口语对话管理综述*

王菁华, 钟义信, 王　枞, 刘建毅
(北京邮电大学 智能科学技术研究中心, 北京 100876)

摘　要: 主要介绍了口语对话系统中对话管理的作用、基本问题和设计方法。对话管理在整个对话系统中处于核心地位,控制整个对话的进行,负责对用户输入的理解以及根据领域内容决定系统对用户的反应。对话管理的设计主要有基于状态图的结构(有限状态机)、填充槽结构和基于任务的结构三种方法,提出了一种基于逻辑表达式的结构,并设计了状态图/逻辑表达式双层结构。

关键词: 口语对话系统; 对话管理
中图法分类号: TP315　　文献标识码: A　　文章编号: 1001-3695(2005)10-0005-04

Overview of Dialogue Management in Spoken Dialogue System

WANG Jing-hua　ZHONG Yi-xin　WANG Cong　LIU Jian-yi
(*Research Center of Intelligence Science & Technology, Beijing University of Posts & Telecommunications, Beijing* 100876, *China*)

Abstract: This paper gives an overview of dialogue management in spoken dialogue system in its function, design and other important problem. Dialogue management plays an important role in the whole system. It controls the process of the dialogue, interprets the input of the user and decides what to do next. There are three basic structures of dialogue management in dialogue system, which are graph-based, slot-filling and task-based. This paper designs the structure based on logic-expression, and designs graph /logic expression dialogue management structure.

Key words: Spoken Dialog System; Dialog Management

引言

随着社会的发展,人与计算机的交互存在于生活的方方面面。在交互的过程中,人们通常要通过键盘、鼠标等输入设备将自己的要求传达给计算机。人们在进行这些操作的时候,通常要参考说明书、帮助文档等,尤其是对于没有使用经验的人来说,这些都是一种障碍。因此,需要一种计算机与人之间的智能的接口,使人可以自然、方便地与计算机进行交流,达到上述目标的一个方法就是人与计算机采用自然语言进行交互。口语对话系统就是在人与计算机之间提供一个通信的桥梁,计算机能够自然、流利、正确地与人进行对话,其中一个最为广泛的应用就是信息查询,用户可以使用口语来查询计算机中的数据。人们从 20世纪 60年代就开始在这方面进行研究,如今已经成功地应用在了如机票查询、剧院订票等方面。

一个对话系统要将用户输入的自然语言进行理解,并根据用户提出的要求在数据库中搜索结果,最后将其转化为自然语言反馈给用户。对话系统(图 1)一般包括:

(1)语音识别(Auto Speech Recognition, ASR)。将用户输入的声音转化为文本。

(2)自然语言分析(Natural Language Understanding, NLU)。对 ASR输出的文本进行分析,建立语义表示。

(3)对话管理(Dialogue Management, DM)。它是对话系统的中心部分,根据 NLU分析出的用户语义控制整个对话过程的进行。

(4)后台数据库(Back-end)。提供查询所需的数据。

(5)自然语言生成(Nature Language Generator, NLG)。将查询结果转换为文本结构的自然语言。

(6)语音合成(Test to Speech, TTS)。将文本结构的回答合成为声音,传递给用户。

从上面的结构分析可以看出,对话管理模块的任务是从 NLU服务器接收分析出来的结果,根据保存的对话状态和对话历史判断对话流程,组织回答发送给 NLG,如果查询条件满足则组织后台数据库查询并接收查询结果。如果在对话过程中出现差错或异常,则进行差错处理使对话继续进行。

对话管理的基本问题

对话管理在对话系统中处于核心地位,其设计优秀与否关系到整个对话系统性能。对话管理的任务是控制对话流程,帮助用户高效自然地完成对话。在对话过程中,用户的回答或提问可能是含糊不清或者是不完整的,对话管理必须引导用户说明自己的意图,并提供完成任务所需要的信息。为了完成和用户的交互行为,对话管理应该根据对话历史建立对话上下文,并根据对话上下文正确理解用户输入。在此基础上,对话管理需要决定如何响应用户,并根据响应的内容修改上下文。对话管理的基本问题主要有:对话策略、领域的可移植性、系统健壮性和校验。

