学 号 2012202130

密 级

哈尔滨工程大学学士学位论文

基于知识库的限定域问答系统的

设计与实现

院（系）名　称：软件学院

专　业　名　称：软件工程（数字新媒体技术）

学　生　姓　名：杨志烨

指　导　教　师：王燕 副教授

哈尔滨工程大学

2016年6月

**基于知识库的限定域问答系统的设计与实现 杨 志 烨 哈尔滨工程大学**

学 号 2012202130

密 级

基于知识库的限定域问答系统的

设计与实现

Design and Implementation of Limited Field Question Answering system Based on Knowledge Base

**学生姓名：**杨志烨

**所在学院：**软件学院

**所在专业：**软件工程（数字新媒体技术）

**指导教师：**王燕

**职称：**副教授

**所在单位：**哈尔滨工程大学

**论文提交日期：**2016年5月

**论文答辩日期：**2016年6月

**学位授予单位：**哈尔滨工程大学

摘要

问答系统作为人工智能和自然语言处理领域中的一个重要分支，具有广泛的发展前景，目前实现问答系统的方式主要是建立知识库，或者构建出知识图谱，搭建搜索引擎，再对输入问题通过知识图谱进行搜索，对搜索结果进行知识抽取即可得出答案。其意义在于将信息与检索进行了高度的整合，使人们可以快速、准确的获得想要的信息，同时也推动了人工智能的发展。

本课题爬取了CK-12网站上6类与科学相关共2139个关键词，通过Wikipedia提供的API下载了对应关键字的维基百科正文形成知识库，使用Lucene搭建了本地搜索引擎，对知识库构建倒排索引，搜索出与输入问题集合中每个问题最相关的前k篇文章，再对文章和问题的答案选项进行语义相似度分析，使用了python语言和Java语言，结合TF-IDF算法、关键词提取算法TextRank、Word2vec工具模型，对该问题的Lucene搜索结果文本与问题的四个选项答案进行语义相似度分析，选择出相似度最高的结果作为最终答案。

本文实现的问答系统已成功完成了测试与提交，Kaggle测评给出了一个比较好的结果，运行结果表明该系统是一个相对稳定、可投入使用的问答系统。

关键词：问答系统；知识库；TF-IDF；Word2vec

**ABSTRACT**

Question Answering system as an important branch of the field of artificial intelligence and Natural Language Processing, its development very promising. Now mainly to build knowledge base, construct knowledge graph and build a search engine to implement it. Then search the input problems by knowledge graph and extraction knowledge by search results can get the answer. the meaning is high degree of integration to information and retrieval, it make people can quickly and accurately get the information what they want, but also promote the development of artificial intelligence.

The topic has crawled six kind of key words of science in CK-12 website, and has downloaded the Wikipedia content through the Wikipedia tool. This topic has used Lucene to build a local search engine, created inverted index for knowledge base to search k articles on the most relevant to each of the problems for the input problem set, then did semantic similarity analysis of articles and four answers. This topic uses the python language and the Java language, combines with TF-IDF algorithm, keyword extraction algorithm, Word2vec tool model, the text and the answer to semantic similarity analysis, chooses the highest similarity result as the answer.

In this paper, the question and answer system has been successfully completed. Kaggle website gave a good result, showing that the system is a relatively stable and practical Question Answering system.

**Key words:** question answering system; knowledge base; TF-IDF; Word2vec

目　　录

[第1章 绪论 1](#_Toc453698145)

[1.1 课题研究背景与意义 1](#_Toc453698146)

[1.2 课题主要内容 2](#_Toc453698147)

[1.3 论文组织结构 2](#_Toc453698148)

[第2章 相关技术概述 4](#_Toc453698149)

[2.1 网络爬虫 4](#_Toc453698150)

[2.1.1 爬虫技术简介 4](#_Toc453698151)

[2.1.2 urllib2网络爬虫 5](#_Toc453698152)

[2.1.3 request网络爬虫 5](#_Toc453698153)

[2.2 Lucene搜索引擎 6](#_Toc453698154)

[2.2.1 Lucene简介 6](#_Toc453698155)

[2.2.2 倒排索引 7](#_Toc453698156)

[2.2.3 检索 8](#_Toc453698157)

[2.3 计算语义相似度 9](#_Toc453698158)

[2.3.1 关键词提取 9](#_Toc453698159)

[2.3.2 TF-IDF算法 10](#_Toc453698160)

[2.3.3 Word2vec模型 11](#_Toc453698161)

[2.4 Django框架 14](#_Toc453698162)

[2.5 本章小结 15](#_Toc453698163)

[第3章 搜索引擎模块的设计与实现 16](#_Toc453698164)

[3.1 搜索引擎模块的体系结构设计 16](#_Toc453698165)

[3.2 知识库建立 17](#_Toc453698166)

[3.2.1 关键词爬取 17](#_Toc453698167)

[3.2.2 正文爬取 18](#_Toc453698168)

[3.2.3 关键词提取 19](#_Toc453698169)

[3.3 搭建搜索引擎 20](#_Toc453698170)

[3.3.1 建立索引 20](#_Toc453698171)

[3.3.2 全文检索 20](#_Toc453698172)

[3.4 本章小结 21](#_Toc453698173)

[第4章 语义相似度计算模块的设计与实现 23](#_Toc453698174)

[4.1 语义相似度计算模块的体系结构设计 23](#_Toc453698175)

[4.2 TF-IDF算法计算语义相似度 24](#_Toc453698176)

[4.2.1 TF-IDF算法的改进 24](#_Toc453698177)

[4.2.2 普通文本的语义相似度计算 24](#_Toc453698178)

[4.2.3 关键词文本的语义相似度计算 26](#_Toc453698179)

[4.3 Word2vec模型计算语义相似度 27](#_Toc453698180)

[4.3.1 Word2vec模型的使用 27](#_Toc453698181)

[4.3.2 平均向量法计算语义相似性 28](#_Toc453698182)

[4.3.3 最大维度法计算语义相似性 29](#_Toc453698183)

[4.4 计算语义相似度算法比较 30](#_Toc453698184)

[4.5 测试及界面 31](#_Toc453698185)

[4.6 本章小结 32](#_Toc453698186)

[结　　论 33](#_Toc453698187)

[参考文献 34](#_Toc453698188)

[攻读学士学位期间发表的论文和取得的科研成果 36](#_Toc453698189)

[致　　谢 37](#_Toc453698190)

# 第1章 绪论

## 1.1 课题研究背景与意义

人工智能(artificial intelligence)是计算机科学的一个重要分支，并且在近几年取得了很大的进展[1]，其被广泛的应用于语音识别、图像识别、自然语言处理等领域，早在1950年“计算机科学之父”图灵为了回答机器是否有智能的问题，就提出了著名的图灵测试：电脑如果能回答一系列人类的问题，而且能让提问者不能分辨是电脑的答案还是人的答案，那么电脑就通过了图灵测试。今年3月份的AlphaGo击败了人类围棋高手李世石，再次说明了人工智能在未来具有非常广泛的发展前景，并且可以应用在各行各业，随着近年来计算机硬件的飞速发展以及深度学习在各个研究领域上的大热，模仿人类神经元的神经网络算法突破了硬件限制，再次出现在人们的视野中，越来越多基于神经网络的深度学习模型可以运用到自然语言处理、语音识别、机器翻译等领域中，人工智能的发展必将又被推到一个新的高度[2]。

而问答系统则是人工智能中不可或缺的一环，市面上的人工智能产品，从SIRI到百度机器人，再到IBM的Watson，都是以问答系统为核心再结合语音识别、动作识别等技术进行设计的。问答系统主要由对问题进行分析、检索知识库中的问题得到结果和对检索结果进行分析提取出答案三个部分组成，它的主要技术就是先从语料库中整理问答中需要的知识，然后将知识高度抽象成利于搜索的数据结构，形成知识库，对于用户的输入，经过词法、句法等自然语言处理技术后送入搭建在问答系统中的内部搜索引擎，对搜索的结果进行处理，如知识推理、答案抽取、相似度计算等，即可得到答案[3]。

国内外研究机构在问答系统的研究上也如火如荼，微软、IBM以及很多著名国内外大学都有专门的问答系统研究中心，已经问世的主要成果有Tipask问答、OASK问答等，苹果公司推出的SIRI智能语音助手也已经给人们的生活带来了极大的便利。目前问答系统的蓬勃发展得利于强大的硬件平台、日积月累丰富的语料及知识资源以及深层的问答技术，其中涉及自然语言处理技术、机器学习技术、知识推理技术等。随着时间的推移，硬件设备的更新换代，分布式的处理高并发的大数据已经不是难题，知识语料也只会越来越丰富，越来越规范，深度学习等前沿技术运用到自然语言处理等领域的模型更是层出不穷，这些因素都将推动着问答系统的发展，将会设计出高准确率的开放域的问答机器人为人类服务。

## 1.2 课题主要内容

在充分分析了Kaggle评测网站给出的课题开发需求后，调研并学习了python语言语法及其对网络爬虫、数据处理和自然语言处理的相关工具包的基本API，同时学习了使用Java语言下的Lucene工具包搭建本地搜索引擎、自然语言处理与机器学习常用算法以及深度学习模型的应用，最终本课题确定了以下几点主要内容：

1、构建知识库

问答系统中知识库是非常重要的，它是问答系统的基石，如果根基打的不牢，那么之后无论用什么算法都不会得到令人满意的结果。Kaggle评测网站并没有给出限定的语料库，所以这是一个开放的部分，由于没有本地资料，所以学习并使用了python下的request工具包和urllib2工具包，针对美国教育类网站CK-12中的八年级科学类关键词部分开发了一套网络爬虫脚本，可以将八年级所有科学类关键词爬取下来。再使用python下的wikipedia工具包将这些关键词的维基百科正文部分按照文本方式爬取到本地，构建成知识库。

2、搭建本地搜索引擎

通过与搜索引擎相关的教程与论文，发现现在最流行的搜索引擎如谷歌、必应、百度[4]，都是将构建搜索引擎分为两步，即索引和搜素，索引是用爬虫技术爬取到网页信息之后，将所有网页通过一种算法索引起来，使网页方便查找，搜索又包含输入分析、网页排名等部分。比较筛选各种搭建搜索引擎的方式之后决定使用Java语言开发的Lucene工具针对已在本地构建成功的知识库搭建本地的搜索引擎。

3、搜索结果分析

通过机器学习、深度学习在自然语言处理方面的应用，经过比较和筛选，最终选择TF-IDF算法和Word2vec深度学习工具模型对搜索结果进行分析，主要采用了计算语义相似度的策略计算结果，TF-IDF算法在计算语义相似度表现良好，虽然Word2vec模型没有达到预期效果，但也算是一种新的尝试。

4、搭建web页面

通过python语言的web框架Django编写了简单的web界面，可以检测到用户输入的美国八年级相关问题和其四个答案选项，通过后台数据库进行检索给出答案。

## 1.3 论文组织结构

本文拟解决的问题主要有：使用python语言中的爬虫工具包解析网页并爬取关键词，并使用python中wikipedia提供的API爬取对应的维基百科正文，构建成知识库。使用python语言中的TextBlob工具包对普通文本进行关键词提取形成关键词文本。使用Java语言中的Lucene工具搭建本地搜索引擎，对知识库建立索引，并对输入问题进行搜索返回最相关的k条文本记录。对这k条文本记录，分别实现改进后的TF-IDF算法和Word2vec工具中的两种算法计算与对应四条答案记录的语义相似性。抽取文本关键词之后再用以上算法计算语义相似性。最后使用python的Django Web框架设计界面，将围绕以上问题，本文的结构组织如下：

