**作者姓名 李斌**

**指导教师姓名、职称 黄健斌 教授**

**企业导师姓名、职称 王文磊 高工**

**申请学位类别 工程硕士**

**智能问诊系统的设计与实现**

**学　号** 1510122545

**密　级　 公开**

**作者姓名：**李斌

**领　　域：**软件工程

**学位类别：**工程硕士

**指导教师姓名、职称：**黄健斌 教授

**企业导师姓名、职称：**王文磊 高工

**学　　院：** 软件学院

**提交日期：**2018年6月

**学校代码 10701**

**分类号 TP311.5**

**智能问诊系统的设计与实现**

**西安电子科技大学**

**硕士学位论文**

By

Li Bin

June 2018

Supervisor: Huang Jianbin

Supervisor: Wang Wenlei

Title: Professor

Title: Senior Engineer

**Design and Implementation of Intelligent Inquiry System.**

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Software Engineering

**西安电子科技大学**

**学位论文独创性（或创新性）声明**

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名： 日 期：

**西安电子科技大学**

**关于论文使用授权的说明**

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在 年解密后适用本授权书。

本人签名： 导师签名：

日 期： 日 期：

摘要

当前已经迈进了医疗问诊的大数据时代，需要对这些问诊大数据进行价值挖掘。而问诊大数据的价值挖掘，会使用一些智能化的手段，解决医疗问诊行业面临的一些问题。比如随着在线咨询人数的快速增加，导致医生不能及时回复，患者在线问诊体验下降。另外，如何从问诊数据中挖掘有用的知识，也是目前急需解决的问题。同时，病患通常是通过医生简介来选择挂号的医生，有时候并不能选择到合适的医生，从而导致就诊效率低下。

在对上述问题的初步探索中，本文充分利用病患通过线上的方式向医生提出问题以及医生对相应问题的回答的问诊记录，实现通过病症名称或者患者病痛部位的描述对现有的所有问诊记录进行精准匹配和查询，并匹配到相应问题的的详细问诊记录；利用现有医护人员的总结的相关症状描述以及相应解决方案的知识卡，快速匹配相关病症的解决方案和利用深度学习网络实现通过对症状的描述或者相关医生的输入实现问诊医生的推荐等功能的基础上，构建了智能问诊系统，通过对该系统的应用场景的了解和分析，本文设计和实现了问诊记录详情的搜索功能，知识卡检索功能以及医生推荐系统。论文在阐述了ElasticSearch搜索引擎、推荐系统、SOA架构等相关技术和理论的背景上，介绍了智能问诊系统相应的业务需求并进行了建模，完成了需求分析。在实现智能系统的过程中，首先进行了整体实现技术的架构和数据库的的设计，然后按照SOA架构的服务划分分别对数据预处理服务、模型训练服务、RPC服务和ElasticSearch服务以及Web服务等基础与应用服务进行了设计和实现，其中在数据预处理阶段将数据库中存储的问诊记录提取整合得到相应的用户医生表和医生问诊表，在推荐系统的模型训练服务中采用Word2Vec算法中的Skip-Gram模型，作为实现医生推荐系统的理论基础，在ElasticSearch服务中设计实现搜索索引的同步构建，在RPC服务实现RPC客户端中调用服务以及在RPC服务端注册服务，在Web服务中重构框架的组织架构利用RESTful协议调用其他各个模块的服务，在上述基础与应用服务实现的基础上对实际的业务进行划分并分别实现了问诊记录搜索，知识卡检索与智能医生推荐的API设计和具体功能实现，以及测试用例编写。

通过基于问诊数据的智能问诊系统的整体设计和实现，以及对各个服务模块的功能测试和性能测试，表明了该系统可以通过系统用户的输入病症的描述和病症的名称可以正常检索该病症的问诊详情以及该病症的相关知识卡，而且通过病症的描述和医生姓名的输入可以获取相关问诊医生的推荐，同时系统的性能指标也达到了需求要求。

**关键词**：医生推荐，ElasticSearch，SOA，Word2Vec，推荐系统，Skip-Gram

ABSTRACT

At present, the society has entered the era of big data for medical inquiry. It is very necessary to excavate the value of inquiring big data. Some intelligent means will be used in the inquiry of big data value mining to solve some problems which the medical inquiry industry is facing. For example, due to the rapid increase of online consultation, the doctor failed to respond promptly, and finally the patient's online inquiry experience declined. In addition, the urgent problem is to mine useful knowledge from interrogation data. At the same time, the introduction of doctors is the most common method for patients to choose registered doctors. Sometimes it is difficult for a patient to choose the right doctor, which leads to low efficiency.

In the study of the questions mentioned, this article collects and deals with patient's online enquiry records of questions and corresponding answers from doctors and realizes the accurate matching and inquiry of all the existing medical records through the description of the disease or the description of the patient's pain to find the detailed inquiry records of the corresponding problems. Based on the description of the related symptoms summarized by the existing medical staff, the knowledge cards of the corresponding solutions, fast matching solutions for related diseases and the use of a deep learning network to implement the functions of doctor's recommendation by describing the symptoms or using the input of the relevant doctor, this paper constructs an intelligent inquiry system. By understanding and analyzing the application scene of this system, this paper preliminarily designs and completes the search function of the details of the medical records, the retrieval function of knowledge card and the doctor recommendation system. After explaining the background of ElasticSearch search engine, recommendation system, SOA architecture and other related technologies and theories, the corresponding business requirements of the intelligent inquiry system were introduced, and the modeling was carried out, and the requirement analysis was completed. In the process of the preliminary completion of the intelligent system, the whole implementation technology architecture and the database are designed, and then the basis of data preprocessing service, model training service, RPC service and ElasticSearch service, and Web service are designed and implemented according to the service division of SOA architecture. In the data preprocessing stage, extract the query records that are stored in the database to obtain the corresponding user doctor tables and doctor's inquiry tables.In the corresponding model training service of the recommended system, the Skip-Gram model is used as the theoretical basis for the implementation of the doctor recommendation system because of the Word2Vec algorithm. The synchronous construction of the search index has been designed and implemented in ElasticSearch services. In the RPC service, it implements the calling service in the RPC client and the registration service on the RPC server. In Web services, the framework of the refactoring framework is used to invoke the services of other modules by the RESTful protocol. In the case of the implementation of the above basic and application services, the corresponding business is integrated and the API design and specific function implementation recommended by the knowledge card retrieval and the intelligent doctor are implemented respectively, and the test cases are written.

Because the design and completion of the intelligent inquiry system are based on the inquiry data, and the functional tests and performance tests have been completed, the system can normally retrieve the details of the diagnosis and related knowledge cards of the disease through the description of the symptoms of the input disease of the system user and the name of the disease. By describing the symptoms and the name of the doctor, the doctor's recommendation can also be easily obtained, and the performance of the system also meets the requirements.

**Keywords**: Doctor Recommended, ElasiticSearch, SOA, Word2Vec, Recommendtion System, Skip-Gram

插图索引

[图2.1 ElasticSearch的分布式调用架构 6](#_Toc516932083)

[图2.2 SOA架构图 8](#_Toc516932084)

[图3.1 智能问诊业务框架图 11](#_Toc516932085)

[图3.2 问诊记录搜索业务活动图 12](#_Toc516932086)

[图3.3 知识卡检索活动图 13](#_Toc516932087)

[图3.4 智能医生推荐业务活动图 14](#_Toc516932088)

[图3.5 智能问诊系统整体需求用例图 15](#_Toc516932089)

[图3.6 问诊记录搜索需求用例图 16](#_Toc516932090)

[图3.7 知识卡检索需求用例图 17](#_Toc516932091)

[图3.8 智能医生推荐系统需求用例图 19](#_Toc516932092)

[图4.1 智能问诊系统SOA架构 21](#_Toc516932093)

[图4.2 智能问诊系统E-R图 23](#_Toc516932094)

[图4.3 智能问诊系统UML数据库表示 23](#_Toc516932095)

[图4.4 智能问诊系统整体框架 24](#_Toc516932096)

[图4.5 数据预处理模块类图 26](#_Toc516932097)

[图4.6 数据预处理模块时序图 27](#_Toc516932098)

[图4.7 基于Skip-Gram的医生问诊推荐模型 29](#_Toc516932099)

[图4.8 基于Skip-Gram用户医生推荐模型 30](#_Toc516932100)

[图4.9 模型训练模块的类图 30](#_Toc516932101)

[图4.10 模型训练模型时序图 31](#_Toc516932102)

[图4.11 JSON-RPC服务模块服务端的类图 33](#_Toc516932103)

[图4.12 RPC客户端类图 33](#_Toc516932104)

[图4.13 RPC服务时序图 34](#_Toc516932105)

[图4.14 构建ElaticSearch的索引的活动图 35](#_Toc516932106)

[图4.15 WebServer的实现类图 37](#_Toc516932107)

[图4.16 问诊记录搜索类图 39](#_Toc516932108)

[图4.17 问诊记录搜索时序图 39](#_Toc516932109)

[图4.18 知识卡检索类图 41](#_Toc516932110)

[图4.19 知识卡检索时序图 41](#_Toc516932111)

[图4.20 智能医生推荐类图 43](#_Toc516932112)

[图4.21 智能医生推荐系统时序图 43](#_Toc516932113)

[图5.1 测试环境拓扑图 45](#_Toc516932114)

[图5.2 新的问诊记录表 49](#_Toc516932115)

[图5.3 用户医生csv表 49](#_Toc516932116)

[图5.4 医生问诊csv表 49](#_Toc516932117)

[图5.5 医生问诊分词表 50](#_Toc516932118)

[图5.6 用户医生分词表 51](#_Toc516932119)

[图5.7 问诊记录测试结果界面 52](#_Toc516932120)

[图5.8 问诊记录详情展示界面 52](#_Toc516932121)

[图5.9 知识卡检索页面展示 53](#_Toc516932122)

[图5.10 输入症状的医生推荐 54](#_Toc516932123)

[图5.11 输入医生姓名的医生推荐 55](#_Toc516932124)

[图5.12 压力测试中的各个服务器CPU使用率 56](#_Toc516932125)

[图5.13 压力测试中的平均响应时间对比 57](#_Toc516932126)

表格索引

[表3.1 问诊记录搜索用例规格说明 16](#_Toc516931950)

[表3.2 知识卡检索用例规格说明 17](#_Toc516931951)

[表3.3 智能医生推荐系统用例规格说明 18](#_Toc516931952)

[表4.1 合并后问诊记录字段表 26](#_Toc516931953)

[表4.2 Web服务和功能服务 36](#_Toc516931954)

[表4.3 Server目录和功能服务 37](#_Toc516931955)

[表4.4 问诊记录搜索路由表 38](#_Toc516931956)

[表4.5 知识卡检索路由表 40](#_Toc516931957)

[表4.6 智能医生推荐路由表 42](#_Toc516931958)

[表5.1 Web服务的ECS服务器硬件与软件配置 46](#_Toc516931959)

[表5.2 Web服务器的ECS软件配置 46](#_Toc516931960)

[表5.3 RPC服务的ECS服务器硬件与软件配置 46](#_Toc516931961)

[表5.4 RPC服务器的ECS软件配置 46](#_Toc516931962)

[表5.5 ElasticSearch服务的ECS服务器硬件与软件配置 47](#_Toc516931963)

[表5.6 ElasticSearch服务器的ECS软件配置 47](#_Toc516931964)

[表5.7 数据预处理管理功能测试用例表 48](#_Toc516931965)

[表5.8 模型训练服务测试用例表 50](#_Toc516931966)

[表5.9 问诊记录功能测试用例表 51](#_Toc516931967)

[表5.10 知识卡功能测试用例表 53](#_Toc516931968)

[表5.11 智能医生推荐功能测试用例表 54](#_Toc516931969)

[表5.12 性能测试用例表 55](#_Toc516931970)

符号对照表

|  |  |
| --- | --- |
| 符号 | 符号名称 |
| Gbps  GB  GHz  IOPS | 十亿位每秒  十亿比特  十亿赫兹  每秒进行读写的次数 |

缩略语对照表

缩略语 英文全称 中文对照

API Applicatoin Interface 应用程序接口

ES ElasticSearch 弹性搜索引擎

RPC Remote [Procedure](https://baike.sogou.com/v303132.htm) Call Protocol 远程过程调用

RESTful Representational State Transfer 状态表征转移

SOA Service-Oriented Architecture 面向服务的架构

目录

[摘要 I](#_Toc516931971)

[ABSTRACT III](#_Toc516931972)

[插图索引 V](#_Toc516931973)

[表格索引 VII](#_Toc516931974)

[符号对照表 IX](#_Toc516931975)

[缩略语对照表 XI](#_Toc516931976)

[第一章 绪论 1](#_Toc516931977)

[1.1 选题背景及意义 1](#_Toc516931978)

[1.2 国内外现状分析 2](#_Toc516931979)

[1.3 论文工作内容 3](#_Toc516931980)

[1.4 论文组织结构 3](#_Toc516931981)

[第二章 相关技术概述 5](#_Toc516931982)

[2.1 ElasticSearch搜索引擎 5](#_Toc516931983)

[2.1.1 ElasticSearch技术介绍 5](#_Toc516931984)

[2.1.2 ElasticSearch工作原理介绍 5](#_Toc516931985)

[2.2 推荐系统技术 6](#_Toc516931986)

[2.2.1 推荐系统技术概述 6](#_Toc516931987)

[2.2.2 基于Word2Vec算法的推荐系统 6](#_Toc516931988)

[2.3 SOA架构 7](#_Toc516931989)

[2.3.1 SOA架构概述 7](#_Toc516931990)

[2.3.2 SOA架构面临的主要的问题 8](#_Toc516931991)

