误率。选择出验证集上最小错误率对应的k值。然后再使用最优的k值计算测试集上的错误率。
6. 对于 WMD_BM25 算法。首先将上一步中 BM25 和 WMD_IDF 在验证集上的相似度得分保存下来，枚举$\lambda \in \{0.1,0.2,0.3 \ldots 0.9\}$，然后根据公式 3.8 重新计算得分。根据新的得分重新对文档进行排序，然后枚举$k \in \{1,5,8,10,13,17,19,25,30\}$得到最小的错误率对应的$\lambda$，$k$。然后使用最优的$\lambda$，$k$计算测试集上的错误率。

实验结果：

图 3.5 和图 3.6 分别是 BM25，WMD，WMD_IDF 在两个数据集上，在不同k值上的错误率变化曲线。横轴代表k值，

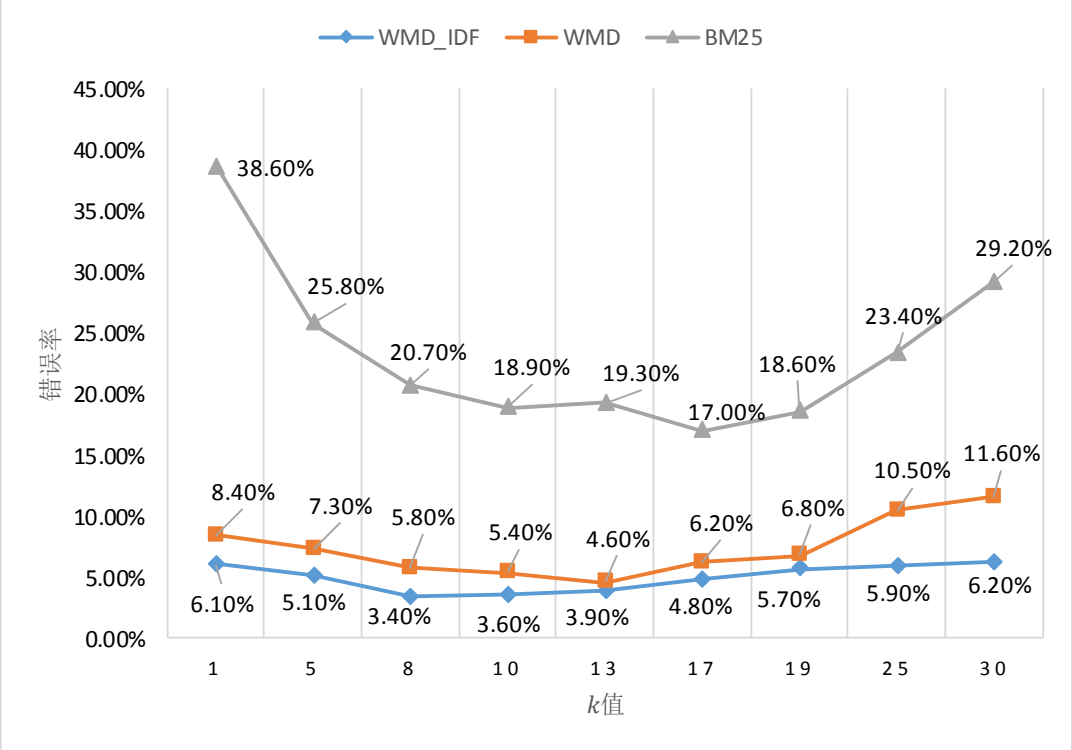

图 3.5 在 BBCSPORT 上取不同k值时错误率变化曲线

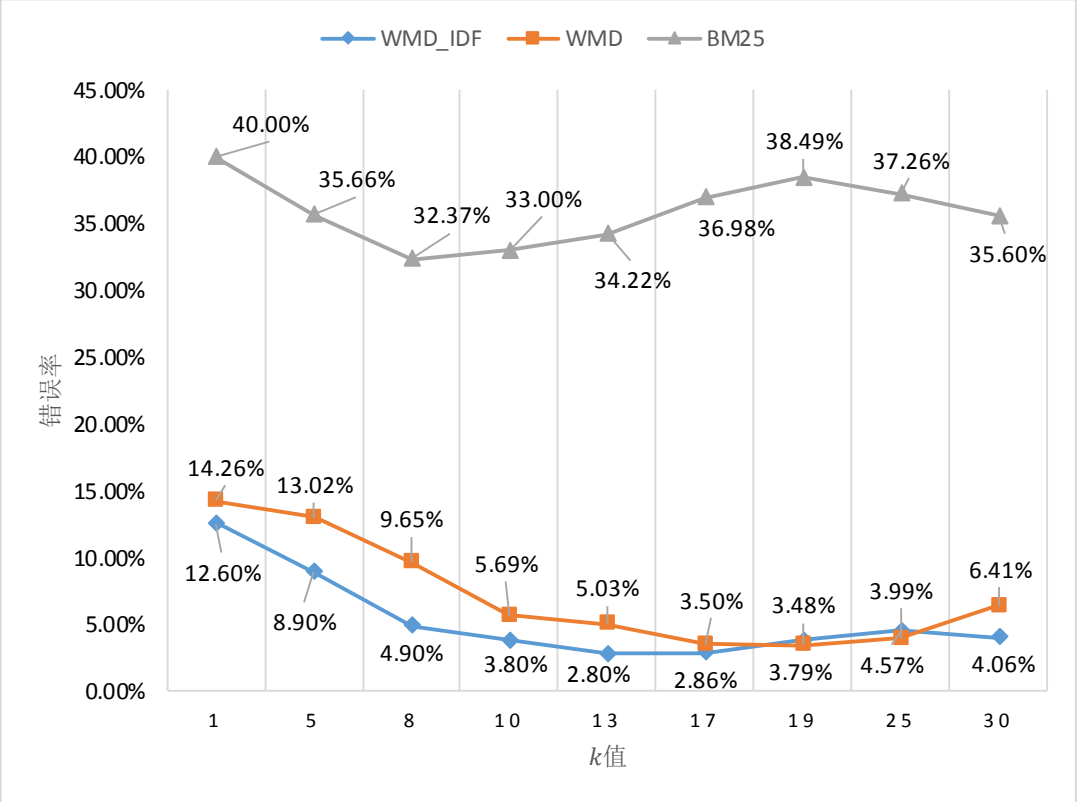

图 3.6 在 REUTERS 上取不同k值时错误率变化曲线

从实验结果中可以看出，BM25，WMD，WMD_IDF 分别在k等于 17,13,8 时在 BBCSPORT 数据集上面取得最小错误率。分别在k等于 10,13,17 时在 REUTERS 上取得最小错误率。从实验结果看出随着k值的变化，三种算法的错误率都是先下降后上升，使用 IDF 改进后的 WMD 始终比原始的 WMD 有更低的错误率。

图 3.7 是 WMD_BM25 算法在两个数据集上，当$\lambda \in \{0.1,0.2,0.3 \ldots 0.9\}$时的错误率变化曲线。图中的每个$\lambda$对应的错误率均是通过枚举$k \in \{1,5,8,10,13,17,19,25,30\}$后选出的最小的错误率。WMD_BM25 算法在$\lambda = 0.9$的时候在 BBCSPORT 数据集上取得了最小的错误率，比 WMD_IDF 最小错误率低了 0.11 个百分点。在$\lambda = 0.8$的时候在 REUTERS 数据集上取得了最小的错误率，比 WMD_IDF 最小错误率低了 0.16 个百分点。图 3.8 表示的是在两个数据集上不同算法的最小错误率对比。

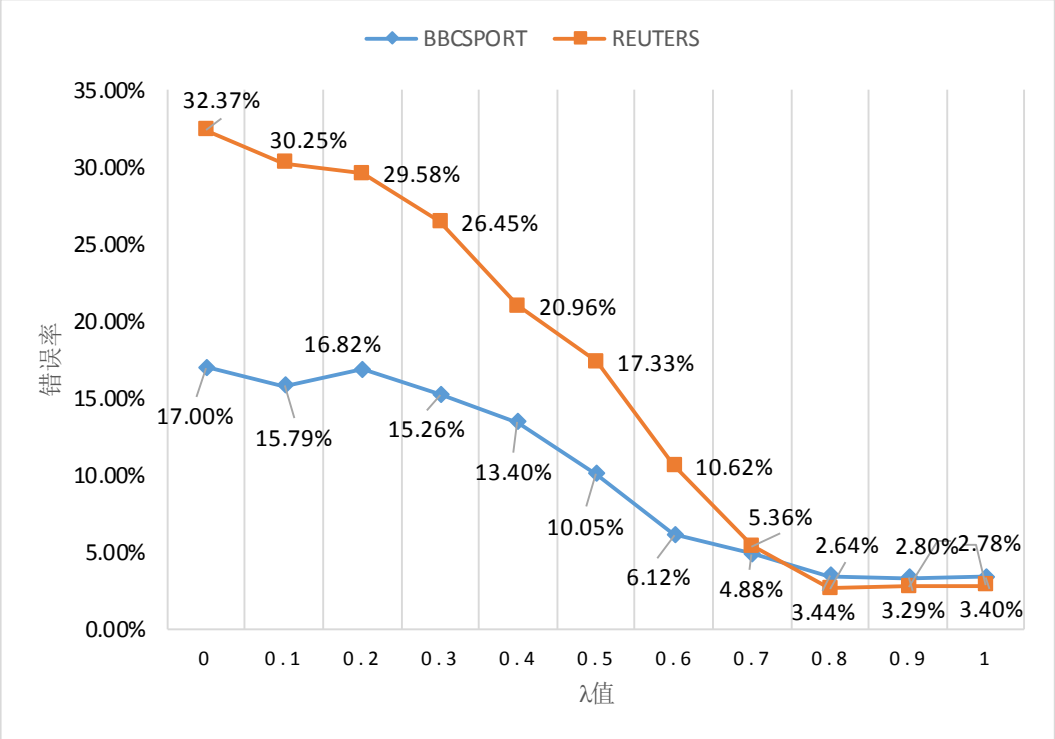

图 3.7 WMD_BM25 算法取不同 λ 值时错误率变化曲线

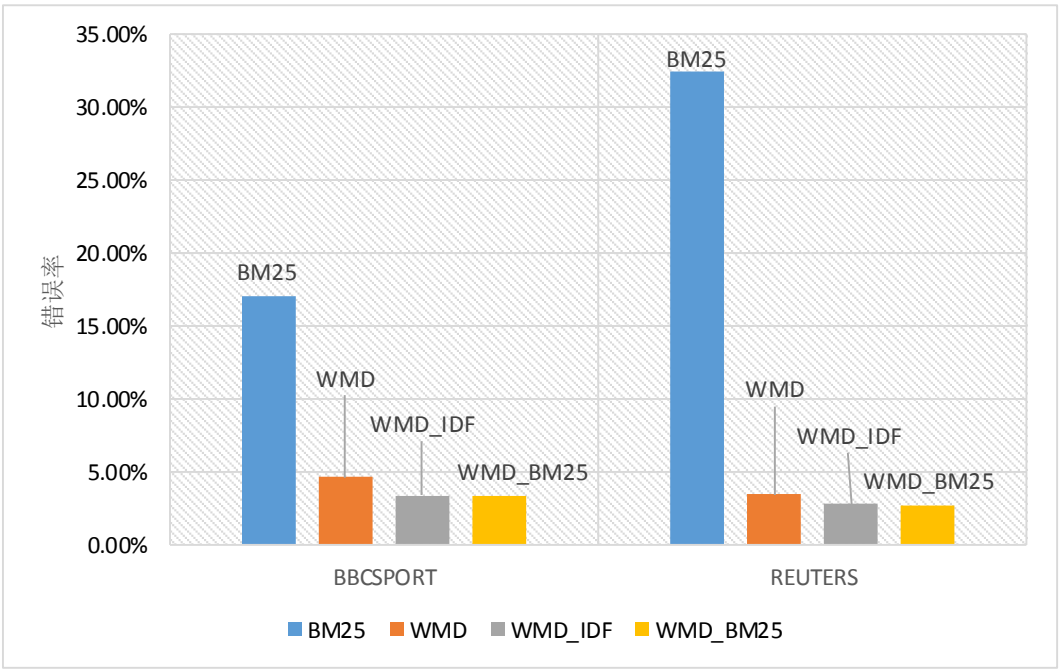

图 3.8 各个算法在两个数据集上错误率对比

2. 文本排序任务

这里的文档排序任务是基于 SQuAD 数据集。SQuAD 数据集由一系列维基百科的段落组成，每一个段落同时又对应了一系列的问题。这里我们需要计算段落与问题之间的相似度得分，并根据相似度得分进行排序，找到问题对应的正确段落所在的排名。然后计算测试集上的 MRR 以及 p@1 和 p@3。p@1 代表正确答案排名第一的比例，p@3 代表正确答案排名在前三的比例。