第1章对本文的研究背景以及研究意义进行了分析说明，阐述了国内外相关技术的发展现状，并简要介绍了本文的主要工作内容以及论文阐述的组织结构。

第2章主要介绍了本课题中用到的相关技术，主要有python爬虫技术、Lucene搜索引擎的搭建技术，以及使用TextRank算法进行关键词提取、TF-IDF算法和Word2vec工具模型分析语义相似性的可行性与数学推导，并详细介绍了Word2vec工具中的Hierarchical Softmax模型和Negative Sampling模型，最后简要介绍的python 的Django框架。

第3章具体介绍了本地搜索引擎搭建的体系结构设计与实现过程，首先介绍如何构建本地知识库，介绍了如何爬取CK-12网站上的与科学相关的关键词，以及如何爬取关键词的对应维基百科正文，并使用实现了Textrank算法的TextBlob工具包对文本进行关键词提取。介绍了Lucene搜索引擎的搭建，其中又包括建立倒排索引和全文搜索两部分。

第4章具体介绍了语义相似度计算的设计与实现，首先使用TF-IDF算法，结合需求对TD-IDF算法行了改进，并实现TF-IDF算法对普通文本文件和关键词文本文件分别计算语义相似度并得出结果。然后使用Word2vec算法，分别实现了平均向量法和最大维度法计算语义相似度，对各种方法进行整理并比较，又研究了一些准确率高的模型的做法并进行学习。然后介绍了使用Django框架搭建本课题的web界面。

最后的结论部分对本文的整个工作内容进行了全面的总结，对四种计算语义相似度的算法进行了对比，并提出了进一步进行课题研究的方向。

# 第2章 相关技术概述

本章介绍了实现本课题所使用的技术，其中包括网络爬虫技术，搭建Lucene所涉及的倒排索引、全文检索、网页排名技术以及关键词提取的实现，TF-IDF算法和Word2vec模型在语义相似度计算方面的分析设计与实现，以及Django Web框架的搭建。

## 2.1 网络爬虫

### 2.1.1 爬虫技术简介

网络爬虫，是按照人为需求通过程序设定的一系列规则，在互联网上自动抓取所需要信息的脚本或者程序，所以它也是搜索引擎抓取部分中非常重要的一部分。网络爬虫的主要目的就是将互联网上的所有网站内容下载到本地形成一个互联网内容集合的镜像备份[5]，为方便用户搜索，搜索引擎通常再对网络爬虫爬取的互联网内容的镜像备份建立倒排索引，提供接口供用户进行搜索。如图2.1所示。

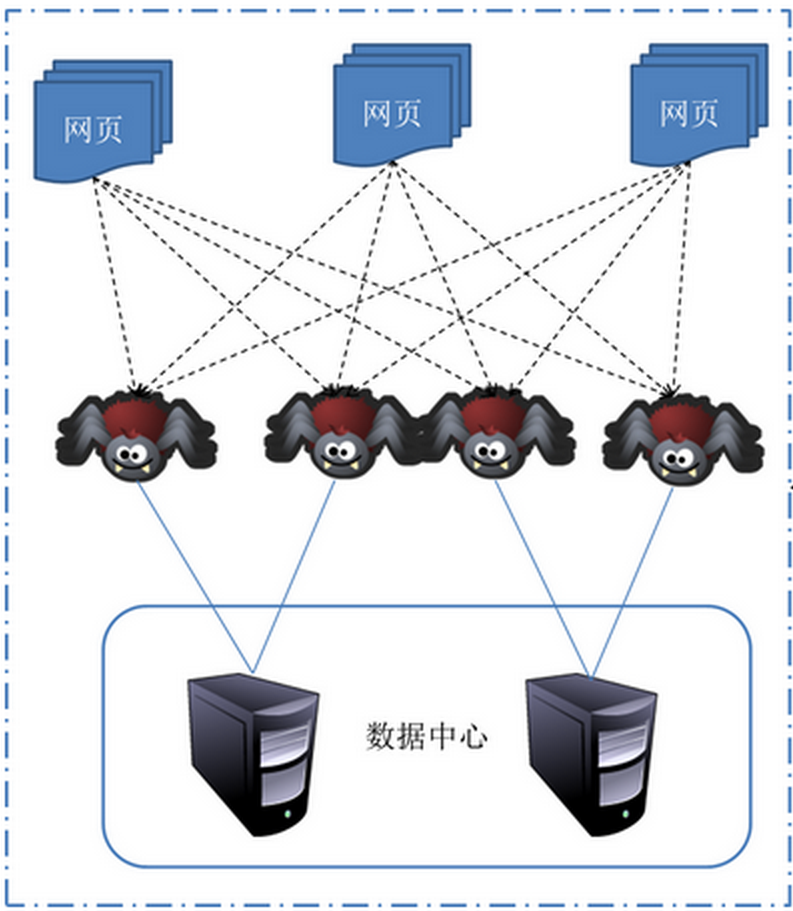


图2.1 网络爬虫功能结构图

图2.1展示了网络爬虫承上启下的功能结构，最上层是互联网，互联网由很多个URL组成，中间层由许多个网络爬虫组成，而且每一个爬虫按照人为的设定，都会去爬取所有的URL，并且将爬取结果分布的存放在数据中心中。

由于Kaggle网站并没有提供知识库，所以需要为构建本地知识库，所以使用了网络爬虫技术，其主要功能就是对美国教育类网站CK-12中与八年级科学相关的六个相关页面的html进行解析，得到与八年级科学相关的关键词。再使用这些关键词去维基百科官网爬取对应关键词的正文部分，形成本地知识库。

### 2.1.2 urllib2网络爬虫

urllib2是众多网络爬虫技术中的一种，它是一个由python实现的获取URLs的组件，并以urlopen函数的形式为用户提供了一个非常简单的接口。同样urllib2也提供了一些复杂的接口用来处理一般情况，例如：基础验证、cookies、代理和其他。由于用python实现，所以代码简洁，其主要步骤为：

(1) 首先使用urllib2库的提供的API接口urlopen获取URLs，返回response，再用response提供的read方法，可以获取页面的html代码；

(2) 使用python提供的BeautifulSoup工具包，通过parser对html进行预处理，相当于将html代码储存到一种自定义的html树中；

(3) 在网页中寻找并定位目标关键词，通过urllib2提供的find\_all函数对预处理后的html进行解析，返回关键词列表。

### 2.1.3 request网络爬虫

request也是一种基于python开发的网络爬虫技术，与urllib2的原理相同，但不同的是它的条理更加清晰，因为使用了用面向对象的思想来设 计爬虫类非常易于理解，这也是本课题采用这种爬虫方法的原因之一，其工作流程如图2.2所示。



图2.2 request爬虫工作流程图

这是一个爬虫类Spider，其中共实现了4个函数，与urllib2的爬取顺序类似，都是先获取URL的html代码，然后定位关键词，解析出目标关键词，具体操作如下：

(1) 首先使用getsourse函数用来获取目标URL页面的html代码，所以输入是URL，通过get方法可以得到目标页面的html对象，html对象的text成员即为该页面的html代码；

(2) 然后使用geteveryclass函数获取关键词所在的类，得到html正文，定位关键词地址，使用python的正则表达式re，用其findall方法对目标html代码进行解析，得到html标签中的外层类；

(3) 传入每一个类，使用getinfo函数，对每一个类里的title进行解析，返回关键词信息；

(4) 使用saveinfo函数用于接收title列表，并将关键词列表存入到文本文件中。

## 2.2 Lucene搜索引擎

### 2.2.1 Lucene简介

Lucene是一个基于Java的开源工具包，其内部实现了搜索引擎的架构，用户可以使用Lucene提供的接口构建出自己的搜索引擎[6]。Lucene由Java开发，被移植到了许多语言中，比如python语言的pyLucene，而且python还提供了一个可以搭建搜索引擎的工具包whoosh，通过多方面的比较，为了更好的原生性和可拓展性，本课题仍采用Java语言开发的Lucene3.6.0版本。Lucene免费为用户提供了一套简单却强大的API接口，可以让用户自己根据需求在本地搭建一个搜索引擎。其系统架构如图2.3所示。



图2.3 Lucene系统架构图

如图，Lucene通过Indexation模块建立索引，对Doucment进行抽象并解析成索引结构，通过Database、File system或者RAM将索引存储起来。另一方面用户进行Searching，对用户的搜索进行解析并搜索，通过排名算法即可返回结果列表，Lucene作为一个全文检索引擎，其具有如下突出的优点：

(1) 索引文件格式独立于应用平台。它专门定义了一套索引方式，无论在Windows、Linux、MacOS上都可以访问这套索引[7]；

(2) 传统的搜索引擎大多使用倒排索引技术建立索引，Lucene不仅也使用了倒排索引，而且又实现了分块索引技术。每一次索引过程可以在之前的索引上进行叠加，也能够针对新的文件建立小文件索引，这大大减少了建立索引的时间，而且新建立的索引可以与原来的索引合并，而不是有新的数据之后需要对所有数据重新建立索引；

(3) 设计并继承了Java语言面向对象的特征，只要熟悉Java就非常简单易于上手Lucene，并且很方便用户在类中添加新的方法；

(4) 用户只需要根据自身的应用场景实现Lucene提供的文本分析接口，就可以创建索引文件，没有其他的复杂操作就能写出创建索引的算法[8]；

(5) 内嵌了全文检索的大部分算法，形成了功能非常强大的搜索引擎。而且搜索排名这部分对于用户来说是透明的，不需要纠结于搜索结果的排名算法[9]。

### 2.2.2 倒排索引

大多数搜索引擎由如下几部分组成：使用爬虫爬取url、分布式存储网页镜像、对页面进行预处理、创建索引、全文检索。用户要在庞大的线下存储库中查找用户需要的网页犹如大海捞针，从用户体验的角度搜索引擎必须在毫秒级别给予用户满意的结果，否则用户不会使用这款搜索引擎，因为需要等待的时间与搜索的信息价值不对等，所以为了弥补这种时间上的损失，多数搜索引擎都通过以下流程使用建立索引技术，最著名的就是倒排索引技术，如图2.4所示。



图2.4 搜索引擎建立索引顺序图

在互联网中，先通过之前介绍过的网络爬虫技术爬取所有网页的镜像存储在线下服务器中，然后对页面做分析，找到这个页面的各个部分，比如title、content、link等等信息，把这些信息储存起来，因为信息可以代表这个页面，然后对这个页面做自然语言处理，使其成为标准的文本格式，最后就可以建立索引了，本课题将讲解倒排索引的建立过程。

倒排列表用来记录有哪些网站包含了某个单词，一般在网页集合中会有很多网页包含某个单词，每个网页会记录网页编号，单词在这个网页出现的次数及单词在网页中哪些位置出现过等信息[10]，这样与一个网页相关的信息被称作倒排索引项，包含这个单词的一系列倒排索引形成了列表结构，这就是某个单词对应的倒排列表。其组织结构如图2.5所示。

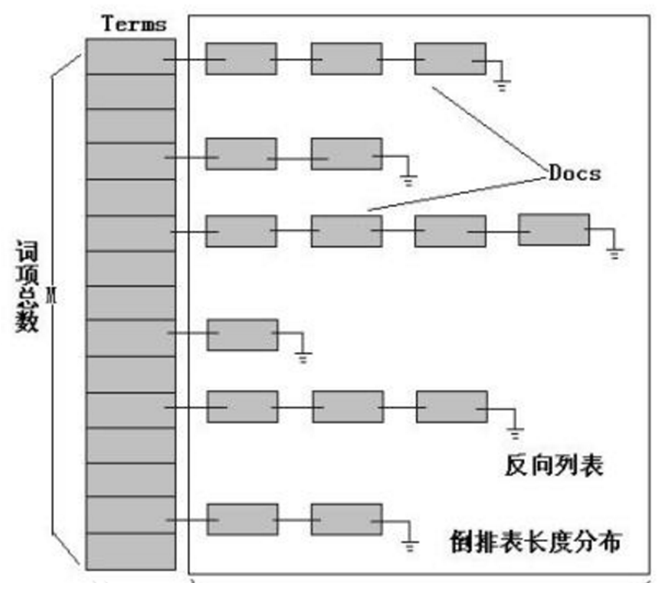


图2.5 倒排索引结构示意图

### 2.2.3 检索

检索部分是搜索引擎非常重要的一部分，它提供了面向用户的接口，对用户的输入表达式进行解析，解析成关键字词，再通过建立的索引进行检索，通过倒排索引等索引技术返回得到相关网页，对网页再使用排名算法，即可呈现显示给用户。