[2.4 本章小结 9](#_Toc516931992)

[第三章 系统需求分析 11](#_Toc516931993)

[3.1 系统业务分析 11](#_Toc516931994)

[3.1.1 问诊记录搜索业务分析 12](#_Toc516931995)

[3.1.2 知识卡检索业务分析 12](#_Toc516931996)

[3.1.3 智能医生推荐服务业务分析 13](#_Toc516931997)

[3.2 系统需求建模 14](#_Toc516931998)

[3.2.1 问诊记录搜索服务的需求 15](#_Toc516931999)

[3.2.2 知识卡检索服务的需求 16](#_Toc516932000)

[3.2.3 智能医生推荐系统的需求 18](#_Toc516932001)

[3.3 智能问诊系统性能需求 19](#_Toc516932002)

[3.4 本章小结 19](#_Toc516932003)

[第四章 系统设计与实现 21](#_Toc516932004)

[4.1 系统的SOA架构设计 21](#_Toc516932005)

[4.2 系统的数据库设计 22](#_Toc516932006)

[4.3 系统整体框架和实现技术 24](#_Toc516932007)

[4.4 基础与应用服务设计与实现 25](#_Toc516932008)

[4.4.1 数据预处理服务的设计与实现 25](#_Toc516932009)

[4.4.2 模型训练服务的设计与实现 28](#_Toc516932010)

[4.4.3 RPC(Remote Procedure Call Protocol)服务的设计与实现 32](#_Toc516932011)

[4.4.4 ElasticSearch搜索服务的设计与实现 34](#_Toc516932012)

[4.4.5 Web服务的设计与实现 36](#_Toc516932013)

[4.5 业务服务的设计与实现 38](#_Toc516932014)

[4.5.1 问诊记录搜素 38](#_Toc516932015)

[4.5.2 知识卡检索 40](#_Toc516932016)

[4.5.3 智能医生推荐 42](#_Toc516932017)

[4.6 本章小结 44](#_Toc516932018)

[第五章 系统测试及分析 45](#_Toc516932019)

[5.1 系统测试环境的配置与搭建 45](#_Toc516932020)

[5.1.1 系统测试环境配置 45](#_Toc516932021)

[5.1.2 系统测试环境搭建 47](#_Toc516932022)

[5.2 系统功能测试与分析 47](#_Toc516932023)

[5.2.1 数据预处理服务功能测试与分析 48](#_Toc516932024)

[5.2.2 模块训练测试与分析 49](#_Toc516932025)

[5.2.3 问诊记录搜索功能测试与分析 51](#_Toc516932026)

[5.2.4 知识卡检索功能测试与分析 52](#_Toc516932027)

[5.2.5 智能医生推荐功能测试与分析 53](#_Toc516932028)

[5.3 系统性能测试与分析 55](#_Toc516932029)

[5.3.1 性能测试用例设计 55](#_Toc516932030)

[5.3.2 性能测试中CPU使用率 56](#_Toc516932031)

[5.3.3 性能测试中平均响应时间 56](#_Toc516932032)

[5.4 本章小结 57](#_Toc516932033)

[第六章 结论和展望 59](#_Toc516932034)

[6.1 论文工作总结 59](#_Toc516932035)

[6.2 后续工作展望 59](#_Toc516932036)

[参考文献 61](#_Toc516932037)

[致谢 63](#_Toc516932038)

[作者简介 65](#_Toc516932039)

# 绪论

## 选题背景及意义

随着“互联网+医疗”[1]模式的公司出现，传统的医疗服务行业进入了互联网时代，在最近国内出现了很多新型问诊类型的互联网医疗的创业公司[2]。春雨医生，平安好医生，丁香园等创业公司陆续推出在线问诊服务。它们给患者提供了更加方便、快捷的获取医疗咨询服务的途径，病人看病不用想过去一样只能到实体医院排队挂号，有了在线问诊平台，互联网医疗用户可以足不出户通过手机给医生发送图文消息，让医生了解自己的病情，并提供在线问诊。

新兴的在线问诊服务能够方便、快捷的为患者提供高效优质的医疗服务。然而，随着在线问诊服务在给患者提供便利的同时也积累了海量的数据，而如何有效的利用这些海量数据，为互联网医疗用户的提供优质服务，并且辅助医生做出合理的诊断决策就变成了一个亟待解决的问题[3][4][5]。比如，如何让医生搜索到某一类病患的相关问题；医生常常有自己的知识库，如何电子化这些知识库，方便医生检索；如何利用现有的问诊数据，向用户推荐合适的问诊医生等一系列问题。

基于上述问题，就需要利用大数据时代下搜索引擎和人工智能技术，搜索引擎一直在海量数据中挖掘优质信息为用户提供优质服务。在大数据时代，新的数据给予了搜索引擎新的篇章。大数据决定了数据具有大而全的特性，在传统计算模式中，通过抽样数据分析得到结果已经不如通过大数据分析得出的结果精确。搜索引擎成为了一个数据加工工厂，经过大数据的挖掘，抽象结构化有价值的数据，加速数据信息流动，使得搜索为用户提供更多的服务和更高的价值。大数据使得人工智能技术开始新的发展，而人工智能技术也深入影响着搜索引擎。大数据下的人工智能在应用层包括机器学习、人工神经网络、深度学习、自然语言处理、知识表示与知识库技术、推理技术、搜索技术等。为了解决论文文中的搜索和推荐问题，需要利用自然语言处理技术，不仅需要对用户的搜索进行分词，还需要对检索语句做语义分析，甚至信息拆解。知识表示和知识库技术的主要目标就是存储知识，一个好的存储方式可以让程序更加灵活，方便快速检索。

因此，本文基于母婴领域的问诊数据，通过研究搜索引擎平台和自然语言处理的基础之上，实现智能搜索服务，可以有效的发觉在线问诊平台所积累的大量数据，从而更好的服务互联网病患群体。

## 国内外现状分析

随着互联网的快速发展，人类面临前所未有的信息过载问题。为了从海量数据中快速方便精准的检索准确的信息，Web搜索引擎应运而生。在互联网发展的早期，数据量较小，单机索引就能支撑一个完整的搜索应用。但是随着数据量的快速膨胀，此时对搜索引擎提出了分布式、准实时、高容错、可扩展、易于交互等诸多要求。ElasticSearch[6]的诞生很好的满足了上述大数据时代的搜索产品的需求。ElasticSearch是一款基础Apache Lucene[7]的Java开源搜索产品，其最早于2010年发布。之后ElasticSearch具有开源、分布式、准实时、RESTful、便于二次开发等特点。而随着数据量提升，之后又发展了出了推荐系统[8]。推荐系统是一种信息过滤和精准信息推荐系统，用于预测用户对物品的评分。现有的推荐系统产生推荐列表的方式主要有以下3个方面：协同过滤[9]、基于内容的推荐、混合推荐解决方案。协同过滤方法可用于预测用户对哪些物品感兴趣，基于内容推荐是利用一些有关物品的离散特征，推荐出具有类似性质的相似物品，两种方法的相互结合即为混合推荐系统。

最近，研究员更多的关注与基于矩阵分解[9]的手段去解决上述问题。这种手段可以将输入数据有效的降维。近期，有人又提出利用词嵌入[10]的手段去做推荐，词嵌入是自然语言处理（NLP）中语言模型与表征学习技术的统称，它是指把一个维数为所有词的数量的高维空间嵌入到一个维数低得多的连续向量空间中，每个单词或词组被映射为实数域上的向量。而相比于矩阵分解的方式，词嵌入可以用低维的向量去表征输入的数据元素。13年，谷歌发表Word2Vec[11]算法，是用来产生词嵌入的相关模型，其中，词嵌入是指词的向量表示，向量的每个维度表示一个特征向量。这类模型包括CBOW和Skip-Gram，且它们均为双层的神经网络，用来训练以重新建构语文本中的单个词，利用词向量去预测周围词向量。之后，有人又利用词嵌入技术做文本推荐，直到16年的时候，又有研究人员利用词嵌入的方式去做LBSNs(基于地理位置的社交网络)的推荐系统[12]，它利用Word2Vec基于词向量的去预测上下文，和根据上下文去推测当前词的特性，去做基于历史访问轨迹，将这些历史访问轨迹进行和相应的用户直接送入Word2Vec模型，用输出的模型和最近邻算法直接用来预测和推荐结果的排序，取得了不错的效果。一般情况下在技术上，Word2Vec通常被认为是一种深度学习技术。而深度学习技术在推荐系统领域方面的应用有限，通常被用的只有Restricted Boltzmann Machines (RBM’s) 去构建评分相关性模型和分层贝叶斯模型去学习内容和偏好模型。

基于上述研究现状，本文提出利用ElasticSearch搜索数据库和Word2Vec算法，依据用户向那些医生发起的提问的问诊记录，去构建问题模型和访问历史模型，进而去做问诊记录搜索和推荐系统。

## 论文工作内容

本文以Elasticsearch搜索引擎和Word2Vec的Skip-Gram算法以及Node.js提供Web服务为基础，充分利用现有在线问诊平台的问诊数据和医生构建的知识库，分别对其进行检索，之后，利用问诊数据建模输出医生推荐模型，并根据终端用户对症状的描述或者相关医生姓名，为终端用户推荐合适的医生。

本文主要研究内容如下：

1)设计和实现整个系统的基础应用服务模块

基于与应用服务总体包括以下几个模块：数据预处理模块，搜索引擎模块，模型训练模块，RPC服务模块，Web服务模块。数据预处理，就是将原本的数据格式转变为构建适合之后文本检索和模型训练的格式；搜索引擎，则主要负责索引数据同步，实现高效检索；模型训练主要利用数据预处理阶段处理的数据作为Word2Vec算法的输入来构建医生推荐系统的模型；RPC服务模块是将模型输出的结构注册到RPC服务供Web服务调用；Web服务提供整个后端服务的调用和操作页面的显示。

2)设计和实现整个系统业务服务模块

推荐系统总体包括以下模块：问诊记录搜索，知识卡检索，智能医生推荐。问诊记录搜索，是通过症状名称检索相关问诊记录详情；知识检索，是通过输入症状名称检索相关症状的描述和常见解决办法；智能医生推荐，是指通过终端用户的对病症的描述为其推荐合适的医生。

## 论文组织结构

本文的主要章节组织结构安排如下：

第一章：绪论。首先介绍问诊平台搜索场景和推荐系统的选题背景和发展现状，随后阐述了论文的内容和结构安排。

第二章：相关技术概述。分别对论文内容中相关技术进行了介绍，其中包括Elaticsearch搜索引擎、Word2Vec算法、SOA服务架构，为后文研究奠定了基础。

第三章：智能问诊系统的业务场景需求分析。首先对总体业务进行分析，然后进行需求分析建模。

第四章：智能问诊系统统设计与实现。首先进行总体设计和数据库设计，并且介绍了系统整体框架和实现技术，然后进行了按照基础和应用服务与企业和业务服务分别按模块详细介绍了内部的设计与实现。

第五章：系统测试与分析。包括测试环境配置和搭建、功能测试和性能测试。

第六章：结论和展望。对本论文工作进行总结，提出论文研究问题过程中的不足之处以及对后续工作进行的展望。

# 相关技术概述

## ElasticSearch搜索引擎

### ElasticSearch技术介绍

ElasticSearch(ES)是一个基于Lucene构建的分布式、RESTful的文本数据搜索引擎，ElasticSearch不仅是一个全文本数据库搜索引擎，还是一个分布式文档存储数据库，其中每个字段均被建立倒排索引，通过这些倒排索引可以快速的检索的符合需求的数据。

ES的基本概念与常见的关系型数据库类比如下：

索引(Index)是ES将数据存储于一个或多个索引中，索引[13]是一类具有相同相似特征的文本文档，相当于关系型数据库中“数据库”。

类型(Type)是索引数据库内部的逻辑分区。因此，索引数据库中可以定义一个或多个类型(type)，类型相当于“数据表”。

文档(Document)是关系型数据库的“记录”类似，一般是用JSON格式表示。

映射(Mapping)：mapping一般是相当于编程语言中的类型定义，确定文档中各个字段的数据类型，以及它所需要用的分词器。

集群是一个或多个节点的集合，共同存储了所有数据集合，提供了联合索引以及可跨所有节点的搜索功能，实现了ES的高可用和分布式特性。

节点是集群中的一个成员节点是存储数据地方，它可以完成集群索引及搜索操作。

分片(Shard)是实际存储数据的地方，是物理存储。

### ElasticSearch工作原理介绍

ES集群主要负责存储写入的document和吐出应用程序所搜索的数据，如图2.1所示，圆圈1，表示请求一个节点(并非是master节点，每个节点均有可能)；圆圈2，表示自动请求路由，利用RESTful协议定位到资源；圆圈3，表示响应，即返回搜索结果。

ElasticSearch具体搜索过程实现基本分为两步：创建索引和搜索。

创建索引：以Document为对象，以默认分词器为例。“I like pingpong”，这个文档首先被分词为“I”，“like”，“pingpong”，然后倒排索引，就是这3个词分别指向了这个文档。

搜索：搜索的时候，在上面的例子中搜索“I pingpong”，会被分词器分词为“I”，“pingpong”，然后ES会用这个搜索说索引与现有文档的倒排索引比较，发现对应“I like pingpong”中的两个分词，于是返回“I like pingpong”这个结果。