数据集：

Stanford Question Answering Dataset（SQuAD）[19]数据集是挑选了 536 篇维基百科的文章，并将这些文章切分为段落，同时删除掉少于 500 个字符的段落。共生成了 23215 个段落。对于每一个段落，通过众包的方式人工提问不超过 5 个问题，并在段落中标记出答案，总共有超过十万个问题。因此，SQuAD 是一个段落和该段落对应问题和相关答案的数据集，其规模庞大，是一个非常难得的数据集。数据集预先已经分成了训练集，验证集和测试集，比例是 8:1:1，但是数据集作者只公布了训练集和验证集。

基准实验：

1. TFIDF（term frequency-inverse document frequency）[44]：将一句话用 bag-of-words 方法向量化，然后除以每个单词在文档中的频率。
2. BM25 算法
3. LDA（Latent Dirichlet Allocation）[45]：一种产生式主题模型，每个句子表示成一些主题构成一个概率分布。
4. 原始 WMD 算法

本次实验对比的都是非常简单的计算句子相似度的算法，并没有对比最新的，复杂的相似度算法，主要原因是句子筛选只是筛选出句子候选集同时要保证算法运行速度，并不需要精确排序各个候选句子。后续的句子排序模块是一个更为精确的排序模型，可以选择出答案最可能出现的句子。

实验步骤和实验配置：

1. 源数据集是 JSON 格式，首先进行解析。在进行 WMD，WMD_IDF 实验时，

所有的数据集在预处理的时候过滤掉停止词（SMART stop word list）。BM25 实验不做处理。

2. 词向量选择免费公开的 word2vec 词向量，基于谷歌新闻训练，包含了 300 万个单词和词组，词向量维度是 300 维。数据集中没有出现在该词向量中的单词将会被删除。
3. 分词工具选择斯坦福的 CoreNLP。
4. 由于数据集过于庞大，本文随机从训练集中随机选择 1000 个段落，每个段落随机选择两个问题，共 2000 个问题。重复采样 5 次，并行运行实验，最后对 5 次实验结果求平均。对每一个问题都要计算 1000 个段落的相似性得分，并排序，找到正确段落对应的次序，计算 MRR 以及 p@1 和 p@3。
5. 对于基准实验和 WMD_IDF 算法，按照第四步计算 MRR 以及 p@1 和 p@3。
6. 对于 WMD_BM25 算法，我们需要确定λ，从验证集随机选择 1000 个段落，每个段落随机选择两个问题来调整超参数λ，其中$\lambda \in \{0.1,0.2,0.3 \ldots 0.9\}$，选出最好的$\lambda$，然后按照第四步计算 MRR 以及 p@1 和 p@3。

表 3.2 各个算法在三种评价指标上的性能对比

	MRR	p@1	p@3
TFIDF	42.33%	30.26%	50.97%
BM25	68.91%	60.30%	74.33%
LDA	54.36%	47.52%	62.88%
WMD	52.39%	41.67%	62.41%
WMD_IDF	59.32%	47.25%	62.37%
WMD_BM25	**76.44%**	**68.22%**	**83.45%**

表 3.2 是文本排序实验的实验结果，实验结果显示 BM25 算法比 WMD_IDF 的表现要好不少，这点与文本分类实验结果相反，经过分析作者认为，该数据集非常适合关键词匹配，因为不同文章中都包含有很多不同的专有名词，因此BM25 算法本身能够通过匹配达到比较高的精确度。其实由于问题和段落长度差别比较大，可能会干扰 WMD 算法的优化过程。通过文本分类和文本排序两个实验的评价，我们证明了改进的WMD 以及混合型的模型在文档相似度计算方面有更好的

性能。

3.4 本章小结

本章介绍了基于句子相似度的句子筛选算法。句子筛选是在文档处理模块返回的结果之后进行的。文档处理模块返回的是排序好的段落的集合。在得到排序后的段落后，很多自动问答系统就直接选择排名最高的段落，并将其传递给答案处理模块，答案处理模块一般是通过一些启发式的匹配模式来抽取答案，或者是将段落进行分词，然后将答案抽取转换成一个排序问题。这里存在一些问题，首先只选择排序最高的段落并不能保证答案就在这个段落中，为了保证比较高的可能性，我们必须选择多个段落作为候选，这样又使得候选答案集比较大，无论是用模式匹配还是用些分类算法都不能达到很高的准确率。因此，我们在段落集合的基础上进一步做了句子筛选。

我们首先介绍了使用非常普遍，非常经典的基于关键词匹配的BM25算法。同时根据 BM25 算法固有的一些缺陷，我们又详细介绍了基于词向量转移的WMD 算法。原始的 WMD 算法给一个文档中的每一个单词赋予了相同的权重，这一点是不合理的，因此本文通过赋予每个单词 IDF 权重来改进 WMD 算法。BM25 和 WMD 两个算法各有优缺点，有一定的互补性，基于此本文提出一种简单的线性模型，混合了两种算法。本文分别在文本分类和文本排序两个实验上进行评价各个模型。实验结果表明改进后的WMD算法和混合模型取得了更好的效果。其中，混合模型获得了最好的效果。

句子筛选能够有效降低答案句子候选集的大小，为后续的答案抽取打下基础，是基于非结构化文档的开放域自动问答系统中非常重要的一环。

第4章 基于多级特征的句子排序模型

4.1 任务描述

在基于非结构化文档的开放域自动问答系统中，文档处理模块搜索相关文档并进行排序，通过段落筛选和段落排序，输出一系列排序的段落，输出的段落经过上一章的句子筛选，输出少量的候选句子。接下来的任务就是对这部分少量的候选句子进行排序，选出答案最有可能出现的句子，然后传递给后面的答案处理模块进行解析并找出答案。图 4.1 描述了句子筛选和句子排序的功能。从图中可以看出句子筛选和句子排序是在文档处理模块和答案处理模块之间的处理过程，主要目的是减小候选答案的规模。句子筛选是通过相似度匹配算法筛选出答案候选句子，将段落集合缩减为句子集合。句子排序是对答案句子候选集中的句子进行评分和排序，选择出答案最可能出现的句子，将句子集合缩减为一个句子。

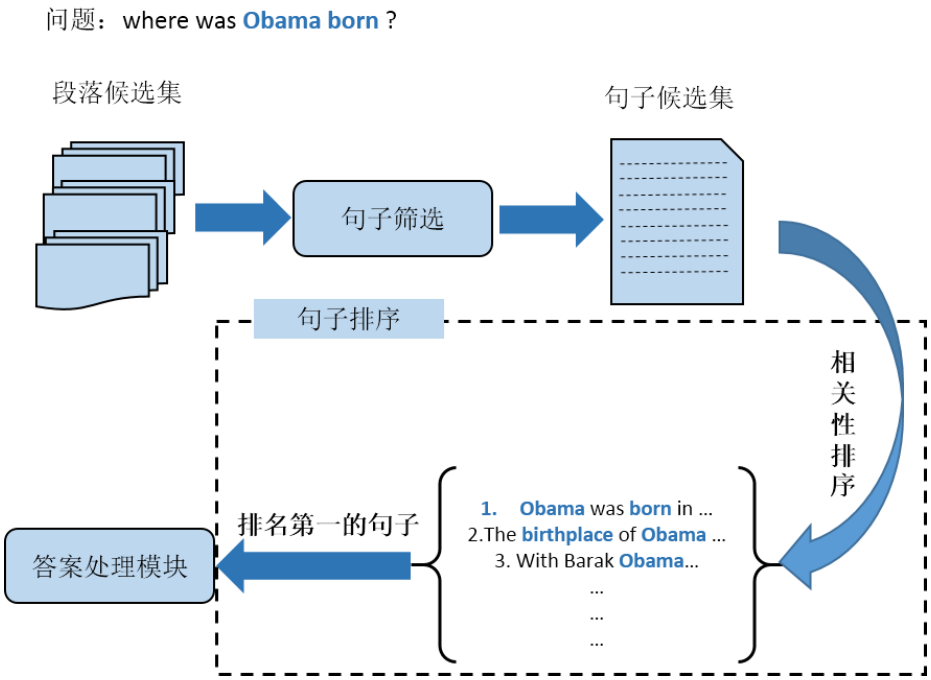

图 4.1 句子排序功能图

4.2 多级特征排序模型

给定一个问题句子Q和一个候选答案句子S，定义衡量两个句子相关性的函数为$R(Q,S)$，$R(Q,S)$是一个线性函数，是多个相关性得分函数的集成，公式如下：

$$R(Q,S) = \sum_{k} \lambda_k \cdot h_k(S,Q) \qquad \text{公式（4.1）}$$

这里的$h_k(S,Q)$代表第k个相关性函数。有了总的相关性函数之后，就可以通过计算候选集中所有句子与问题的相关性得分进行排序。

我们设计了五种特征来衡量两个句子之间的相关性，这些特征包含了不同的级别，分别是单词特征、短语特征、句子语义特征、句子结构特征、答案类型特征。我们将该模型称为 Multiple Level Feature Rank(MLFR)模型，接下来我们详细介绍每一个特征。

4.2.1 单词特征

单词级别的特征是从单词出发，计算两个句子在单词方面相似程度。本文使用了四种单词级别的特征，分别如下：

1. 共同的单词数特征，每个单词共现次数都要乘以 IDF 权重。公式如下：

$$h_{WM}(Q,S) = count\_common\_word\_(Q,S) \qquad \text{公式（4.2）}$$

2. 单词翻译特征，首先介绍单词对齐的概念，单词对齐是机器翻译中的概念，也是机器翻译中的最基本的一个工作。IBM 模型 1 中在建模翻译模型的时候，首先引入了这个概念。对齐指的是两个句子之间的映射关系，如图 4.2 所示：

