|  |  |
| --- | --- |
| 航班预订示例   |  | | --- | | 第一个[逐步示例](http://www.opendial-toolkit.net/user-manual/step-by-step-example) 介绍了OpenDial的最基本功能。在这个第二个更广泛的例子中，我们在*插槽填充应用程序*的上下文中查看了一些更高级的OpenDial功能。这个例子中的对话系统的目的是帮助用户预订挪威机场之间的机票。  [http://www.opendial-toolkit.net/_/rsrc/1437078540916/user-manual/flight-booking-example/flight-booking.png](http://www.opendial-toolkit.net/user-manual/flight-booking-example/flight-booking.png?attredirects=0)  *与航班预订系统进行互动的示例*  为系统设计的完整对话域可在文件[domains / examples / example-flightbooking.xml中找到](https://github.com/plison/opendial/blob/master/domains/examples/example-flightbooking.xml)。像往常一样，对话域的XML规范可以分为自然语言理解，对话管理和生成。在这个对话域中，这些在单独的XML文件中指定（在顶级XML文件example-flightbooking.xml中引用）。  对话状态包含与要填写的各种插槽有关的各种变量（飞行目的地和出发地，旅行日期，返回日期（如果有的话）以及要预订的票数）。对话通过一系列步骤进行，当前步骤记录在变量current\_step中。以下步骤是：   1. 询问用户的目的地 2. 要求用户出发 3. 询问用户的航班日期 4. 询问用户是否他/她想要回程 5. 如果是的话，要求返回日期 6. 检查所需航班的价格并通知用户该价格 7. 询问用户他是否想要订票 8. 询问预订旅程的门票数量 9. 要求用户确认上次提供的所有信息 10. 如果用户确认，预订所有票据 11. 询问用户是否他/她想要预订额外的门票。   **1.自然语言理解**  理解自然语言的模型用[domains / examples / example-flightbooking\_nlu.xml表示](https://github.com/plison/opendial/blob/master/domains/examples/example-flightbooking_nlu.xml)。该文件包含两个不同的模式：一个模型来识别用户对话动作a\_u从原始使用者说话u\_u，和一个模型更新四个插槽（目的地，出发，日期，ReturnDate和NbTickets）按照这个对话行为a\_u。  *对话行为识别模型*  第一个模型将用户对话行为与其基础用户话语之间的映射表示为概率分布P（a\_u | u\_u）。与每个话语相关联的用户对话行为在领域中被表示为一*组*基本对话行为（诸如“确认”或“通知（ReturnDate，March 28）”）。这种对话的表达方式充当了一系列要素，使我们能够捕捉到这样的事实：诸如“ 是，3张票 ” 这样的话语包含确认和关于票数量的新信息。为了允许对话行为包含多个元素，规则的集合运算符包括exclusive = false属性（请参见参考资料）[高级建模：操作元素集以](http://www.opendial-toolkit.net/user-manual/advanced-modelling-1#5._Manipulating_sets_of_elements)获取详细信息）。  对话行为识别模式依赖于复杂的条件，例如：    <if var="u\_u" relation="contains" value="(to|from)? {Airport}"/>   <if var="Airport" relation="in" value="[Oslo,Bergen,Trondheim,   Stavanger,Moss,Sandefjord,Tromsø,Bodø,Kristiansand,Ålesund]"/>  上述条件检查提及用户话语中的机场（在可用机场列表中）。请注意，依赖正则表达式语法来捕获可选元素或替代元素，例如（to | from）？（请参阅[高级建模：字符串匹配](http://www.opendial-toolkit.net/user-manual/advanced-modelling-1#4._String_matching)以了解详细信息）。  模型的最后一条规则是优先级较低的规则（优先级为2，而其他规则的默认优先级为1），该规则用于指定缺省对话行为，以防未找到其他值。如果用户话语是N最佳列表{ 我想飞往奥斯陆（0.7）; 我想明天飞行（0.2） }，结果对话行为a\_u将等于{ [通知（机场，奥斯陆）（0.7）; [其他]（0.2） }}。  *插槽填充模型*  第二个模型捕获用户对话行为a\_u与特定于领域的插槽Destination，Departure，Date，ReturnDate和NbTickets之间的映射。确切的填充位置取决于对话框中的当前步骤：如果当前步骤是要求用户输入目的地并且答案是奥斯陆，则变量Destination将填充值Oslo。如果当前步骤是关于出发的，则对于时间段出发将发生相同的情况。  该模型的第二个规则指出，如果最后一个系统动作是针对时隙{Slot}的特定值{Value}的确认请求，并且用户发出确认，则该时隙应该被设置为{Value}（在其他单词，它会增加{Value} 的概率并降低替代假设的概率）。