ElasticSearch的分布式调用架构

## 推荐系统技术

### 推荐系统技术概述

推荐系统其实就是一种预测用户或对物品的打分高低，一般情况下，得分越高用户对该物品越感兴趣。一般系统通过分析用户访问物品的历史记录，或者用户提供的搜索文本，来构建用户兴趣模型。推荐系统把用户模型[14]中的用户感兴趣信息和推荐对象模型中的特征信息匹配，同时使用相应的推荐算法进行计算筛选，找到用户可能感兴趣的推荐对象，然后推荐给用户，所以推荐系统所要解决的具体问题总体有两个，分别是预测(Prediction)和推荐(Recommendation)两个部分。预测所要解决的主要问题就是通过模型的输出去推测每一个用户对每一个项目的的喜爱程度，而推荐所要解决的主要问题就是根据预测阶段所得到的各个用户对各个物品的打分向某一个用户推荐他没有实际打分的物品，一般情况按照得分的从高到低进行排序。

### 基于Word2Vec算法的推荐系统

Word2Vec算法是2013年Google公司发布的开源词向量工具包，这个项目的算法原理参考了Bengio在2003年设计的神经网络语言模型，由于这个模型存在参数多，效率慢，不适合大语料。Google团队对其做了简化，实现了Word2Vec词向量模型。Word2Vec模型一般基于如下假设：要比较两个词在语义上的相似性，决定于它们的邻居词分布是否类似。Word2Vec是一种神经网络架构的概率模型，最初应用于学习单词嵌入，就是单词的向量化表示，近年来，这一技术也被应用于推荐系统中，不过一般是基于内容的。这种神经网络通过输入的大量文本，分析它们，并对每个单词产生一个表示该单词的数字向量。这个向量是极其重要的，因为它们编码了这个词的意义信息和它出现上下问的关系。一般地，Word2Vec包含两种算法一种是连续词袋（CBOW）模型[15]，一种是Skip-Gram[15]。CBOW的意义是，输入是周围的词的词向量，而输出是当前的词的词向量，也就说是，通过周围的词来预测当前的词，而Skip-Gram与CBOW正好相反，就是输入的是当前词的词向量，而输出是周围词的词向量。也就是说，通过当前词来预测周围的词。

利用词嵌入(Word2Vec)方法的推荐系统就是将输入的文本通过学习转为低维度的向量空间，这种方法通常被用来学习语言的规律和文本数据集的语义信息[16]。而在本文中，作者使用Word2Vec的Skip-Gram模型做推荐系统，采用KNI(K-nearst items)，就是K个余弦相似度最近的项目来做为推荐项目，在其中，论文中分别先预处理问诊记录和用户访问医生记录，之后进行命名实体识别[17][18]，专有名词提取，将它们作为文本输入给Word2Vec算法，Word2Vec算法的输出模型可以作为推荐模型，推荐模型的输出中可以提取医生姓名作为推荐项目。

## SOA架构

### SOA架构概述

SOA(Service-Oriented Architecture)，是面向服务的一种软件架构方法，这里的服务，不限于业务服务，企业服务，功能服务以及基础等服务，它是一种松耦合的服务架构，要求软件项目在开发时，照相关标准或者各种网络协议，进行分层开发[19][20]。通过这种分层设计使得软件项目的鲁棒性变高，且尽可能集成其他软件项目，以达到快速扩展，满足软件项目需求的多变性。

一般情况下，基于服务的架构的共性就是分布式架构，也就是服务组件之间的数据交换都是通过各种网络访问协议来实现。在基于服务的SOA架构中，“模块化”指的是将应用的各个服务功能各自进行封装的方法，具体实践如图2.2所示，一般，将WebUI和手机终端通过前端网络服务访问功能模块，前端网络服务功能则与各个具体功能通过各类协议调用，通过拆分后的每个服务模块都可以单独进行服务设计、功能开发、功能和性能测试，以及单机和多机的分布式部署，与其他组件或服务之间的依赖性和耦合度都很低。模块化的架构可以用重写的方式来维护组件健壮性。随着业务增长，架构可以逐渐地、以很小的部件为单位进行重构或者替换，而不是整个应用进行重构或者重新自底向上进行架构重写，一般情况下，在一个软件项目利用SOA架构，就是采用前端服务负责处理所有的外部交互，后端负责实现和前端的对话协议，所有的客户端是一种业务上的抽象，进一步需要按服务进行的抽象的化，分别业务服务、企业服务、应用服务、基础服务。而在一般实践中，分别前端页面展示，API功能服务以及基础服务如数据存储服务进行快速开发。

虽然，SOA架构的分层式和解耦开发可以快速做出鲁棒性高的软件项目，但是，它依然存在这各个服务之间如何协同工作问题以及服务是否能做出正确响应的问题。



SOA架构图

### SOA架构面临的主要的问题

服务合约，是服务供应者（供应商）和使用者（消费者）相互进行数据交换合约语言，它规定数据输入与输出的协议，现在一般使用JSON或者RESTful规范作为服务合约语言[21]。

服务可用性（Availability）和服务响应能力（Responsiveness）是两个在服务设计实现阶段需要面对的问题，可用性，是要确保服务是经过功能测试，确保该服务可用，而服务的响应，是指，服务被使用者能被正常响应返回结果，且响应时间是可以接受的。这两者都涉及服务客户与远程服务之间通信的能力，因此一般情况下客户一般会采用不同方式来解决这两个问题[21]。

基于服务的架构相对于单体式应用[22]来说是一种进步，但是同时也带来比较突出的三个问题，服务合约的选择以及服务可用性以及服务的响应能力。而转向基于服务的SOA架构，面临着必须在服务合约的选择和服务可用以及响应的解决方案中做出权衡。而在本文所述的智能问诊系统中，分别采用RESTful协议[23][24]与JSONRPC[25]协议，解决了服务合约、服务可用性、服务响应能力等问题。

## 本章小结

在本章中介绍了论文中相关技术，首先是搜索引擎和推荐系统技术的介绍，并介绍了基于Word2Vec的推荐系统设计，然后重点介绍了SOA架构的概念，及其所面临的问题，而文中智能问诊系统之所以采用SOA架构，是因为涉及多种语言开发，各个服务之间的调用协议也不同，采用SOA架构开发可以尽可能的解耦系统模块，方便开发和部署，为后文的研究奠定了基础。

# 系统需求分析

在第一章中已经提到，本文主要设计和实现基于问诊数据的智能搜索平台，首先是要设计和实现问答库搜索，然后是要设计和实现知识卡搜索，最后实现通过问题或者医生搜索推荐相关医生。本章详细介绍了各个服务的功能需求。

## 系统业务分析

在“十月呵护”母婴领域的在线问诊平台中，积累了超过3000万条的问答记录，为了充分挖掘这些问答记录的价值，需要设计一个智能搜索平台，尽可能的挖掘问诊数据的价值。

在智能问诊系统中，问诊记录搜索服务是主要是为了在海量在线问诊数据中快速检索出相关人员想要的问诊数据；知识卡构建和检索服务主要是因为医生通常会总结各个疾病或者病患的医疗卡片包含这些常见疾病以及其解决方案，如何把这些知识卡电子化，并辅助医生快速问诊，最后，医生推荐服务是因为互联网病患群体一般知道症状，但是不知道该去挂哪个医生的号，因此需要一个医生推荐系统，当病患描述自己的症状时候，通过病患的描述，向其推荐合适的医生。通过研究和考察，得到其中的核心业务框架如图3.1所示。



智能问诊业务框架图

从图3.1和上文可以得知，该系统的主要用户是医护人员，和病患用户。用户分别通过问题搜索、知识卡构建以及搜索、以及辅助病患快速找到合适的医生。

### 问诊记录搜索业务分析

问诊记录搜索业务是指医护人员通过输入的症状名称或者症状，包括分词，检索，展示相关关键词，显示相关结果等。



问诊记录搜索业务活动图

从图3.2可以得知，问诊记录的业务整体流程如下：

1)首先在输入症状名称或者症状表述。

2)当输入是症状名称时候，直接检索ES(ElasticSearch)数据库，当每次重启ES服务的时候，系统会自动将MongoDB数据库存储的问诊记录数据与ES数据库同步，并更新ES数据库中问诊记录的索引，用来检测问诊记录。当输入是症状的时候，先进行分词，选出其中的名词，然后检索相关词模型，找出关键词，如果没有关键词，则业务结束，然后检索ES数据库。

3)检索ES数据库时，如果找到相关问诊记录索引，则按照ES默认的TF/IDF算法进行排序展示。如果没找到，则业务结束。

4)如果用户想显示单个搜索结果，则可以点击单个问题卡，进行展示。

### 知识卡检索业务分析

知识卡管理业务是指对问诊中医生总结的知识卡进行构建和检索的业务，其中包括构建知识卡，同步知识卡，检索知识卡等。知识卡业务活动图如图3.3所示。



知识卡检索活动图

从图3.3可以得知，知识卡检索的业务整体流程如下：

输入相关症状名称，送入ES数据库，当检索ES数据库时候，ES会检索相关症状的索引，该索引是每次重启ES数据库的时候，系统会将MongoDB数据库中存储的知识卡同步构建到ES数据库中，如果有查询结果，点击症状后可以展示结果，如果没有则退出。

### 智能医生推荐服务业务分析

智能医生推荐系统，包括当终端用户输入症状描述或者医生姓名的时候，经过关键词词典提取，送入相关模型，匹配到最终推荐的医生。具体业务活动图如图3.4所示。

从图3.4可以得知，业务整体流程如下：

1)用户输入症状的描述，或者输入医生的姓名

2)当输入是症状的时候，检索“医生问诊”模型；当用户输入里包含医生姓名的时候，检索 “用户医生”的模型。

3)如果没有匹配到相关医生，业务结束。

4)如果匹配到医生，当用户选择某一医生时候，展示医生详细信息。



智能医生推荐业务活动图

## 系统需求建模

智能问诊系统是为了有效应用目前在线问诊所积累的庞大问诊记录数据，这类数据中包含终端用户对于常见疾病或者健康状况的咨询，以及各位医护人员为其制定的解决方案。而建立一个搜索系统，是提取这些问诊数据价值的初步手段，基于这个搜索系统，可以利用一些深度学习的神经网络，使得整个系统变得更加智能化，通过这些智能化的赋能，能够使得这些数据的价值最大化的被挖掘，从而更好的为病患服务，进一步，提升整个医疗服务行业的效率，进而可以推动整个医疗服务行业的发展。

从图3.5可以看出， 智能问诊系统服务整体划分为3个业务服务，分别是问诊记录搜索、知识卡构建和检索、智能医生推荐服务。

问诊记录搜索：对问诊记录查询检索。包括原问诊模型的数据预处理、同步更新构建ES索引、全部查询结果、单条记录查询。

知识卡构建检索：利用问诊系统的知识库构建在线可检索的知识卡。包括知识卡的创建、同步更新构建ES索引、获取结果展示。

智能医生推荐：通过用户医生推荐模型和医生问题模型，预处理当前用户的输入，获取专有名词和医生姓名来推荐相关的医生。包括对输入数据的预处理、问诊记录的预处理、模型的构建、获取相关结果、医生的展示等功能。



智能问诊系统整体需求用例图

### 问诊记录搜索服务的需求

问诊记录搜索服务需求主要包括对原问诊记录的重新整理合并，并构建搜索引擎数据库ElasticSearch的问诊记录索引实现搜索，问诊记录搜索服务的需求用例图如图3.8所示。问诊记录的合并，主要是因为依据现有的实际需求，想减少数据库的访问压力，为了快速构建精准的搜索引擎数据库的索引和之后为医生推荐模型训练阶段更容易的提取字段所服务，因此将原有问诊记录按照，每个终端用户所咨询医生的方式进行合并，合并之后给整个问诊记录同步映射到数据搜索引擎ElasticSearch。

从图3.6可以看出，在问诊记录搜索的需求中，用户一般为医护人员，且医护人员可以通过输入，后调用相关API，正常在前端展示。

用例规格说明如表3.1所示，在满足预设条件系统各项基础服务正常运行，数据预处理完毕以及问诊记录字段的索引在ES中构建完毕的情况下，医护人员按照表3.1中事件流顺序依次执行每个事件。



问诊记录搜索需求用例图

问诊记录搜索用例规格说明

|  |  |
| --- | --- |
| 用例名称 | 问诊记录搜索 |
| 参与者 | 医护人员 |
| 前置条件 | 服务正常运行。  数据预处理结束。  构建索引全字段索引。 |
| 事件流 | 1.当用户输入症状名称 2.输入相关关键字，和相关问诊记录 3.点击”相关关键字”，更新相关问诊记录  4.点击”单个问诊记录”，切换问诊详情界面 |

### 知识卡检索服务的需求

知识卡检索需求主要包括通过对利用现有医生总结的疾病分类知识库，当医护人员输入症状名称的时候，通过数据搜素引擎ElasticSearch，检索出相关疾病知识卡，并显示某一知识卡的详情。



知识卡检索需求用例图

知识卡检索用例规格说明

|  |  |
| --- | --- |
| 用例名称 | 知识卡检索 |
| 参与者 | 医护人员1 |
| 前置条件 | 服务正常运行。  知识卡构建服务已经完成。 |
| 事件流 | 1.当输入“症状名称”时，调用知识卡查询API，显示相关病症分类。  2.当点击“相关病症分类的”，调用知识卡显示API。 3.显示知识卡，点击某一具体问题时候，显示解决方法。 |

从图3.7可以看出，在知识卡检索的需求中，用户一般是医护人员，且其可以构建知识卡，具体业务描述，就是将线下医生利用Excel表格工具总结的病症分类，描述，常见问题及其解决方法等知识库，存入MongoDB数据库，之后和ES数据库建立索引，方便之后检索。

知识卡检索的服务的用例规格说明如表3.2所示。从表3.2可以看出，用户可以通过输入病症名称可以检索到相关病症分类及其解决方案。

### 智能医生推荐系统的需求

智能医生推荐系统需求的主要用户是各种终端用（手机、PC等），当用户输入症状信息时，系统会根据从症状信息中提取的名词来，利用Word2Vec算法构建的医生问诊模型，来预测和该名词相近的医生，并将选择排名前9的医生，推荐给用户。