从图 4.2 中我们可以看出单词对齐其实就是一种映射关系。第一个例子是翻译模型，机器翻译首先需要在大量的双语平行语料库的基础上训练学习得到这种映射关系，然后基于这种映射关系以及单词之间的翻译概率来对测试集上的句子进行打分，作为翻译模型的概率，也可以称为 IBM Model 1 得分。第二个例子是问题到和相关问题之间的映射关系。从图中可以看出，这种映射关系可以学习到同义单词和词组之间，相同单词不同形态之间的映射关系。这一点是非常重要的，因为问题句子可能是对答案句子的一个简单变形。相似问题的映射关系可以使用

问题和相关问题的数据集来训练得到。我们直接使用成熟的翻译模型 GIZA++，以及问题和相关问题的语料库来得到问题句子和答案句子之间的翻译概率，以此作为两个句子之间的相关性得分。相关性函数定义为$h_{WT}(Q,S)$。

3. 词向量平均特征，将每个句子中的单词对应的词向量相加求平均作为该句子的句子向量，然后求答案句子向量和问题句子向量之间的余弦距离。相关性函数定义为$h_{WV}(Q,S)$。

4. 词向量转移距离(WMD)特征，第三章提到的 WMD 计算相似度的方法。两个句子去除停止词，然后计算词向量转移的最小距离作为两个句子之间的相关性。相关性函数定义为$h_{WMD}(Q,S)$。

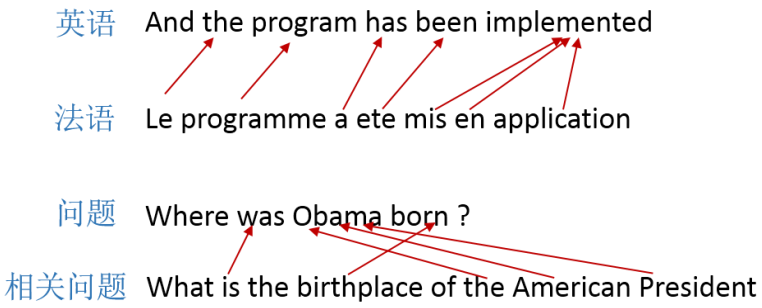

图 4.2 单词对齐示意图

4.2.2 短语特征

以单词为最小单位进行匹配有很多缺陷，不能考虑到上下文，比如有些固定搭配和固定词组同时出现时有特殊含义，如果分开出现在一个句子中就没有了特殊含义，比如“see book”，“watch TV”，“Chinese dragon（麒麟）”，“English disease（软骨病）”等。基于短语的方法更容易处理局部上下文依赖问题，能够很好的匹配到习语和常用词的搭配。基于以上分析，本文用 Zhao Yan 等人论文中的短语特征[29]。

本文直接使用 MOSES[5]来抽取短语列表。短语列表是基于问题答案对的语料

[5] 数据来源 http://www.statmt.org/moses/

库抽取的。$PP = \{\langle s_i, t_i, p(t_i|s_i), p(s_i, t_i)\rangle\}$是一个短语表，其中$s_i$是答案句子中的一个短语，$t_i$是问题句子中的一个短语，$p(t_i|s_i)$和$p(s_i, t_i)$分别代表从$s_i$翻译成$t_i$和从$t_i$翻译成$s_i$的概率。

接着，定义基于短语的相似性得分函数如下：

$$h_{PP(S,Q)} = \frac{\sum_{n=1}^{N} \frac{\sum_{j=0}^{|S|} Count(S_j^{j+n-1}, Q)}{|S| - n + 1}}{N} \qquad \text{公式（4.3）}$$

其中，S_j^{j+n-1}定义了连续的S中从S_j到S_j^{j+n-1}的连续的单词或者短语序列。N定义的是最大的 *n-gram* 值，本文中$N = 3$。$Count(S_j^{j+n-1}, Q)$有如下的定义：

1. 如果$S_j^{j+n-1} \in Q$，则$Count\left(S_j^{j+n-1}, Q\right) = 1$。
2. 否则如果有$\langle S_j^{j+n-1}, s, p\left(S_j^{j+n-1}\middle|s\right), p(s|S_j^{j+n-1})\rangle \in PP$ && $s \in Q$ 则$Count\left(S_j^{j+n-1}, Q\right) = p\left(S_j^{j+n-1}\middle|s\right) \cdot p(s|S_j^{j+n-1})$。
3. 否则，$Count\left(S_j^{j+n-1}, Q\right) = 0$。

简单可以描述为，当答案句子中的短语直接出现在问题句子中时，该短语的得分就是 1，如果该短语与问题句子中的某些短语出现在短语表中，意味着两个短语是同义的短语或者相关短语时，该短语得分就是短语表中短语互相翻译概率的乘积，是一个 0,1 之间的值。如果该短语不满足以上两种情况，那么该短语的得分就是0。计算答案句子中所有的$1 - garm$到$N - gram$短语与问题句子的相关性得分，最后对N求平均。

4.2.3 句子语义特征

单词特征衡量的是两句子之间单词的相关性，短语特征衡量的是两句子之间短语的相关性。都只能衡量局部的信息，不能够衡量两个句子的语义相关性。本文利用最新的基于深度学习的计算两个句子相似度的模型[12]来获得语义相似度得分。该模型首先将问题句子和答案句子分别用 Bi-LSTM（bidirectional long short term memory）计算两个句子每个位置的向量表达，两个句子的不同位置进行交互形成新的矩阵和张量，然后接k-Max 采样层和多层感知机进行降维。最后输出两

个句子的相似度。句子语义相关性函数定义为$h_{S(S,Q)}$。

首先，本文先介绍 LSTM（long short term memory）和双向 LSTM。LSTM 是循环神经网络 RNN（Recurrent Neural Network）的一种变形。RNN 本身适合对序列化的输入进行建模，比如语音，语言句子等。但是传统的 RNN 存在的一个严重的问题就是后续的输入会覆盖掉前面的内容，导致后面序列的产生只受前一个时刻输入的影响，更早时刻的输入信息不能被保留下来。更加数学化的解释是传统的 RNN 的激活函数使用的是 *sigmod* 或者 *tanh*，模型的训练过程中参数的更新是通过目标函数误差反向传播进行的，其实也就是梯度的反向传播。由于 *sigmod* 和 *tanh* 的导数在 0 到 0.25 之间，当网络层数比较多时，或者说当输入序列比较长时，后面层的梯度每往前传一层，都需要乘上一个 0 到 0.25 之间的小数，因此梯度是呈指数递减的，前面层的参数很难更新，这也就导致了长期依赖问题。为了解决长期依赖的问题，Hochreiter 和 Schmidhuber[30]在 1997 年提出了 LSTM，LSTM 由一个中心单元和三个门组成，中心单元可以用来保存前面序列的信息和传递反向传播时的梯度，因此解决了梯度消失的问题。

给定一个句子$S = (x_0, x_1, x_2, \dots, x_T)$，其中$x_t$是$t$时刻的单词词向量的输入。LSTM 为每一个时刻都输出一个向量表达h_t:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad \text{公式（4.4）}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \quad \text{公式（4.5）}$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad \text{公式（4.6）}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad \text{公式（4.7）}$$

$$h_t = o_t tanh(c_t) \quad \text{公式（4.8）}$$

其中，i，f，o分别是输入门，遗忘门和输出门。输入门决定当前时刻信息输入多少，遗忘门决定以前时刻信息保留多少，输出门决定当前时刻中心单元信息输出多少。c代表的是中心单元中的信息，h是每个时刻的向量表达。

类比于 LSTM，Bi-LSTM 是将句子序列正向输入一次 LSTM，然后再反向输入一次 LSTM。这样每个时刻都有两个向量表达，分别是前向的$\overrightarrow{h_t}$和反向的$\overleftarrow{h_t}$。最重要的是每一个时刻的向量表达都包含了整个句子的信息。Bi-LSTM 中每个时

刻的向量表达就是两个 LSTM 向量的拼接，即$p_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]^T$。

首先，将问题句子和答案句子输入到 Bi-LSTM 中，得到每个时刻的向量表达。然后将两个句子的各个时刻的向量进行交互。原论文中介绍了三种计算两个相似度的函数。分别是余弦距离函数，双线性函数和张量函数。由于本文只用到了张量函数，所以这里也只介绍张量函数。余弦距离函数和双线性函数都是把两个向量映射成一个标量，张量函数是把两个向量映射成一个向量，公式如下：

$$s(u,v) = f(u^T M^{[1:c]} v + W_{uv} \begin{bmatrix} u \\ v \end{bmatrix} + b) \qquad \text{公式（4.9）}$$

其中，u, v分别代表两个向量，$M^i, i \in [1, \ldots, c]$是张量的一个切片。$W_{uv}$和$b$是线性变换的参数。$f$是非线性函数，原论文中使用$f(z) = max(0, z)$，主要是保证输出始终是非负的值。经过张量函数变换后的结果是一个向量。图 4.3 描述了模型的架构，该图是论文[12]中的原图。

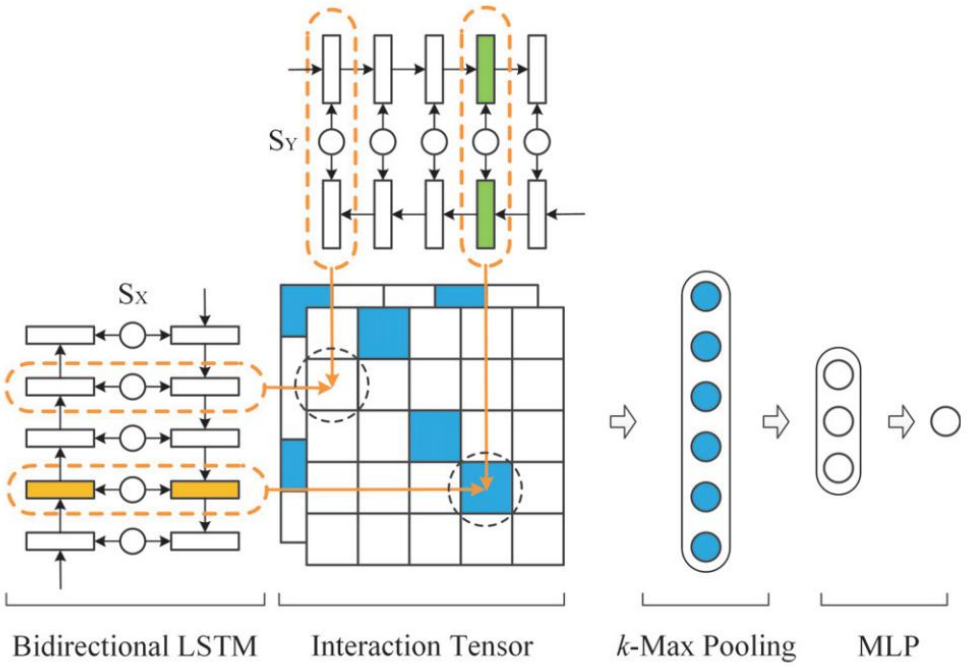

图 4.3 句子语义特征模型架构图[12]