. 对话策略

对话系统要对用户提出的问题进行回答,所以通常需要系统的引导使对话在某一个领域内进行;当用户的回答模糊或者

收稿日期: 2004-10-27; 修返日期: 2004-12-28
基金项目: 国家"863"计划资助项目(2002AA117010-07)

缺少关键信息的时候，也需要系统的提示。这样就使得在对话的效率和用户的自由度方面存在矛盾，所以需要采用适当的对话策略使这个问题得到折中的解决。对话策略分为三种[1]：①系统主导的会话是指由系统向用户提出一系列的问题，根据用户的回答来提供信息。直接的提问（如“你要问哪个城市的天气？”）通常都可以得到用户的明确、简洁的回答，从而有着较好的查询效果，但是系统主导的会话束缚了用户，使对话过程不自然。②用户主导方式是指在对话过程中，用户是对话的主导者，可以非常自由地按照自己的意愿来提问。③用户主导和系统主导的混合使用，即系统可以提问要求用户回答，用户可以回答问题；也可以按照自己的意愿提出问题，要求系统回答。该种方式具有更大的灵活性，可以处理更加复杂的用户输入，与用户的交流更加流畅[2]，因此是较好的对话策略。

. 领域可移植性

领域可移植性是口语对话系统设计中需要着重考虑的一个问题。对话系统是在某一领域内回答用户的问题，对话系统需要具有该领域的专门数据或知识。例如电影服务，系统必须有各个电影院所放映的电影时间、电影内容介绍和电影院的介绍等数据。每个领域都要有不同的控制策略。在以往的设计中，系统的反应行为被直接代码化，然而，当要移植到新领域时，就需要完全重新设计对话管理器。一般的解决方法是将对话管理分成领域相关和领域无关两个模块。与领域相关的部分被抽取出来单独设计，将领域知识存储在配置文件，如脚本、数据库或知识库中；与领域无关的模块则被设计成通用模块，此模块可以轻易地移植到其他领域。领域相关的模块和领域无关的模块通过配置文件联系起来，通过调用不同的配置文件，可以驱动不同领域的对话系统。当需要移植到其他领域时，只需要修改配置文件即可。

. 系统健壮性与校验

用户对话的自由度和整个对话系统的回答的准确度很难达到统一。用户在口语的对话中常有省略、重复和一些感叹词等，这些都给对话系统的设计带来一些困难。其中最主要的是系统在对话中要正确地推测出用户的省略，这样就带来了信息确认的问题。系统推测出来的信息没有经过用户的确认，可信度比较低，在必要的时候就需要用户进行确认。在整个系统设计策略制定的时候这也是必要的一方面。

另外，在实际的交互过程中，用户输入的内容有的时候是有错误或输入条件之间有矛盾的，如误输入“北京时代电影城”（应为“首都时代电影城”）、“八月十三日星期六”（应为星期五）等，系统均要能识别出来，提示用户出错需要改正。

对话管理设计方法综述

一个对话管理系统要做到能够在与用户多次交互的情况下保持回答的连续性和合理性，并且能够处理用户在交互过程中转变提问目的的情况。在已经实现并应用的对话管理的设计中，主要有基于状态图的结构、填充槽结构和基于任务的结构。

. 基于状态图的结构 – 或 – –

基于状态图的结构采用有限状态机来控制对话的进行[3]。有限状态机系统可由图 2表示。其中，S表示状态的集合；X表示输入集合；Y表示输出集合；λ，μ分别表示两种映射关系，可表示为 λ：$S\times X\rightarrow S$，μ：$S\times X\rightarrow Y$。因此，λ可看作是状态迁移映射，而μ则可认为是输出映射，在时序关系上，又可表示成如下形式：

$$s(t+1)=\lambda(s(t),x(t)),\quad y(t)=\mu(s(t),x(t))$$

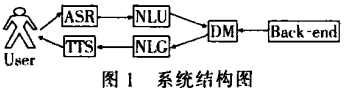

图 1 系统结构图

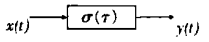

图 2 有限状态机

有限状态机的工作原理为：给定状态集 S和输入集 X，并且确定输入 $x(t)$和状态 $s(t)$，在映射 λ和 μ的作用下，有限状态机的下一状态为 $s(t+1)$，输出为 $y(t)$[4]。在对话系统中，$x(t)$为用户对系统的输入，$y(t)$为系统对用户的输出。