当用户输入的信息里包含医生姓名的时候，这时候的推荐是依据用户医生模型，来预测与该医生相关的医生，并将排名前9的医生，推荐给用户。智能医生推荐系统的需求用例图如图3.8所示。

从图3.8可以看出，在智能医生推荐系统的需求中，用户一般是想要在线问诊的用户，且这类用户想要通过简要的描述症状信息，就可以快速的匹配到合适医生，为他们提供专业的建议和指导。但是，要满足这个需求，需要以下前置性服务：问诊数据预处理，将问诊记录表进行合并使得新表中包含某一用户对应医生的全部问诊记录，不需要频繁进行联合查询，浪费MongoDB的内存和IO开销；用户医生推荐模型的生成服务，要实现根据输入医生姓名进行推荐与其相关的其他医生，需要构建此推荐模型；医生问诊推荐模型的统建，就是根据用户输入对症状的描述，为其推荐合适的医生。

智能医生推荐系统的用例规格说明如表3.3所示，标注的前置条件中服务正常运行，是指当前系统的各项基础服务包括MongoDB服务、Web服务与RPC服务正常运行，以及数据预处理和各个Word2Vec算法模型构建完毕，是指用户医生推荐模型与医生问诊推荐模型构建完成，通过用户医生和医生问诊正常输出推荐的医生姓名。

智能医生推荐系统用例规格说明

|  |  |
| --- | --- |
| 用例名称 | 智能医生推荐系统 |
| 参与者 | 终端用户 |
| 前置条件 | 服务正常运行。  模型构建完毕，且正常运行。 |
| 事件流 | 1.输入症状描述或医生姓名，正常提交输入， 调用推荐医生API。  2.点击“医生姓名”。  3.调用医生详情API，显示医生详细信息。 |



智能医生推荐系统需求用例图

## 智能问诊系统性能需求

智能问诊系统的应用场景中，需要对Web服务，ElasticSearch服务以及RPC服务的性能进行需求分析，这些性能指标包括：CPU使用率和服务平均响应时间，具体性能需求如下：

CPU使用率：Web服务器、ElasticSearch服务器与RPC服务器的在并发请求最大数达到1000的情况下，各个服务器的CPU使用率不超过80%，且在并发数500的时候，各个服务器的CPU使用率应不超过40%。

平均响应时间：Web服务、ElasticSearch服务和RPC服务在并发请求数最大1000的条件下，各个服务器的平均响应时间应不超过1.5秒，而在并发请求数500的时候，各个服务器平均响应时间应不超过0.3秒。

## 本章小结

本章主要介绍了智能问诊系统的整个系统的业务进行需求分析，就是从系统所需要实现的业务层面给出分析。该章节需要实现问诊记录搜索、知识卡检索服务、智能医生推荐系统的业务分析，然后对各个需求进行了分析建模，并最终完成整体的业务梳理。

# 系统设计与实现

在第三章中已经对智能问诊搜索服务平台的需求进行了分析，这一章将针对需求进行总体设计和分模块设计，并且最终实现基于问诊数据的智能问诊平台。由于智能推荐医生服务的内部实现较为复杂且与依赖于推荐模型的实现，因此在模型训练阶段会详细对Word2Vec算法中Skip-Gram模型应用于医生推荐模型的实现进行阐述。

## 系统的SOA架构设计

智能问诊系统架构设计如图4.1所示。整个系统采用SOA这种面向服务的架构。按照SOA架构的四种基本服务类型，构建如下的基本服务架构。一般SOA架构定义了四种基本的服务类型包括：业务服务，企业服务，应用服务，基础服务。其中，业务服务一般指抽象的、高层级、粗粒度的服务。一般只包括服务名字，期望的输入和输出。企业服务是具体的、企业层级的、粗粒度的，用以实现业务服务所定义的功能。企业服务可以与业务服务之间存在一对一或一对多的对应关系。应用服务，是细粒度的，特定于具体应用的服务，与某个特定应用的语境相关。应用服务提供在企业服务中没有特定的业务功能。基础服务，实现的是非功能性业务。



智能问诊系统SOA架构

从图4.1可以看出，利用SOA架构的四种服务类型分类，将智能搜索平台业务进行拆解。

业务服务主要包括该系统需要实现的3大业务，其中包括，问诊记录搜索服务，知识卡检索服务，智能医生推荐服务。

企业服务提供了业务服务中各个模块所需要的API(应用程序接口)。

应用服务是整个各个功能实现的核心。这一类服务中，实现了问诊记录的查询服务，这个服务包括，问诊记录索引的构建，同步。知识卡的构建，和实现检索查询功能，并将查询结果排序。主要实现智能医生查询的，各个功能模块，如模型训练服务，数据预处理服务，检索医生服务，实现将这些服务注册的RPC服务的应用服务。

基础服务，主要包括，搜索引擎和Web服务的日志服务，一些安全策略服务，数据库服务，和ES(ElasticSearch搜索)服务。

## 系统的数据库设计

在对智能问诊搜索系统的功能和性能做了详细的需求分析之后，从软件的应用场景考察，对软件中需要管理的数据进行考察和分析，对业务进一步细化，确定软件中数据实体需要涉及的属性和范围，为后续的功能设计和实现提供强有力的支持和参考。

问诊记录搜索和智能医生推荐系统的基础的是基于问诊记录表的，问诊记录表原来的设计是包含：一个第一次提问问题也就是根问题的ID，用户ID，医生ID，问诊内容，图片信息列表，创建时间，类型，以及是来自谁的提问或回答一个叫FROM的字段，这个字段包括两个类型一个是用户，一个是医生。问诊记录关联的表有医生表和用户表，医生表包含以下字段：姓名，职称，图片，医院，标签，描述，创建时间；用户表包含以下字段：姓名，电话，创建时间。为了之后便于数据处理和数据检索，需要将问诊记录表的数据进行整合构建一张新表，新表的结构需要包含以下字段：第一次提问问题ID；标签；医生姓名；日志列表包含FROM，内容，图片数组；创建时间；用户名。

知识卡检索的基础是线下医生总结的知识文档，需要将这些文档存入数据库中，然后构建ElasticSearch的索引，方便检索。主要是包含以下字段：科室、分类、标签、症状、症状图片、简要描述、常见问答（包含：标题、内容、图片等）。主要需要同步症状名称、内容、描述等ElasticSearch这个搜索引擎数据库的索引。这个知识卡只是根据现有业务的初步探索，以后随着病症的多样化和特定格式的需求，需要做进一步更改。

根据以上分析，可以得到智能问诊系统的问诊记录数据库E-R图如图4.2所示。



智能问诊系统E-R图

从图4.2可以看出，智能问诊搜索服务的数据库包含的主要信息有用户，医生，问诊记录等。一个问诊记录至少包含一个医生和一个用户，而一个用户或者一个医生对应多个问诊，一个知识卡主关联多个常见问答，它们是1对多的关系，通常某种病对应一系列常见解决方法，而一个医生管理可以管理多个知识卡。

整个系统采用MongoDB作为后台数据库，数据库关系如图4.3所示。



智能问诊系统UML数据库表示

从图4.3可以看出，InquiryChatLog是问诊记录表，Doctor是医生表，User是问诊用户表，KnowlegeCard是知识卡，Body是问诊详情。

## 系统整体框架和实现技术

系统整体实现框架图如图4.4所示，其具体描述如下：

Node.js Server主要负责服务端渲染，进行操作页面展示和调用ES(ElasticSearch)服务，数据库服务，RPC服务。

ElasticSearch Server主要负责和底层的MongoDB数据库同步数据，并向上层的Node.js Server 提供检索RESTful API，供Node.js Server调用。

Offline Data Process主要负责线下的数据预处理任务，包括，问诊记录数据的合并，负责专有名词识别，并按照识别出来的专有名词进行分词，进而生成用户医生csv表和医生问诊csv表。

Offline ML Trainning Service主要负责利用数据预处理任务提供的csv表，利用Word2Vec模型的训练算法，生成用户医生模型，医生问诊模型，为预测推荐做好准备。

RPC Server将训练好的推荐模型以及一些工具函数注册到RPC Server中，在Node.js Server中调用在RPC注册的方法。

MongoDB主要提供数据库服务。



智能问诊系统整体框架

上述各个模块的调用关系自底向上的表述如下，MongoDB作为底层数据库支撑，搜索引擎ElasticSearch 负责同步构建数据库中问诊记录和知识卡索引，向Web服务提供RESTful路由；MongoDB向Web服务提供单条记录的详情查询，知识卡详情以及医生详情等信息；Offiline Data Process线下数据预处理服务，该服务主要利用MongoDB里存储的合并后的问诊记录，提取出医生问诊csv表以及用户医生csv表，csv表作为Offline ML Trainning Service 线下模型训练的输入，模型训练的输出的推荐模型被注册到RPC服务中供Node.js提供Web服务调用。

整体系统服务的实现技术：

编程语言：Node.js，Python。

技术框架：Nuxt.js，JSONRPC，Gensim。

数据库中间件：Mongoose。

后端数据库：MongoDB，ElasticSearch

定时任务：crontab。

模型训练算法：Word2Vec。

## 基础与应用服务设计与实现

根据上文的需求分析和系统功能设计，本章节会逐一介绍系统SOA架构中主要基础与应用服务的实现。对于模型训练服务的实现，将着重阐述如何将Word2Vec算法的Skip-Gram模型应用于智能医生推荐服务。

### 数据预处理服务的设计与实现

数据预处理服务是应用服务，因为之前的问诊记录存储的记录的是关联表，为了提取出只需要满足当前业务所需要的数据，需要重新定义这个问诊记录的数据结构，因此，对原问诊记录存储表的格式转换如下，另外需要说明下，因为MongoDB是文档型数据库，因此它支持字段嵌套，因此可以按如下方式设计表结构。在表4.1中主要是将用户所咨询单个医生的完整对话记录合并到一张表内，方便问诊记录数据提取和展示。

具体从图4.5可以看出整个数据预处理的类图，它的具体表示如下所示：

1)DataProcess是数据预处理类，有一个私有属性db，负责连接MongoDB数据库，有3个公共方法，负责合并用户groupUser方法，是从新问诊记录模型中获取每个用户问了哪些医生，构成一张只有用户和医生的两栏的csv数据表；负责构建合并后问诊模型combineQuestionLog方法，就是合并旧的的问诊记录创建新的问诊记录；负责提取医生问诊记录的csvQuestionLog方法，是生成一张只有当前医生的问诊记录的csv表。

2)MongooseService是主要是当前数据库处理的业务方法，其中，connect方法是判连接数据库的具体方法，返回数据库是否正确连接；createNewChatLog方法是创建上面所示表的数据表NewChatLog（新的问诊记录表）；findChatLog方法查询NewChatLog表。

3)csvjsonService是实现将JSON格式的数据转换成csv格式的文本文件，是辅助工具类。

合并后问诊记录字段表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | | 字段含义 | | 数据类型 | | 是否允许空 | |
| ROOT | | 主问题 | | ObjectID | | 否 | |
| TAGS | | 标签 | | String | | 是 | |
| TITLE | | 职称 | | String | | 是 | |
| DESCRIPTION | | 描述 | | String | | 是 | |
| DOCTOR | | 医生 | | String | | 是 | |
| USER | | 用户 | | String | | 是 | |
|  | FROM |  | 叙述人 |  | String |  | 是 |
| LOGS | CONTTENT | 问诊记录 | 内容 | Array | String | 是 | 是 |
|  | IMAGES |  | 图片 |  | Array |  | 是 |



数据预处理模块类图

上述程序的具体执行顺序为例，数据预处理模块的时序图如图4.6所示。由图4.6可知，数据预处理的过程中包含了如下几个步骤：

1)定时任务的执行程序首先初始化DataPrcess类

2)DataProcess的实例默认调用Mongoose提供的数据库连接方法connect，返回数据库是否正确连接的状态给任务执行程序。

3)创建新的问诊记录，调用当前DataProcess的实例的combineQuestionLog方法。之后内部调用MongooseService的findChatLog以及createNewChat来构建新的表。

4)当新的问诊记录创建完成，由定时任务执行程序调用DataProcess类的公共方法csvQuestionLog去构建医生问诊csv表的操作，此时构建的csv表主要包含医生和问诊记录。

5)定制任务执行程序调用DataProcess类的公共方法groupUser去构建用户医生csv表，主要是查询新构建的表提取用户和医生字段，一个用户包含多个多医生。

6)之后返回是否数据库是否成功响应，csv表格是否正常生成的响应。



数据预处理模块时序图

### 模型训练服务的设计与实现

模型训练服务是应用服务，它是对数据预处理服务中提取出的用户医生csv表和医生问诊记录csv表，利用分词器提取专有名词后，接着利用Word2Vec算法的Skip-Gram进行推荐模型建模。以下分3部分分别简绍其中的关键技术点，并最终设计出与实现该模块的相关架构。

一是分词，该训练模型中的分词主要使用中文的jieba分词工具，jieba分词是采用张华平的Nshort算法作为主要的分词算法，该算法原理简单、容易理解、便于训练、大规模分词效率较高、模型支持增量扩展、模型占用资源低等优势。由于只想提取出来疾病关键字和病症部位，因此需要建立给jieba做一个中文分词器，主要是利用Unicode编码将文本中的中文提取出来，接着需要确定一些问诊类项目的停用词，该词表文件一般由医护人员维护，将一些常见不需分词器分词的词语写入该停用词表。