接来下利用k-Max采样从交互矩阵或者交互张量中选出值最大的区域。对于

交互矩阵，直接选择出最大的k个值组成一个向量，对于交互张量，从每一个切片中选择出k个最大的值，然后将所有切片的值拼接成一个向量。

最后需要用多层感知机将高维的向量降到一维的标量，这个标量经过线性变换后就是两个句子的相似度得分。公式如下：

$$r = f(W_r q + b_r) \quad \text{公式（4.10）}$$

$$s = W_s r + b_s \quad \text{公式（4.11）}$$

其中，W_r和W_s是参数矩阵，b_r和b_s是相应的偏置向量。

最后，我们需要设计损失函数。本文的任务是排序，因此直接使用合页损失函数（hinge loss）。给定一个三元组(S_X, S_Y^+, S_Y^-)，其中S_X代表问题句子，S_Y^+代表正确的答案句子，S_Y^-代表错误的答案句子。损失函数可以定义为：

$$L(S_X, S_Y^+, S_Y^-) = max(0, 1 - s(S_X, S_Y^+) + s(S_X, S_Y^-)) \quad \text{公式（4.12）}$$

其中，$s(S_X, S_Y^+)$和$s(S_X, S_Y^-)$是相应的相似度函数。

4.2.4 句子结构特征

语言本身具有一定的语法结构和规律，这也使得问句和答案句子之间在语法结构上存在着一定的相似性。在自然语言处理领域，句法分析是非常关键的技术，其基本任务是确定句子的句法结构和句子中词汇之间的依存关系。一般来说，句法分析并不是一个自然语言处理任务的最终目标，但是，它往往是实现最终目标的重要环节。句法分析分为句法结构分析和依存关系分析两种。句法结构分析又可称为短语结构分析。依存关系分析又可称为依存分析[31]。

句法结构分析是指对输入的单词序列（一般为句子）判断其构成是否合乎给定的语法，分析出合乎语法的句子的句法结构。句法结构一般用树状结构表示，通常称为句法分析树，简称分析树。对于英文句子“The can can hold the water”，其对应的分析树如 4.4 所示。图中根节点“S”代表句子，两种非叶子节点“VP”和“NP”分别代表动词短语和名词短语。由于词法歧义和句法歧义在自然语言中普遍存在，因此一般来说，存在一个句子有多种句法结构表示的情况。这种情况出现时，句法分析器应该分析出最有可能的结构。

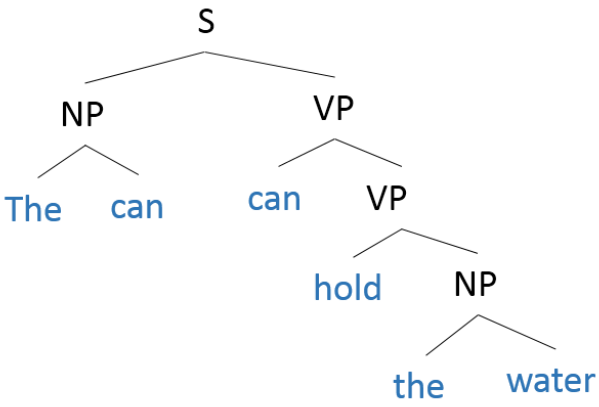

图 4.4 句法分析树示例

依存语法是指用词与词之间的依存关系来描述语言结构的框架，又称为从属关系语法。利用依存语法进行句法分析也是自然语言理解的重要手段。在依存语法的理论中，“依存”就是指词与词之间支配与被支配的关系，这种关系不是对等的，而是有方向的。通常使用带有方向的弧表示依存关系，支配者在有向弧的发出端，被支配者在有向弧的接受端。英文句子“The can can hold the water”，其对应的依存关系如 4.5 所示。

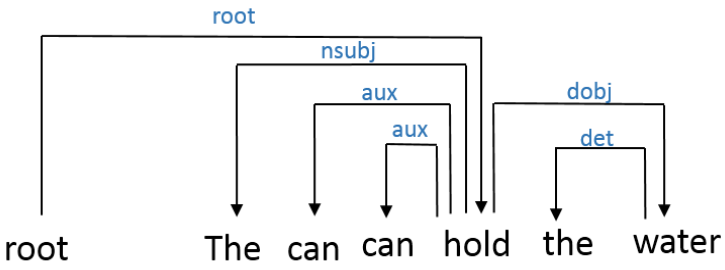

图 4.5 依存关系示例

我们发现，在实际的问答当中，问题句子和答案句子的依存关系往往存在相似性，比如对于问题句子“Where was Obama born？”和答案句子“Obama was born in Honolulu.”。图 4.6 显示了两个句子的依存关系。图中红色的线代表不同的依存关系。可以看出两句话的依存关系基本一样。“where”和“Honolulu”刚

好对应，都是“born”的修饰词。

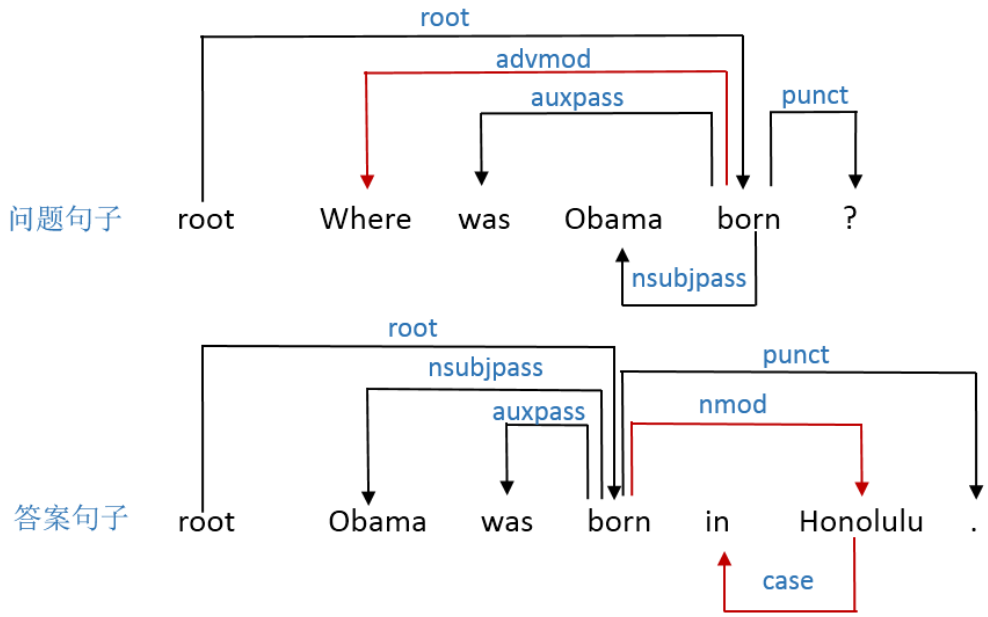

图 4.6 依存关系对比示例

基于以上的分析，本文设计了两个句子结构特征：

1. 依存树根匹配

定义为$h_{RM(S,Q)}$，这是一个取值为 0,1,2 的特征。如果问题句子和答案句子对应的依存关系拥有相同的根，则$h_{RM(S,Q)} = 2$，否则如果答案句子包含问题句子的根或者问题句子包含答案句子的根，则$h_{RM(S,Q)} = 1$。如果上面两种情况都不存在则，$h_{RM(S,Q)} = 0$。

2. 依存关系匹配。定义为$h_{DM(S,Q)}$，算法描述如下：

- 首先找到两个句子中共同的词，这里称为一对锚点。在两个句子中可能会出现多对锚点。
- 然后分别计算出两个句子的依存关系。
- 统计两个依存树从根出发到锚点的相同依存关系的数目。

值得一提的是两个句子依存树的根并不一定相同，因此这里的相同的依存关系指的是关系，而忽略词汇的差异。以图 4.6 为例，两个句子的拥有相同的根，

因此$h_{RM(S,Q)}=2$，两个句子除根外有两个共同单词，而且根到这两对锚点的依存关系相同，因此$h_{DM(S,Q)}=2$。

4.2.5 答案类型特征

由第二章的综述可知，问题处理模块首先对问题进行分析，根据问题的疑问词来判断问题类型。同时根据事先定义好的分类目录表来找到所需的答案类型。因此，一个直观的特征是，答案句子中是否包含有问题所需的答案类型的单词。如果不包含，那显然该候选句子并不是答案句子。如果包含，该候选句子才有可能是答案句子。基于这种分析，本文定义了答案类型匹配特征$h_{AM(S,Q)}$。算法步骤如下：

1. 首先，问题处理模块输出问题所需答案类型。
2. 对答案句子进行命名实体识别和词性标注。
3. 判断答案句子中命名实体是否包含问题所需答案类型，如果包含则，$h_{AM(S,Q)}=1$，如果不包含，$h_{AM(S,Q)}=0$。
4. 对于有比较明确的答案类型，比如“Money, Number, Person”等，可以根据命名实体识别来识别，对于“NNP”类型的答案，可以根据词性标注进行识别。最后，对于命名实体识别和词性标注都无法确定的答案类型，比如“Reason”或者“Manner”等默认$h_{AM(S,Q)}=1$。

4.3 实验分析

本节使用 SQuAD 数据集对 MLFR 模型进行性能评价。第三章虽然已经简单介绍过该数据集，但不够具体。本节首先详细介绍下该数据集，然后介绍数据集的预处理工作，以及每个特征具体训练的细节和使用到的数据集。

实验任务是候选句子的排序，对于每个问题都有一个规模很小的候选答案句子集（段落）。在这个候选集中只有一个答案句子是正确的。这里我们将数据集定义为一个三元组的形式(S,Q,y)，S代表候选句子，Q代表问题，y是一个布尔类型的标签，只取 0,1。0 代表不是正确答案句子，1 代表正确答案句子。我们使用MRR和p@1、p@3来衡量排序算法的性能。

1. SQuAD 数据集

SQuAD 数据集本身是一个用于阅读理解的数据集。该数据集包含了 536 篇维基百科中的文章，共 23215 个段落，每个段落都人工提问不超过 5 个问题。总共的问题数超过十万，是一个非常难得的语料库，既可以用于阅读理解领域，也可以用于自动问答领域。图 4.7 是该数据集中的一个段落，也是一个样本。

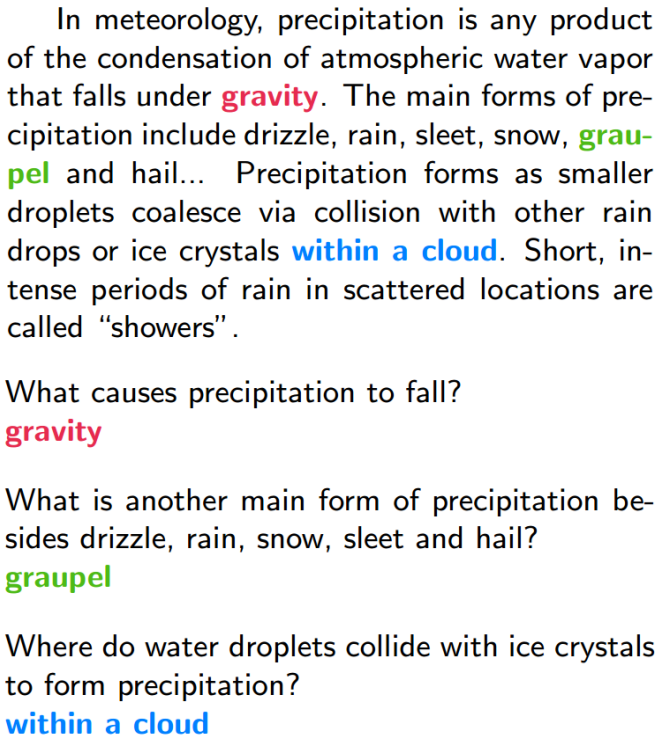

图 4.7 SQuAD 一个示例[19]