有限状态机也可以采用状态转移图的形式表示。每个对话片段的情况可以看成是一个一个的状态，将对话过程的每一次交互都看作是一次状态的跳转，即每一个状态节点都表示着当时对话的信息状态和系统动作，每一个连接弧表示用户的每次操作。因此，整个对话的过程，从开始到结束可以看成是在状态图中的一个连接开始节点和结束节点的状态转移的路径。

图 3是一个简单的范例，state1 ~ state 4表示对话系统存在的四个状态，action1 ~ action3是系统定义用户的全部动作。该状态转移图表示了所有状态与动作之间的关系，系统根据用户的动作和当前状态转到下一个状态，并根据每个状态的定义对用户作出反应。整个对话的过程就在状态的转移中实现。

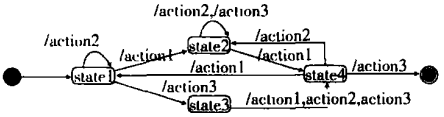

图 3 状态转移图

这种对话管理结构要求设计者要在设计时预计出所有可能的对话状态和用户可能的操作，即所有状态之间的转移条件。从工程实现的角度来讲，由于此种结构要求对于每一个状态用户的任何操作都要有一个跳转的规定，因此这种结构在对话清晰明确的时候有着很好的应用。如果领域的内容复杂则状态图很难保证没有任何的纰漏，实现起来要耗费大量的人力；并且有限状态的结构有着其必然的缺点，即难以应付没有预测到的情况，如果用户的反应完全超乎设计师的预计，则对话必然不能正常地进行，并且这个缺点在一个以用户为对话主导的互动系统中会更加突显出来。

. 填充槽结构 –

填充槽结构采用一个多维特征向量来表示对话的情况，并且在对话的过程中不断地修改向量的值[3]。特征向量通常是由从用户接收到的信息和一些状态标志组成，根据特征向量的值来决定下一步的操作。这种方法与上一种基于状态图的方法的最大区别在于：对于操作的顺序没有严格的限制，即只关心当前对话的状态信息，根据现在的状态作出反应，然后根据用户的回答或系统的反应修改特征向量。因为这种结构不考虑整个对话的顺序，所以比基于状态图的结构适应更多的对话类型。同样，这种结构也有着自己的适应范围。①与基于状态图的结构一样，也要列出所有的可能状态，即所有可能的特征向量。②由于填充槽的结构要求列出所有的槽来表示状态，所

以槽的数目要有一定的限制, 这也是对其可以实现的系统范围的一个约束; 并且由于它只记录信息存在的状态, 所以对于多提问目标的情况就难以应对。假设在电影院订票的对话系统中, 如目前的信息状态是用户提供了一个剧院的名字, 如首都时代电影城, 但是根据用户的目的——查询首都时代电影城的介绍或是查询首都时代电影城上映的电影要组织出完全不同的答案, 而只根据用户提供的名称是区别不出来的。

. 基于任务的结构

基于任务的结构是一种目前最受瞩目的结构, 并且适应的范围也最为广泛。任务是指用户为达到某种目的而采取的一系列的操作或对话[5], 如"要买两张《特洛伊》的电影票"就是一个任务。一般来讲, 任务包括进度表(Plan)和目标。目标就是用户想要达到的目的, 在上一个例子中, 买电影票就是用户的目标。通常来讲, 系统要通过一系列的步骤与用户交互才能完成特定的任务, 这些交互的步骤就构成进度表。例如上例中, 为了达到上面的任务, 系统要与用户交互确定电影的时间、要求的电影院等, 用户可能要求系统提供影片介绍, 系统要根据用户提出的新的要求不断地修改进度表, 最终完成任务。同时系统还要能够支持在对话过程中任务的突然跳转。