而排名算法中的关键就是pagerank分数，pagerank分数越高，则搜索结果越排在前面，不同的搜索引擎对pagerank的评定方法不同，但通常受关键词分数、域名权重、外链接分数、用户数据、内容质量分数以及人工干预影响[11]。

Lucene中的检索部分需要一个Query对象作为参数，那么也就是将Expression组合成Query的过程，这里边有一个对象叫QueryParser，它将前面传过来的规则的解析成对象然后进行查询。其工作原理如图2.6所示。



图2.6 Lucene搜索结构图

## 2.3 计算语义相似度

### 2.3.1 关键词提取

关键词提取采用TextRank算法，它可以用于自动为文本生成关键字和文本摘要。其设计思想是受PageRank算法所启发。PageRank算法是应用于搜索引擎的检索部分的重要算法，用来为网页打分排序，它将所有网页看成一张有向无权图，每个网页都是一个节点，如果网页A存在指向网页B的链接，那么节点A就有一条有向边，如图2.7所示。



图2.7 网页关系示意图

图中显示A、B、C都为网页，网页B存在指向网页A的链接，网页C也存在一个指向网页A的链接，可以使用PageRank计算这三个网页的重要程度，通过PageRank算法公式的反复迭代，直至收敛，可以得出迭代100次的结果，网页A的权重为1.4595，网页B和网页C的权重都为0.7703，这与人们肉眼观察得到的结果一致[12]，在这三个网页所组成的集合中，网页A应该具有最高的权重。

而TextRank汲取了PageRank的思想，将文本拆分成句子，删除掉句子中的停用词，并且可以只保留指定词性的单词，将每个单词作为一个节点，设置窗口大小，在一个窗口中的任意两个单词对应的节点之间都存在一条无向无权的边构成图[13]，这样就可以利用PageRank算法提供的打分公式计算图中每个单词节点的重要性，为每一个单词进行打分，直到分数收敛。返回前k个最重要的词或者设计重要度阈值即可得出文本中的关键词。

本课题中采用了python语言中的TextBlob工具包实现了关键词提取功能，TextBlob是在python语言中的NLTK工具包和pattern工具包的基础上构建的工具，它实现了使用TextRank算法来计算词的权重，并提取关键词，非常方便并且易于使用。

### 2.3.2 TF-IDF算法

TF-IDF(term frequency-inverse doucment frequency)算法是统计学、信息检索和数据挖掘等领域非常重要的算法，应用于许多领域[14]，但它并没有复杂的数学的公式推导与演算，因为只需要统计目标词词数、文章词总数，所以实现起来也非常简单，但却在各个领域中发挥着不可替代的作用。其中TF代表词频，IDF代表逆向文件频率，该算法的主要思想是如果一个单词或者短语在一篇文章中出现的次数除以词的总数很高[15]，也就是词频很高，并且在其他文章中出现很少，也就是逆向文件频率很高，即TF和IDF都越大，也就是TF乘以IDF的值越大[16]，相当于TF-IDF算法给予的评分很高，那么说明这个词或者短语具有很强的区分能力，更适合用来分类。

TF-IDF算法应用非常广泛，在计算搜索排名的时候也可以采用，计算用户输入的关键词在所有相关网页中的TF/IDF，得到的结果值越大，就意味着用户输入的关键词在网页中出现的频率很大，并且在其他网页中出现的频率很小，那么这个词就具有很好的分类特性，也代表用户输入与这个网页越相关。在提取关键词中也有TF-IDF算法的一席之地，删除停止词后，在一篇文章中某个词的词频非常高，那么也可以说明这个词很能代表这篇文章，也就是这篇文章的关键词。

本课题中使用TF-IDF算法主要是为给答案打分，对于每一个选择题答案，提取出关键词，然后以Lucene返回回来的前k条最相关文本为语料为每个关键词计算并记录的TF，然后除以关键词个数得到平均词频，关键词平均词频最高的答案即为结果，这里根据具体问题的不同而使用的是优化后的TF/IDF算法。

TF-IDF的优点是实现简单但效果却还不错，所以具有很高的性价比，但如果只依靠该算法对文章进行分析，那么会忽略词与词之间的位置关系等重要信息，而一般文章中句首的词通常应该具有更高的权重，每一段落的第一句很可能是这个段落的核心句，这些都是TF-IDF算法所忽略掉的重要信息，所以不能单单靠该算法解决所有的情况。为了弥补这种情况，本课题使用了在与语义层面上对词可以进行表示计算的可以计算出词向量的模型Word2vec。

### 2.3.3 Word2vec模型

Word2vec是Google在2013年推出的可以计算出每个单词的词向量的一个工具包，它的效率很高并且效果也很好，所以很多人将它应用在自然语言处理的各个方面，使用起来比较简单，因为Google官方给出了一个通过大量语料并且训练了很长时间的300维的词向量语言模型，并且通过C++、python等语言都可以解析这个bin文件。但要想理解这个Word2vec工具的内在原理还是需要下功夫的[17]。

统计语言模型是自然语言处理中非常重要的部分，通过贝叶斯公式结合马尔科夫模型，采用最大对数似然估计，只需要根据语料库求出语言模型的参数，就可以计算出语言模型。计算语言模型参数的方法有很多，其中神经概率语言模型是比较好的一种[18]，它也是Word2vec工具思想的基础。其计算原理如图2.8所示。

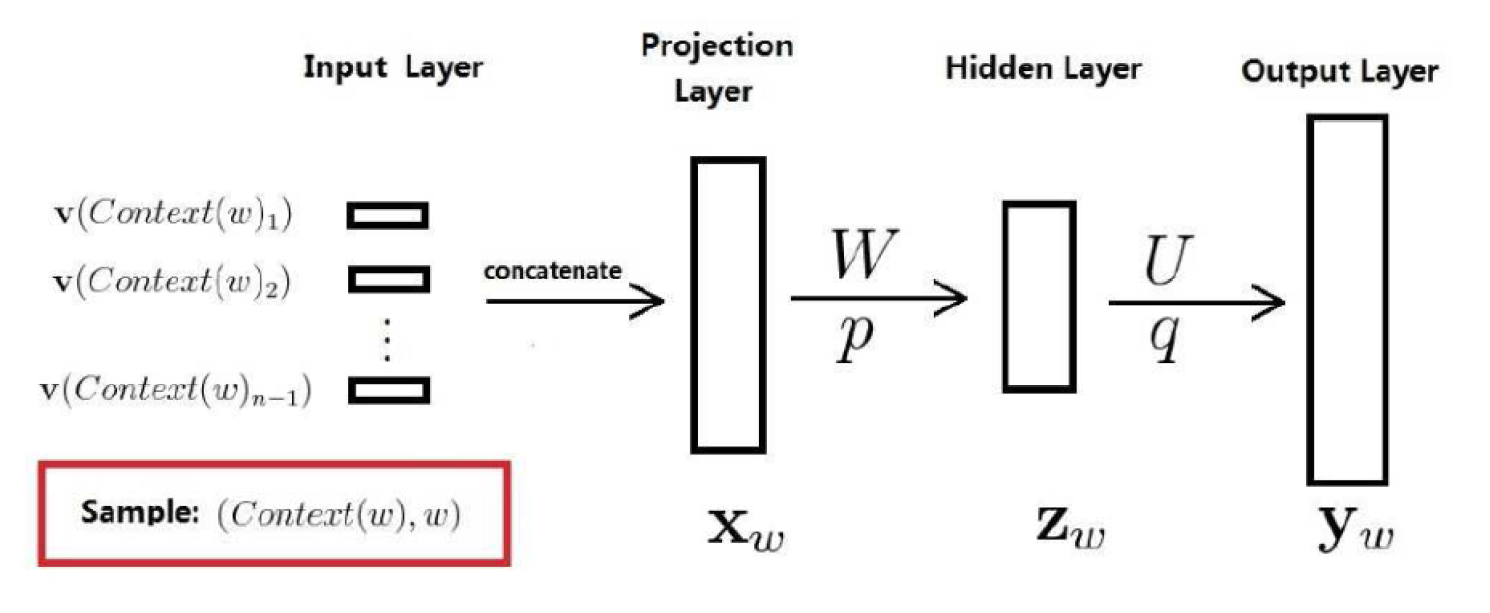


图2.8 神经网络结构示意图

如图所示这是一个四层的神经网络模型，输入层、映射层、隐藏层和输出层，其中输入层为对于语料中的任意一个单词w，Context(w)为其前面的n-1个词，这样的二元对就是一个训练样本。其中W,U分别为投影层与隐藏层以及隐藏层和输出层之间的权值矩阵，p和q分别为隐藏层和输出层上的偏置向量，再使用tanh双曲正切函数作为隐藏层的激活函数，其作用是在向量上表示它作用在向量的每一分量上，最终在输出层做softmax归一化，就可以计算出p(w|Context(w))的表达式，其中输出的值为：

 (2-1)

经过softmax归一化之后，其分量就可以代表当上下文为Context(w)时，下一个词为词典中第i个词的概率，就可以带入最大似然对数函数，使用随机梯度下降法估计参数，调节模型。

这种神经网络模型相比于原始的n-gram模型有很大优势，首先词语之间的相似性可以通过计算词向量的欧氏距离来计算，其实，神经网络模型计算的概率结果自带平滑效果，不会出现0或者1的结果。但明显的不足就是整个模型的计算都集中在隐藏层和输出层之间的矩阵向量运算，以及输出层之上的softmax归一化处理，而且这部分的复杂度是非常高的，所以Word2vec针对这部分进行了优化。

Word2vec用到了两个重要的模型，CBOW(Continuous Bag-of-Words Model)模型和Skip-gram模型[19]，如图2.9所示。

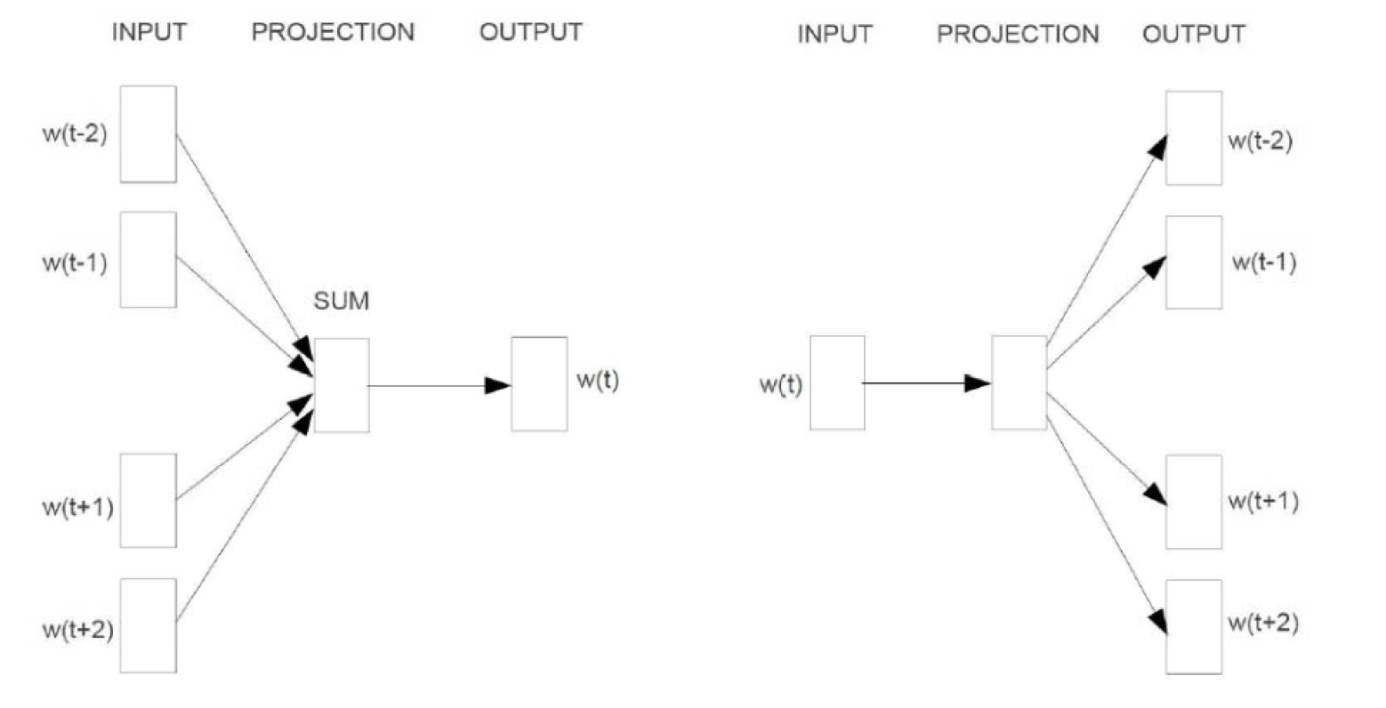


图2.9 CBOW模型结构图(左)和Skip-gram模型结构图(右)