图 4.7 中是 SQuAD 中的一个段落，同时段落下面有三个人工提问的问题以及这个问题的答案。答案需要在段落中标记出来。图 4.7 来源与论文[19]中的原图。该数据集涵盖的主题广泛，问题涉及到的答案类型也各种各样，表 4.1 是 SQuAD 中的问题所涉及到的答案类型的一个比例分布。从 4.1 表中可以看出，SQuAD 数据集涵盖的答案类型非常广泛也分布也比较均匀，满足开放域非结构化文档的要求。

表 4.1 SQuAD 数据集答案类型比例分布

Answer type	Percentage	Example
Date	8.9%	19 October 1512
Other Numeric	10.9%	12
Person	12.9%	Thomas Coke
Location	4.4%	Germany
Other Entity	15.3%	ABC Sport
Common Noun Phrase	31.8%	Property damage
Adjective Phrase	3.9%	Second-largest
Verb Phrase	5.5%	Returned to earth
Clause	3.7%	To avoid trivialization
Other	2.7%	quiety

2. 数据预处理：

原数据集只公开了训练集和验证集。为了完成模型的评价，本文将原有的训练集和验证集进行合并，重新按照 8:1:1 的比例划分训练集，验证集和测试集。划分过程中每个段落和这个段落的所有问题是一个基本的划分单元。训练集、验证集、测试集的统计信息见表 4.2。

将段落切分成句子，将每个段落切分成答案句子候选集，这样每个问题只需要对所属段落中的句子进行排序来找到答案句子，候选集规模比较小。虽然数据集中并没有给出问题所对应的答案句子。但是给出了每个问题答案的起始位置，我们根据答案的起始位置可以设计算法定位该问题所对应的句子。本文使用了多个自然语言处理工具进行句子切分效果对比，包括 NLTK、OpenNLP 和 CoreNLP。切分效果的评价标准是判断该问题的答案是否包含在我们预测的答案句子里。经过对比分析，本文发现 CoreNLP 和 OpenNLP 的切分效果更好，NLTK 的切分规则过于简单，切分错误率太高。本文最终使用 CoreNLP 对段落进行句子的切分。

使用 CoreNLP 对段落中的句子以及该段落的问题进行分词、词性标注、命名实体识别、句法分析树和依存关系分析。本文使用 Java 语言调用公开的 CoreNLP

工具包。

表 4.2 训练集、验证集、测试集统计信息

	段落数	问题数
训练集	16630	81345
验证集	2078	10048
测试集	2078	10288

3. 各个特征函数的实现细节

首先对于单词特征函数$h_{WT}(Q,S)$，我们使用 GIZA++来训练一个有 11.6M 个问题和相关问题的平行语料库[35]，该语料库是从 WikiAnswers 网站上爬取的。对于$h_{WV}(Q,S)$函数，本文直接使用 Word2Vec 模型训练维基百科语料库。这里并没有直接使用谷歌新闻训练的词向量的原因是 SQuAD 本身是维基百科的文章。

对于短语特征函数$h_{PP(S,Q)}$，我们直接使用 MOSES 来训练 SQuAD 数据集。SQuAD 数据集先经过预处理步骤，然后将训练集中的每个问题和对应的答案句子放在一起形成一个问题到答案的平行语料库，共有 81345 对句子。然后输入到 MOSES 中训练得到短语表。

对于句子语义特征函数$h_{S(S,Q)}$，本文首先构建正负样本的集合。对于一个问题，其对应的正确句子为正样本，对应段落中的其他句子作为负样本。训练集中每个问题有一个正样本，同时随机采样两个负样本。这样每个问题对应了三个训练样本，共 244035 个样本。验证集和测试集采取同样的方式，分别有 30144 和 30864 个样本。我们使用基于 Theano 的 Keras[6]实现双向的 LSTM 模型。训练过程中批的大小为 128，优化器选择 Adagrad。模型训练的终止条件是验证集上的错误率不再下降。

句子结构特征函数$h_{RM(S,Q)}$和$h_{DM(S,Q)}$。数据集预处理之后，每个句子都有一个依存关系，然后统计出这两个结构特征函数。同样对于答案类型匹配特征

[6] 数据来源 https://keras.io/

$h_{AM(S,Q)}$。数据集预处理之后，每个句子都进行了命名实体识别，问题句子在问题处理模块中也有相应的答案类型。根据简单的匹配就可以计算出来。

以上是所有特征函数的实现细节，在所有特征函数训练和计算完成后，我们需要把这些特征函数进行线性加权融合。线性模型的参数为每个特征函数的权重值，见公式（4.1）。目标函数和公式（4.12）相同。训练集是将验证集中所有的问题的一个正样本，并随机采样两个负样本，共 30144 个正负样本。

4. 基准实验

本次实验对比的基准实验有四个，分别是：

- BM25 算法
- 谷歌 DeepMind 团队 2014 年在 NIPS 上发表的论文中提到的二元文法的卷积神经网络（bigram-CNN）[46]，该论文算得上是将 CNN 用到句子匹配上的开山之作。bigram-CNN 模型只有一个卷积层和一个平均采样层，bigram 的意思是卷积窗口每次包含两个单词，窗口移动不重叠。问句和候选句子分别被向量化，然后计算余弦距离，利用逻辑回归训练得到每个候选句子是正确答案句子的概率。
- 本章 4.2 节中句子语义特征部分使用的深度学习模型 MV-LSTM[12]。该模型首先将问题句子和答案句子分别用 Bi-LSTM 计算两个句子每个位置的向量表达，两个句子的不同位置进行交互形成新的矩阵和张量，然后接k-Max 采样层和多层感知机进行降维。最后输出两个句子的相似度。
- 德国慕尼黑大学和 IBM Waston 团队在 2015 年提出的的 ABCNN（Attention-Based Convolutional Neural Network）句子匹配模型[47]。根据该论文中的实验结果，本文选用性能最好的 ABCNN-2 模型，即有两个卷积层和两个采样层。

5. 实验结果

实验结果见表 4.3，BM25 算法的 MRR 能达到 79.44%，充分说明 SQuAD 数据集非常适合于关键词的匹配。bigram-CNN 模型的性能比 BM25 要差，本文认为一个主要原因是 SQuAD 数据集非常适合于关键词匹配算法，bigram-CNN 模型

简单，只用最后一层的输出作为句子的向量表达，并不能突出单词之间的匹配关系。MV-LSTM 模型和 ABCNN-2 模型的性能基本接近，比 BM25 要好不少，主要原因是这两个模型都有 Attention 机制，能够表达单词级别的匹配关系。本文的 MLFR 模型在三个评价标准上都要比基准模型好，充分说明了使用多级特征能够更好的计算句子之间的相关性。MLFR 的 p@1 达到了 87.73%。也就是说给定一个问题和一个段落，MLFR 模型选择出包括答案的句子的概率为 87.73%。这为后面答案抽取打下了基础。

为了探究出哪种特征函数对结果影响比较大，本文逐个去掉各个特征函数，然后分别评价 MRR，结果如表 4.4 所示。第一行是包括所有特征的模型，后面行依次代表从模型中减去相应特征。MRR 降低率中的值为负代表降低。我们发现每个特征都很重要，其中单词特征和句子语义特征对模型影响最大，句子结构特征对模型影响最小。

表 4.3 各个模型在三种评价指标上性能对比

模型	MRR	p@1	p@3
BM25	79.44%	66.82%	92.16%
bigram-CNN[46]	72.38%	59.12%	84.73%
MV-LSTM[12]	86.32%	80.61%	93.77%
ABCNN[47]	89.56%	83.92%	94.75%
MLFR	**92.31%**	**87.73%**	**99.85%**

表 4.4 各个特征对 MLFR 模型性能影响

模型	MRR	MRR 降低率
MLFR	92.31%	
MLFR-F_w	85.82%	-6.49%
MLFR-F_p	88.06%	-4.25%
MLFR-F_{sem}	85.98%	-6.33%

（续表 4.4）

模型	MRR	MRR 降低率
MLFR-F_{stru}	90.21%	-2.10%
MLFR-$F_{awstype}$	87.12%	-5.19%

4.4 本章小结

句子排序是句子筛选和答案抽取之间的工作，主要任务是将答案句子候选集进行排序，找出最可能包含答案的句子，并将这个句子传递个答案处理模块。

本文设计了 MLFR 模型，该模型融合了多级特征函数，包括单词级别、短语级别、句子语义级别、句子结构级别和答案类型级别。在 SQuAD 数据集上对该模型进行了评价，评价标准有三个，分别是 MRR，p@1 和 p@3，并与最新的基于深度学习的句子排序模型进行了对比。实验结果表明，MLFR 能够有效提高问题和答案句子的匹配准确度。MLFR 在测试集上的 p@1 达到了 87.73%。也就是说给定一个问题和一个段落，MLFR 模型选择出包括答案句子的概率为 87.73%。这为后面答案抽取打下了基础。

第5章 基于端到端深度神经网络的答案抽取

5.1 任务描述

答案抽取是基于非结构化文档的开放域自动问答系统中最后一个模块，输入是一个包含答案的句子，输出是一个具体的答案。图 5.1 描述了答案抽取模块的功能和流程。答案抽取的技术有很多，很多论文将句子切分成候选答案的单词，然后使用人工设计的特征，比如词性标注、命名实体识别、依赖关系等来衡量候选答案与问题之间的相关性，从而找出最可能的答案。这种方式比较繁琐，灵活性不高。因此，本文直接使用端到端的深度神经网络来直接产生出答案。