对于一个应用的领域, 通常采用树型结构来描述任务。在表示领域的根节点下面的第一层子节点是任务节点, 任务节点的子节点表示解决这个任务所可能用到的信息要素, 一个信息要素节点的子节点表示这个信息要素的子要素。要素之间的关系, 如"与"、"或"等, 在节点关系中体现出来[6]。若两个节点之间是"与"的关系, 则表示两个节点要同时满足才能完成这两个节点的父节点; 若节点之间是"或"的关系, 则表示两个节点只要满足一个即可。若领域包含的信息元素间的关系比较复杂, 则还会包括其他节点间关系。当用户与系统交互的时候, 系统首先要判断用户的任务是什么, 即要达到什么样的交互目的; 然后找到相对应的任务树, 将用户提供的信息填进各个信息要素的节点中。根据节点间的逻辑关系判断目前所拥有的信息量是否足够完成该任务, 如果不能, 找到缺少信息的节点, 根据节点所定义的提问方法对用户进行提问, 要求用户对该节点的信息进行补充, 即根据任务树来不断地制定修改进度表。

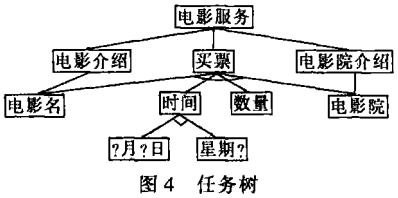

图 4　任务树

下面以电影服务领域的任务树(图 4)为例进行说明。电影服务领域共可以完成三种任务: 电影介绍、电影院介绍和买票。图 4中电影介绍、买票、电影院介绍为任务节点, 其余的电影名等为信息要素节点, 折线连接表示"或"关系, 即其中之一满足即可; 弧线连接表示"与"关系, 即必须全部满足。如果用户想要买电影票, 则系统必须得到"买票"下面的四个子节点表示的信息要素, 如果该要素不满足, 则系统就对用户提问要求补充信息。如果用户在交互过程中改变任务, 则系统根据系统设计对原来得到的信息保留或者删除, 然后根据新的任务树继续交互。

对话管理新方法的设计

基于状态图的结构、填充槽结构这两种方法都是预先设定所有可能的对话过程或状态, 虽然速度快, 但是无法应对没有预计到的情况, 灵活性差。在实际工程应用的时候, 很难预计出全部可能情况, 如果领域内容复杂、对话内容多变则实现起来需要耗费大量的人力。基于任务的结构以任务数为基础, 采用树型结构表达领域内的要素关系, 回答灵活, 是主流的设计方法。但是这种结构同时也受树型结构的限制, 因此, 本文提出了一种基于逻辑表达式(Logic expression)的结构, 对树型结构进行抽象, 不仅保留了任务树的优点, 而且使系统回答更加地灵活。由于基于状态图的结构和基于逻辑表达式的结构在性能上具有互补性, 进一步将基于状态图的结构和基于逻辑表达式的结构相结合, 采用双层结构, 充分发挥这两种对话管理机制的长处, 使整个对话系统既有灵活性又有较高的执行速度。

. 基于逻辑表达式的结构

基于任务的结构有着良好的灵活性, 是主流的对话管理方法, 基于逻辑表达式的结构是对基于任务的结构的提炼抽象。在基于任务的对话管理中, 一个领域内定义了多种任务, 每个任务所需要的信息要素用树型结构表示。而基于逻辑表达式的结构将领域内任务完成所需要的条件组合看成一系列的布尔变量组成的逻辑表达式:

$$D\begin{cases} T1=F_1(p_1, p_2, \cdots, p_m) \\ T2=F_2(p_1, p_2, \cdots, p_m) \\ T3=F_3(p_1, p_2, \cdots, p_m) \\ \vdots \qquad\qquad \vdots \\ T_n=F_n(p_1, p_2, \cdots, p_m) \end{cases}$$

其中, T_1, T_2, …, T_n 表示在该领域内定义的任务; p_1, p_2, …, p_m 是布尔变量, 其值是由其代表的信息要素是否完备决定的, 若该信息要素系统得到了, 其值为 1, 否则为 0。F_1, F_2, …, F_n 则是用∧, ∨ 等逻辑关系将 p_1, p_2, …, p_m 相连接的逻辑表达式。若 F_k($1 \leqslant k \leqslant n$)的值为 1, 即为真, 则说明系统完成该任务所需的信息要素已经全部满足, 可以给用户以回答; 若为 0, 即为假, 则说明信息要素不全, 需要用户的进一步补充。F_1, F_2, …, F_n 可以采用树型结构进行表达, 即为任务树。