如图所示，CBOW模型和Skip-gram模型都只包含三层神经网络：输入层、映射层和输出层。CBOW模型是在已知目标词的上下文词的前提下预测目标词，而Skip-gram模型相反，是在已知目标词的情况下预测该词的上下文词。Word2vec共给出了两套框架，它们分别基于Hierarchical Softmax和Negative Sampling进行设计，其中Hierarchical Softmax框架的CBOW模型，如图2.10所示。

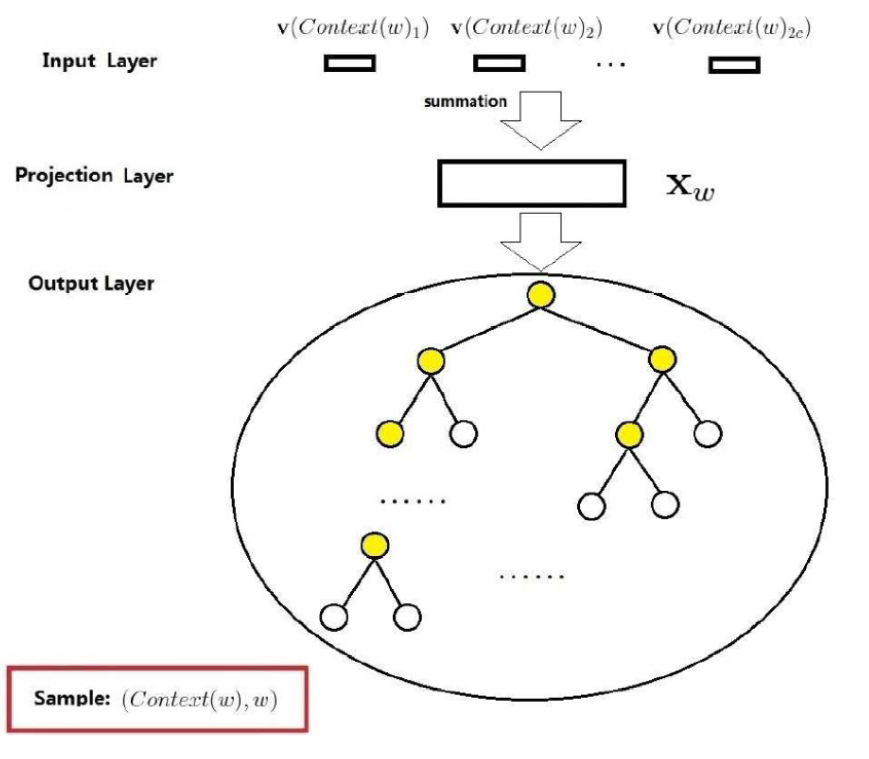


图2.10 CBOW模型的网络结构示意图

如图，与传统的神经网络语言模型不同的是其输入层不在是向量的简单拼接而是词向量的累加求和，并且删除掉了隐藏层，输出层也并没有采用softmax归一化，而是采用Hierarchical Softmax，其用赫夫曼树代替了原来的线性结构，由于神经网络模型中绝大多数的复杂计算都集中在隐藏层和输出层之间的矩阵向量运算，以及输出层上的softmax归一化运算，所以这样优化会很大程度的减少了复杂度。

将赫夫曼树的每一层都看成是一个二分类问题，通过sigmoid期望函数将每一个二分类的概率相乘，就可以得到最终叶子节点出现的概率。通过最大对数似然的思想，使这个概率最大，所以就采用随机梯度下降法通过迭代求得语言模型的参数，而期望得到的词向量就是语言模型的中间产物。

第二种模型就是使用Skip-gram模型实现了Word2vec，和CBOW模型的网络结构一样，它也包括输入层、投影层和输出层，以样本(w,Context(w))为例，对这个三层的神经网络做出简要说明，如图2.11所示。

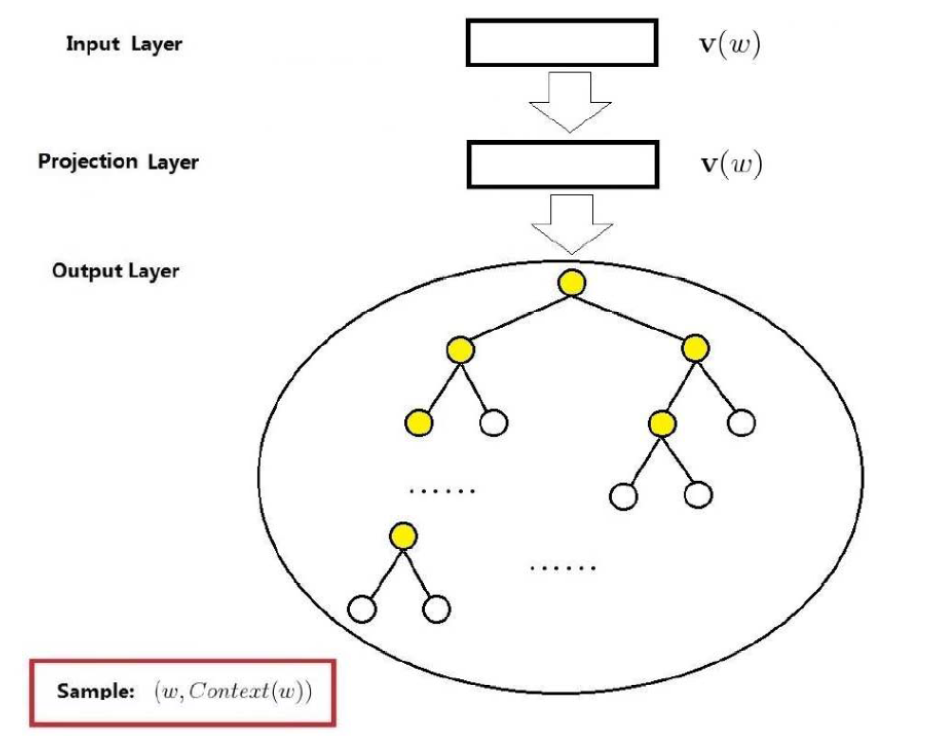


图2.11 Skip-gram模型的网络结构示意图

如图，输入层只包含当前样本的中心词w的词向量，这里投影层是一个恒等投影，把v(w)投影到v(w)[20]，所以这个投影层其实是多余的，这里之所以保留投影层主要是方便和CBOW模型的网络结构做对比，其输出层和CBOW模型一样都是一颗哈夫曼树。

CBOW模型和Skip-gram模型可以理解为是两个相反的求解过程，CBOW是根据目标词的上下文词求出目标词的词向量，而Skip-gram则是根据一个词汇的词向量求解其上下文词汇的词向量，但最终的目的都是计算出每个词的词向量，而且都是构造出最大似然对数，然后通过二分类的概率乘积得到Context(w)和w的关系，通过随机梯度下降法进行迭代直至模型收敛，然后得到这套参数构建语言模型，所以使用哪一种模型都可以[21]。

## 2.4 Django框架

Django是python中的一个定制web框架，也是一个基于MVC构造的框架，但是Django更关注的是模型、模板和视图，这也是一个MTV模式。它源于一个新闻站点，本课题使用Django不仅因为它功能强大，也是它结构简单易于上手。由于也是python语言，所以对本项目的结合效果非常好。本课题使用python2.7和Django1.7.6搭建了一个web工程，其目录结构如图2.12所示。

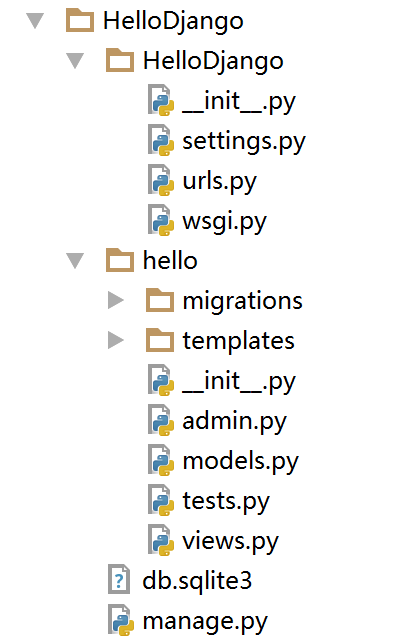


图2.12 Django项目结构图

如图，这是一个名为HelloDjango的项目，其中hello是它的一个app，其中urls.py是网址的入口，它会关联到对应的view.py中的一个函数。view.py用来处理用户发出的请求，比如提交表单、用户请求数据等操作。model.py是用来管理数据库的脚本，templates文件夹中保存着html模板，可以用来缓存来提高速度。admin.py可以使用很少量的代码就能拥有一个强大的后台。setting.py里包含Django的设置，配置文件，DEBUG的开关，静态文件的设置等。

## 2.5 本章小结

本章介绍了本课题中使用到的或相关的技术，有网络爬虫技术、Lucene搜索引擎技术、用于提取关键词的TextRank算法、TF-IDF算法和Word2vec工具，讲解了Word2vec的设计思想与实现过程以及Django框架的使用，并着重介绍了本课题中使用的方法。

# 第3章 搜索引擎模块的设计与实现

本章介绍了问答系统搜索引擎模块的设计与实现，首先对本模块的体系结构进行概述，其次对知识库建立的过程进行详细介绍，最后阐述搜索引擎的建立索引与检索部分，并且进行测试与总结。

## 3.1 搜索引擎模块的体系结构设计

搜索引擎模块是本课题的第一部分，对之后的语义相似性分析打下了基础，这部分产生结果的优劣也直接应该着语义相似性分析的准确率。在整个搜索引擎模块中，主要包含两大部分，第一部分是知识库的爬取与创建，第二部分是Lucene搜索引擎的搭建，包括创建索引与全文检索，如图3.1所示。



图3.1 搜索引擎模块体系结构图

如图所示，该模块由两大部分组成，爬虫部分和Lucene搜索引擎部分。在实现本课题时先处理的是爬虫部分，以为设计并实现一个问答系统，首先就要有知识库，否则无法建立索引，搭建搜素引擎。建立完知识库之后，使用了Lucene为知识库建立了倒排索引。解析测试集的输入，并进行预处理，然后使用Lucene提供的API对索引进行检索，规定得出结果的格式，即得到了与目标问题相关的前k个文本文件的路径。下面分别详细介绍知识库的建立过程和搜索引擎的搭建过程。