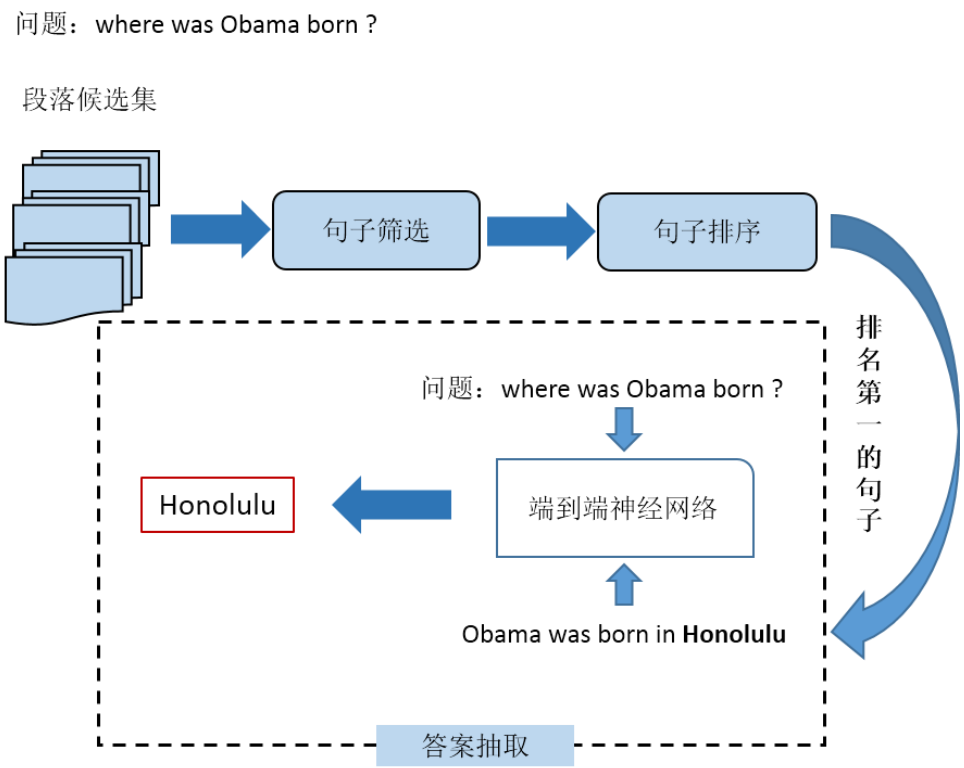

图 5.1 答案抽取流程图

5.2 端到端神经网络模型

本文直接使用 Shuohang Wang 论文中的端到端的模型[32]进行答案抽取。该模型训练时的输入是一系列三元组(S, Q, a)，S代表答案句子，Q代表问题句子，a代表精确的答案，其中a是从S抽取出来的。测试的时候输入只有S, Q，输出是a。该模型包括两个部分，首先借鉴[33]中的 Match-LSTM 网络将S中的每一个词与Q进行匹配，同时为每一个词生成一个匹配向量。第二部分借鉴[34]中的 Pointer Net，将第一部分产生的匹配向量作为输入，每一步预测一个S中的词，最终产生出答案。原论文中提出了两种模型的变种，分别称为序列模型和边界模型。两个变种模型适用于不同的任务，序列模型不要求答案是连续的子句，边界模型要求答案是原句子中连续的子句。原论文在 SQuAD 数据集上分别测试了两个变种模型，边界模型性能更好，因此本文只使用边界模型。具体模型框架如图 5.2。该图来自于论文[32]中原图。从图 5.2 中可以看出，该模型自下往上一共有三层，最下面的是 LSTM 预处理层，中间是 Match-LSTM 层，最上面是答案定位层。本文分别介绍这三个层的原理。

1. LSTM 预处理层

使用单向的 LSTM 将Q和S分别进行编码，主要目的是为了将文本信息向量化，同时为句子中的每个位置都产生一个隐向量，该隐向量融合了上下文的信息，方便进行后续的带有 Attention 机制匹配。LSTM 的原理在第四章中已经介绍，这里不做阐述。数学描述为：

$$H^s = \overrightarrow{LSTM}(S) \qquad \text{公式（5.1）}$$

$$H^q = \overrightarrow{LSTM}(Q) \qquad \text{公式（5.2）}$$

其中$H^s \in \mathbb{R}^{l\times m}$，$H^q \in \mathbb{R}^{l\times n}$分别代表答案句子和问题句子的向量表达，$l$代表隐向量维度，$m, n$分别代表答案句子和问题句子的的长度。$H^s$和$H^q$的第$i$个列向量$h_i^s$和$h_i^q$分别代表答案句子和问题句子的前$i$个单词的向量表达。

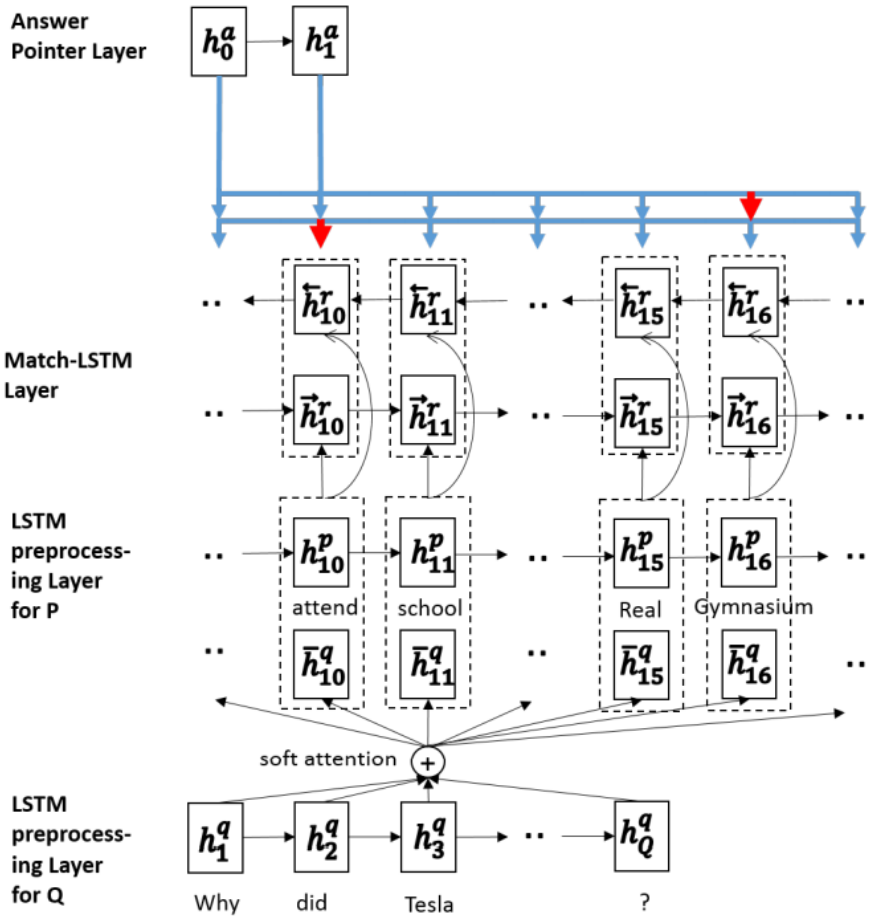

图 5.2 边界模型框图[32]

2. Match-LSTM 层

Match-LSTM 层的目的是为了将S中的每个词与Q进行匹配。由常识可知，S中每个词只会跟Q中某些词关系比较密切，也就是说每个词都有自己的“关注点”，每个词只“关注”Q中的某个部分。因此论文中引入了 Attention 机制。

定义$\alpha_i \in \mathbb{R}^n$为$S$中第$i$个词对应的 Attention 权重向量，$\alpha_{ij}$代表$S$中第$i$个词与$Q$中第$j$个单词的关联度。接着用$\alpha_i$来获得一个加权版本的问题向量表达，并与当前时刻的答案句子表达h_i^s进行拼接，形成一个新的向量。

$$z_i = \begin{bmatrix} h_i^s \\ H^q \alpha_i^T \end{bmatrix} \qquad \text{公式（5.3）}$$

向量z_i被当做另外一个 LSTM 模型第i个时刻的输入，这个 LSTM 就被称为

Match-LSTM:

$$\overrightarrow{h_i^r} = \overrightarrow{LSTM}\left(z_i, \overrightarrow{h_{i-1}^r}\right) \quad \text{公式（5.4）}$$

其中$\overrightarrow{h_i^r} \in \mathbb{R}^l$，这里的箭头代表正向的 LSTM。同理，使用方向 LSTM 进行上面的步骤就可以得到$\overleftarrow{h_i^r}$

定义$\overrightarrow{H^r} \in \mathbb{R}^{l\times m}$代表正向 LSTM 的隐向量集合$\left[\overrightarrow{h_1^r}, \overrightarrow{h_2^r}, \dots, \overrightarrow{h_m^r}\right]$，定义$\overleftarrow{H^r} \in \mathbb{R}^{l\times m}$代表反向 LSTM 的隐向量的集合$\left[\overleftarrow{h_1^r}, \overleftarrow{h_2^r}, \dots, \overleftarrow{h_m^r}\right]$。则 Match-LSTM 层的输出是：

$$H^r = \begin{bmatrix} \overrightarrow{H^r} \\ \overleftarrow{H^r} \end{bmatrix} \quad \text{公式（5.5）}$$

3. 答案定位层（Ans-Ptr layer）

答案定位层也是一个 LSTM 模型。答案定位层很像 seq2seq 模型。seq2seq 模型是先用 LSTM 对问题进行编码，然后再用一个 LSTM 模型进行解码，解码 LSTM 每个时间步都产生一个答案单词。类比于 seq2seq 模型，答案定位层也是一个解码的过程，只不过编码过程在 Match-LSTM 中完成，H^r就是我们的编码，代表的是答案句子的不同位置与问题句子的相关程度。解码过程中输出的是答案句子中单词的索引$\mathrm{a} = (a_1, a_2, \dots)$，每个时间步产生一个索引。对于边界模型，也就是答案是连续的情况，我们只需产生两个索引，分别是开始单词索引和结尾答案索引，然后默认两个索引中间的单词都是答案。解码过程需要用到前面的编码矩阵H^r，同样使用 Attention 机制来确定解码的每一步跟哪些位置的匹配向量关系密切。假设解码过程中需要确定第k个时间步产生的索引，首先计算当前时刻对应的 Attention 权重向量$\beta_k \in \mathbb{R}^m$，$\beta_{k,j}$代表第k个时间步产生答案句子第j个单词的概率。β_k的计算公式如下：

$$F_k = \tanh(VH^r + (W^a h_{k-1}^a + b^a) \otimes e_m) \quad \text{公式（5.6）}$$

$$\beta_k = softmax(v^T F_k + c \otimes e_m) \quad \text{公式（5.7）}$$

其中，$V \in \mathbb{R}^{l\times 2l}$，$W^a \in \mathbb{R}^{l\times l}$，$\mathrm{v} \in \mathbb{R}^l$，$c \in \mathbb{R}$。$\otimes e_m$代表将一个向量或者标量重复$m$次形成矩阵或者向量。$h_{k-1}^a$为另外一个 LSTM 上一时刻的隐形量。

$$h_k^a = \overrightarrow{LSTM}(H^r \beta_k^T, h_{k-1}^a) \quad \text{公式（5.8）}$$

对于边界模型，我们只需要产生两次输出，分别是起始单词和结束单词，分别定义为a_s，a_e。模型产生答案序列的概率是：

$$p(a|H^r) = p(a_s|H^r)p(a_e|a_sH^r) \quad \text{公式（5.9）}$$

并且

$$p(a_s|H^r) = \beta_{1,a_s}, p(a_e|a_sH^r) = \beta_{2,a_e} \quad \text{公式（5.10）}$$

模型训练过程中使用的目标函数是：

$$-\sum_{n=1}^{N} logp(a_n|S_n, Q_n) \quad \text{公式（5.11）}$$

5.3 整体性能测试

5.3.1 实验介绍

本章将句子筛选、句子排序和端到端神经网络三个模块整合在一起进行性能测试。如图 5.3 所示，虚框中的部分是将三个模块进行整合，是本次实验的测试对象，输入是一系列段落集合，输出是精准答案（词或者短语等）。