第三条规则在系统发出接地动作后将给定的插槽值标记为确定。最后，第四条规则在没有附加信息的情况下转移槽的当前分配。  请注意，槽填充模型描述了分布P（槽| a\_u）。另一种建模方法是对传统POMDP方法的“用户行为模型”中经常进行的逆分布P（a\_u | slots）建模。然而，后一种方法需要明确地事先分配时隙的所有可能值，这在我们的领域中很难表达，因为航班的可能日期的数量在理论上是无限的。  **2.对话管理**  对话管理分为三种模式，如[domains / examples / example-flightbooking\_dm.xml中所示](https://github.com/plison/opendial/blob/master/domains/examples/example-flightbooking_dm.xml)：   * 给定最后一个用户对话行为的动作选择模型，以找到最佳系统动作，要填充的时隙的值以及交互中的当前步骤。 * 一个转换模型，指定所选动作如何修改当前对话状态（尤其是当前步骤） * 预测目的地分布，离开和下一个用户对话行为的预测模型。   *行动选择模型*  此模型指定了各种系统操作的实用程序。在对话期间的每一步，该系统有4个可供选择的选择：   * 为当前插槽找到最可能的答案并继续下一步 * 针对插槽的特定值发出确认请求 * 要求用户重复 * 什么也不做。   将槽S的特定值X接地的初始效用设置为-4.5，如果槽S确实等于该值X，则加上5 。换句话说，这意味着为了接地给定答案，时隙值的概率至少应为0.9（自-4.5 + 5 \* 0.9 = 0），否则接地动作将具有负面效用。  确认请求的初始效用设置为-0.3加0.5，如果该插槽确实等于请求确认的值。换句话说，时隙值的概率至少需要0.6，否则确认将会产生负面效用。  最后，要求用户重复可以在所有情况下完成，前提是用户话语不等于None值。  *转换和预测模型*  转换模型指定了特定系统操作的选择如何影响当前对话状态，特别是对话中的当前步骤。例如，如果目的地时隙正在接地，则系统可以移动到下一个时隙，即离开。  预测模型表示出发和目的地的先前可能性，以及给定上下文的下一个用户响应的先前可能性。这些预测允许系统增加用户对话行为发生的可能性，并降低不太可能响应的可能性。在这个例子中，先验分布是手工确定的，但也可以根据数据估计（参见[参数估计](http://www.opendial-toolkit.net/user-manual/parameter-estimation)的相应部分）。  **3.自然语言生成**  最后，[domains / examples / example-flightbooking\_nlg.xml中](https://github.com/plison/opendial/blob/master/domains/examples/example-flightbooking_nlg.xml)的生成模型指定当前对话状态与实际系统响应之间的映射。系统可以在当前步骤发生变化或选择新系统动作时产生新的系统话语。  请注意，许多生成规则依赖于模板来填充生成的响应的某些部分。例如，下面的规则情况描述如果当前对话步骤是LastConfirm并且书籍旅程是单个单向票证，则可以基于变量Departure，Destination，Date和TotalCost中的值构建系统响应：  <case>   <condition>     <if var="current\_step" value="LastConfirm" />     <if var="ReturnDate" value="NoReturn" />     <if var="NbTickets" value="1"/>   </condition>   <effect util="1">     <set var="u\_m" value="You are ordering one one-way ticket from  {Departure} to {Destination} on {Date} for a total cost of  {TotalCost} EUR. Shall I confirm your order?" />   </effect> </case>  （请参阅[高级建模：](http://www.opendial-toolkit.net/user-manual/advanced-modelling-1#1._References)有关此类参考资料的更多详细信息）。  还值得注意的是，生成的话语可以根据上下文进行调整。例如，交流行为a\_m = AskRepeat取决于当前的对话步骤以及在前一回合中是否已经执行了AskRepeat动作，具有不同的实现。  **4.外部模块**  航班预订系统需要访问外部数据库以检查特定航线的价格并执行机票预订。模块 [src / opendial / modules / examples / FlightBookingExample.java](https://github.com/plison/opendial/blob/master/src/opendial/modules/examples/FlightBookingExample.java) 提供了这样一个连接的一个简单伪造的例子。模块监控对话状态，并在更新系统操作a\_m时触发。当系统操作等于FindOffer时，模块检查目的地，出发地和日期的填充值，并通过在对话状态中添加新的行动MakeOffer（价格）来返回相关的价格。  同样，如果系统操作是预订，系统可以检查目的地，出发日期和票数的填充值，并执行预订。 | |