## 3.2 知识库建立

一个问答系统产生结果的好坏，很大程度上受与其相关的语料库影响，本课题的用户需求输入是美国八年级科学类的选择题，所以应该对八年级科学类选择题构建知识库，为问答系统储备知识。

### 3.2.1 关键词爬取

Kaggle网站本课题比赛的官网没有给出本问答系统所需的知识库，而是建议参赛者自己构建知识库，在论坛中有网友分享了CK-12网站，这是一个面向美国教师和学生的学习类网站，里面有义务教育阶段的所有学习资料，所以决定爬取CK-12网站上与美国八年级科学相关的所有关键字。

该网站与科学相关的页面共有6个，分别是earth-science、life-science、physical-science、biology、chemistry和physics。本课题中使用了之前介绍的python语言实现的requests工具包来爬取这些关键字。首先定义了爬虫类Spider，在爬虫类中共实现了4个函数，它们的作用分别是获取页面的html、或者包含目标 关键词的类、从类中解析出关键词获取关键词信息列表、存储关键词列表到本地文本文件，文件格式只有一列，列中每一项代表一个关键词。最终结果如图3.2所示。

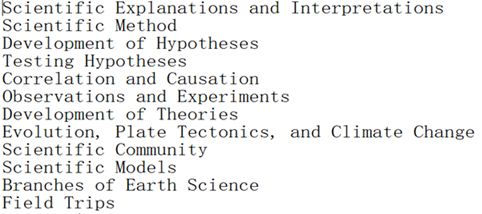


图3.2 关键词列表图

如图所示，最终将关键词列表都存放在一个文本文件中，共有2139个与美国八年级科学相关的关键词，下面将围绕着这些信息构建本地知识库。

### 3.2.2 正文爬取

只爬取了相关的关键词是不够的，因为还并没有形成可供检索知识库，所以还需要在网络中爬取与这些关键词相关的知识，有了关键词之后要爬取知识库，也就是爬取与这些关键词相关的文章，或者说就是爬取以目标关键词为标题的文章，所以可以爬取各大网站的百科，通过比较后发现，维基百科对英文关键词很友好而且强大，并且维基百科官方为python语言提供了强大的wikipedia工具包，这个工具包提供的API可以模拟网页中关键词的输入，通过提供的page方法返回指定关键词的content、url和title等信息，其中的正文就是本课题中所需要的目标关键词的正文信息。所以使用wikipeida爬取下的关键词正文建立知识库，知识库组织结构如图3.3所示。



图3.3 知识库结构示意图

最终通过使用python语言中的wikipedia工具提供的API爬取这2139个关键词的维基百科正文，将爬取结果以关键词名为文本文件名称，文本内容为爬取内容正文进行存储，形成未经过处理的知识库，其具体形式如图3.4所示。



图3.4 普通文本知识库列表图

共用了十多个小时的时间爬取维基百科正文，并且按照不同的文本文件进行存储，但有一个问题就是只爬取到了1469条数据，这是因为一些关键词在维基百科中并没有与之相对应的词条，所以也就无法获取关键词的正文，所以只能先使用这1469个文本文件建立知识库，但显然这会影响最终的结果，因为会有一些用户输入的问题可能搜索不到与其关键词相关的文本文件。

### 3.2.3 关键词提取

从维基百科中爬取目标关键词的正文之后，虽然是人易于理解维基百科正文，但对于计算机是不易于计算的，所以要对文本进行预处理操作，删除掉文本里面冗余的停用词等无关信息。

本课题使用了python语言提供的BeautifulSoup工具对文本进行了清理，清除了里面的html冗余信息，将所有字母转换为小写字母，删除掉各种符号信息，因为计算机无法理解这些信息。

因为最终要比较语义相似性，那么如果把文本信息中的关键词提取出来，会一定程度上减少其他无用词的噪声影响，所以在这里使用两个知识库，分别对他们进行存储和建立索引，这里需要用到关键词提取的技术。第二章已经提到过本课题使用的是python语言中的TextBlob工具包，它实现了TextRank算法，计算了每个词的权重，设置阈值删除掉了重要程度较小的一部分词，但返回结果多数是单词短语，所以又对文件进行分割，让每一个词以逗号分割，形成第二套知识库，其中知识库组织结构与图3.3一样，其具体形式如图3.5所示。

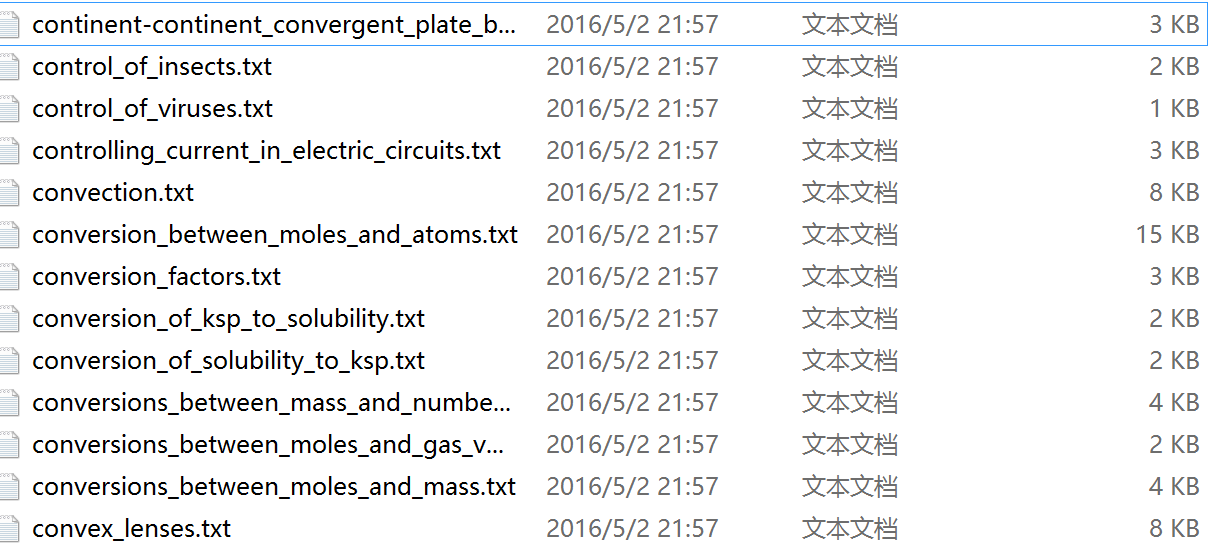


图3.5 关键词文本知识库列表图

可以看出，每一个文本文件都变小了很多，因为这里面存储的都是原来维基百科正文的关键词，有了这两套知识库，再使用Lucene分别对其建立索引，最终使用语义相似度的计算比较究竟哪一个语料库的效果更好。

## 3.3 搭建搜索引擎

### 3.3.1 建立索引

Lucene中搭建索引的相关技术部分在第二章也已经介绍过了，这部分只需要介绍具体实现细节。本课题使用了Java语言中的Lucene工具包搭建了一个本地搜索引擎，这部分着重介绍搭建搜索引擎中建立索引的部分。

在Java中建立了一个Lucene类，这个类实现了两个主要函数，一个是index也就是建立索引的操作，另一个是searcher，也就是进行搜索的操作。在建立索引的过程中，先创建索引地址，将索引地址存储在一个directory中，然后创建一个Document类型的对象，根据需求为这个对象添加三个域，分别是正文域、文件名域和文件路径域。然后使用Lucene提供的对应版本对应解释器的writer对象将文本文件都写入到Document中[22]。这样就为维基百科正文的文本文件建立了索引文件，如图3.6所示。

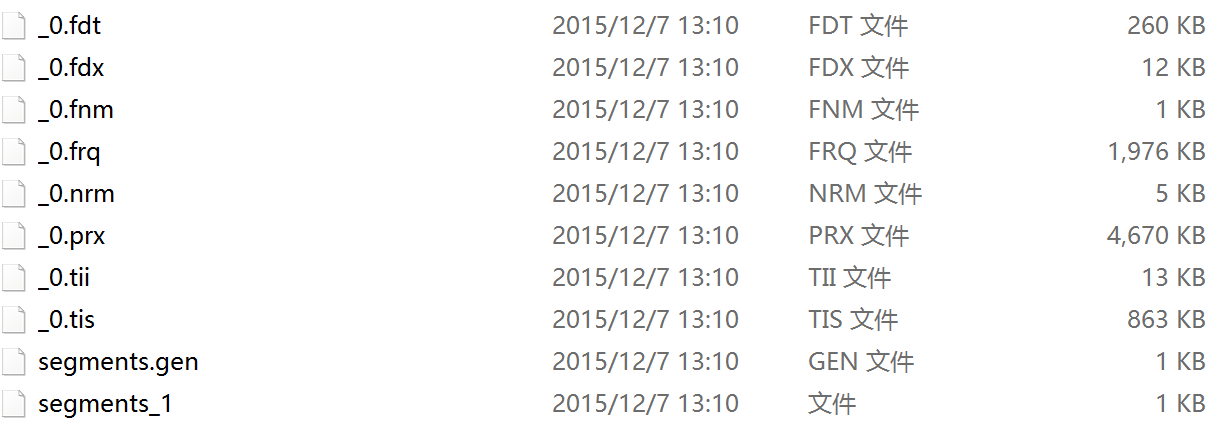


图3.6 Lucene索引结构图

通过Lucene创建出如图所示的索引结构，其中包含了段、文章、词、域等信息，通过Lucene的检索部分可以对索引进行检索，得出结果。

### 3.3.2 全文检索

Lucene建立完索引之后就要根据用户的输入进行检索了，所以对于用户的输入需要进行语法分析、句法分析及自然语言处理，形成几个与搜索相关的关键词。通常在搜索引擎进行全文检索，首先在反向索引表中，分别找出包含这几个关键词的文档链表，然后对这些链表进行合并与差操作得到目标文档，之后的任务就是对文档与文档的相关性进行打分，分数高的相关性好，就应该排在前面。

为了后续计算语义相似度的工作能更好的组织并检索结果，将Lucene搜索结果按照图3.7的结构进行设计。



图3.7 Lucene搜索结果结构图

这个文本文件的格式如图所示，每一个元组由两部分组成，第一部分是目标问题，然后以制表符分割，第二部分是Lucene对这个问题进行搜索，得到的与这个问题最相关的10条文本文件的绝对路径。

Lucene搜索结果按照图3.7的格式被存放到一个很大的文本文件中，这便于最终对所有问题共同计算语义相似性，因为这样存储减少了文件的打开关闭操作。测试集中共有2500对问题与答案，所以这样的二元组共有2500个。之所以这样设计的原因也是为后续语义相似度分析中的关键数据结构二级字典的遍历提供的方便。对于一个问题没有10个与之相关的文本文件时，以空路径作为填充，由于知识库相对充足，所以这种情况并不多见。其Lucene搜索结果如图3.8所示。

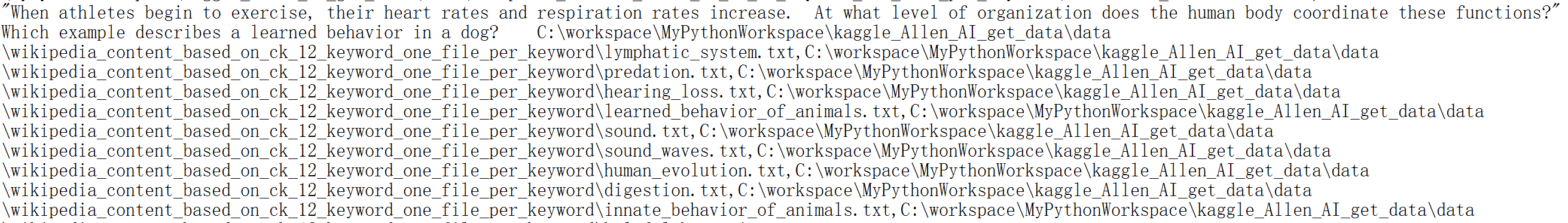


图3.8 普通文本Lucene搜索结果图

## 3.4 本章小结

本章详细介绍了知识库在互联网上的爬取过程与预处理过程，其中包括对CK-12网站的关键字进行爬取、对目标关键字的维基百科正文进行爬取以及提取文本中的关键词。又详细介绍了Lucene搜索引擎建立索引和全文检索的过程，这两部分的组合内容是本课题的根基，只有根基打的好，下一步语义相似度的检测才更有意义，才能更加准确，否则就是对着垃圾数据做无用的语义分析。

