第六届全国人机语音通讯学术会议，135-138 页，2001 年 11 月 20-22 日，深圳

口语对话系统中动态查询组织和应答生成的设计

李芳，吴文虎，郑方，黄寅飞，苏毅

(清华大学计算机科学与技术系, 智能技术与系统国家重点实验室语音技术中心, 北京 100084)
Lifang@sp.cs.tsinghua.edu.cn

摘要

EasyNav 是一个用于清华校园导游的口语对话系统，这个系统已经在 ICSLP2000 上介绍过[1]。最近，为了解决一些原来版本无法解决的问题，如复杂句型的分析、上下文处理、友好回答等，对其中的查询和应答模块进行了重新设计，开发出 *EasyNav*2.0。这篇文章主要介绍了 *EasyNav*2.0 的动态查询组织和应答生成。动态查询框架是根据查询模块的需求设计的，这种新的查询机制不仅能处理更多的句型，而且能使查询更有效。另外，如果引入查询的优先级，则可以使应答生成模块的性能得到提高。这种新的方法已经在系统中实现并取得很好的实验效果。

1. 简介

语音识别技术进入九十年代之后得到了快速的发展，并逐渐进入了办公听写、人机对话系统的应用领域。近十年来，发达国家投入了大量的人力、物力、财力来研究人机口语对话系统，像美国 DARPA 的 Communicator 计划、欧洲的 ARISE 计划、REWARD 计划、VERBMOBIL 计划。许多著名的学府和研究机构也竞相开展这项研究，比如 MIT 的 SLS 实验室[2]、CMU 的 ISL 实验室、Lucent-Bell 实验室、日本的 ATR 实验室、德国的 Erlangen-Nuremberg 大学和 Philips 公司等。在国内，中科院自动化所[3]、清华大学、香港中文大学、台湾大学等也都投入了相当大的精力进行这方面的研究。

对话系统研究目标是让人和计算机之间能通过自然语言进行交流，从而使人能更方便有效地利用计算机这个工具去驾驭广泛的资源。以人为本是对话系统的设计目标，所以要求对话系统具备以下特点：1）自然——不需限制说话方式，让用户直接使用自然语言交流；2）智能——能对交互中出现的问题，如矛盾、含糊、偏题等进行纠正，以确保交互无歧义地顺利完成；3）友好——给予用户较大的交互自由度，并随时将系统的“理解”反馈给用户。作为口语对话系统，还需研究如何集成语音识别和语言理解的成果，以及处理口语现象。目前，已经有一些较为实用的口语对话系统作为研究人类语言的测试平台，例如 MIT 的 GALAXY I 和 GALAXY II，以它们为核心的航空售票系统、网上售车系统等已经证明了口语对话系统应用的潜能。

本文介绍的校园导游系统 *EasyNav*，是提供清华大学校园内地点信息查询服务的口语对话系统。用户可以询问特定地点信息，查询满足要求的地点，询问到特定地点的走法，或进行其它校园信息的查询。1999 年 9 月实现第一版的原型系统，只支持一问一答。该系统在 PC 上运行，提供文本输入，文本输出，并且将有关地点位置在地图上标示。允许用户询问清华大学校园内与地点有关的问题，但不限定询问的句型。目前已实现第二版，在保持原有用户主导查询的基础上，重新设计一些模块，以支持带上下文的多句对话和智能的查询与应答。

本文主要介绍 *EasyNav*2.0 中有关查询的动态组织和应答生成的设计。该系统 2.0 版本对有关查询部分做了很大的改进，整个查询过程是根据实际的用户询问进行动态组织。另外在应答方面，为了使应答更友好，尤其是针对查询失败的情况，应答生成部分做了进一步的设计，能够较为智能且友好的为用户提供一些可能有用的信息。目前该系统作为实验室的演示系统，收到了良好的效果