逻辑表达式是对任务树的抽象, 信息要素之间的关系表达更为简洁、明确。若领域的关系复杂, 则更可以通过逻辑学的理论进行进一步的推理和化简信息元素之间的关系。在实际的工程应用中, 逻辑表达式将任务表达树这种树型的层层深入的结构表示成一种平面的结构, 我们只要判断如何得到信息可以使表达式的逻辑值为 1即可。在对用户提问的方面也比树型结构的着眼范围更加地广阔, 给用户更多的选择范围。

比如飞机服务领域中的一棵任务树如图 5所示。其转化成逻辑表达式为

$$D_{\text{飞机服务}}\begin{cases} T_{\text{订飞机票}}=(P_{\text{航班号}} \vee (P_{\text{出发地}} \wedge P_{\text{目的地}})) \wedge P_{\text{时间}} \wedge P_{\text{数量}} \\ T_{\text{航班信息}}=P_{\text{航班号}} \end{cases}$$

在任务树的模型中, 当用户给出时间、数量和出发地的时候, 根据节点之间的与关系, 系统会提问"请问你要买到哪里的飞机票?"提示用户。

而采用逻辑表达式描述关系后，将各个元素的值带入：$T_{订飞机票}(0\vee(1\wedge 0))\wedge 1\wedge 1=0$ 结果为 0说明信息不足，需要对用户提问。采用逻辑推理的方法，可以得到 $P_{航班号}$ 或 $P_{目的地}$ 为 1 即可使表达式值为 1。于是向用户提问“请问您要买到哪里的飞机票或者航班号？”，这样用户就可以选择提供航班号的方法来查询，给用户提供了更大的提示范围。在用户由于口音等问题不能明确表达某些信息(如目的地)的时候，具有重要的意义。

. 状态图 逻辑表达式( )双层管理结构

从状态机和逻辑表达式法的特点来看，这两种方法有着良好的互补性。有限状态机灵活性差，但是速度快，而逻辑表达式法则正好有良好的灵活性。因此本文采用将两种方法结合在一起的层次结构，如图 6所示。

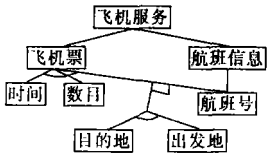

图 5 任务树

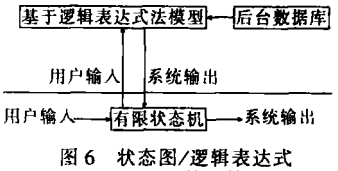

图 6 状态图/逻辑表达式(G/L)双层管理结构

在对话过程中，用户的输入可以分为两种类型：①简单地对答系统的命令表达自己的感受、心情，如“请重新输出一次”、“谢谢”、“再见”等；②提出要求进行信息的查询，如“明天的天气如何？”、“我要买两张明天到北京的飞机票”。第一种用户的输入简单、易于处理，可以系统地划分为重复、致谢、道别等几大类，可以预先定义出来。而第二种输入，用户可以将各种信息要素任意合理地组合，提出各种要求，是完全无法预计的，回答需要充分的灵活性。从对用户输入特点的分析来看，上边两种回答分别适合基于状态图的结构和基于逻辑表达式的结构进行管理。因此采用双层结构将这两种方法相结合。

有限状态机用处理低层次的对话行为控制，如回答用户的道别、致谢等行为。这些行为决策过程简单，采用有限状态机控制健壮高效。而当用户输入第二种问题，即需要对用户高层次的行为进行系统应对决策的时候，则使用基于逻辑表达式的模型，如回答用户提问，系统需要对用户提供的信息要素的状态进行判断，决定是否能够回答用户，若能，如何回答；若不能，则判断缺少什么信息，要求用户补充。

总结

对话管理在口语对话系统中处于核心的地位，其性能关系到整个系统的灵活性与健壮性。在实际的对话系统设计中，既要使系统有良好的灵活性，与用户交互自然流畅，也要使系统健壮，在多次交互的前提下仍然能够正确地判断用户的省略重复等动作。目前的会话系统一般都是采用上述设计方法之一进行的。采用基于状态图的结构处理速度快，但灵活性较差；而基于任务的设计则与之相反[7]。在基于任务的结构的基础上提出了基于逻辑表达式的结构，进一步提高了系统的灵活性，并采用状态图 逻辑表达式(G/L)双层管理结构使其与基于状态图的结构相结合，得到相辅相成的效果。

参考文献：

[1] 王显芳．汉语口语对话系统的语言处理和对话管理方法研究[D]．北京：中科院声学研究所，2002.