# 第4章 语义相似度计算模块的设计与实现

通过之前的模块已经对测试集中2500道美国八年级科学类选择题分别返回了前k篇最相关文档的绝对路径，这些信息保存在一个文本文档中，本章的内容就是分析这些文档与四个答案选项之间的语义相似性，为每一个答案选项进行打分，分数越高则它是准确答案的概率就越大[23]。

## 4.1 语义相似度计算模块的体系结构设计

语义相似性计算模块包括两部分输入，第一部分是与问题最相关的前k篇普通文档和问题的四个答案选项作为输入，第二部分是与问题最相关的前k篇只包含关键词的文档和问题的四个答案选项作为输入。本课题共使用两种算法对语义相似度进行分析，第一种算法是TF-IDF算法，对TF-IDF算法进行一些改进之后应用到本课题求解问题答案与相关文档的相似性，第二种方法是使用了谷歌2013年推出的Word2vec工具，这是一个优化后的神经网络语言模型，本课题主要使用了该模型的中间产物词向量，词向量是将每个单词映射成300维的空间向量，使用了平均向量法和最大维度法进行语义相似度分析，语义相似度模块的体系结构图如图4.1所示。



图4.1 语义相似度计算模块的体系结构图

## [4.2 TF-IDF算法计算语义相似度](file:///E:\2015毕业设计\0毕设\张晓丽版本-04.doc#_Toc327175924)

本节主要描述TF-IDF算法在本课题的应用，使用改进之后的TF-IDF算法，计算普

通文本与答案、关键词文本与答案之间的语义相似度，为答案进行打分，求出分数最高的答案。

### [4.2.1 TF-IDF算法的改进](file:///E:\2015毕业设计\0毕设\张晓丽版本-04.doc#_Toc327175925)

TF-IDF算法用于评估一个词对于一个语料库中的其中一份文件的重要程度，该词的重要性随着它在该份文件中出现的频率成正比增加，又同时随着它在语料库中其他文件中出现的频率成反比下降。但在本课题中，由于是互不相干文本文件组成的知识库，所以一组词对于一篇文章的重要程度虽然和在该文本中词的频率有关，但和其他文本中出现这些词的频率无关，因为这些答案中的词本身也是关键词，在其他文本中出现这些关键词只能说明其他文章与这个答案的相似度也很高，所以本课题并没有采用原始的TF-IDF算法，而是对TF-IDF算法做了针对本课题的需求进行了改进，实质也就是没有使用逆向文件频率来计算语义相似性。

 (4-1)

改进后的TF-IDF算法公式见(4-1)，式中分子是该词在在文件d中出现的次数，而分母则是在文件d中所有字词出现次数之和。这样可以计算出每个关键词的tf值，也就是每个词对于这几篇文本的分数，由于答案中每个词都是关键词，所以不会出现冗余的停止词，直接将四个答案选项中每个词的tf评分相加再除以答案选项中的关键词数，就可以分别求出四个答案选项的平均tf评分值，对于评分最高的答案进行输出。

### [4.2.2](file:///E:\2015毕业设计\0毕设\张晓丽版本-04.doc#_Toc327175926) 普通文本的语义相似度计算

本部分详细阐述使用改进之后的TF-IDF算法，计算四个答案选项和普通文本之间的语义相似度，因为Lucene搜索引擎部分已经返回了2500条问题及与其最相关的10个文本的绝对路径构成的二元组，所以需要一种数据结构将这些数据读取到内存中。本课题中使用python语言的字典对这些数据进行存储。

而TF-IDF算法中最重要的就是词的统计，每一篇文章包含了多少种词，每种词又分别出现了多少次，这些信息也都需要读取到内存中，本课题中使用二维字典对这些信息进行存储，这也是本部分最重要的数据结构，在TF-IDF算法中和Word2vec工具中都需要使用，二维字典结构图如图4.2所示。



图4.2 二维字典结构图

二维字典示意图如图4.3所示。首先外层的键是文件的路径，图中文件是resonance.txt的绝对路径，值又是一个字典，这个字典的键是单词，值是这个单词在本篇文章中出现的次数。

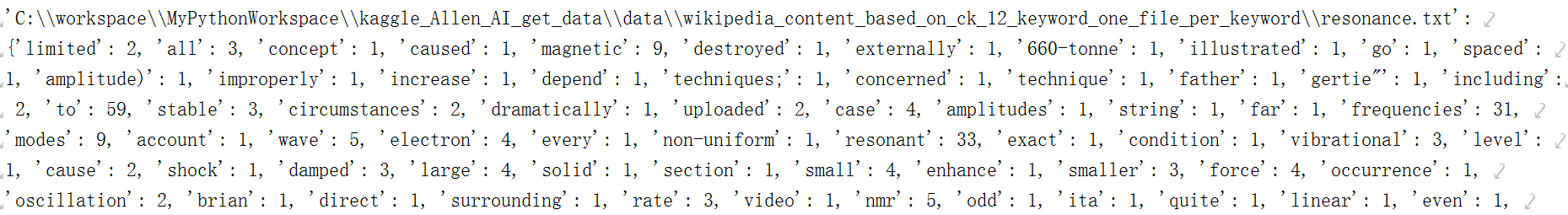


图4.3 二维字典示意图

具体实现过程为首先读取问题，然后读取该问题对应的文件列表，把四个答案选项里的关键词记录下来，使用二维字典遍历路径列表，查询每篇文章中是否含有答案中的关键词，如果有的话计数器累加count，没有则不进行累加。这样会得出一个总数，这个总数就是答案选项中的词出现在这几篇最相关文章中的词数，再计算答案选项中的次数，进行除法运算就可以计算出每一个答案的平均词频打分值，分数最高的答案认为它是与这几篇文档最相关的答案，即为最终答案。按照这种算法，可以计算出2500条答案记录，这其中一个问题就是对于用户输入的答案，Lucene有可能返回小于k篇相关文档，不到10篇的部分由空路径填充，甚至由于没有检索到而没有返回。

另外对于每一个答案的选项，可能文章中并没有包含这些词，这样根据TF-IDF算法的定义无法区分这四个答案的权重，对于这两种情况，都默认选择C，据统计这种情况出现的并不多，小于十次，按照Kaggle给出的输出要求，构建如图4.4的输出结构。



图4.4 输出答案结构图

最终输出结果如图4.5所示。由于进行输出的是测试集，测试题中的2500道题都有正确的答案，所以本课题使用了python写了一个比较答案正确的脚本文件，这种使用改进的TF-IDF算法与普通文本文档知识库的方法最终得到了百分之33.48的正确率。

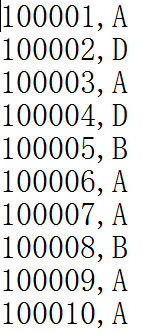


图4.5 输出答案格式

### [4.2.3 关键词文本的语义相似度计算](file:///E:\2015毕业设计\0毕设\张晓丽版本-04.doc#_Toc327175906)

本节详细介绍了改进的TF-IDF算法与关键词文本进行语义相似度计算，其原理与上一节普通文本文档计算语义相似度的原理是一样的。本课题使用Lucene对关键词文本又单独创建了一份索引，而且在Lucene检索部分中，也是使用关键词文本这份语料库对问题输入进行检索，检索结果的格式与普通文本一样，如图4.6所示。

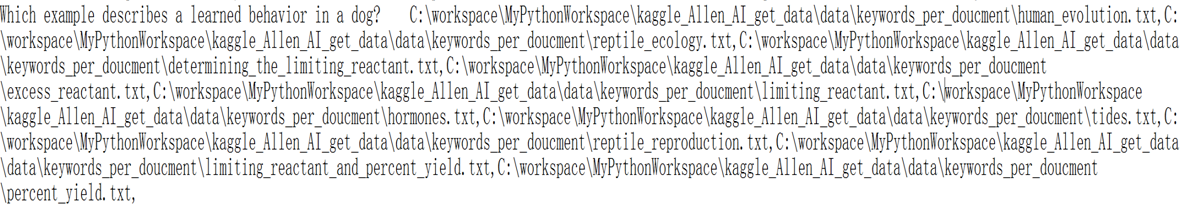


图4.6 关键词文本Lucene搜索结果图

Lucene搜索结果中包含2500个元组，该元组第一部分是问题，第二部分是与该问题最相关的10条只包含关键词的文本的绝对路径，使用与普通文本计算语义相似度一样的方法，最终得出百分之32.44的准确率。

使用关键词文本计算语义相似度的结果与使用普通维基百科正文计算语义相似度的结果相差不多，甚至还变低了，经过分析，主要原因是在抽取关键词的过程中虽然会减少无关词的噪声，但也极其可能删除掉了与答案有关的关键词，这种情况下，统计答案词频的时候就会减小对该答案的打分，导致结果错误。但提取出关键词再进行语义相似度计算也是一种可以尝试的方法。

## [4.3](file:///E:\2015毕业设计\0毕设\张晓丽版本-04.doc#_Toc327175927) Word2vec模型计算语义相似度

本节采用Word2vec模型计算语义相似度，第二章已经介绍过Word2vec模型工具，使用Word2vec生成语言模型的过程中会产生中间产物，也就是本课题需要使用的词向量，每一个词在向量空间上都对应着一个向量，词的意思越相近，那么这两个词对应的向量的欧式距离就越小，或者说这两个词对应的向量之间的夹角就越小，正是由于词向量的这个特征，本课题就采用了Word2vec所生成的词向量的运算生成能代表文档的向量，使用这些向量再与选项答案的句子向量进行语义相似度的计算。

### [4.3.1 Word2vec模型的使用](file:///E:\2015毕业设计\0毕设\张晓丽版本-04.doc#_Toc327175928)

本课题使用了谷歌提供的已经训练完成的语言模型GoogleNews-vector-negative300，顾名思义，其训练的语料库为GoogleNews，训练的维度为300维，但采用的并不是Hierarchical Softmax的模型，而是采用的基于Negative Sampling的模型。Negative Sampling是Tomas Mikolov等人提出的，它是Noise Contrastive Estimation的一个简化版本，目的是用来提高训练速度并改善所得词向量的质量。与Hierarchical Softmax模型相比，Negative Sampling不再使用复杂的哈夫曼树，而是利用相对来说比较简单的随机负采样，使用随机负采样算法能大幅度提高性能，因此可以作为Hierarchical Softmax的一种代替。

Noise Contrastive Estimation的细节比较复杂，其本质是利用已知的概率密度函数来估计未知的概率密度函数[24]，简单来说，假设未知的概率密度函数为X，已知的概率密度为Y，如果得到了X和Y的关系，那么X也就可以求出来了。

CBOW模型和Skip-gram模型在第二章都已经介绍过了，本节着重介绍一下负采样，对于任意一个给定的单词w，目标是计算这个单词的负采样NEG(w)。词典D中的词在语料C中出现的次数有多有少，对于那些高频率的词，被选为负采样本的概率就应该比较大，反之，对于那些低频率的词，其被选中的概率就应该比较小，这就是负采样过程的一个大致要求，其本质上就是一个带权采样的问题。谷歌正是使用这样的方法对文章进行训练，得出语言模型和中简产物词向量所构成的模型。

python语言中的gensim工具包里提供了Word2vec模型的实现，它可以训练自己的Word2vec模型，也可以直接使用别人已经训练好并且已保存的模型，本课题中使用的就是谷歌官方推出的模型，因为它训练的语料库足够强大，训练的时间也足够充足，并且已经有很多优秀的尝试都是基于这个模型。