二是提取专有名词，为了更好的训练模型，需要正确的分词，因此需要利用用户字典，这个用户字典是用户自定义，一般由医护人员维护，他们来确定某种词是不用分词的，而一般分词会将代词，动词单独分出词，因此，利用句法依存关系算法，提取出问诊记录中专有名词。

最后就是利用Word2Vec算法的Skip-Gram模型和KNI最近邻算法实现推荐系统。Word2Vec的Skip-Gram算法之所以可以用来做推荐系统，以及如何利用Word2Vec算法解决当前系统的智能医生推荐问题，是因为，在一般情况下，Word2Vec算法选用的文本内容作为输入，而在本文中分别采用医生加问诊内容与用户和他们咨询过的医生分别作为输入，来进行医生推荐系统的建模。

通过两个实际问诊记录例和用户访问医生记录为例说明实现思想。例如一段包含医生问诊记录，经过分词和提取专有名词后，变做如下格式：“<符娟> 儿科医生 湿疹 小儿 疝气 腿 儿童 皮肤病 儿童 外科 疾病 家庭 护理 拉肚子 感觉 吃奶 精神 问题 问题 酸菜鱼 拉肚子 吃奶 精神状态 大 性状 图片 程度 饮食 关系 妈妈 饮食 食物 甜食 腹部 <符娟>”，先对这个格式做以下简要说明，为什么要在分词后的结果中加上”<医生姓名>”，因为需要将这条问诊记录和当前医生相关联，而且可以通过放大预测的范围，通过正则匹配的方式的获取到这个医生的名字。

比如，当输入“湿疹”的时候，那么，“<符娟>”、“儿科医生”、“小儿”、“疝气”就是想要的词。那么其Word2Vec的神经网络如图4.7所示，其中W1和W2代表权重矩阵，它控制了在输入上施加的连续变换的权重来得到输出。训练神经网络包括学习那些权重矩阵的值，这些值使得输出与所提供的训练数据最接近。给定一个输入词“湿疹”，先通过网络进行第一个正向传播，用来得到输出词是根据训练数据所期望的概率。由于，肯定的知道对输出的期望是什么，所以可以测量预测中的误差，并利用反向传播将这个错误通过网络传播，并通过随机梯度下降来调整权重。通过这个步骤，就可以调整W1和W2矩阵的值，这样它们就能更准确的预测出想要的输出词。当完成这个步骤的时候，将上下文窗口移动到下一个单词，然后重复上述的步骤。当对训练语料里的句子重复了上述步骤，这些都完成的时候，权重矩阵将会集中到产生最准确预测的那个值上。如果，两个不同的词语出现在相似的环境中，且把这两个词中任何一个作为输入，那么神经网络也会给出非常相似的预测结果。需要特别说明下，W1这个权重矩阵，它的行数和词汇表中的单词一样多，每一行都包含了一个与特定单词相关的权重。因此，由于相似的词所需要输入类似的预测，所以矩阵W1应该类似。这个矩阵中每个词的权重都是用来表示这个词的嵌入。



基于Skip-Gram的医生问诊推荐模型

因此，当放大上述语料的相似词范围时，就可预测出想要的包含医生姓名的特定格式的分词结构，而该预测相似过程的实现是利用KNI最近邻算法。

接下来，系统又想利用用户和其咨询过的医生来，通过输入医生姓名来预测，常和该医生一起被咨询医生，整个神经网络的架构如图4.9所示。

这样系统在构建输入的时候，就不需要分词，提取关键字这样去构建语料，可以通过以下这个方式来构建“续姣 周小晴 符娟 曹雅茹 赵志花 符娟 周小晴 凌万里 王倩”，其中第一栏是用户的名称，因为系统是通过微信号获取的用户信息，一般是当前用户所使用的微信账户昵称作为用户的名称，后面几栏是当前用户咨询过的医生记录。之所以会用这样的方式来构造输入语料，因为一般的基于Word2Vec的推荐系统，都是采用第一种方式，是基于内容，然将这些内容进行分词，而不是通过其他特征来作为输入，这种数据输入一般叫做用户和项目输入，这种数据也是可以作为Word2Vec的输入，来进行类似第一种方式的预测。一般，输入的数据用来通过Word2Vec来构建模型，而输出的模型可以用来做推荐系统。上面构造的句子结构中，其实就是推荐系统中，用户和项目作为输入，之后可以用连续的向量表示这些医生和用户。通过某一医生的向量表示，可以用来预测那哪些医生的向量和它接近。

基于上面的叙述，按照Word2Vec的Skip-Gram模型训练数据，用KNI最近邻算法来预测，哪些医生时常一起出现。比如当输入“赵志花”的时候，那么，通过上述模型和最近邻算法，就可以预测出，“符娟”和“曹雅茹”会在其附近一块出现，从而，进行医生的推荐。那么其神经网络图就会如图4.8所示，它的计算步骤与之前上述描述过程一样，通过这种方式来进行较为准确的预测。



基于Skip-Gram用户医生推荐模型



模型训练模块的类图

由图4.9可知，模型训练服务的类图中各个类的职责如下：

1)ChineseTokenizer是中文分词器，这里主要分出利用中文Unicode编码和正则匹配出中文，自定义一些停用词和专有名词表。主要是重写了原来jieba分词中调用方法。

2)phraseExtractor是提取专有名词，这个模块中主要包含了generate\_trees方法，它是利用NLTK(Python的一个自然语言处理包)生成语法树；leaves方法是获取语法树中的叶子节点；get\_terms方法是提取叶子节点中单词；get\_phrase方法是获取要提取的专有名词。

3)MySentence是为了读取预处理阶段所构建医生问诊记录的csv表，并构造上述第一个例子中那样的数据，用来送入Word2Vec模型

4)BuildSeg是构建Word2Vec的输入，主要是利用之前数据预处理阶段产出的csv数据表，分别调用doctor\_question\_seg方法产生这样的数据“<符娟> 儿科医生 湿疹 小儿 疝气 腿 儿童 饮食 关系 妈妈 饮食 食物 甜食 腹部 <符娟>”集合和user\_doctor\_seg方法来产出“续姣 周小晴 符娟 曹雅茹 赵志花”这样的数据集合。

5)BuildWord2Vec 类是包含一个gensim（python的高效自然语言包，包含了一个高效的Word2Vec的实现）的Word2Vec的Skip-Gram模型和构造其输入LineSentence属性，然后分别利用之前产出的分词表来作为参数输入，利用buildWord2Vec方法来构造训练模型。

以输入为医生问诊记录csv表为例子来，模型训练模块的时序图如图4.10所示：



模型训练模型时序图

由图4.10看出，当定时任务程序执行者获取数据处理模块处理好的医生问诊记录的csv表，将传入到BuildSeg模块中doctor\_question\_seg方法中获取到构造好的分词表，然后送入BuildWord2Vec去训练医生问诊模型，这个训练好输出的模型可以用来做推荐系统。

由于智能医生推荐系统在对某个专有名词进行匹配时，会匹配出多个医生，因此该系统在设计时，选取9个医生姓名作为推荐项目，因此查准率的衡量制定如下标准：

|  |  |  |
| --- | --- | --- |
|  |  | (4-1) |

在公式(4-1)中，表示查准率，表示真正预测正确的医生姓名的数量，9表示一次医生推荐系统输出9个医生的姓名。

接着，对智能医生推荐系统查全率的衡量制定如下标准：

|  |  |  |
| --- | --- | --- |
|  |  | (4-2) |

在公式(4-2)中，表示查全率，表示真正预测正确的医生姓名的数量，表示该专有名词对应的医生姓名的数量。

而在第五章实际测试的时候，对于查全率以及查准率的计算，通过多轮计算取平均值的方法，作为该推荐系统的衡量标准。

### RPC(Remote [Procedure](https://baike.sogou.com/v303132.htm) Call Protocol)服务的设计与实现

RPC(Remote [Procedure](https://baike.sogou.com/v303132.htm) Call Protocol)，是一种远程过程调用模型，它是一种通过网络从远程[计算机程序](https://baike.sogou.com/v157144.htm)上请求服务，而不需要了解底层网络技术的协议。该协议允许运行于一台计算机的程序调用另一台计算机的[子程序](https://baike.sogou.com/v7539248.htm)，而程序员无需额外地为这个交互作用编程。RPC协议假定某些传输协议的存在，如TCP或UDP，为通信程序之间携带信息数据。在OSI网络通信模型中，RPC跨越了[传输层](https://baike.sogou.com/v16962.htm)和[应用层](https://baike.sogou.com/v16954.htm)。

当前设计的RPC服务模块是将之前在模型训练阶段，训练好的模型，封装成服务，方便其他前端模块进行调用的模型模块。简要说一下，之所以用RPC调用，而不是封装成RESTful风格的API调用是因为，该模块提供的服务，在其他前端模块的多个业务API中均有使用，为了提高应用更好的效率，高效的使用推荐系统的服务，需要将其封装成RPC调用[26]。而在本文中使用是JSONRPC，主要是因为JSONRPC是基于json的跨语言远程调用协议，比XML-RPC[27]等基于文本的协议传输数据格小；相对Hessian、Java-RPC[28]等二进制协议便于调试、实现、扩展，是非常优秀的一种远程调用协议。

本文中主要是利用的Python的JSON-RPC实现，用来作为RPC服务，方便方法注册。下面分别对于RPC的服务端和RPC的客户端的设计与实现，进行描述。



JSON-RPC服务模块服务端的类图

从图4.11可以看出，

1)Helper类是一个工具类，包含一个公开方法filterDoctor，该方法主要是把“<医生姓名>”中的“医生姓名提取出来”，并将它们追加到一个结果列表里。

2)Operations类是忘JSONRPCServer里注册方法的集合，主要包括通过在模型训练阶段生成医生问题模型与KNI最近邻算法来预测相关关键词predict\_simular\_words方法和预测问题医生predict\_question\_doctor方法，和通过用户医生模型与KNI最近算法来预测用户医生的，这里是要通过输入医生名字来预测相关医生的predict\_user\_doctor方法。

3)JSONRPCServer是调用了基于Python的jsonrpc实现，并利用werzeug向外暴露接口，提供服务，主要包含一个应用注册方法application，和应用启动run方法。

调用JSONRPCServer的的客户端是利用Node.js的jayson包实现，它的构造类图如下所示：



RPC客户端类图

如图4.12所示，它这个RPClient类主要包含一个client私有属性，这个属性定义RPC服务端的请求地址和端口号，并将其初始化为一个jayson的client对象。之后包含一个返回promise对象的request请求方法，主要是对client.request方法的封装。

以请求调用predict\_doctor\_question为例，来介绍整个RPC服务模块时序图如图4.13所示。



RPC服务时序图

由图4.13可知，在RPC服务调用的过程中主要包含了如下几个步骤：

1)由RPCClient端发起远程调用，这时需要给request方法出入要调用的函数名“predict\_doctor\_question”以及它的参数“[“小儿麻痹”]”。

2)JSONRPCServer接收到远程请求，解析它的参数，知道需要调用application这个程序。

3)application这个方法在接收到JSONRPCServer解析出来的文本后，确定了需要调用“predict\_doctor\_question”这个方法，然后通过JSONRPCServer，将结果异步返回给了客户端。

### ElasticSearch搜索服务的设计与实现

ElasticSearch搜索引擎模块是属于基础服务，在ElasticSearch搜索引擎模块中，由于ElasticSearch是开源的分布式Java开发的高效搜索引擎，本文所需要解决的主要是以下两点，一是部署，二是如何同步数据，构建索引。

安装部署ElasticSearch设计如下：

由于ElasticSearch插件丰富，而且原生的版本对中文的支持并不是特别友好，因此选用ElasticSearch-rtf作为本文选用ElasticSearch的发行版本。

在安装ElasticSearch之前，确保先安装JDK，可以去Oracle官方网站内下载安装java，通过在终端中输入“java --version”来确定java是1.8以上的版本。之后可以开始安装ElasticSearch-rtf，它对汉语的分词，安装了很多插件，比如分词插件等，系统内存要在2G以上。通过git命令下载ElasticSearch的github的仓库，之后整个项目的文件被下载下来。

现在对项目文件的bin目录和config目录以及plugins目录略做说明，其中bin目录包含了该项目运行时候的执行文件，config目录中包含该项目的配置文件，plugins目录包含了第三方插件的文件。最后，进入bin目录，直接运行elasticsearch文件，之后注意观察终端窗口的提示，ElasticSearch监听在9200端口，服务正常启动。



构建ElaticSearch的索引的活动图

之后将MongoDB中数据同步到ES中并构建ES的索引[29]，这里采用开源的项目叫做“mongoosatic”，这个项目可以自动将利用mongoose构建的MongoDB模型上定义类型mapping，并自动导入到ElasticSearch，创建它的index，ElasticSearch的index相当于关系型数据库的数据库。ElasticSearch支持RESTful接口，支持GET，POST，PUT，DELETE等方法，本文中主要使用POST方法创建index，GET方法查询index，注意ElasticSearch中的索引index是相当于关系型数据库中的数据库。

如图4.14所示，在mongoose定义的MongoDB的model（模型）中，预先定义ElasticSearch的type，这个type类似于关系型数据库的表，需要确定哪些字段需要建立表，如在问诊知识卡，需要为症状，症状描述等分别建立type。之后，选择以ik作为中文分词器，在ElasticSearch中以ik[30][31]作为中文分词器的准确度和效率都非常高，因此选择ik这个分词插件，同步的可以执行活动，schema加载mongosastic插件，连接ElasticSearch服务，需要明确地址和端口，并选择混用状态。最终就可以同步构建ElasticSearch的索引。

