第29卷 第15期 计 算 机 工 程 2003年9月
Vol.29 № 15 Computer Engineering September 2003
·基金项目论文· 文章编号：1000—3428(2003)15—0084—02 文献标识码：A 中图分类号： TP391

基于互联网的中文问答系统

张永奎，赵辄谦，白丽君，陈鑫卿
（山西大学计算机科学系，太原 030006）

摘 要：搜索引擎(如Google等)返回的是与用户查询相关的文档集，并不是所提出的问题的答案。该文提出了一个基于互联网的中文问答系统，用来增强已有的搜索引擎的功能，使它们能够支持自然语言的回答。

关键词：问答系统；搜索引擎；命名实体

Internet-based Chinese Question-answering System

ZHANG Yongkui，ZHAO Zheqian，BAI Lijun，CHEN Xinqing
(Department of Computer Science, Shanxi University, Taiyuan 030006)

【Abstract】 Web-based search engines such as Google return documents that are relevant to a user query, not answers to user questions. The authors develop a Chinese question-answering system that enhances the functions of existing search engines, so that they support natural language question answering.

【Key words】 Question-answering; Search engine; Named-entity

随着互联网的高速发展，网上的信息越来越多，如何在这些海量信息中快速准确地找到所需要的信息也越来越困难。虽然现在的搜索引擎(如Google等)已经取得了很大的成功，但是这些搜索引擎是被设计用来获取与用户查询请求相关的文档的，因此其查询序列是一系列关键词的组合，而不是以自然语言的形式提供的，同时其返回的结果是与查询相关的网页的列表，其中只有一小部分是用户需要的信息，而且用户必须自己从这些文档中找到相关的信息。事实上，用户可能更习惯于用自然语言来描述一个问题而不是用一系列的关键词，例如使用“世界最高峰是什么？”而不是“世界 AND 最高 AND 山峰”，通常情况下用户所需要的只是问题的确切答案，而不是与该问题相关的一系列网页。另一方面，传统的问答系统虽然可以对用户提出的问题给出确定的答案，但是这些问答系统的知识库是基于一个固定的文档集合，尚且不能满足用户的各种各样的需求。由于互联网信息的丰富多样性，毫无疑问可以作为问答系统知识库的理想资源，因此将问答系统与互联网结合起来，就变得非常必要。本文正是介绍了这样一个基于互联网的中文问答系统的结构和实现方法。

1 相关工作

START[1]是最早的问答系统之一，然而它的侧重点是地理知识，使用一个预先构建好的知识库MIT InfoLab来回答问题。其他的一些早期的系统如MURAX[2]，则是使用百科全书作为知识库来回答各种问题。给定一个问题，MURAX使用一个浅层的句法分析器按照百科全书章节中的词与问题中词之间的相似度来抽取潜在的答案。

近来，大量的问答系统开始出现。这些问答系统可以被粗略地分为两类：一类是使用TREC Q&A[3]数据作为测试语料，并且基于该语料构建自己的检索系统和答案抽取系统。例如Webclopedia[4]是基于信息检索（IR）和自然语言处理（NLP）技术的，给定的问题首先经过句法分析形成查询序列来获取最相关的文档，然后这些文档再被分成一个个的片断并且排序，最后从这些片断中抽取潜在的答案并排序。另一类是使用WWW作为知识库并且使用通用的搜索引擎（如Google）来获得与问题相关的信息，然后做进一步的处理来抽取出问题的答案。例如MULDER[5]利用了与传统问答系统相同的技术，首先问题经过句法分析得到句法结构，然后分成3类：nominal, numerical, temporal。最后利用句法分析抽取出答案。国内现在也有尤里卡和孙悟空智能搜索引擎提供自然语言的查询界面，但是都是基于自己的索引库，返回结果未经过处理。

本文提出的系统将问答系统的领域扩展到中文，并且使用互联网这个庞大的信息资源。初步的试验表明是有效和可行的。

2 系统的处理流程

系统的主要处理流程如下：

输入查询语句->查询类型的识别->去掉问题类型词(如“是哪里”等)->分词->查询序列的扩展->相关文档的获取->包含答案的句子的获取和排序->答案的抽取->答案的相关度计算和排序等等。

