算法设计文档

项目名：华为云智能问答系统

版本：1.0

编订：王飞鸿

日期：2018-07-20

部署地址：http://58.87.125.79:5000/chat

1.引言

1.1 编写目的

本文档的目的是详细地介绍华为云智能问答系统的算法及设计思路，对本系统的各模块、程序、子系统进行了实现层面上的要求和说明。使系统开发人员清楚认识本系统所需要实现的功能以及功能模块的划分、数据库的表结构、设计思想。

1.2 背景

本文档介绍的产品是华为云智能问答系统，该软件面向所有需要使用华为云产品的人群。该软件基于华为云网站页面作为原始数据由都柏林科技有限公司团队开发。主要方便用户咨询华为云产品的使用，并节省了客服人力。

1.3 软件功能

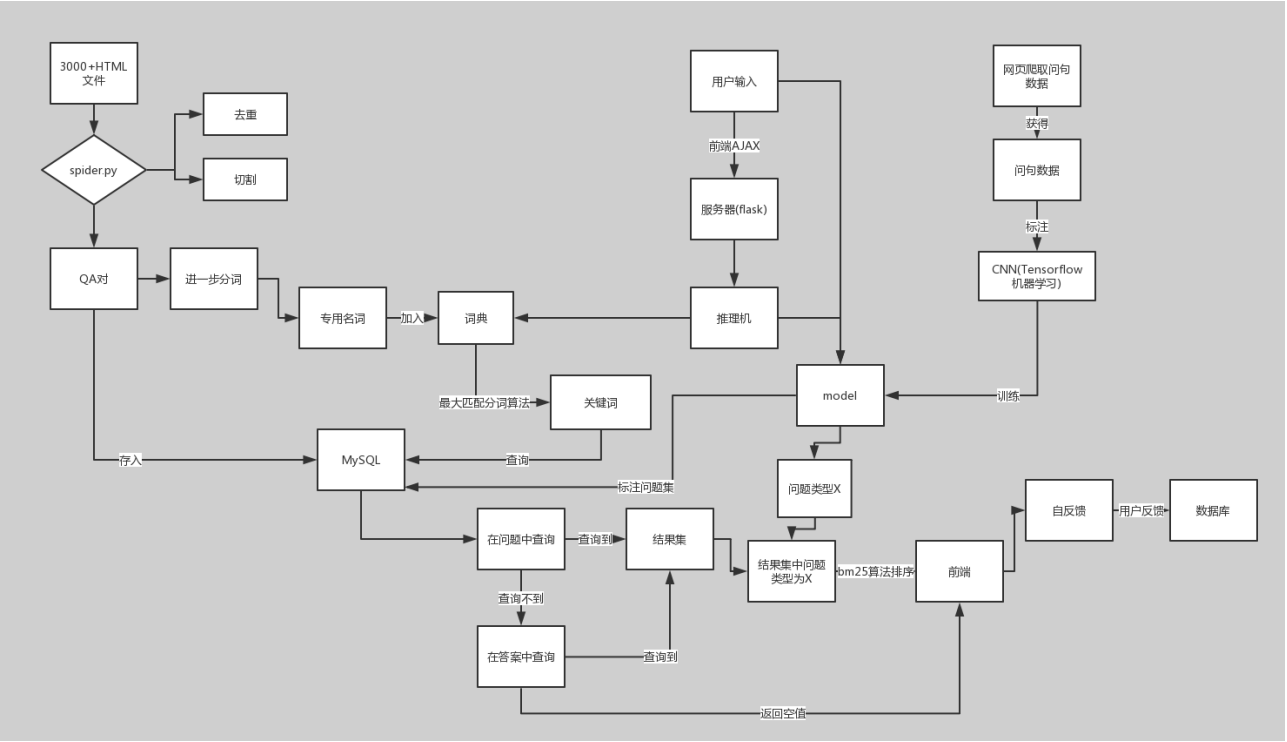
后台实现产品文档(HTML 文件)录入、知识库构建根据录入文档实现“QA 对”自动化生成，知识库实现基本 QA 对删除、增加、查询等操作功能。

前台实现 QA 对话界面，该界面可以基于用户提问，自动连接后台、并从知识库寻找答案，并呈现给用户，前台问题可以是由主题、关键词、短语构成。针对不同的问法的同种问题，可自动识别。