这个工具使用起来就非常简单了，只需要导入必要的文件，使用gensim提供的load函数就可以将二进制Word2vec模型文件读取到内存中，这样这个model就可以做很多事了，求一个单词的词向量是最简单的了，只需要model[word]就可以返回一个300维的词向量，也可以使用这个模型对语义进行分析，找出与目标词汇最接近的k个词，最相反的k的词，或者找出一组词中语义与其他最不相关的一个词，甚至可以在语义基础上为每个词做加法减法等操作，来增加一个词或者减少一个词相对于另外一个词的语义相关性，可以得出新的词汇。

使用了Word2vec提供的大多数API，的确就像一个词汇空间，每一个词都是一个向量，非常强大，可以根据需求在语义程度上做很多事情，而本课题中只使用了Word2vec最基本的功能，也就是获取单词的词向量，因为词向量与语义相关，所以在单词的词向量的基础上，本课题进行分析并计算答案与文本之间的语义相似程度。

### 4.3.2 平均向量法计算语义相似性

本节介绍使用平均向量法计算选项答案与相关文本之间的相似度，了解了Word2vec模型的数学原理之后，具体实现比较简单。沿用TF-IDF算法计算语义相似度的做法，依旧保留着双重字典这个数据结构，对四个答案选项，分别计算其词向量之和，再除以词的个数，如果模型中没有这个词，那么以300维零向量进行补空操作。

本课题使用了python语言中的numpy工具包，这个工具非常强大，可以将300维的列表映射成为向量，并且可以直接进行向量的加减乘除等运算，所以这里300维向量的累加和除权非常容易实现，最终每一个答案选项都会得到一个向量，一共是四个答案向量。

由于计算量太大，前10篇最相关的文本文件在这部分只取前4篇文本进行计算，同样是对每个词的词向量进行累加，这里可以用到之前存储的二维字典这个数据结构，直接遍历字典中的所有词，得到它的词向量，再乘以这个词出现过的次数，累加到和向量上，最终再除以这4篇文本中的总词数，这样对每个词都会得到一个平均向量，假设答案向量为A，文本向量为B，如图4.7所示。

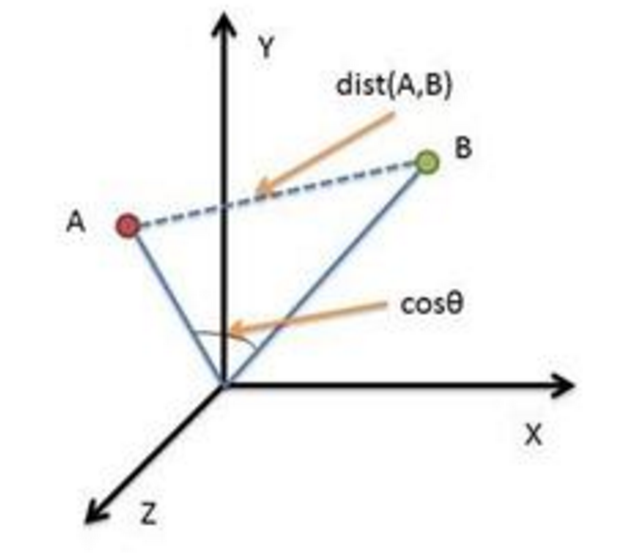


图4.7 向量运算求解语义相似度

假设将300维向量映射到三维空间中形成三维向量，则利用余弦定理即可计算出夹角大小，或者用欧氏距离计算向量距离，就可以计算出四个答案向量与文本向量的距离。与平均文本向量最近的那条选项向量，认为它在语义相似度层面最接近正确答案，所以输出这条向量所代表的答案。

使用之前的答案比较脚本，计算出正确率为百分之24.69，结果并没有达到预期，猜测是词向量如果进行简单的相加相除则会失去本身的语义，而不是代表了更多词的语义，而且忽略的词与词之间的位置关系，所以采用另外一种策略，也就是下一节要介绍的最大维度策略。

### [4.3.3 最大维度法计算语义相似性](file:///E:\2015毕业设计\0毕设\张晓丽版本-04.doc#_Toc327175928)

之前使用的平均向量法表现并不好，因为每一个词的300维的向量，都是使用随机梯度下降法经过反复迭代最终计算得出的，那么也可以说每一个维度都代表了该词在某一方面的特征，这300维的特征在这个向量空间中描绘出了只属于这个词的一条向量，受这种思想启发，本课题又实现了计算最大维度来评判答案与文本之间的语义相似程度。

既然每一维度都代表着该词在某一方面的特征，那么也就是说这个维度的值越大，那么这个词在这方面的特征就越明显，这个词更偏向这个特征，如果某一维度的值很小，那么也就是说这个词在这方面并没有什么表现力，该向量不偏向这个方面。那么也就是可以保存每个词中维数最大的值，比如说两个词累加，那么可以在这300维中的每一维度，比较这两个词在这个维度上面的大小，将大的值赋值到这个向量的这一维度上，这样就放大了单词在维度中的表现力，而最终形成的向量，也是吸取了所有词的表现力所形成的最能表现这些词的特征的向量。

通过这种方法，计算出了四个选项所对应的最大维度向量，也计算出了前4篇文档中的最大维度向量，与上一节一样，通过计算选项向量与文本文档向量之间的欧式距离，得到了结果，通过运行验证脚本计算出准确率为百分之24.85，所以结果也比预期要差一些。

## 4.4 计算语义相似度算法比较

前两节共介绍了四种计算语义相似度的算法，并且都成功实现，实验结果的汇总如表4.1所示。

表4.1 语义相似度结果汇总表

| 算法 | 正确率 |
| --- | --- |
| TF-IDF+普通文本 | 33.48 |
| TF-IDF+关键词文本 | 32.44 |
| Word2vec平均向量法 | 24.69 |
| Word2vec最大维度法 | 24.85 |

由此可见，本课题中TF-IDF算法的表现比较好，结合普通文本可以得到百分之33.48的正确率，预期着结合关键词文本会减少无关词的噪声，但是使用TF-IDF算法并没有预期要好，而且还低了1.04个百分点，分析原因应该是在关键词提取的过程中，虽然删除掉了多数无关词，但也一定误删了许多可以决定答案的词，而误删的词对答案的影响更大一些，所以准确率下降了。使用Word2vec算法表现不好，无论是平均向量法还是最大维度法，都没有得到超过百分之25的正确率，也就是说明Word2vec工具在本课题中无法计算语义相似度，查阅资料分析原因，主要是因为文本中的词太多了，无论是向量累加再除以向量总数还是每一维度都取其最大值来放大特征都会改变向量本身代表的意义，并不像预期的会形成代表文本的一条向量。

Kaggle比赛的榜单上，Cardal代表队得到了第一名，取得了百分之59.30的正确率，在GitHub上也保留了其设计思想和部分代码，研读后发现在其中两部分上做的很好，第一部分是知识库要比本课题丰富的多，他使用了六部分数据组成知识库，分别是allen实验室的数据集、CK-12上的数据、Quizlet学习卡片、Simple维基百科数据、StudyStack语料数据、Wikibooks数据集，丰富的语料库构建出的知识库一定比本课题全面很多，这样就不会出现找不到相关词的情况了。第二部分是特征的提取和模型的构造，他使用的方法是提取特征构建特征模型来计算结果，共提取了27个特征，前20个特征是从知识库中提取的问题答案二元组，后7个特征是使用pyLucene搜索得到的特征，然后使用这些特征训练模型。

排名第四的队伍在Github上给出了全部的处理过程，他使用了Studystack的数据集，这个数据集包括454743个问题答案二元组，并且受到哥伦比亚大学的LSTM深度学习模型的启发，使用了RNN(recurrent neural networks)来获得每一个问题和答案的句子向量，然后与本课题类似使用余弦损失计算问题和答案之间的距离，正确答案与问题的余弦距离一定要比错误答案与问题更近，映射到二维空间如图4.8所示。

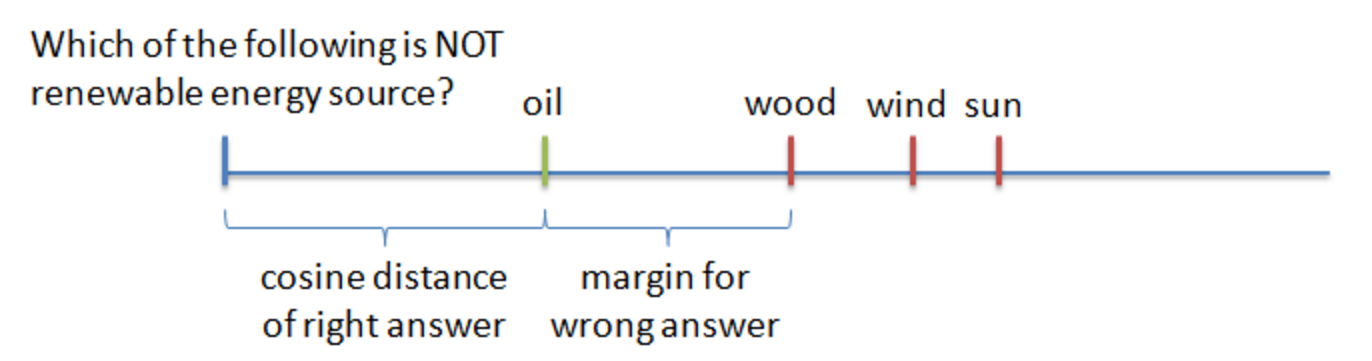


图4.8 余弦距离计算答案

对于这个问题，其中正确答案“oil”距离原点的余弦距离就会比其他三个错误答案要小，事实也是如此，它就是正确的答案。

## 4.5 测试及界面

本课题成功实现了上述所有模块，并使用这四种方法计算语义相似度分别得出了结果，目前TF-IDF算法结合普通文本知识库表现最好，所以将其作为最终数据，送入Django模型，进行界面的编写，交互界面如图4.9所示。