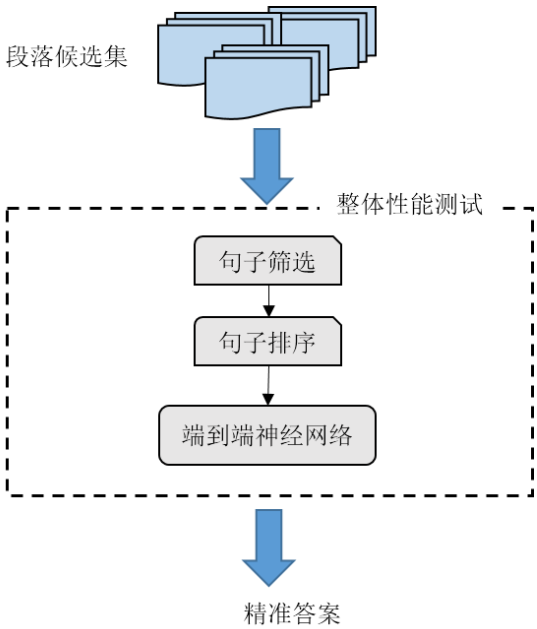

图 5.3 整体性能测试框图

值得说明的是，本文并没有对整个基于非结构化文档的开放域自动问答系统进行测试，只是对本文提出来的模块和模型部分进行测试。

5.3.2 实验设计

本章依然使用 SQuAD 数据集进行实验，前面介绍过该数据集是由段落和相关问题以及问题答案构成。这里需要特别强调一点，由于 SQuAD 数据集中每个问题都对应于一个段落，也就是一个答案句子候选集。因此句子筛选部分可以省略，相当于只测试句子排序和端到端神经网络模型的答案抽取，另外一个原因是很多基于 SQuAD 数据集的模型都是面向于阅读理解任务，比如本章中使用的端到端的神经网络模型[32]。这些模型并不存在句子筛选部分，而是直接从段落中抽取答案的。因此为了方便模型对比，本次实验省略了句子筛选模块，只进行句子排序和基于端到端神经网络的答案抽取模块的性能测试。

本次实验的评价标准有两个，分别是精确匹配准确率和 F1 得分。精确匹配是指将预测答案与真正的答案进行精确匹配，如果不完全相同，则认为预测答案错误。F1 得分是计算预测答案和正确答案单词覆盖的 F1 分数。

数据集中的一个样本可以表示成一个三元组(Q, D, a)，其中Q代表问题，D代表问题所属的段落，a代表精准的答案，a所在的句子定义为S。定义$p(S|Q, D)$代表预测对答案句子的概率，也就是答案排序中 p@1 的精确率。$p(a|Q, S)$代表端到端神经网络模型的答案抽取的准确率。则对于测试集中所有的样本三元组，答案抽取的精确率可以用以下公式表示：

$$p(a|Q, D) = p(S|Q, D)p(a|Q, S) \qquad \text{公式（5.12）}$$

基准实验

本次实验的基准实验是本章 5.2 节的端到端的神经网络模型。基准实验的模型和本文中答案抽取的模型完全相同，区别在于基准实验没有句子排序模块，直接从段落中抽取答案，本文中的模型是先进行句子排序选出答案句子，然后对答案句子进行答案抽取。对比的目的是为了证明，本文中的多级特征排序模型是非常有效的，能够提高答案抽取的准确率。在给定答案句子候选集的情况下，本文

先选择答案句子再抽取答案的流程在基于无机构化文档的开放域自动问答的研究中也是行之有效的。

实验步骤

1. 训练集，验证集和测试集的划分方式跟第四章实验一样。
2. 使用斯坦福的 CoreNLP 将段落切分成句子，并分词。
3. 词向量的选用跟第四章相同。
4. 词典长度为 10000，其余统一标记为“<unknown>”。
5. 首先进行句子排序实验，有第四章可知该实验的 p@1 的精确率为 87%左右，对于答案句子选择错误的样本不进行答案抽取实验。答案抽取实验只在正确选择答案句子的样本中进行。直接使用论文[32]中公开的边界模型的代码[7]，参数与原论文中保持一致。然后计算出测试集上的精确匹配准确率和 F1 得分。不妨将该模型称为（MLFR_Match-LSTM_Ans-Ptr）。
6. 对于基准实验，同样使用边界模型。直接从段落中预测答案，计算测试集上的精确匹配准确率和 F1 得分。不妨将该模型称为（Match-LSTM_Ans-Ptr）

5.3.3 实验结果

表 5.1 是本文模型和基准实验模型在测试集上的精确匹配率和 F1 得分的对比。从表中可以看出，本文先进行句子排序，然后进行答案抽取取得了更好的效果。经过分析，我们认为取得高精确匹配率和 F1 得分的主要的原因在于句子排序可以有效筛选掉很多有干扰的句子，可以明显提高端到端神经网络进行答案抽取时的准确率。虽然本文模型存在级联误差，但是本文中的句子排序模型的 p@1 精确率已经很高，所以级联误差很小，同时输入单个句子能够有效的提升端到端神经网络答案抽取的性能。

图 5.4 是实验结果的一个举例，该例子中问题是“奥巴马出生于哪里？”，候选段落是维基百科中介绍奥巴马的一段描述。Match-LSTM_Ans-Ptr模型直接从段落中抽取答案，得到的答案并不精确。本文的 MLFR_Match-LSTM_Ans-Ptr 模

[7] 代码来源 https://github.com/shuohangwang/SeqMatchSeq

型先选择最可能的答案句子，然后再抽取答案，得到了精确的答案“Honolulu”。

表 5.1 测试集上精确匹配率和 F1 得分对比

	精确匹配率	F1 得分
Match-LSTM_Ans-Ptr	59.3%	68.4%
MLFR_Match-LSTM_Ans-Ptr	**65.7%**	**73.6%**

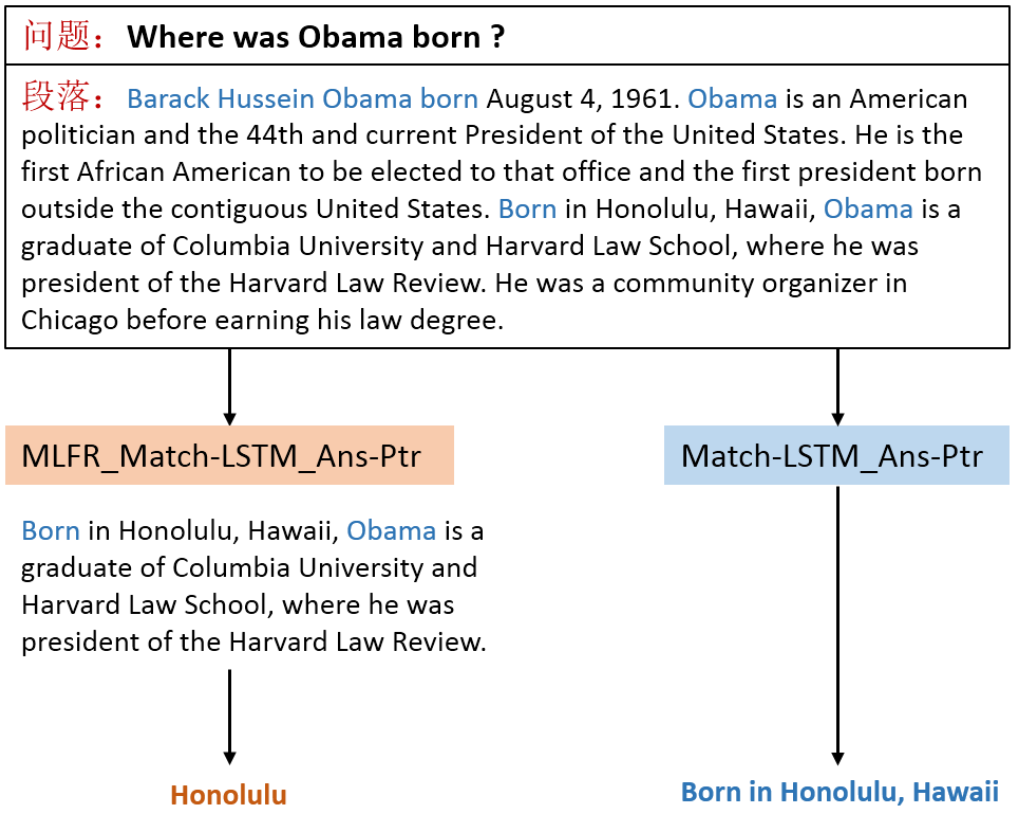

图 5.4 实验结果示例

5.4 本章小结

答案抽取是自动问答系统中最后一个模块，输入是一个包含答案的句子，输出是一个具体的答案。答案抽取的技术有很多，很多论文将句子切分成候选答案的单词，然后使用人工设计的特征，比如词性标注、命名实体识别、依赖关系等

来衡量候选答案与问题之间的相关性，从而找出最可能的答案。随着深度学习在自然语言处理领域的迅猛发展，端到端的深度神经网络可以用来直接产生出答案。本文直接使用论文[32]中的基于深度学习架构的模型。本章首先介绍了该模型的实现原理，并在 SQuAD 数据集上进行实验。本章设计实现了两个实验，第一个是按照本文流程，先使用第四章的句子排序模型选择出答案句子，再使用边界模型进行答案抽取。第二个实验是直接利用边界模型从段落中选择出答案。通过在测试集上的精确匹配率和 F1 得分的对比，我们发现进行答案排序是很有必要的。本文中的多级特征排序模型是非常有效的，能够提高答案抽取的准确率。在给定答案句子候选集的情况下，本文先选择答案句子再抽取答案的流程在基于无机构化文档的开放域自动问答的研究中也是行之有效的。

在实际问题中，基于非结构化文档的封闭域自动问答系统往往更为实用。比如，在线客服、在线医疗问答、技术咨询等。在基于非结构化文档的封闭域自动问答系统中，文本的句式和语法结构相对变化少，因此基于人工设计模板和规则的方法也能取得一定的效果。然而，基于人工设计模板和规则的方法又存在过于繁琐和扩展性差等问题。本文提出的基于非结构化文档的开放域自动问答系统的架构同样适用于封闭域的情况，同时能够解决模板设计繁琐和扩展性差的问题。但依然存在一些挑战和困难，主要有两点：

1. **词向量需要重新训练。**在不同的专业领域中，词汇和语义会有所不同，因此词向量需要使用相关专业领域的语料库重新训练，降低了扩展性。
2. **数据源规模存在限制。**本文中提到的架构使用了很多深度学习的模型，需要用大规模的数据源进行训练。一方面，某些专业领域可能无法提供规模庞大的数据源。另一方面，一些规模庞大的数据源往往掌握在大型企业的手中，比如在线客服和在线医疗问答的数据源。数据源不仅要求有大量的文档，同时要求有大量的问题，这些问题的来源只能通过类似于搜索日志的方式进行收集，同样也只有企业具备这种问题收集的能力。

第6章 总结与展望

6.1 总结

随着互联网的发展与普及，互联网上的信息也在爆炸式的增长。传统的搜索引擎，能够帮助人们快速地找到相关的网页，但是并不能直接找出真正的答案。也就是人们的这种需求推动了自动问题的发展与繁荣。本文介绍了一般的自动问答系统的结构以及实现方法，并指出了可能存在的问题。同时提出了一些改进的方案。

本文的主要贡献在于：

对基于非结构化文档的开放域自动问答系统进行了详细的综述，包括基于非结构化文档的开放域自动问答系统的组成部分，以及各部分的功能介绍。并对相关研究工作进行了归类和综述。

指出了传统的基于非结构化文档的开放域自动问答系统存在的问题，并引入了句子筛选和句子排序模块。句子筛选模块主要功能将段落候选集缩减为句子候选集。句子排序是将句子候选集缩减为一个句子并传递给答案抽取模块，缩减了候选答案的规模后能大大提高答案抽取的准确率。