后台搜索引擎可根据问题的模糊程度返回多条答案或一条精确答案。在每条答案的基础

上，增加自反馈功能，增强用户体验。

2.系统结构



本文的系统设计结构如图所示。

3. 算法设计

本系统（后简称 WEBQA 系统）具有以下几个特征：

（1）自动根据数据集生成知识库，通过文档结构树和自行设计的爬虫框架的方法提取 QA 对。无需从大量篇幅中做语句分析或是实体识别，从而提高系统效率和准确度。

（2）自动收录专业名词训练词库。对用户提出的问题，更准确地提取关键信息，从而使命中问答对数增高。

（3）利用 word2vec 模型处理中文语言特色引起的问题。

（4）可以回答同种问题的不同问法的问句，并准确返回答案。

（5）通过与用户的交互实现知识库自我更新，使其提供用户友好性。

3.1 系统模型

构建 WEBQA 系统模型，主要包括两个部分，一是自动生成知识库，建立问题与答案的

关系；二是创建推理机，知识推理需要知识库所存储的知识作为基础，不同的知识表达方式在一定程度上决定了特定的知识运用方式。

WEBQA 系统的基本设计思路分两个步骤，即知识库构建和推理机：

知识库构建是自动生成的,本通过自设计框架、借助标签从网页中提取 QA 对，并通过文本结构树的思想整合答案的类型。

推理机处理模块通过问题分析、问题分类取得用户的期望问题类型和关键词。用 word2vec 模型进行相似词辅助搜索,已解决中文语言特色带来的问题 。在此基础上，用关键词对知识库中总结的概述词和答案进行文本相似度计算，得到目标答案。

3.2 自动生成知识库

目前，自动生成知识库已作为实现实用的问答系统的一个基本组成部分。WEBQA 系统针对网页形式的数据集自动生成知识库，主要通过网页的标签进行答案的提取，并采用创新的文本结构树算法对答案和问题进行分割，如图 1 所示。

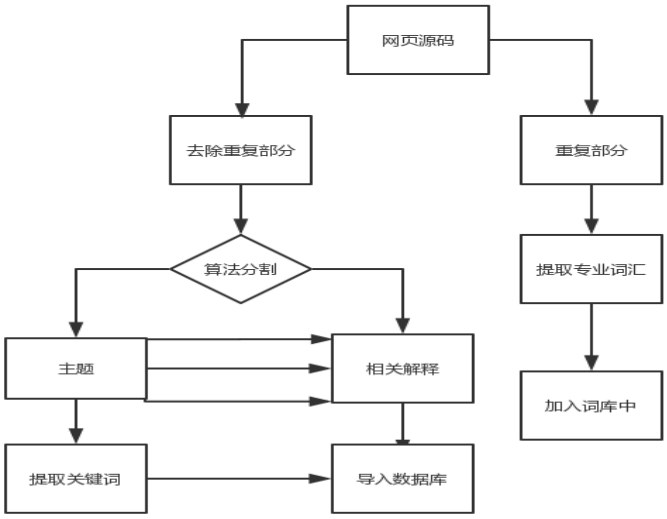


图 1 知识库自动生成流程

WEBQA 系统基于网站生成知识库，利用网页中已有的标签、结构进行 QA 对提取。以

华为云网站为例（图 2）。

图 2 华为云网站页面样例

每个网页都存在这种索引的代码，知识库的自动生成部分可以从中提取专业名词加入词库中。原词库由 jieba（Python 语言中作中文 nlp 分析的库）提供，将提取出的专有名词以中等频率写词库。专有名词能够在每句话中被分出，也可在 TF-IDF 算法中被识别为关键词。

词库应尽可能贴合知识库的方向。比如“云服务器是什么？”，用普通词库分出的关键词是“云”和“服务器”，而想要的是“云服务器”这整个词，分词错误会影响后来的一系列操作，所以应采用相关的词库，将描述的词加进词库。把词频设为中等频率，因为过大的词频会影响推理机部分 TF-IDF 算法提取关键词，过小的词频又不能被分出。

尽管网页比纯文本的结构层次分明，但是网页结构缺乏一定规律和统一性。

本系统的爬虫放弃对网页标签结构的研究，即把一个网页当成纯文本形式，将所有 h 标签作为问题。h 标签和 h 标签中间必然存在解释前者的内容，将其分割后，一一对应提取出来，作为常用答案库对加入数据库.该框架可用性高，爬取速度快，并可自行处理去重，乱码等问题。比基于 NLP 或规则爬取的方法准确而且不会遗失 QA 对。这种爬取方法是能爬取到最多 QA 对的方式。最终总计爬取 QA 对 8144 条（文件中的“知识库.xlsx”）。