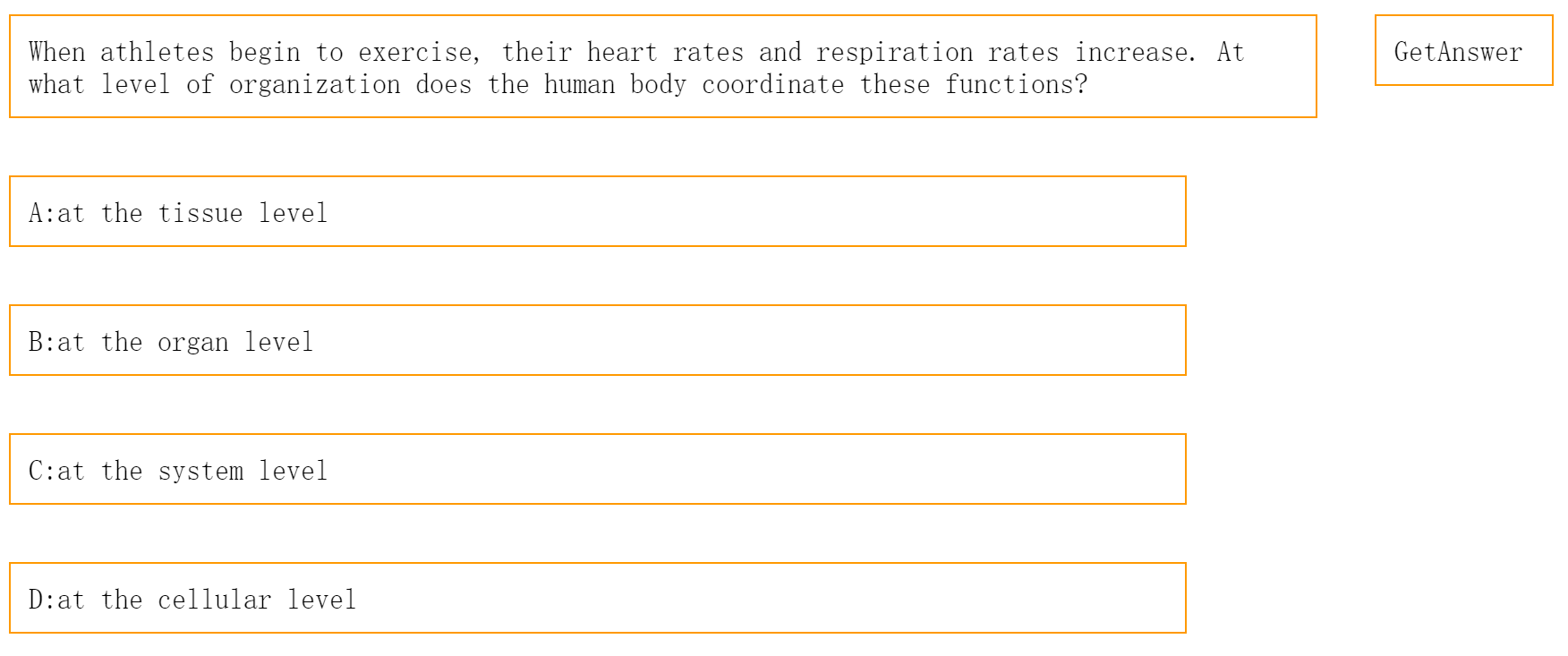


图4.9 问题输入界面图

在第二章已经介绍过，本课题使用python语言的Django Web框架开发页面，由于界面不是本课题的主要部分，所以只进行了简单的设计，如图输入是带求解问题和其四个答案选项，通过Get Answer按钮可以对之前TF-IDF和普通文件结合计算语义相似性的结果进行搜索，搜索到此问题，返回所对应的答案选项，如图4.10所示。

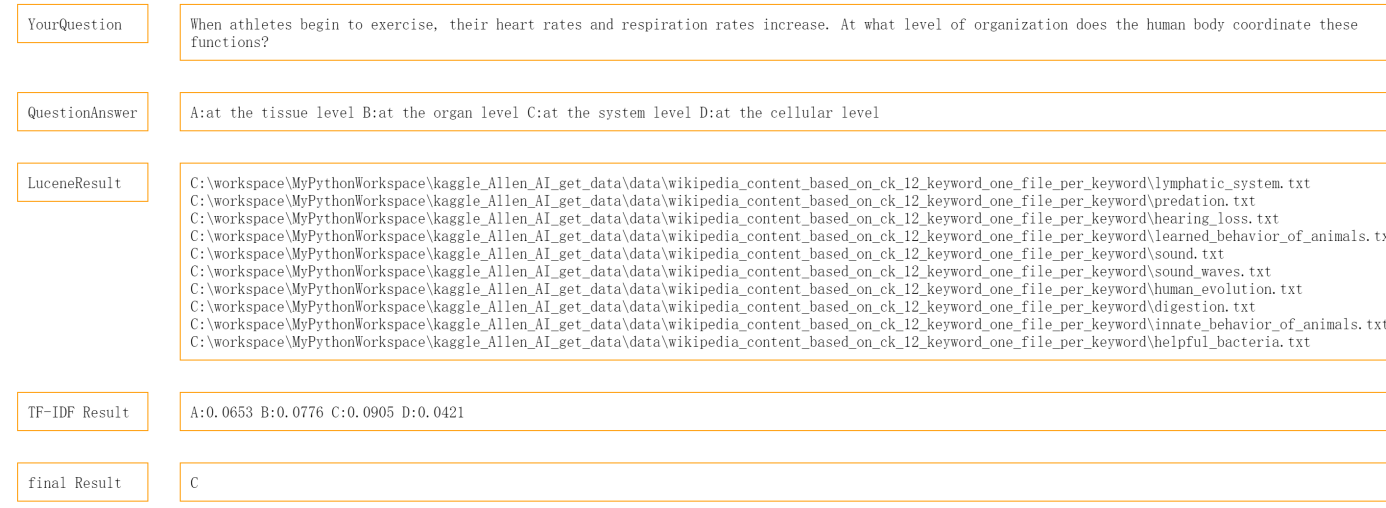


图4.10 结果界面图

## 4.6 本章小结

本章详细介绍了使用TF-IDF算法和Word2vec计算语义相似度过程的设计与实现，其中TF-IDF算法表现良好，但使用普通文本和关键词文本得到的结果差别并不大，但Word2vec工具模型无论是平均向量法还是最大维度法表现都不好，并且对这四种计算语义相似度的方法进行了比较，并且学习了排名前几的队伍的设计思想与模型的构造方法。最后使用了Django进行了页面的开发。

# 结　　论

本课题基于在CK-12网站和维基百科上爬取的美国八年级科学类知识构建了知识库，使用Lucene搭建了本地的搜索引擎，完成了搜索引擎中建立索引和全文检索两大部分，对Lucene返回的结果使用TD-IDF等算法进行语义相似度的计算，为每道选择题得出最匹配的答案。本课题的主要工作成果有以下几点：

(1) 问答系统需要知识储备，需要在语料中抽取出符合本身限定域的知识库，充分了解并学习了知识表示、知识构造、网络爬虫等知识后，使用python的爬虫技术成功爬取到了CK-12网站上与限定域相关的关键词，又使用wikipedia提供的接口爬取到了对应关键词的维基百科正文；

(2) 充分了解学习了搜索引擎的基本架构设计，学习了搜索引擎中创建索引和全文检索的原理和过程。使用Java语言Lucene工具包在本地对知识库搭建了一个搜索引擎，对输入的问题可以检索到知识库中与此问题最相关的前k篇文本文档；

(3) 深入学习了TF-IDF算法和用来做关键词提取的TextRank算法，使用python实现了TF-IDF算法，并且针对本课题做出了改进，使用TextBlob实现了TextRank算法，实现了对知识库的关键词提取工作，并且将TF-IDF算法分别与普通文本和关键词文本进行结合，计算不同知识库的语义相似度；

(4) 深入研究了谷歌在2013年推出的Word2vec工具，学习了其中的数学推理过程，学习了传统使用神经网络来语言模型的弊端和使用Word2vec对传统神经网络语言模型进行改进，使用Hierarchical Softmax模型和Negative Sampling模型分别实现的CBOW模型和Skip-gram模型，并且研读了Word2vec的C语言源码。

根据课题开发中遇到的困难以及限定域问答系统研究的情况，未来研究工作的开展方向可以概括总结为两点如下：

(1) 本课题中知识库的内容和形式过于简单，只使用了一个网站上的相关关键词，而且在对维基百科正文进行爬取的时候还丢失了很多关键词信息，这样对于很多问题就很可能出现Lucene无法搜索出相关文档的情况。所以可以吸取其他参赛者经验，下载问题答案对以及维基百科知识库，再对其中与本课题相关的知识进行提取，形成一个丰富的知识库；

(2) 本课题正确的做法应该是实体连接、知识表示、知识理解、知识推理，在未来的学习中要使用这些自然语言处理的方法重新实现本课题。

# 参考文献

1. 吴静松,汤跃明.人工智能技术在教育游戏中的应用[D].河南:河南师范大学,2008.
2. 韩彤.基于归纳逻辑的人工智能研究[J].重庆科技学院学报(社会科学版),2010,(3):30-32.
3. 孙宏,周明.基于互联网的自动问答答案抽取的研究[D].天津:天津大学,2014.
4. 何艳玲,李岚.元搜索引擎的排序合成技术研究及实现[D].长沙:湖南大学,2014.
5. 成功,李小正,赵全军.一种网络爬虫系统中URL去重方法的研究[J].中国新技术新产品,2014,(12):23-23.
6. 石军伟.基于Lucene的图书垂直搜索引擎探析[J].科学时代, 2012,(5):144-145.
7. 杜吉梁,李炯.中文搜索引擎的关键技术研究[D].成都:电子科技大学,2009.
8. 彭莹.基于Ajax和Lucene的Web应用研究[D].武汉:武汉理工大学,2008.
9. 孟媛,陈洪亮.基于电路课程的主题搜索引擎设计与实现[D].上海:上海交通大学,2010.
10. 邓丹君, 周彩兰. 基于内容相关性和时间分析的改进PageRank算法[J].计算机与数字工程,2011,(1):25-27.
11. 王艳丽,王静婷.基于长尾关键词的搜索引擎优化研究[J].计算机光盘软件与应用,2012,(24):44-45.
12. 刘金桂,李绪蓉.基于网页相似度的Page Rank算法的改进[J].淮阴工学院学报,2006,(1):8-11.
13. 曲超.利用单词超团的二分图文本聚类算法[D].广州:中山大学, 2007.
14. HC Wu,RWP Luk,KF Wong,KL Kwok.Interpreting TF-IDF term weights as making relevance decisions[J]. Acm Transactions on Information Systems,2008,26(3):55-59.
15. 李宁,李涛.基于文本匹配和用户行为分析组合策略的精准广告投放[D].天津:南开大学,2012.
16. 王沛,陈惠鹏.文本倾向性分析中的评价对象抽取与消歧研究[D].哈尔滨:哈尔滨工业大学,2013.
17. Tomas Mikolov,Ilya Sutskever,Kai Chen,Greg Corrado,Jeffrey Dean.Distributed Representations of Words and Phrases and their Compositionality[J].Advances in Neural Information Processing Systems,2013,26:3111-3119.
18. Yoshua Bengio,Rejean Ducharme,Pascal Vincent,Christian Jauvin.A neural probabilistic language model[J].Journal of Machine Learning Research,2003,3(6):1137-1155
19. Tomas Mikolov,Kai Chen,Greg Corrado,Jeffrey Dean.Efficient Estimation of Word Representations in Vector Space[J].Computer Science,2013,arXiv:1301.3781.
20. 周练.Word2vec的工作原理及其应用探究[J].科技情报开发与经济,2015,(2)：145-148.
21. Tomas Mikolov,Quoc V.Le,Ilya Sutskever.Exploiting Similarities among Languages for Machine Translation[J].Eprint Arxiv,2013, arXiv:1309.4168v1.
22. 郭艳艳,吴扬扬.数据空间查询索引研究[D].厦门:华侨大学, 2010.
23. 宋万鹏,顾乃杰.段文本相似度计算在用户交互式问答系统中的应用[D].合肥:中国科学技术大学,2010.
24. 张玉敏,王熙照.基于不同核函数的概率密度函数估计比较研究[D].保定:河北大学,2010.

# 攻读学士学位期间发表的论文和取得的科研成果

# 致　　谢

在论文的最后，首先要感谢的就是本课题的指导教师王燕老师，在本课题的设计和开发以及最后撰写论文阶段，王老师都给予了极大的帮助，在论文最后的修改阶段王老师逐字逐句的对论文进行斟酌，提出宝贵的修改意见。在开题答辩和中期答辩中王老师也为本课题提出了许多宝贵意见，使本文的结构更加严谨，更有说服力。

其次要感谢我在哈尔滨工业大学的导师孙承杰老师，是孙老师为我选题并且提供了技术的支持。实验室的刘洋师兄、成昊师兄、周博通师兄也都为我解答了很多疑惑，你们也是我能顺利完成毕设的保证。

同样要感谢本科期间我的所有任课教师及给予过我帮助的教师们，是你们一步步带领我走进计算机知识的海洋，让我学会编程，让我爱上编程。是你们四年日积月累的努力才使我有今天的成就。在论文修改阶段，李建利老师的论文格式讲座让我受益匪浅，使我顺利的完成的论文的撰写。

最重要的是感谢哈尔滨工程大学计算机学院、软件学院、国家保密学院。是学院给予了我强大的硬件设施，让我顺利完成学业。

最后要感谢我的父母，感谢你们对我的养育之恩，你们永远是我最坚实的后盾。