针对句子筛选模块。本文改进了一种计算文档相似度的模型 Word Mover's Distance(WMD)，并提出了将 BM25 和 WMD 结合的混合模型。本文分别进行了文档分类和文本排序实验。实验结果表明，混合模型能够在计算文档相似度方面比 BM25 和原有的 WMD 算法有更好的效果。

针对句子排序模块。本文设计了五种特征来衡量两个句子之间的相关性，这些特征包含了不同的级别，分别是单词特征，短语特征，句子语义特征，句子结构特征，答案类型特征。我们将该模型称为 Multiple Level Feature Rank(MLFR)模型。基于 SQuAD 数据集，本文对比了目前最好的基于深度学习的句子匹配模型。实验结果表明，MLFR 模型在各个评价指标上（MRR, p@1 和 p@3）都要比基准

实验有更好的效果。

针对答案抽取模型。本文放弃了传统答案处理模块中繁琐的人工设计特征和模板的抽取算法，引入了用于阅读理解的端到端的深度神经网络模型，该模型能够刚好适用于从短文本中抽取答案的任务。本文中进行了两个实验，一个是先用句子排序模块选择一个答案句子，然后在这个答案句子中抽取答案，另外一个实验是直接在段落中抽取答案。实验结果表明，答案排序模块在提高答案正确率方面是非常有必要的。本文中的多级特征排序模型是非常有效的，能够提高答案抽取的准确率。在给定答案句子候选集的情况下，本文先选择答案句子再抽取答案的流程在基于无机构化文档的开放域自动问答的研究中也是行之有效的。

6.2 展望

本文针对现有的基于非结构化文档的开放域自动问答系统结构中存在的问题，提出了相关解决方案。在文档处理模块之后，进一步使用句子筛选将候选段落集合缩减为候选句子集合，使用句子排序从候选句子集合中选择出一个最相关的句子。由于基于非结构化文档的开放域自动问答系统本身是一个比较复杂的问题，本文认为基于无结构化文档的开放域自动问答系统可以在以下几个方向。

1. 使用端到端的深度学习框架

本文中的基于非结构化文档的开放域自动问答系统在抽取答案之前首先进行了句子排序，选择了一个最可能的答案句子，这种方式存在一些级联误差。目前在自动问答领域和阅读理解领域，有很多端到端的深度学习架构可以直接输入一段话来产生出精确答案，并且取得了很高的精确度。

2. 提高系统交互性

目前的基于非结构化文档的开放域自动问答系统都只是根据当前用户输入的问题来匹配答案，并没有考虑用户本身的个人习惯和以往的搜索记录。基于非结构化文档的开放域自动问答系统应该为每个用户进行建模并且记录用户习惯，并在选择答案的时候体现出来。这样能够提高用户体验。

参考文献

[1] Banerjee, Protima, and Hyoil Han. "Drexel at TREC 2007: Question Answering." TREC. 2007.

[2] 李维. 立委科普：问答系统的前生今世 [OL]. http://blog.sciencenet.cn/home.php?mod=space&uid=362400&do=blog&id=436555. 2011-4-23

[3] Stoyanchev, Svetlana, Young Chol Song, and William Lahti. "Exact phrases in information retrieval for question answering." Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering. Association for Computational Linguistics, 2008.

[4] Ramprasath, Muthukrishnan, and Shanmugasundaram Hariharan. "A survey on question answering system." International Journal of Research and Reviews in Information Sciences 2.1 (2012): 171-179.

[5] Moldovan, Dan I., et al. "Lasso: A tool for surfing the answer net." (1999).

[6] Lampert, Andrew. "A quick introduction to question answering." Dated December (2004).

[7] Voorhees, Ellen M., and Donna Harman. "Overview of TREC 2001." TREC. 2001.

[8] Radev, Dragomir, et al. "Probabilistic question answering on the web." Journal of the American Society for Information Science and Technology 56.6 (2005): 571-583.

[9] Kangavari, Mohammad Reza, Samira Ghandchi, and Manak Golpour. "Information retrieval: improving question answering systems by query reformulation and answer validation." Information Retrieval 2585 (2008): 13824.

[10] Feng, Minwei, et al. "Applying deep learning to answer selection: A study and an open task." Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on. IEEE, 2015.

[11] Yin, Wenpeng, et al. "Abcnn: Attention-based convolutional neural network for modeling sentence pairs." arXiv preprint arXiv:1512.05193 (2015).

[12] Wan, Shengxian, et al. "A deep architecture for semantic matching with multiple

positional sentence representations." arXiv preprint arXiv:1511.08277 (2015).

[13] Ravichandran, Deepak, and Eduard Hovy. "Learning surface text patterns for a question answering system." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.

[14] Xu, Jinxi, Ana Licuanan, and Ralph M. Weischedel. "TREC 2003 QA at BBN: Answering Definitional Questions." TREC. 2003.

[15] Peng, Fuchun, et al. "Combining deep linguistics analysis and surface pattern learning: A hybrid approach to Chinese definitional question answering." Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2005.

[16] Shih, Cheng-Wei, et al. "ASQA: Academia sinica question answering system for NTCIR-5 CLQA." NTCIR-5 Workshop, Tokyo, Japan. 2005.

[17] Yu, Yang, et al. "End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension." arXiv preprint arXiv:1610.09996 (2016).

[18] Wang, Shuohang, and Jing Jiang. "Machine comprehension using match-lstm and answer pointer." arXiv preprint arXiv:1608.07905 (2016).

[19] Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).

[20] Gaizauskas, Robert, and Kevin Humphreys. "A combined IR/NLP approach to question answering against large text collections." Content-Based Multimedia Information Access-Volume 2. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2000.

[21] Harabagiu, Sanda M., et al. "Falcon: Boosting knowledge for answer engines." (2000).

[22] Hermjakob, Ulf. "Parsing and question classification for question answering." Proceedings of the workshop on Open-domain question answering-Volume 12. Association for Computational Linguistics, 2001.

[23] Li, Xin, and Dan Roth. "Learning question classifiers." Proceedings of the 19th international conference on Computational linguistics-Volume 1. Association for

Computational Linguistics, 2002.

[24] Zhang, Dell, and Wee Sun Lee. "Question classification using support vector machines." Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval. ACM, 2003.

[25] Ji, Zongcheng, Zhengdong Lu, and Hang Li. "An information retrieval approach to short text conversation." arXiv preprint arXiv:1408.6988 (2014).

[26] Kusner, Matt J., et al. "From Word Embeddings To Document Distances." ICML. Vol. 15. 2015.

[27] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[28] Cachopo, Ana Margarida de Jesus Cardoso. Improving methods for single-label text categorization. Diss. Universidade Técnica de Lisboa, 2007.

[29] Yan, Zhao, et al. "DocChat: an information retrieval approach for chatbot engines using unstructured documents." ACL, 2016.

[30] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[31] 成庆, and 宗. 统计自然语言处理. 清华大学出版社, 2008.

[32] Wang, Shuohang, and Jing Jiang. "Machine comprehension using match-lstm and answer pointer." arXiv preprint arXiv:1608.07905 (2016).

[33] Wang, Shuohang, and Jing Jiang. "Learning natural language inference with LSTM." arXiv preprint arXiv:1512.08849 (2015).

[34] Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks." Advances in Neural Information Processing Systems. 2015.

[35] Fader, Anthony, Luke S. Zettlemoyer, and Oren Etzioni. "Paraphrase-Driven Learning for Open Question Answering." ACL (1). 2013.

[36] 吴友政, et al. "问答式检索技术及评测研究综述." 中文信息学报 19.3 (2005): 2-14.

[37] 黄鹏程. 面向自然语言查询的知识搜索关键技术研究. MS thesis. 浙江大学, 2016.

[38] Allam, Ali Mohamed Nabil, and Mohamed Hassan Haggag. "The question

answering systems: A survey." International Journal of Research and Reviews in Information Sciences (IJRRIS) 2.3 (2012).

[39] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[40] Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.

[41] Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences." arXiv preprint arXiv:1404.2188 (2014).

[42] Le, Quoc V., and Tomas Mikolov. "Distributed Representations of Sentences and Documents." ICML. Vol. 14. 2014.

[43] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[44] Salton, Gerard, and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." Information processing & management 24.5 (1988): 513-523.

[45] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." Journal of machine Learning research 3.Jan (2003): 993-1022.

[46] Yu, Lei, et al. "Deep learning for answer sentence selection." arXiv preprint arXiv:1412.1632 (2014).

[47] Yin, Wenpeng, et al. "Abcnn: Attention-based convolutional neural network for modeling sentence pairs." arXiv preprint arXiv:1512.05193 (2015).

攻读硕士学位期间主要的研究成果

发明专利申请

[1] 王东辉，李亚楠，蔺越檀，孙欢，黄鹏程，洪高峰，**徐灿**，梁建增，庄越挺，一种基于本体内容的 SPARQL 查询语句生成的系统及方法（申请号：201410852126.1）

[2] 王东辉，李亚楠，蔺越檀，熊奎，黄鹏程，洪高峰，**徐灿**，梁建增，庄越挺，一种基于任务难度与标注能者能力的众包标注数据整合方法（申请号：201410850691.4）

[3] 王东辉，**徐灿**，庄越挺，一种基于多级特征的问题和答案句子相似度计算方法（申请号：201710100418.3）

致谢

至此，两年多的硕士生活就要结束了，人生的一个重要转折点，从校园到职场。告别了大学，也告别自己的青春。

首先衷心的感谢我的导师王东辉老师。王老师在科研上一丝不苟的工作精神和勤奋刻苦的态度深深的鼓舞了我。王老师周末也会来实验室，平日都工作到很晚才回家，我觉得这才是科研的态度，是我学习的榜样。王老师在科研上为我们指明了方向，耐心为我们答疑解惑。在生活中也给予了我们无微不至的关心和帮助。硕士期间能遇到这样的导师真的是无比幸运。

感谢实验室的兄弟姐妹们，感谢他们陪我一同成长，感谢他们给予我的帮助。美丽的大师姐李亚南，经常能看到她在操场晨跑的优美身姿。拥有完美身材，同样毕业于山东大学的大师兄蔺越檀，科研运动两不误。话少但踏实做事的梁建增，我们一起找工作，一起毕业，革命情谊万岁。智商高情商更高的师弟庞章阳，为我解答了很多疑惑，给了我很多帮助，很佩服他。可爱的小师妹夏子琪，给实验室增加不少活力。还有两个人高马大，帅气聪明的小师弟胡焕行和王纪东，以及毕业的师兄熊逵、黄鹏程、洪高峰和师姐孙欢，感谢他们在我找工作时给予的帮助。感谢我的好哥们夏鑫，感谢他一直鼓励着我，给了我许多建议和正能量。感谢我的家人对我一如既往的关爱和支持。最后感谢自己的努力，感谢自己在某些困难时刻能够坚持下来。

我爱浙大，我爱杭州，我爱浙大美丽的早晨，我爱自己塞着耳机走过的每一条路。回忆太多而且都历历在目。未来的生活刚刚开始，我会带着自己的梦想上路，继续努力。长风破浪会有时， 直挂云帆济沧海。

徐灿

二零一七年三月于浙大求是园