为有效管理知识库和,我运用文档结构树的知识，将每条 QA 对一步步细分，以达到方便管理和查询的目的。如“帮助中心>机器学习服务>最佳实践>附录>修订记录”，这是一条 QA 对中问题的格式。是每层一步步细化下来的。在此基础上我设计数据库表用于存储 QA 对。在表设计上我考虑到了效率，将用于搜索引擎中的分词，去除停用词等文本预处理在生成知识库步骤中完成。在后期就不用再次操作了。答案是连同 html 的 tag 标签一起爬取下来的，

所以在前端返回答案时，可以看到表格，列表等网页样式的答案而不是单一的文字。将知识库存入数据库也满足了赛题中“实现 QA 对的增删查改”的要求。

数据库表格成分为：

- 标准问题：通过规则组合关键词生成（基于文档结构树技术）。
- 答案：带网页标签的答案
- url：数据来自的网页
- 描述词：此数据的描述，由网页中提取出（如 3.1 图）。
- 问题词组：通过分词（全模式分词算法）和去除停用词等步骤得到的词组。用于计算文本相似度。
- 答案词组：通过分词（全模式分词算法）和去除停用词等步骤得到的词组。用于计算

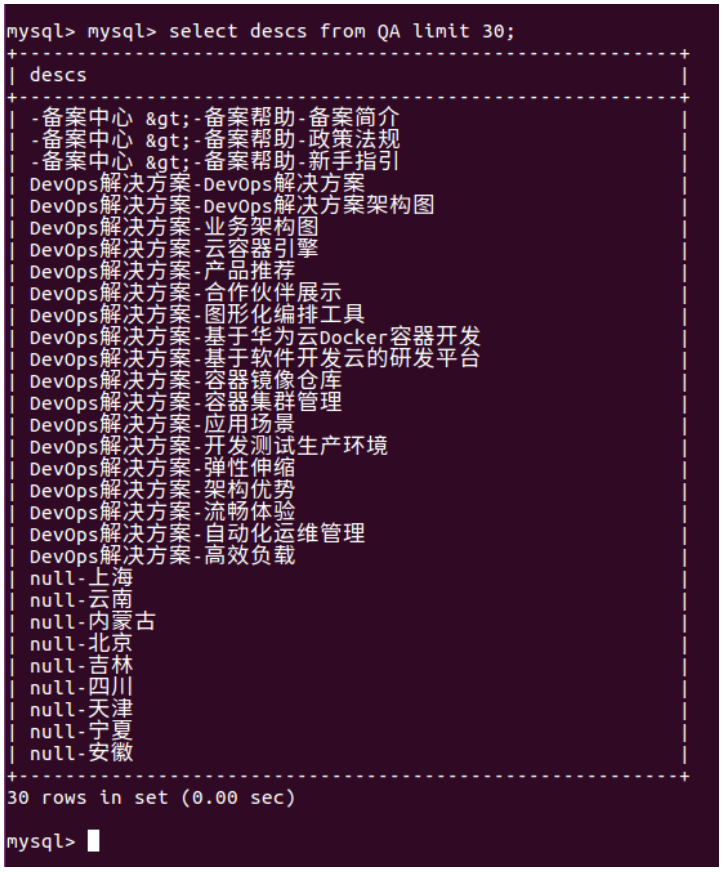


图 3.1 数据库描述词样例

文本相似度。

```
mysql> desc QA;
+------------------+---------------+------+-----+---------+-------+
| Field            | Type          | Null | Key | Default | Extra |
+------------------+---------------+------+-----+---------+-------+
| normal_question  | varchar(255)  | NO   | PRI | NULL    |       |
| type             | varchar(255)  | YES  |     | NULL    |       |
| url              | varchar(1000) | YES  |     | NULL    |       |
| answer           | mediumtext    | YES  |     | NULL    |       |
| descs_words      | mediumtext    | YES  |     | NULL    |       |
| answer_words     | mediumtext    | YES  |     | NULL    |       |
+------------------+---------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