下面分别对上述流程中的主要处理过程加以解释。

3 查询类型的识别

查询类型的识别是指将输入的查询语句划分到一个指定的类型。在抽取问题的答案之前，识别该问题的语义类型可以使抽取的答案更加趋于精确，因此是重要的一个步骤。例如“中国足球队的主教练是谁？”所期望的答案是一个人名，这样在抽取答案时对不是人名的命名实体就可以简单地丢弃掉。目前按照问题的类型可以大致分成9种，分别是：PERSON，LOCATION，ORGANIZATION，DATE，

基金项目：山西省自然科学基金资助项目（991035）
作者简介：张永奎（1945—），男，教授、博导，研究方向：人工智能和中文信息处理；赵辄谦、白丽君、陈鑫卿，硕士生
收稿日期：2002-08-28

TIME，MONEY，PERCENTAGE，NUMBER，OTHER。其中大部分对应于信息抽取系统中的命名实体类别。

查询类型的识别方法有很多，例如句法分析、启发式算法、基于机器学习的算法等。使用的方法是基于模板的匹配方法，即每一种类型对应若干条模板。当前已经初步建立了170个模板。如果有多个模板与问题相匹配，选择最大匹配原则进行选配。例如，一个问题包含“多少元”，那么模板“多少元”和“多少”都和问题相匹配，在这种情况下，选择“多少元”，从而问题类型为MONEY。

表1给出了部分类型的例子。

表1 部分类型的例子

问题类型	模 板
PERSON	多少人
LOCATION	哪个城市
ORGANIZATION	什么组织
DATE	哪一年哪一月哪一天
TIME	什么时间
MONEY	多少元
PERCENTAGE	百分比是什么
NUMBER	多少
OTHER	什么意思

4 查询序列的扩展

考虑到一般用户所提出的查询请求中所包含的信息相对较少，如果直接用于搜索引擎的话会造成检索结果的召回率偏低，因此需要将查询序列扩展使其包含足够的信息。此处使用《同义词词林》[6]对查询中的词进行同义词扩展，使其在检索过程中包含足够的信息。《同义词词林》是一部语义词典，主要选取现代汉语词语。使用了与被扩展词同处于一级的词进行扩展。

5 相关文档的获取

由于问答系统是基于互联网的，因此就可以直接使用已有的搜索引擎(如Google、Excite等)进行相关文档的检索。然后将搜索引擎返回的相关文档列表中排在前面的40个文档下载下来作为检索的结果。不同的搜索引擎适用于检索不同类型的问题，例如对于关于新闻的问题的检索，使用Yahoo的新闻应该是比Google更好的选择，但是由于系统不是用来回答新闻问题，而是一般性知识的，这需要搜索引擎具有全面的索引，Google拥有世界最大的网页索引并且其用于计算结果相关度的PageRank算法具有很高的准确率，因此选择Google作为搜索引擎。查询序列提交到搜索引擎之后，将返回的列表中的前40个网页下载下来作为相关的文档。

6 包含答案的句子的获取和排序

这个处理过程的目的是减少后面答案抽取、排序时的计算量。因为直接从一篇文档中抽取出一个答案的计算量很大，同时也是不必要的。与查询相关的答案总是存在于某个句子之中，只需要将包含答案的句子抽取出来就可以了。

判断一个句子是否与查询序列相关可以通过计算该句子与查询序列的相似度来完成。相似度的计算可以使用N-Gram模型的方法，也可以使用向量空间模型的方法。我们使用的是传统的向量空间模型[7]。考虑到句子长度的影响，相似度计算使用的公式为

$$Score(s)=\sum_{req}\frac{3\times tf\times idf}{0.5+1.5\times\frac{Sentence\_Length}{Sentence\_length\_avg}+tf}$$

其中，*Sentence_Length*是该句所包含的词的个数，*Sentence_length_avg*是40个网页中所有句子包含词个数的平均值，*tf*是词的频度，*idf*是词的反比文档词频。