[2] James R Glass. Challenges for Spoken Dialogue Systems[C]. Proc of IEEE ASRU Workshop, 1999.

[3] Karl Branting, James Lester, Bradford Mott. Dialgue Management for Conversational Case based Reasoning[C]. Proceedings of the 7th European Conference on Case based Reasoning, 2004.

[4] 刘秀罗，黄柯棣，朱小波．有限状态机在 CGF行为建模中的应用[J]．系统仿真学报，2001，13(5)：663-665.

[5] Lars Bo Larsen, Tom Bøndsted, Hans Dybkjaer, *et al*. State of the art of Spoken Language Systems: A Survey[R]. Spoken Language Dialogue System Project Report 1, Aalborg.

[6] Eli Hagen, Fred Popowich. Flexible Speech Act Based Dialogue Management[C]. Hong Kong: Proceedings of the 1st SIGdial Workshop on Discourse and Dialogue, Association for Computational Linguistics, 2000.

[7] Alon Lavie, Lori Levin, Yan Qu, *et al*. Dialogue Processing in a Conversational Speech Translation System[C]. Philadelphia: Proceedings of ICSLP-96, 1996. 554-557.

作者简介：

王菁华(1981-)，女，辽宁人，硕士研究生，主要研究方向为自然语言理解、口语对话系统。

(上接第 4页)

[3] Alexis B Hombrecher. Reconciling Event Taxonomies Across Administrative Domains[D]. University of Cambridge, 2002.

[4] M Addlesee, R Curwen, S Hodges, *et al*. Implementing a Sentient Computing System[J]. IEEE Computer, 2001, 34: 50-56.

[5] David S Rosenblum, Alexander L Wolf. A Framework for Internet-Scale Event Observation and Notification[C]. Zurich: Proceedings of the 6th European Software Engineering Conference/ACM SIGSOFT 5th Symposium on the Foundations of Software Engineering, 1997.

[6] S Gatziu, K R Dittrich. SAMOS: An Active Object-Oriented Database System[J]. IEEE Quarterly Bulletin on Data Engineering: Special Issue on Active Databases, 1992, 15(1-4): 23-26.

[7] E SA-I-Shaer, H M Abdel-Wahab, K Maly. HiFi: A New Monitoring Architecture for Distributed Systems Management[C]. International Conference on Distributed Computing Systems, 1999.

[8] Mansouri-Samani, M Sloman, M GEM. A Generalised Event Monitoring Language for Distributed Systems[C]. IEE/IOP/BCS Distributed Systems Engineering, 1997. 96-108.

[9] S Chakravarthy, V Krishnaprasad, E Anwar, *et al*. Anatomy of a Composite Event Detector[R]. Technical Report UF-CIS-TR-93-039, University of Florida, E470-CSE, Gainesville, FL 32611, 1993.

[10] Peter R Pietzuch, Brian Shand, Jean Bacon. A Framework for Event Composition in Distributed Systems[C]. The 4th ACM/IFIP/USENIX International Conference on Middleware (Middleware'03), Rio de Janeiro, Brazil. LNCS 2672, 2003. 62-82.

[11] Liebig C, Cilia M, Buchmann A. Event Composition in Time-Dependent Distributed Systems[C]. Proc of the 4th Int Conf on Coop Inf Sys, 1999. 70-78.

[12] Gatziu S, Dittrich K R. Detecting Composite Events in Active Database Systems Using Petri Nets[C]. Proc of the 4th RIDE-ADS, 1994. 2-9.

作者简介：

张菊芳，硕士研究生，研究方向为网络分布式计算；魏峻，副研究员，博士，研究方向为分布式计算、软件形式化方法。