图 3.2　数据库表信息

图 3.3　单条数据示例

在创建数据过程中，对爬取到的答案和问题做预处理为推理机部分做准备，进行去除网页标签，分词，去除停用词等操作。其中分词算法较初赛改良为全模式分词。这种分词的特点是把一句话中能分出的词全部分出，优点可以体现在：确保知识库中的词可以匹配上推理机中的词。例如“云服务器”和“服务器”这样的词在相似度算法中并没有匹配上，但实际上这表达了同一种意思。最后将这些分好的词写入每条数据库的 descs_words 和

answer_words 的属性中并作为推理机中用于匹配答案的信息。提前将答案预处理,可以节省后期在加载数据时浪费的时间和资源.

3.3 推理机处理模块

3.3.1 构建推理机

WEBQA 系统的答案提取由常用知识库的简单答案匹配和文本相似度匹配构成。

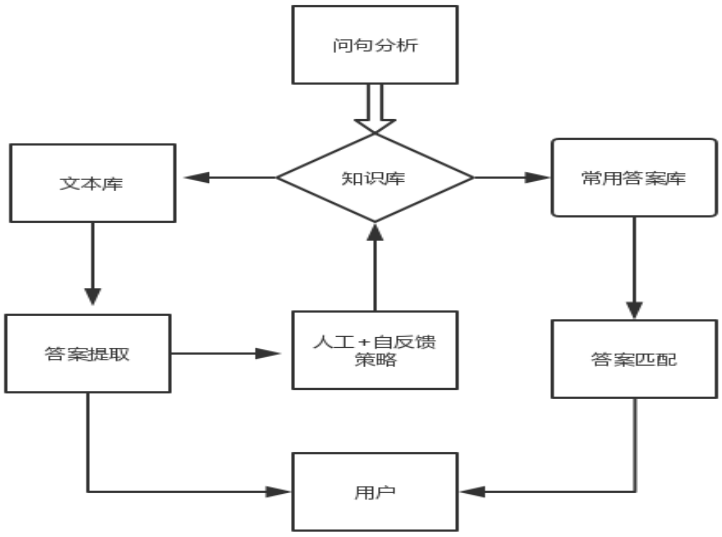


图 4　推理机构建流程图

常用知识库构成包括：文本概述和答案构成。文本概述是通过爬取网页建立的概述词，在一定层面上想比文本相似度更精确，更人性化。因为它本身就源自于人工的总结，所以在答案提取优先使用常用知识库，其实际查询规则是基于文本相似度。

在推理机构建中着重创新解决了以下问题：

判断网页内容是否与用户查询相关，这依赖于搜索引擎所采用的检索模型。检索模型是搜索引擎的理论基础，是对查询词和文档之间进行相似度计算的框架和方法。如图 5 所示，

检索模型所在搜索引擎系统架构位置。

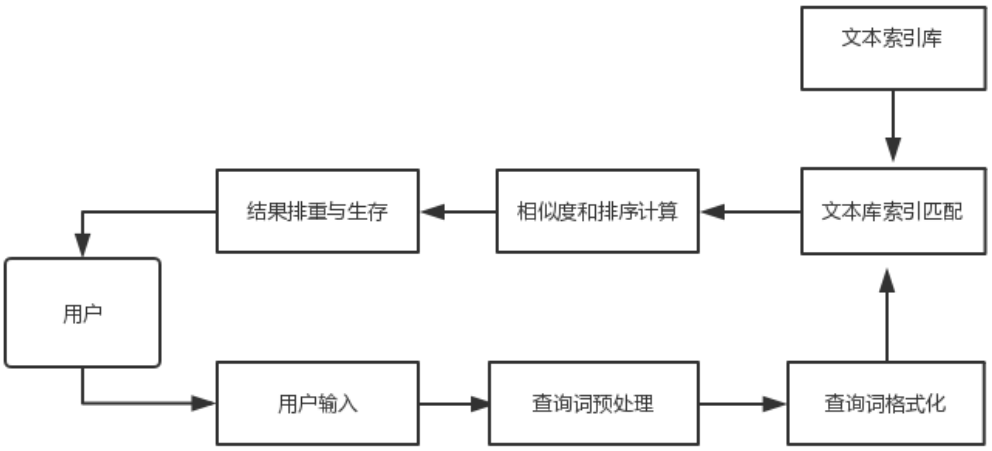


图 5 文本相似度的检索模型运行流程

本系统使用 BM25 模型用作计算文本相似度，使用改良的 BM25 算法对答案问题同时进行打分。通过分析数据，还提高了的文本长度影响分数的权重，因为概括性强的问题往往在比较短的答案中出现。

摒弃基于 TF-IDF 算法提取用户的问题中的关键词。这里我对数据进行分析，发现不适合用 TF-IDF 算法，因为每个词汇都十分重要，可替代性不强，所以删除了原 BM25 的 TF-IDF 部分。TF-IDF 算法的主要思想就是：如果某个词在一篇文档中出现的频率高，并且在语料库中其他文档中很少出现，则认为这个词代表问句的主题，即关键词。由于 WEBQA 词库中已有大量相关网站领域性的词，所以如果用这种方法去提取关键词会丢弃重要的词（同时出现频率也很高的词）。如“云专线”出现的频率很高。但是如果从关键词中排除它就很难定位答案。在用用户输入的关键词匹配答案时，采用同时在答案和问题打分的方式，同时赋予问题更高的权重，最终将两部分打分相加再排序返回前 3 条高分答案。因为答案和问题都有可能包含用户想要的到的信息。而赋予问题更高的权重，是因为问题是人工总结出来的，其概括性更强，更能代表这个 QA 对的主题。本系统还添加了自反馈系统，在返回问答系统回答的问题时，会产生与用户的交互，用户可以根据返回的答案满意与否，点击答案下的按钮，如满意，系统会自动将用户输入的关键词加入到那条数据的 descs_words 属性中。下次搜索时，更新过的关键词就会影响到得分。从人的角度增加了知识库的可用性。