利用以上公式计算出40个网页中所有句子的相关度，然后将这些句子按照相关度排序，保留100个排在前面的句子作为包含答案的候选句子。

7 最终答案的抽取

这个处理过程就是从上一步得到的候选句子中找到正确的答案。这需要使用信息抽取中命名实体的抽取方法。命名实体识别可以被看作是一个分类问题，每个词被划分到某一实体类型或者不属于任何类型。使用的算法是IdentiFinder[8]，它是由Daniel.M.Bikel和RichardSchwariz提出的一种命名实体抽取算法。近年来隐马尔科夫模型在其他的分类问题中已经获得了许多成功的应用，其中最著名的就是词性标注问题。有了这些成功的使用，并且由于在局部范围之内实体的可见性，因此就可以利用隐马尔科夫模型来进行命名实体的识别问题。IdentiFinder就是基于隐马尔科夫模型的，它使用一个改进的算法，能够通过机器学习对人名、日期、时间、数字等进行正确地识别和分类。

在抽取答案时，首先用IdentiFinder对候选句子进行处理找出所有的命名实体并识别其类型，然后将这些命名实体中与查询类型相同的作为问题的候选答案。

8 答案的相关度计算和排序

抽取出的答案与问题的相关度计算使用其在所有句子中出现的分值的总和来计算，公式为

$$TRDR=\sum_{i}^{N}\frac{1}{rank_i}$$

其中*N*是句子总数，*ranki*为出现该答案的句子序号的倒数。

例如，如果总的句子数是10，该实体在第2、8、10个句子中出现，则

TRDR = 1/2 + 1/8 + 1/10 = 0.725

计算出所有候选答案与问题的相关度之后，按照相关度大小进行排序，取相关度最高的5个答案作为最终结果并以列表的形式提供给用户。

9 试验结果和分析

我们使用了包含9种类型50个问题来做试验。获取的答案的相关度在前5位的有23个，占46%，答案出现在第一位有18个，占36%。试验结果表明，本文提出的方法比一般的方法获取的答案的精确性有明显的提高。

尽管本文提出的只是一个初步的系统，但是也说明了英文问答系统处理的方法也同样适用于中文问答系统。进一步的工作包括提高查询序列的转换的效率，例如使用语义网HOWNET来进行查询序列的扩展，以使搜索引擎返回更加全面的网页。另外现在的查询类型模板是手工编写的，下一步可以采用基于机器学习的分类方法来代替等。

参考文献

1 Katz B. From Sentence Processing to Information Access on the World Wide Web. In Natural Language Processing for the World Wide Web: Papers from the 1997 AAAI Spring Symposium, 1997: 77-94

2 Kupiec J. MURAX: A Robust Linguistic Approach for Question Answering Using an On-line Encyclopedia. In Korfhage R, Rasmussen E M, Willett P, Editors. Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, 2001-06-27: 181-190

（下转封三）

如图2所示，分两个部分对其说明：(1) 包捕捉模块。这里要提到bpf过滤机制。它的改进版是一种基于寄存器设计，采用非共享缓存的工作机制，因此，它的效率很高。它包括如图2所示的网络分接头 (networking tap) 和过滤器 (networking filter)两个部分，完成从网络驱动程序拷贝数据到根据过滤条件选择是否丢弃数据包的功能。(2) 包转发模块。它利用pcap_sendpacket()函数调用可以完成向网络中发送数据包，而无须调用系统API。这里需要说明：这个函数是Windows版本的Libpcap所特有的，只能用于开发Windows下的应用程序。

3 设计与实现

该设计的关键是在网关上实现数据包捕捉、改写和转发。下面分别对这3个部分作具体分析：

(1) 包捕捉模块

由于该设计采用Libpcap函数库，因此较一般的网关有了更大的灵活性和可选择性。使得网关可以在非交换的网络结构的任何位置，甚至可以不让其它主机知道网关的具体位置（地址）。而且利用对bpf的设置，为NAT网关增加了更多的控制：可以设定只转发什么类型的包（TCP或UDP）；只转发什么地址的包(包括对源地址和目标地址的设定)；甚至对包的长度都可以作相应的限制。它的实现如下：