2. 系统设计

EasyNav 系统的结构如图 1 所示。文本输入的句子经过分词、词性标注、句法分析和语义分析模块的处理后，系统充分理解用户的意图（Intention），并将抽取出来的语义信息组织成查询框架，对地图数据库进行查询，根据查询结果生成应答输出。

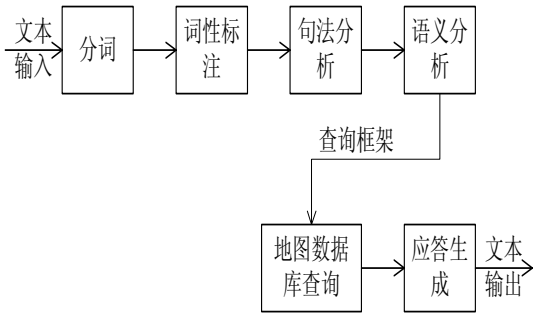

图 1　EasyNav 系统流程图

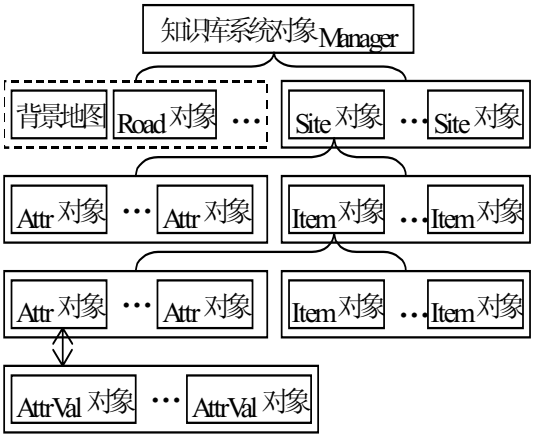

图 2　EasyNav 地图数据库的数据结构

与 GALAXY 系统的较为复杂的 client-server 架构不同[4]，*EasyNav* 原型系统采用了较为精简的单机模式，设计时把更多的注意力投入到了对汉语特殊现象的处理上。例如分词系统是针对汉语词与词之间没有分隔符而设计的，语义分析模块对处理了汉语中的习惯用语和省略句式进行了特殊处理。地图数据库是根据校园导游这个特定的实用环境而设计，采用的是与数据无关的多叉树层次结构模型的知识表示方法，这种方法具有很强的表达能力并且易于扩展。*EasyNav* 地图信息库的数据结构如图 2 所示[5]。

3. 基于动态 QueryFrame 的查询组织

3.1 概述

在一般的对话系统中，对用户提问的问句多是采取整句分析，分析过程多是以关键词为核心，然后将分析得到的围绕中心词的其它信息填在一个预定的表格（这里暂时称为静态 QueryFrame）中，然后交由查询模块进行知识库查询，并且将查询结果填充在表格中的相应位置以备以后的应答生成和记录历史信息所用。在 *EasyNav*1.0 中，就是采用这种静态 QueryFrame 方法。这种分析方法有以下几点不足：（1）对于语义分析模块来说，由于静态 QueryFrame 的局限性，导致一些复杂的句子无法分析；（2）对于查询模块来说，静态 QueryFrame 没有为知识库查询提供必要的指导，可能出现信息冗余，或者信息不完全（例如对于句子中各成分之间的关系，很难用这种方法表达出来），使基于静态 QueryFrame 的知识库查询实际上还需要做进一步的语义分析，导致整个系统各功能模块的划分有些混乱，不利于系统的维护和扩展。

为了解决上述问题，在 *EasyNav*2.0 中，设计了一种动态 QueryFrame，目标是得到与查询知识库任务联系更紧密的有关的语义信息。这种查询框架是嵌套的框架—槽结构，它是在语义分析的过程中动态生成的（我们称其为动态 QueryFrame），能够胜任复杂的句型，而且形成了一个我们称之为查询链的结构，这个查询链是从框架中抽取出来的与查询有关的一系列函数和函数之间的连接，这种结构对知识库的查询提供了强有力的指引，查询模块要做的仅仅是沿着这个查询链，对知识库进行一些原子查询，从而得到满足用户需求的信息。