### Web服务的设计与实现

Web服务主要是基于Nuxt.js的Koa模板实现的，下面分别介绍下具体实现，以及作者对这个模板所做的修改。

Koa是基于Node.js[32][33][34]的HTTP框架，它使用了一些Node.js的原生API，对其做了一些封装，使其变得更好用。

Web服务和功能服务

| 目录文件名 | 中文名 | 功能 |
| --- | --- | --- |
| Pages | 页面目录 | 主要包含问题检索，知识卡检索，医生推荐 |
| Plugins | 插件目录 | 主要导入一个前端的vuetify的组件库 |
| Static | 静态资源目录 | 主要存储静态资源图片，公用css，公用js文件 |
| Store | 前端存储目录 | 主要是前端数据存储 |
| Server | 服务端目录 | 利用koa提供后端API服务 |
| nuxt.config.js | 配置文件 | 指定插件配置，全局样式配置 |
| start.js | 启动文件 | 启动项目的入口文件 |

Nuxt.js是一个基于Vue.js[35]的通用框架，通过对客户端/服务端基础架构的抽象组织，基于它可以初始化新项目的基础结构代码，而且它这个解决方案存在的意义就是帮助使用解决方案的开发者便面复杂重复的工作，绝大多数时候已经帮助开发这完成了大量与业务不相关的工作，当然对使用该解决方案的开发者来说，这个框架具备了以下的要素：一是对使用的开发者透明(无需知道框架内部的具体实践就可以使用)；二是基于Node.js技术做了上层抽象封装；三是提供了配置、插件机制使得框架和具体的业务实现分离解耦。Nuxt.js正是基于以上3点，大大提高了开发者的效率。它正是符合了软件工程上模块化、低耦合、高内聚的设计，正是基于以上原因促使作者使用该框架，在具体实践中，论文采用Vuex作为前端数据管理方案，方便整个前端数据渲染的统一管理，采用Vue的开源组件，做为前端组件库，方便快速构建前端界面架构，并根据实际界面需求改写部分表单提交组件。

下面，按照Nuxt框架基于Koa生成目录做以下说明，在表4.2中，本文增加了start.js这个启动文件，并重构了server目录的内部组织结构，方便之后开发。

Server目录和功能服务

| 目录文件名 | 中文名 | 功能 |
| --- | --- | --- |
| Config | 配置目录 | 主要是配置项如数据库地址等 |
| database | 数据库模型目录 | 主要定义数据库模型的目录 |
| Routes | 路由目录 | 后端路由定义目录 |
| middlewares | 中间件目录 | 后端中间件 |
| controllers | 控制器目录 | 后端控制器目录 |
| Api | API目录 | 后端控制器具体实现的目录 |
| Decorator | 装饰器目录 | 为了实现使用decorator功能目录 |
| rpc\_client | rpc客户端 | 实现rpc客户端功能目录 |
| index.js | 启动文件 | Koa服务启动文件 |



WebServer的实现类图

着重对server目录做如表4.3中定义的功能性实现，其主要实现的功能是，在index.js文件中，本文以类的方式重写改写server启动文件的实现，它的实现类图如图4.15描述，其中Koa是底层的Web框架，负责HTTP处理请求和响应，文中将Koa实例在Server类的构造函数里实例化，接着立即执行它的中间建函数useMiddlewares方法，一次性载入database、common和router，完成整个后端Server的所需要功能的加载，其中databse负责连接数据库和读取之前在database目录中定义的模型，common负责一些常用session类中间件，而router是加载在routes目录中定义的路由文件。最后在Server类内定义公开的启动start方法，这个方法主要负责实例化Nuxt框架，加载其配置，然后利用Koa实例的监听函数，确定监听端口和地址，从而启动整个HTTP服务。

## 业务服务的设计与实现

本章节会逐一介绍系统SOA架构中主要业务服务的实现，一般在实际实践开发中主要关注业务服务的实现。

### 问诊记录搜素

问诊记录搜索功能的实现，是基于第3章中对问诊记录搜索业务的描述，主要是通过分析医护人员输入的症状名称或者症状部位以及病情描述来利用ES搜索服务和模型训练阶段关键词模型遴选出对应的问诊记录列表，当终端用户点击选择某一具体问诊记录卡的时候，向终端用户展示该问诊记录的详情。

问诊记录搜索路由表

| 路由名 | 请求方式 | 功能表述 |
| --- | --- | --- |
| /questions?term= | GET | 获取医护人员输入实现问诊检索 |
| /words?sentence= | GET | 获取医护人员输入输出相关词 |
| /question?id= | GET | 获取某一具体问诊记录详情 |

按照表4.4来设计问诊记录搜索服务的前端路由表示和功能描述。接着在图4.16中，QuestionQuery提取前端请求数据，它具体包含了以下3个公共方法包括getQuestions获取问诊列表，getWords获取关键字，getQuestion获取某个问诊详情；QuestionQueryAPI是问诊记录搜索业务实现类，包含了4个具体的业务实现方法，processQuestion处理前端展示问诊数据格式方法，question方法是获取单个问诊详情，words是获取关键字方法，MongoAndESService是数据提取类，包含了2个方法，findOneQuestion是找个某个具体问诊详情，queryQuestion是遴选出问诊列表方法；RPCClient是RPC客户端调用RPC服务端中注册的predict\_simular\_words方法。



问诊记录搜索类图



问诊记录搜索时序图

下面以医护人员输入一个疾病名称为例，来描述下问诊搜索业务的调用时序图如图4.17所示。

首先，当前医护人员访问一个获取相关专有名词的请求，该请求会通过路由QuestionQuery映射到getWords方法，该方法通过解析“/words?sentence=湿疹”，获取到query的参数，然后将其送到QuestionQueryAPI中的words方法，之后调用RPC中注册的预测相关专有名词，最后将结果通过HTTP的response响应给前端。

其次，当医护人员利用上一步中返回的相关专有名词结果去检索相关问诊记录的时候，比如医护人员选择“热疹”的时候，会访问”/questions?term=热疹”这个路径，那么，它会通过路由QuestionQuery映射到getQuestions方法，这个方法会同通过解析路由的query，获取到“热疹”，接着调用QuestionQueryAPI中的questions方法，这个方法会去调用MongoAndESService中的queryQuestion方法，这个方法调用ElasticSearch的搜索服务，通过ik分词器，检索之前在mapping中定义的数据类型，按照TF/IDF的评分公式来进行相似评分，这个评分方式有一些基本原则，包括：越多越罕见的词被匹配上，文档得分越高；文档字段越短（包含更少的词），文档得分越高；权重越高（不论是索引期还是查询期间赋予的权重值），文档得分越高的排序方式，按照相似度从高到低，依次将搜索匹配出来的问诊记录排序。

最后，当医护人员想要查看某一具体的问诊的详情的时候，通过访问之前获取问诊的ID值，这个ID是该记录的唯一标识，方便了数据的检索，通过访问QuestionQuery的getQuesion方法，去请求QuestionQueryAPI的question方法，在其中调用数据库服务中提供的findOneQuesion方法，来获取结果。

### 知识卡检索

问诊记录搜索功能的实现，是基于第3章中对知识卡检索业务的描述，主要是通过分析医护人员输入的症状名称来选择出对应的症状，当选择某个症状的时候，显示其它的详情，并获取常见的问题，及其解决方案。在表4.5中详细列出了知识卡检索服务的前端路由访问路径和功能描述。

知识卡检索路由表

| 路由名 | 请求方式 | 功能表述 |
| --- | --- | --- |
| /kds?sentence= | GET | 获取医护人员输入实现知识卡检索 |
| /kd?id= | GET | 获取某一知识卡的详情显示 |

知识卡检索服务的主要功能是为了完成表4.5中HTTP服务的编写，它是实现类图如图4.18所示。在图4.18中，KDQuery是路由类，主要映射上表中列出的两个访问路径，其中getKD方法获取用户输入，getKDOne方法获取知识卡ID，然后KDQuery类中这两个方法的进一步实现分别依赖了KDQueryAPI类KD和KDOne方法，它们分别是调用数据库和ES服务提供的queryKD方法来，完整之前提出知识卡检索和显示某一知识卡详情显示。



知识卡检索类图



知识卡检索时序图

之后以医护人员查询“湿疹”方面的相关知识卡为例子的时序图如图4.19所示，这个知识卡检索的业务主要包含以下两个业务，首先当医护人员输入“湿疹”这个症状名称的时候，发起GET请求，解析到query的参数，然后传递给知识卡查询KDQuery这个路由类的getKD这个方法，这个方法调用了KDQueryAPI这个类中KDS方法，在这个方法里，调用了MongoAndESService的queryKD方法，这个个方法里面，请求了ElasticSearch数据库中的之前同步的知识卡的索引，检索到的内容依旧安装TF/IDF评分公式进行评分，按照得分从高到低排序，然后将结果返回给前端进行展示。接着，当医护人员选择某一具体的知识卡，比如“湿疹相关”的时候，根据返回的结果的ID值，发起“/kd?id=”的请求，请求KDQuery这个路由类中获取某一知识卡详情getKDOne这个方法获取ID值，并将ID值传入KDQueryAPI中KDOne方法，其中KDOne方法调用了MongoAndESService中findKDOne方法，返回“湿疹相关”的知识卡详情知识。

### 智能医生推荐

智能医生推荐的服务的实现，是基于第3章中对医生推荐系统的描述，作者主要是通过分析问诊用户输入的症状简单描述或者当用户输入相关医生姓名的来做出出对应推荐的医生，在智能问诊系统一般只默认推荐9个医生，这9个医生的排序是按照模型训练阶段按照最近邻排序算法得到，之后当选择这9个医生中某个医生的姓名的时候，系统自动显示医生的详情，供终端用户参考。

先按照表4.6来设计相关Restful API业务的表征，在表中确定智能医生推荐服务前端请求的访问路径的访问方式，和各个访问路径的功能描述。

智能医生推荐路由表

| 路由名 | 请求方式 | 功能表述 |
| --- | --- | --- |
| /recommend | POST | 提交用户的对症状的描述，获得推荐的医生 |
| /doctor?name= | GET | 显示医生的详细信息 |

为了实现上述两个HTTP请求，实现智能医生推荐系统的类图如图4.20所示，从这个类图中，DoctorRecommend是路由类，包括后去推荐医生的getRecommend方法，和获取医生详情的getDoctor方法，之后DoctorRecommendAPI是路由类中对应方法的具体实现，其其中getRecommed方法中利用调用RPCClient去调用RPCServer中定义的获取专有名词和医生姓名的get\_phrase方法，获取通过输入医生姓名以及一些疾病的专有名词，之后分别利用医生姓名和专有名词去调用相关医生的预测方法predict\_user\_doctor和predict\_doctor\_question方法，如果上述预测方法，没有返回值为空，则在其对应路由类中分别处理这些空值。

当终端用户输入一段症状的描述或医生姓名，它的时序图如图4.21所示，当终端用户在表单内提交一段病症的描述的时候，会发起一个HTTP的POST请求，请求“/recommend”路径，DoctorRecommend路由类将其映射到对应获取推荐医生的路由方法getRecomend，然后在此方法中简析HTTP的POST请求中的Body体，获取终端用户在表单内提交的内容，然后将此内容传递该路由函数的具体实现DoctorRecommendAPI中的getRecommend方法中，此方法会调用之前在RPC模块中，JSONRPCServer中注册的3个方法，通过调用get\_phrase方法是获取专有名词或者医生姓名，当检测到提交内容中有医生姓名的时候，通过RPCClient调用predict\_user\_doctor方法，获取到推荐的医生，当检测到提交内容中有专有名词时候，调用预测医生问诊模型输出的方法predict\_doctor\_question，得到推荐的医生。最后，当终端用户点击返回的医生姓名，会调用DoctorRecommed类中getDoctor路由，并调用它的实现API类中getDoctor方法，接着请求MongoService服务中findOneDoctor，返回当前选择医生的详细信息，供前端页面展示。



智能医生推荐类图



智能医生推荐系统时序图

## 本章小结

本章主要设计和实现了整个智能问诊搜索系统的基础、应用、业务服务，包括对整个系统的总体设计，并设计了数据库基础服务的数据库，介绍了系统整体框架和实现技术，然后就主要服务模块数据预处理、模型训练、RPC服务、ES搜索服务、Web服务、问诊记录搜索、知识卡检索、智能医生推荐等进行了服务的设计和实现进行了详细描述。

# 系统测试及分析

经过前面章节的分析与设计，本章将对基于问诊记录的智能问诊系统进行系统的测试。首先搭建系统运行测试环境，然后设计实验测试系统，最后对整个系统性能进行分析和总结。

## 系统测试环境的配置与搭建

### 系统测试环境配置

智能问诊系统是以阿里云平台作为云服务提供平台，分模块按照Web服务，RPC服务，ElasticSearch搜索服务，MongoDB数据库分别部署了整个项目。

测试环境拓扑图如图5.1所示。



测试环境拓扑图

从图5.1可以看出，搭建整个智能问诊系统服务的测试环境，需要一台部署Node项目Web服务器，一台部署了Python的RPC服务器，一台数据搜索引擎的ElasticSearch服务器，部署了MongoDB的的非关系型数据库的服务器。

测试环境使用3台阿里云ECS服务器和阿里云的MongoDB数据库服务搭建。

下面分别列出各个阿里云ECS配置和MongoDB配置，其中Web服务器的软硬件配置如表5.1和表5.2所示。由于Web服务所需要的内存较大，因此在选型的时候选择内存型的ECS服务。

Web服务的ECS服务器硬件与软件配置

|  |  |
| --- | --- |
| 硬件类型 | 硬件型号或配置 |
| CPU | 4核Intel Xeon(Skylake) Platnum 8163 2.5GHz |
| 内存 | 32GB |
| 硬盘 | 20GB |
| 内网带宽 | 1.5Gbps |