1) 利用函数pcap_lookupdev() 找到可用的网络设备。

2) 利用函数pcap_live_open()初始化网络设备，并返回一个pcap_t结构的指针。

3) 再利用pcap_compile()和pcap_setfilter()设置过滤器，就是上面提到的各种过滤策略的具体实现。

4) 最后，调用pcap_loop（）函数不断地获取数据包并放入缓存中供应用程序处理。

(2) 数据包改写

这里是NAT网关的核心。它有两个辅助条件：其一，必须建立一张连接状态表（如前文提到）。其二，必须重新计算改写后的数据校验和。下面作进一步分析。

1) 连接状态的数据结构

```
connect_stat {
long  ip_internal;  //内部主机地址
short  port_intenal;  //内部主机端口
long  ip_external;  //外部主机地址
short  port_external;  //外部主机端口
short  port_gw;     //修改后的网关端口
long  time ;       //定时器
char   proto ;      //协议类型 }
```

2) 校验和算法

因为修改了地址和端口，使包头发生了变化，所以必须重新计算校验和。一般的校验和算法是把被校验的数据16位进行累加，然后取反码，若数据字节长度为奇数，则数据尾部补一个字节的０以凑成偶数。算法如下：

```
long sum=0;
while (len>1) {
  sum+=*((unsigned short *)ip)++;
  if (sum&0x80000000)
    sum=(sum&0xffff)+(sum>>16);
  len-=2;  }
  if (len)
 sum+=(unsigned short)*(unsigned char*)ip;
 while (sum>>16)
   sum=(sum&0xffff)+(sum>>16);
 return(~sum);
```

当然，为了提高转发速度，可以不要全部重新计算校验和，只须对修改后的差值进行校验，即只须将发生变化的值累加进去即可。

3) 改写过程算法

```
if (数据包来自internet) {
包头的目的地址=connect_stat->ip_internal;
tcp目的端口号=connect_stat->port_internal;
}
else   //来自内部主机
{包头源地址=connect_stat->ip_gw;//网关地址
tcp源端口号=connect_stat->port_gw;//随机选择空闲独端口
}
```

其中，要用到定时器来限制每项连接状态表的生命周期，如TCP连接，在一定时间内将自动删除，用这种方式来检测连接的活动性。

(3) 数据包转发

这一步可以用系统的API调用来实现。本文讨论利用Libpcap提供的pcap_sendpacket()来实现。具体如下：

```
int pcap_sendpacket(pcap_t *p,u_char *buf,int size)
```

其中，p是用于发送数据包的网络接口，buf用于存储要发送的各种数据包，size指出buf的大小。由于pcap_sendpacket提供的功能极简单，还必须借助循环语句来提供其连续性。

4 结束语

用Libpcap函数来实现NAT网关提供了一种灵活、快速的设计方法。它不仅具有NAT 网关所具有的共享网关、包过滤、计费等功能，而且能够在此基础上集成网络嗅探器。当然由于Libpcap函数的本身复杂性设计时可能遇到各种问题，读者可查阅Libpcap函数手册。在试验性网关中主要用的是Windows下的Libpcap函数库，它的优点在设计时得到了体现。

参考文献

1 Egevang , Francis. The IP Network Address Translator(NAT). RFC1631, 1994

2 Loris Windump Manul. http:// www.tcpdump.org, 2002

3 Stevens W R. TCP/IP Illustrated(VoL 1,2). Addison-Wesley, 1995

4 McCanne S, Jacobson V. The BSD Packet Filter:A New Architecture for User Level Packet Capture. Proceedings of the 1993 Winter USENIX Conference, Sandiego,Calif,1993:259-269

☆☆☆

（上接第85页）

3 Voorhees E, Tice D. The TREC-8 Question Answering Track Evaluation. In Text Retrieval Conference TREC-8, Gaithersburg, MD, 2000

4 Hovy E, Gerber L, Hermjakob U, et al. Question Answering in Webclopedia. In NIST Special Publication 500-249; The Ninth Text Retrieval Conference(TREC9), 2000: 655-664

5 Kwok C, Etzioni O, Weld D S. Scaling Question Answering to the Web. In the Proceedings of the 10th World Wide Web Conference (WWW2001), HongKong, 2001

6 梅家驹. 同义词词林. 上海: 上海辞书出版社，1989-10

7 吴立德. 大规模文本处理. 上海: 复旦大学出版社, 1997

8 Bikel D, Schwartz R, Weischedel R. An Algorithm that Learns Whats in a Name. Machine Learinging-spcial Issue on NL Learning, 1999,34: 1-3