3.2 例子

下面通过对例句“哪里可以买到清华出版社出版的书”进行对比分析。

在 EasyNav1.0 中，通过前面模块的处理，得到如图 3 的一个静态 QueryFrame。在这个表格中，可以看到对于查询如何进行没有任何指导，另外“清华大学出版社”是属于谁的属性就会有歧义，它可能是主语的，也可能是宾语的。这样的信息传送给查询模块，查询模块必须做进一步的语义理解。

图 3　静态 QueryFrame

句式：哪里 主语：空 动作：买 宾语：书 一般属性：清华大学出版社

基于静态 QueryFrame 方式的查询在得到图 3 的 QueryFrame 后，就需要首先扫描 Frame 中的各个槽，以确定诸如句式、主语、动作、宾语、属性等的内容以确定采取哪种预定的查询方案，遇到一个属性限制还要确定它的作用对象。尽管表面上静态 QueryFrame 中的信息包含了用户提问中所有的词语，似乎是完整的，但是它却漏掉了本应在语义分析模块中明确给出的各个词语之间的关系，这样就会使得查询模块要做进一步的语义分析，另外对于一些复杂的提问，查询就需要有特殊的处理才可以获得满足用户需求的应答，这样做，不利于模块间彼此的独立性从而影响到今后的扩展。所以，对这种较简单的 QueryFrame 必须加以改进。

在 *EasyNav*2.0 中，我们使用了动态 QueryFrame（图 4）：语义分析模块采用部分分析的方法，对输入句子的各个部分使用相应的规则生成各自的语义框架，整个句子就可以表示成为框架的序列，而各个部分之间语义上的连接关系体现在框架之间的连接上。对应不同的用户输入，产生的 QureyFrame 的框架数目和每个框架的内容也不相同，所以称之为动态 QueryFrame。图 4 中示出了例句“哪里可以买到清华出版社出版的书”经分析后所形成的动态 QueryFrame。可以看出，原输入中的词语之间的修饰关系都通过这种结构化的表示形式非常清晰的表达出来了。例如，对于“清华大学出版社”修饰“出版”这个在静态 QueryFrame 中很难判定的关系，现在是一目了然的。并且每一个需要查询数据库的地方都对应一个原子查询函数，使得查询模块不用再预先设定任何查询方案，一切查询都由输入的句子决定。从这

种意义上来说，基于动态 QueryFrame 的查询模块也是动态进行组织的。

```
哪里可以买到清华出版社出版的书
{
  [应答内容]={1}.地点名
  [应答内容]={1}.地点位置

{1
  [地点位置]=Query 地点位置( [地点名] )
  [地点名]=SelectOne( [地点名集合] )
  [地点名集合]={2}.[地点名集合]
 }

{2
  [地点名集合]=Query 地点名( [动作] ) <={3}.[设置受事]
  [动作]=买
 }

{3
  [设置受事]=Set 受事( [受事] ) <= {4}.[设置物品属性]
  [受事]=书
 }

{4
  [设置物品属性]=Set 物品属性( [动作] ) <= {5}.[设置施事]
  [动作]=出版
 }

{5
  [设置施事]=Set 施事( [施事] )
  [施事]=清华大学出版社
 }

}
```