通过分析中文语言特色，发现用户会使用相近的意思的词语代替知识库中的词搜索，因此做了词向量模型的训练，帮助解决这样的问题。词向量模型用于计算关键词的相似度，帮助比较每种近义词搜索的得分。但需要注意的是，专有名词并不能用近义词去替代搜索，因

为其可替代性很低，如果选择近义词，就不能代表原来意思。如“服务器”和“电脑”会被词向量模型标记为近义词。而在专业领域，这两个词词义差异很大。

本系统还尝试用CNN算法训练分类器模型，也是为了增加准确度。最终整合达到最高的准确度。在问句分析中运用问题分类。经统计，用户在提出问题时，习惯性用口语形式，问句形式不规范，进而无法准确分析作者的期望答案类型。因此，在知识库中，对一类问题用规范统一的问法保存。在推理机中，通过机器学习训练出的分类模型对用户输入的问句进行分类，用问句的类别去匹配数据库中这类别的答案，以更好的提供回答。用于训练问题分类模型的数据来自网络，通过爬虫技术获取，并自行标注，利用CNN算法训练。如表1所示，

用户输入“云服务器多少钱”转换为知识库中更为专业性的问句“云服务器的价格”。

专业性问句	口语性问句 1	口语性问句 2	问句分类
云服务器价格	云服务器多少钱？	云服务器多贵？	价格
云服务器功能	云服务器是什么？	云服务器有什么用？	功能

表 1　问句分类样例

4．测试

测试使用每个网页中的“引导标签”，通过基于规则的方式生成测试数据，首先筛除部分对问题生成无用的标签，如：“帮助中心”、“概览”、“产品简介”、“快速入门”等等，接着通过一些规则自动生成问题，规则包括：

1. “引导标签”的最后一个标签以“什么”、“怎么”或“如何”开头，则不进行语法更改，直接使用最后一个标签作为生成的问题，如：

帮助中心>云服务器备份>产品介绍>什么是云服务器备份

->

“什么是云服务器备份？”

2. 最后一个“引导标签”以动词开头且这个动词不做形容词用，用“怎么”加最后一个标签生成问题，如：

帮助中心>弹性云服务器>快速入门>注册公有云

->

“怎么注册公有云？”

3. 最后一个“引导标签”以动词做形容词开头（这些词组包括”使用限制”,”计费方式”,”操作指南”,”开发指南”），则以主标签加“的”加最后一个标签加“有哪些”生成问题，如：

帮助中心 > MapReduce 服务 > 开发指南

->

“MapReduce 服务的开发指南有哪些？”

测试使用的规则不止这 3 条，但是这 3 条比较独特或常见，测试选取了随机 1000 个网页并生成了问题，这些问题在 作者的 Github 上可以看到。测试从中抽取了 100 条不重复的合理问题输入机器进行了黑盒测试，并人工判断输出的合理性。结果比较令人满意，100 个问题中超过 90 个能给出合理的答案。