Web服务器的ECS软件配置

|  |  |
| --- | --- |
| 名称 | 版本 |
| 操作系统 | Ubuntu 16.04 64位 |
| Node.js | V8.9.0 |
| Ngix | V1.13.11 |

RPC服务器的的软硬件配置则如表5.3和表5.4所示。而由于RPC服务器中包含了模型训练，因此它偏向于计算型的，因此选用ESC的计算性服务。

RPC服务的ECS服务器硬件与软件配置

|  |  |
| --- | --- |
| 硬件类型 | 硬件型号或配置 |
| CPU | 8核Intel Xeon(Skylake) Platnum 8163 2.5GHz |
| 内存 | 16GB |
| 硬盘 | 20GB |
| 内网带宽 | 2.5Gbps |
| 内网收发包 | 80万PPS |

RPC服务器的ECS软件配置

|  |  |
| --- | --- |
| 名称 | 版本 |
| 操作系统 | Ubuntu 16.04 64位 |
| Python | V3.6.0 |
| Jsonrpc | V1.2 |
| Jieba | V0.39 |

ElasticSearch服务也是偏内存型服务，因此选用ECS的内存型，它的配置如表5.5和表5.6所示。

ElasticSearch服务的ECS服务器硬件与软件配置

|  |  |
| --- | --- |
| 硬件类型 | 硬件型号或配置 |
| CPU | 4核Intel Xeon(Skylake) Platnum 8163 2.5GHz |
| 内存 | 32GB |
| 硬盘 | 20GB |
| 内网带宽 | 1.5Gbps |

ElasticSearch服务器的ECS软件配置

|  |  |
| --- | --- |
| 名称 | 版本 |
| 操作系统 | Ubuntu 16.04 64位 |
| Java | V1.8.0\_51 |
| ElasticSearch | V5.1.1 |

而对于MongoDB服务器的选择，选择阿里云提供的三节点副本集版，选择2核4G，支持最大连接数1000，支持最大IOPS达2000，存储空间选择60GB。

### 系统测试环境搭建

1)在Web服务器中Node.js和Nginx的安装选用Ubuntu的安装包管理工具apt进行安装，注意更新apt源。

2)在RPC服务器中，更新Python的版本为3.6.0，依旧是利用Ubuntu系统自带的安装包管理工具apt进行安装。

3)在ElasticSearch服务其中，需要确保当前Ubuntu系统安装git工具，如果没有安装先利用apt进行安装，之后在下载官方java版本，确定JDK版本在8以上，系统当前内存2G以上，之后利用git去下载elasticsearch-rtf的github仓库，下载之后，运行bin目录下elasticsearch文件。

4)MongoDB服务则直接利用阿里云提供的地址和端口进行连接即可。

## 系统功能测试与分析

基于问诊数据的智能问诊搜索服务的功能测试过程中，主要包括数据预处理模块测试、模型训练测试、问诊记录搜索测试、知识卡检索测试以及智能医生推荐测试。对于其他模块，由于篇幅限制，就不一一详细列出。本节将针对以上的测试工作进行系统地用例设计、测试过程设计以及最终的结果进行分析。

### 数据预处理服务功能测试与分析

数据预处理功能包含构建新的问诊记录表、输出用户医生csv表、输出医生问诊csv表实例等功能，表5.7是数据预处理管理功能测试用例表，预设条件是进行此用例时需要满足的条件，主要是指智能问诊系统的MongoDB数据库正常运行，且之前以单个问诊回合记录存储的问诊数据保存完整，测试步骤指明了测试的方法，就是通过实际运行各个脚本文件生成上述结果，通过期望结果得到最后的合并后问诊记录表，用户医生csv表，医生问诊记录csv表和实际输出结果的对比，来判断该数据预处理模块测试用例是否通过，相关功能是否满足要求。

数据预处理管理功能测试用例表

|  |  |  |
| --- | --- | --- |
| 测试用例 | 数据预处理 | |
| 用例1:  创建新问诊记录表 | 用例目的 | 测试新问诊记录表是否被正确创建。 |
| 预设条件 | MongoDB数据库运行正常，旧的问诊表和医生用户表。 |
| 测试步骤 | 在守护程序运行数据处理的主文件。 |
| 期望结果 | 当程序执行完毕，MongoDB数据库中生成新的问诊记录表。 |
| 用例2:  创建用户医生csv表 | 用例目的 | 测试用户医生csv表是否被正确创建。 |
| 预设条件 | MongoDB数据库正常运行，新的问诊记录表已经生成。 |
| 测试步骤 | 运行数据处理的生成表文件。 |
| 期望结果 | 用户，医生数据字段被正常提取。 |
| 用例3:  创建医生问题csv表 | 用例目的 | 测试医生问题csv表是否被正确创建。 |
| 预设条件 | MongoDB数据库正常运行，新的问诊记录表已经生成。 |
| 测试步骤 | 运行数据处理的生成表文件。 |
| 期望结果 | 医生的主要字段和问答内容被正常提取。 |

测试用例结果分析如下：

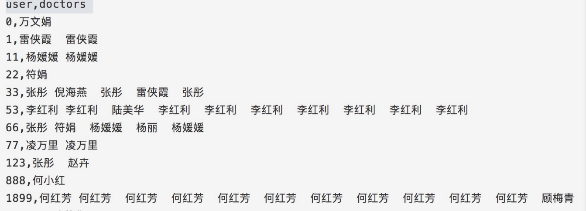
测试用例1：测试结果截图如图5.2所示，测试结果与期望结果一致，本测试用例通过，新问诊数据表正常生成。

测试用例2：测试结果与期望一直，依靠新的问诊数据表，新的用户医生csv表正常生成。测试结果截图如图5.3所示，测试结果与期望结果一致，本测试用例通过。

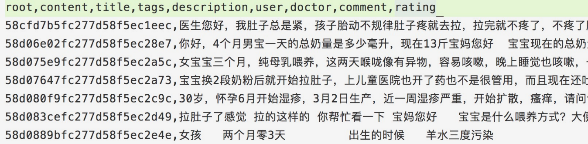
测试用例3：测试结果与期望一直，依靠新的问诊数据表，新的医生问诊csv表正常生成。测试结果截图如图5.4所示，测试结果与期望结果一致，本测试用例通过。



新的问诊记录表



用户医生csv表



医生问诊csv表

### 模块训练测试与分析

模型功能包含创建分出专有名词和用户医生表、分别构建用户医生，医生问诊模型功能，表5.8是模块训练的测试用例表，预设条件是进行此用例时需要满足的条件，测试步骤指明了测试的方法，通过期望结果和实际结果的对比，判断测试用例是否通过，功能是否满足要求。

测试用例结果分析如下：

测试用例1：检查分词表的结果，是否与正常按需求分期中格式创建，并输出。测试结果截图如图5.5和图5.6所示，测试结果与期望结果一致，本测试用例通过。

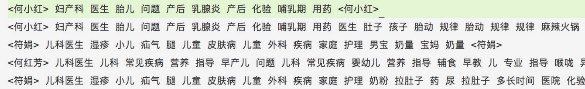
模型训练服务测试用例表

|  |  |  |
| --- | --- | --- |
| 测试用例 | 模型训练服务 | |
| 用例1:  构建分词表 | 用例目的 | 测试Word2Vec输入词表的正常生成。 |
| 预设条件 | 数据预处理阶段的csv表，分别是用户医生，医生问诊csv表。 |
| 测试步骤 | 运行buildSeg文件。 |
| 期望结果 | 输入分词后的结果表。 |
| 用例2：  模型训练 | 用例目的 | 测试用户医生，医生问诊模型是否正常生成。 |
| 预设条件 | 用户医生，医生问诊的分词表已经生成。 |
| 测试步骤 | 运行buildwWord2Vec文件。 |
| 期望结果 | 输出用户医生模型和医生问诊模型。 |
| 用例3：  用户医生模型评价 | 用例目的 | 测试用户医生模型的相似度 |
| 预设条件 | 用户医生模型。 |
| 测试步骤 | 加载用户医生模型，随机构建医生姓名作为输入。 |
| 期望结果 | 相似度达到90%以上。 |
| 用例4：  医生问诊模型评价 | 用例目的 | 测试医生问诊模型的查全率以及查准率。 |
| 预设条件 | 医生问诊模型 |
| 测试步骤 | 将20%训练集的医生标签去除，利用生成的医生问诊模型进行预测，只有相似度最高的医生与原标签进行比较。 |
| 期望结果 | 查准率以及查全率均达到70%。 |

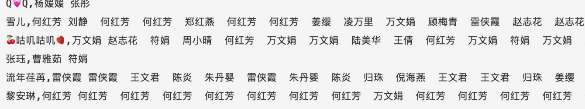
测试用例2：测试结果正常用户医生以及医生问诊均可正常加载，本测试用例通过。

测试用例3：测试结果表明的输出医生与输入医生相似度均达到97%左右，本测试用例通过。

测试用例4：通过测试结果表明该推荐模型的查准率85.18%，查全率71.21%符合该测试用例期望结果中所定的标准，本测试用例通过。



医生问诊分词表



用户医生分词表

### 问诊记录搜索功能测试与分析

问诊记录搜索功能的测试包含问诊记录搜索API的测试、问诊记录页面展示等功能，表5.9是测试用例表，预设条件是进行此用例时需要满足的条件，测试步骤指明了测试的方法，通过期望结果和实际结果的对比，判断测试用例是否通过，功能是否满足要求。

对于这部分的测试，需要额外说明下，在4章问诊记录搜索的实现中，论文中提出了以下3个HTTP请求的API, 包括“/question?term=”、“/words?sentence=”、“/question?id=”等。对于这类基于HTTP的API，本文采用Postman工具进行测试。这里需要简要说下这个工具的安装使用。首先，安装chorme浏览器，在它的插件市场中搜索postman工具，搜索到之后，点击安装插件即可。Postman工具提供的最基础的功能就是发起HTTP请求，当用户填写URL、HEADER、Body等，就可以发送一个请求，所以，当做一些简单的HTTP请求测试是可以满足需求的。

测试用例结果分析如下：

测试用例1：利用Postman发起请求得到结果与之前期望的一致，本测试用例通过。

测试用例2：测试结果如图5.7和图5.8所示，通过浏览器访问地址，得到的各个问诊记录搜索页面，与之前期望一致，本测试用例通过。

问诊记录功能测试用例表

|  |  |  |
| --- | --- | --- |
| 测试用例 | 问诊记录管理 | |
| 用例1:  问诊记录搜索API结果测试 | 用例目的 | 测试Web后端问诊记录搜索业务API返回结果是否正确。 |
| 预设条件 | 整个系统正常运行。 |
| 测试步骤 | 利用Postman输入URL进行测试。 |
| 期望结果 | 输入对应的URL得到对应的输出结果。 |
| 用例2：  问诊记录搜索界面显示 | 用例目的 | 测试问诊记录搜索的前端业务界面是否正常显示。 |
| 预设条件 | 整个系统正常运行。 |
| 测试步骤 | 在首页检索栏内输入“湿疹”。 |
| 期望结果 | 相关关键词，相关问诊，问诊详情正常展示。 |



问诊记录测试结果界面



问诊记录详情展示界面

### 知识卡检索功能测试与分析

知识卡检索搜索功能的测试包含了知识卡API的测试、知识卡页面展示等功能，表5.10是测试用例表。

测试用例结果分析如下：

测试用例1：通过Postman工具测试对应的知识卡业务API后。实际结果与期望结果符合，测试用例通过。测试结果与期望结果一致，本测试用例通过。

测试用例2：如图5.9所示，利用浏览器访问对应的地址，选择知识卡检索栏后得到页面展示与期望结果一致，本测试用例通过。

知识卡功能测试用例表

|  |  |  |
| --- | --- | --- |
| 测试用例 | 知识卡检索 | |
| 用例1:  知识卡检索API | 用例目的 | 测试知识卡检索业务API正常返回结果。 |
| 预设条件 | 系统正常运行。 |
| 测试步骤 | Postman输入对应URL。 |
| 期望结果 | 根据输入的URL得到的结果是知识卡。 |
| 用例2：  知识卡检索页面展示 | 用例目的 | 测试知识卡检索业务的界面正常展示。 |
| 预设条件 | 系统正常运行。 |
| 测试步骤 | 选中知识卡检索栏，在检索栏中输入“湿疹”。 |
| 期望结果 | 正常输出相关症状名称，以及知识卡详情。 |



知识卡检索页面展示

### 智能医生推荐功能测试与分析

智能医生功能的测试包含智能医生推荐API的测试、智能医生推荐页面展示等功能，表5.11是测试用例表，预设条件是进行此用例时需要满足的条件，测试步骤指明了测试的方法，通过期望结果和实际结果的对比，判断测试用例是否通过，功能是否满足要求。

测试用例结果分析如下：

测试用例1：通过Postman工具输出结果，发现测试结果与期望结果一致，本测试用例通过。

测试用例2：测试结果如图5.10所示，在浏览器中观测当前智能医生推荐系统，发现测试结果与期望结果一致。

测试用例3：测试结果如图5.11所示，在浏览器中观测当前智能医生推荐系统，发现测试结果与期望结果一致。

智能医生推荐功能测试用例表

|  |  |  |
| --- | --- | --- |
| 测试用例 | 智能医生推荐 | |
| 用例1:  智能医生推荐API测试 | 用例目的 | 测试智能医生推荐的业务API。 |
| 预设条件 | 整个系统正常运行。 |
| 测试步骤 | Postman工具输入URL。 |
| 期望结果 | 对应URL对应的结果与预测的结果一致。 |
| 用例2：  输入症状描述的推荐页面结果展示 | 用例目的 | 测试智能医生推荐的页面结果展示是否正确。 |
| 预设条件 | 整个系统正常运行。 |
| 测试步骤 | 在医生推荐系统表单提交框内里，先输入症状描述“我儿子拉肚子了，1周多都没好怎么办”，如果出现医生姓名，则单击选择医生姓名。 |
| 期望结果 | 正常输出推荐的医生的姓名，并显示医生的详情。 |
| 用例3：  输入医生姓名的推荐页面结果展示 | 用例目的 | 测试智能医生推荐的页面结果展示是否正确。 |
| 预设条件 | 整个系统正常运行。 |
| 测试步骤 | 在医生推荐系统表单提交框内里，先输入医生姓名，如果出现医生姓名，则单击选择医生姓名。 |
| 期望结果 | 正常输出推荐的医生的姓名，并显示医生的详情。 |