图 4　动态 QueryFram

3.3 动态 QueryFrame 的结构

动态 QueryFrame 由若干个框架构成，每个框架又由若干个槽构成，这些槽可以是以下 3 种类型中的任意一种：（1）值槽（value-slot）；(2) 指针槽（link-slot）；(3) 函数槽（func-slot）. 值槽包含一个用户询问句子中的关键字（如图 4 中框架 5 中的槽[施事]为“清华大学出版社”）；指针槽说明此槽包含一个指向其它槽的指针（如图 4 中框架 1 的槽[地点名集合]指向框架 2 中的槽[地点名集合]），查询必须沿着指针进行进一步的推导，最终可以得到当前槽的真正值；函数槽说明此槽对应于一个数据库的原子查询（如图 4 中框架 2 的槽[地点名集合]），需要调用在此槽中引用的查询函数对知识库进行查询以求得当前槽的真正值。函数槽所对应的查询函数都是一些对校园信息库的原子查询，它是根据本系统知识库结构与实际查询请求的种类之间的对应关系设计的，这使得数据库模块从语义分析中彻底脱离开来，在查询中再也不用对当前的问句做过多的理解了。

在将动态 QueryFrame 传入查询模块时，必须同时指定一个入口框架，也即查询链的开始，从而指导查询模块沿着查询链进行查询。在实际设计查询模块时，利用动态 QueryFrame 的框架与框架之间彼此相接的特性，可以很方便的利用递归的方法从入口框架开始实现对整个查询链的遍历，在遍历的过程中自然而然完成了查询的任务。

利用这个动态 QueryFrame 的特性，对于上下文信息的查询与储存同样是非常方便的。由于框架各自的独立性，不必从顶层框架入口进行查询，从任何一个框架入口都可以形成一个独立的查询链，沿着查询链进行查询就可以得到本框架所需要的查询结果。所以当需要保存某个框架的内容作为历史纪录时，只需从相应的框架入口进行查询即可。

3.4 查询模块的设计

由于动态 QueryFrame 提供了大量查询所需要的信息以及入口框架，所以使用递归的思想对动态 QueryFrame 进行遍历是很自然的。递归函数每层的任务就是对当前的槽进行分析，对应 3 种可能的槽，分别进行相应的操作。对于 value 槽，直接把它的值返回给上层递归即可；对于 link 槽，把它指向的框架和槽作为下层递归的入口；对于 func 槽，由于查询函数有可能有参数，这些参数实际上是属于 link 类型的指针，只不过限制它们只能指向本框架内部其它的槽，即是近指针。所以利用类似的递归方法可以得到它们的真正值，之后再调用查询函数进行查询。图 4 中的<=表示本槽有先决条件的限制，实际上每个先决条件对应于一个或多个查询函数，作为一种对当前槽的限制（如最近、附近、出版社等），同样可以很方便的用递归的方法加以实现。

一般情况下，用户提问中的查询条件是作用于同一个主体的，例如：

“离这里最近的校门在哪儿”
查询条件为：（类别 = 校门） and （特殊属性 = 离这里最近），即这些限制是作用于同一主体的，但是对应的查询出现在动态 QueryFrame 的不同槽也就是不同的递归层次上。为了能够对查询作必要的优化，必须为每一次提问维护一个查询结果链，使上一次查询得到的结果作为下一次查询的搜索集，从而避免了再次搜索全部数据库。例如，图 4 中，查询的顺序是：

Query 地点名(“买”),
Set 受事(“书”),
Set 物品属性(“出版”),
Set 施事(“清华大学出版社”),

这些查询彼此之间是“与”的关系，所以可以引入查询结果链，前一个查询结果链作为下一个查询的搜索集，这样使搜索的范围随着查询的过程一步步缩小，最终得到需要的结果。

但是，确实存在用户的提问包含的查询条件是作用于不同的主体的情况，例如：

“图书馆前面那个是大礼堂吗？”
这句话中包含了两个主体之间的比较（图书馆前面的建筑物，大礼堂），对于这种情况，由于在对应的动态 QueryFrame 中的入口处有：

“[确认结果]=Query 是否匹配() <= {1}.[设置确认], {4}.[设置确认]”
的明确指示，可以对这种判断匹配的查询是用特殊的方式，保存 2 个结果链，分别对应两个主体的查找，然后再进行比较。如果存在更复杂的句型，使用类似的方法在动态 QueryFrame 中标明需要 N 个彼此独立的结果链即可。

4. 应答生成

应答生成模块的主要功能是把查询得到的信息加以组织，作为直接的文字输出或合成语音反馈给用户。在准确的应答生成基础上，回答的智能和友好也是非常重要的。一方面需要查询模块在返回结果时能够给出尽可能多的信息，这就要求不但在查询成功的时候能反馈满足用户需求的信息；而且当查询失败时，也要指出导致失败的原因，这样就能尽量友好的与用户建立交流。另一方面如果用户的需求有很多限制，就可能使查询失败（即找不到满足用户所有要求的信息），例如下面要讲的例子：“这附近有没有便宜的食堂”，如果用户所在的地方的附近有一些食堂，但都比较贵时，那么只回答“没有”和告诉用户一些附近的食堂，后者的回答显然比较好。这就是说，应答生成模块可以指导查询模块减少一些查询限制，这样就可以提供给用户一些相关信息，而这些信息可能对用户是有用的。

下面用一个例子详细说明这种方法（图 5）：
对查询框架的遍历查询后，假如查询成功，由于动态 QueryFrame 中指出了需要回答的内容，所以利用已有的信息可以很方便的进行相应提问句型的回答；假如查询失败，为了能够使得应答更加智能和友好，可以引入对于查询的优先级的设置。在查询模块中使用了结果链保存当前提问的查询结果，如果同时规定优先级：种类查询 > 一般属性查询 > 设置范（here, 附近），则当查询结果为空时，去掉优先级最低的查询：设置范围（here, [附近]），如果结果不为空（假设结果为 x, y），则生成应答：“附近没有便宜的食堂，但是远一些，八食堂、教工食堂都比较便宜”；假如结果仍然为空，则接着去掉属性查询（便宜），以此类推。通过这种方式，能够使许多原本返回为空的查询得到中间结果，这些结果可能对用户有用，再通过应答生成模块的组织，使得原本生硬的回答更加智能化，从而很大的提高了对话系统的性能。

图 5　应答生成优先级

这附近有没有便宜的食堂？

查询函数：Query 类别（[食堂]）
ResultList：八食堂，七食堂，教工食堂，西餐厅……
查询函数：Set 一般属性（[便宜]）
ResultList：八食堂，教工食堂
查询函数：Set 范围（这里，[附近]）
ResultList：NULL

5. 结论

这篇文章主要介绍了 EasyNav 系统动态查询组织和应答生成。对于用户提问的句子，经过语义分析形成动态 QueryFrame，查询模块根据抽取出的查询链对校园信息库进行查询，从而得到满足用户意图的信息。这种新的设计思想可以处理复杂的句型，完成上下文分析等问题。另外，通过引入了查询优先级思想，使得系统的应答更友好，更智能，大大提高系统的性能。目前本系统已取得很好的实验效果。

6. 参考文献

[1] HUANG Y.F., “Language Understanding Component for Chinese Dialogue System,” ICSLP’2000, pp. 1053-11056, Beijing, China, Oct.2000

[2] D. Goddeau, E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue, “GALAXY: A Human Language Interface to Online Travel Information,” Proc. *ICSLP’94*, pp.707-710, Yokohama, Japan, Sept. 1994.

[3] Chao HUANG, Peng XU, et al. “LODESTAR: A Mandarin Spoken Dialogue System for Travel Information Retrieval” EUROPEAN CONFERENCE ON SPEECH COMMUNICATION AND TECHNOLOGY, EuroSpeech, 1999, v3, p 1159-1162

[4] S. Sneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, “Galaxy-II: A Reference Architecture for Conversational System Development,” ICSLP’98, pp. 931-934, Sydney, Australia, December, 1998

[5] 燕鹏举,郑方, "口语对话系统中的词类概率模型和知识表示", 清华大学学报, p.69-72，2001 年第 41 卷第一期