输入症状的医生推荐



输入医生姓名的医生推荐

## 系统性能测试与分析

### 性能测试用例设计

系统的性能测试内容包括CPU使用率测试以及平均响应时间等方面的测试。在测试过程中，通过使用压力测试工具对搭建的测试环境进行性能测试，并根据测试的结果对系统的性能情况进行评价和评估，具体的性能测试用例设计如表5.12所示。

性能测试用例表

|  |  |
| --- | --- |
| 测试用例名称 | 系统性能测试 |
| 测试目的 | 测试智能问诊系统各个服务器的性能。 |
| 预设条件 | 系统正常运行。 |
| 测试步骤 | 利用ab工具和top命令工具向分别向Web服务器，RPC服务器，ElasticSerach服务器发送500到1000个并发请求，分别观察系统内CPU使用率，系统内平均响应时间。 |
| 期望结果 | CPU使用率和平均响应时间在一个可接受的范围内，一般指不长时间维持在90%左右的使用率。 |

测试用例结果分析如下：

测试用例：从CPU使用率测试以及平均响应时间等各方面都在一个可以接受的范围内。测试结果与期望结果一致，本测试用例通过。

### 性能测试中CPU使用率

Web服务器、RPC服务器以及ES服务器所有服务器的平均CPU使用率的压力测试中的变化如图5.12所示。

压力测试中的各个服务器CPU使用率

从图5.12可以看出，当Web服务器采用的是Node.js作为Web服务开发的基础语言，它是基于chorme的V8引擎的，是C++编写的，因此它在处理并发发面的CPU使用率会低于基于Java开发的ElasticSearch搜索引擎和基于Python开发的JSONRPC服务，而上面所示情况的CPU使用情况，均可在正常系统运行的接收范围内。Python开发的RPC服务器，之所以初始500个链接请求的时候CPU使用率较高，是因为它主要做计算类服务，在每一次调用智能医生推荐服务的时候，都会利用KNI最近邻算法的对Word2Vec算法模型生成单词向量，进行大规模科学计算，所以该服务的CPU开销会比较大，因此在选型了阿里云提供的计算型服务器，使得RPC服务器的CPU使用率即使在1000的并发下其CPU使用率也只有75%。

### 性能测试中平均响应时间

基于Node.js的Web服务器，基于Java的ElasticSearch服务器服务器与基于Python的RPC服务器的平均对请求的响应时间在压力测试如图5.13所示。

从图5.13可以看出，请求数达到1000时，平均响应时间已经接近1秒，响应时间已经很长。但是，实际上，一般，现有根据业务系统的业务量，并发最高峰也就是处于500左右并发请求数时，所以当前各个服务器的配置是可以接受的，而且是经济的选择。

压力测试中的平均响应时间对比

## 本章小结

本章主要对基于问诊数据的智能问诊搜索系统，进行了环境搭建、和主要的数据预处理、模块训练、问诊记录搜索、知识卡检索、智能医生推荐做了功能测试，而后对Web服务器、ElasticSearch服务器、RPC服务器进行了性能测试。通过实验结果，可以发现整个系统的核心功能工作良好，在一些高并发访问的情况，能够保证各个服务模块服务的健壮性和稳定性。

# 结论和展望

## 论文工作总结

智能问诊搜索系统的之所以存在是因为现有的在线问诊系统，积累了几千万条大量的问诊记录数据。如何发掘这些数据的价值，使得医护人员和终端用户能够更有效的利用这些数据，创造出优秀的应用服务，正是此系统对这个问题的探索，作者根据实际的业务需求，先提出问诊记录检索，知识卡的检索和利用问诊记录做出一个基于文本和基于用户访问医生记录做出一个智能医生推荐系统，而各个服务的搭建是基于SOA的面向服务的软件架构，实现了低耦合、健壮性强、鲁棒性强的系统。本文的主要工作包括：

1)基础与应用服务的实现

根据之前提出总体包括以下几个模块：数据预处理服务，模型训练服务，RPC功能，ElasticSearch搜索引擎服务，Web服务。这个基础与应用服务是为了业务服务提供了它们所需的各个基本的服务，有些服务是串联起来，相互依赖的。上面基础与应用服务中，数据预处理将处理好的数据，传递给模型训练服务，当模型训练完毕，将输出模型，注册进RPC服务器。而ElasticSearch服务是和MongoDB服务相关联的，数据被同步构建生成ElasticSearch的索引，最后，Web服务统筹调用各个基础应用服务。

2)业务服务的实现

根据实际的业务需求，主要要实现以下3个业务：问诊记录搜索，知识卡检索，和智能医生推荐服务。问诊记录检索是通过获取医护人员输入的症状的专有名词，来检索相关的关键词，并通过这些关键词去检索相关的问诊记录。知识卡检索，是通过医护人员病症的专有名词，来获取这类症状的相关症状总结，进而显示症状的解决方案。最后，智能医生推荐服务，是通过获取终端用户对病症的描述，或者终端用户输入某个医生的姓名来推荐与之相关的医生。

## 后续工作展望

本文中完成了基于问诊记录的智能问诊搜索系统的设计和实现了整个系统的基础和应用服务，还有业务服务的各个子模块的实现。根据现有的技术选型实现一个比较高效的解决方案。但是，由于时间仓促、开发时间较短等原因，整个系统仍然不够完善，系统中存在着一些不足之处，对以下两个方面提出进一步的优化和改进：

1)目前的系统只是对在线问诊数据做了一些初步的探索，只是根据当前业务实际需求，提出了一个搜索问诊记录，检索知识卡和医生推荐系统。随着在线问诊数据的进一步积累，系统将拥有海量的知识，可以利用这些知识，构建知识图谱，但是，在前期的探索中，发现医疗问答的知识图谱构建繁琐，需要大量的人工干预，去构建知识3元组，它就是“实体-关系-实体”，其中，实体可以理解成医学方面的专有名词，关系，就是，它们之间是医疗关系。这种3元组，应该是可以自动化构建，但是基于目前自动化构建技术，并不能很好的，去提取出这样的知识结构。作者希望在下一步工作中能攻克这个难点，实现自动构建知识图谱，为实现智能问诊做好基础。

2)为了提高整个系统的健壮性和高可用性，一些数据的之间的调用关系，其实是基于生产者消费者模型的，比如，数据预处理阶段生成的数据被模型训练阶段所消费，正是基于这种情况，可以用Redis这个内存数据库提供的订阅模型实现系统的在分布式环境下的高可用性。

参考文献

1. 孙国强, 由丽孪, 陈思,等. 互联网+医疗模式的初步探索[J]. 中国数字医学, 2015(6):15-18.
2. 贺文. 春雨医生:回到“问诊”初心[J]. IT经理世界, 2016(17):26-27.
3. 汪祥松. 互联网医疗背景下在线问诊服务咨询量影响因素研究[D]. 西安电子科技大学, 2015.
4. 黄高明. 数据挖掘及其在医疗卫生领域中的应用[J]. 广西医学, 2006, 28(2):161-164.
5. 袁占花, 李祥生. 数据挖掘在医学信息系统中的应用[J]. 电脑开发与应用, 2009, 22(7):55-57.
6. Gormley C, Tong Z. Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine[M]. " O'Reilly Media, Inc.", 2015.
7. 张阳. 基于Lucene技术搜索引擎设计与实现[D]. 吉林大学, 2014.
8. Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions[J]. IEEE transactions on knowledge and data engineering, 2005, 17(6): 734-749.
9. Ma H, Zhou D, Liu C, et al. Recommender systems with social regularization[C] Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011: 287-296.
10. Musto C, Semeraro G, De Gemmis M, et al. Word Embedding Techniques for Content-based Recommender Systems: An Empirical Evaluation[C]//RecSys Posters. 2015.
11. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.
12. Ozsoy M G. From word embeddings to item recommendation[J]. arXiv preprint arXiv:1601.01356, 2016.
13. 谢同. 基于文本的Web图片搜索引擎的研究与实现[D]. 电子科技大学, 2007.
14. 王国霞, 刘贺平. 个性化推荐系统综述[J]. 计算机工程与应用, 2012, 48(7).
15. 唐明, 朱磊, 邹显春. 基于 Word2Vec 的一种文档向量表示[J]. 计算机科学, 2016, 43(6): 214-217.
16. 吴轲. 基于深度学习的中文自然语言处理[D]. 东南大学, 2014.
17. 张晓艳, 王挺, 陈火旺. 命名实体识别研究[J]. 计算机科学, 2005, 32(4):44-48.
18. 杨锦锋, 于秋滨, 关毅,等. 电子病历命名实体识别和实体关系抽取研究综述[J]. 自动化学报, 2014, 40(8):1537-1562.
19. 谭永明, 苏斌. 面向服务架构体系的研究[J]. 计算机技术与发展, 2007, 17(3): 132-134.
20. 方伟忠. 基于 SOA 架构的通用数据交换平台的设计与实现[J]. 计算机时代, 2008 (8): 29-32.
21. 黄嘉东, 徐兵元, 叶向阳. 企业级应用系统SOA架构建设研究与实践[J]. 中国高新技术企业, 2016(2):159-161.
22. 王帅. 面向服务体系架构软件平台应用研究[J]. 电子技术与软件工程, 2016(20):68-68.
23. 王素娟. 一种基于Spring MVC的RESTful服务设计方法:, CN 104639603 A[P]. 2015.
24. 杨勇. 构件化系统中实现调用RESTful服务的方法:, CN 106201744 A[P]. 2016.
25. 江谭森, 李升林, 黄玉标,等. 一种用于分布式系统的远程过程调用处理方法和系统:, CN106713226A[P]. 2017.
26. 冯新扬, 沈建京. REST和RPC:两种Web服务架构风格比较分析[J]. 小型微型计算机系统, 2010, 31(7):1393-1395.
27. 张华, 魏俊, 马卫东. XML-RPC在跨平台RPC中应用[J]. 科技创业月刊, 2007(7):168-169.
28. Tu J, Stewart C. Replication for Predictability in a Java RPC Framework[C]// IEEE International Conference on Autonomic Computing. IEEE, 2015:163-164.
29. 黄振文. 企业微博系统的设计与实现[J]. 企业微博, 2015.
30. 孟帮杰, 王占刚, MengBangjie,等. 两种中文分词算法在云计算平台上的实现及比较[J]. 网络安全技术与应用, 2014(12):67-67.
31. 柴洁. 基于IKAnalyzer和Lucene的地理编码中文搜索引擎的研究与实现[J]. 城市勘测, 2014(6):45-50.
32. Tilkov S, Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs[J]. IEEE Internet Computing, 2010, 14(6):80-83.
33. 王金龙, 宋斌, 丁锐. Node.js:一种新的Web应用构建技术[J]. 现代电子技术, 2015, v.38;No.437(6):70-73.
34. Teixeira P. Professional Node.js: Building Javascript Based Scalable Software[J]. 2012.
35. 李嘉, 赵凯强, 李长云. Web前端开发技术的演化与MVVM设计模式研究[J]. 电脑知识与技术, 2018(2).

致谢

时间飞逝，光阴似箭，难忘又充实的研究生生活已接近了尾声。流转的时光，褪色的过往，回首这三年的学习和生活，使我受益良多。在此，我要向曾经无私关心和帮助过我的人表达我内心真诚的感谢和由衷的谢意。

首先，我要向我的研究生导师黄健斌老师致以深深的感谢！黄老师对我的论文写作进行了耐心的指导并且给予我许多的宝贵建议，让我从中受益良多。从论文选题和开题到论文中期答辩，再到论文初稿和定稿，最后到论文最终完成，黄老师都一丝不苟地对我的论文写作进行了严格的审查，使我顺利完成了论文写作。老师严谨的治学态度、开阔的学术眼界以及对我研究课题的独特的见解，常常令我感到茅塞顿开，眼前一亮，黄老师老师为我后期论文的撰写提供了极大的帮助。

感谢我的实习公司西安图灵网络技术有限公司，感谢开发组和测试组的亲爱同事们，特别感谢王文磊、吕新桥、刘飞雁三人，感谢你们在我实习的一年时间里对我实习工作上的帮助和指导，让我的职业技能水平有了很大的提升。

感谢我舍友郝嘉伟、李国乐、李承泽和其他同班同学，你们对我生活和学习上提供了极大的帮助，永远怀念和你们在一起的快乐时光。

我还要感谢我的父母，感谢你们辛苦地将我抚养长大，并对我的生活无微不至的关心，谢谢你们支持我并尽你们最大努力地为我创造良好的学习环境和氛围。您们对我的支持是我今后在人生道路上不断前进的最大精神动力。

最后再次感谢所有关心、爱护和曾经帮助过我的人！谢谢你们！

作者简介

1. **基本情况**

李斌，男，陕西省榆林市人，1991年9月出生，西安电子科技大学软件学院软件工程专业2015级硕士研究生。

1. **教育背景**

2009.09～2013.07 西安电子科技大学，本科，专业：软件工程

2015.09～至今 西安电子科技大学，硕士研究生，专业：软件工程