# 一个基于文本输入的口语对话系统的新的实现策略

刘智博[1] Michael Brasser[2] 郑 方[1] 徐明星[1]
(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 北京 100084)[1]
(北京得意音通技术有限公司)[2]

摘 要 口语对话系统(Spoken Dialogue Systems, SDS)现在已经越来越多地应用于实际生活之中,然而,当前对于普通人来说要开发处理英文对话的功能强大的系统通常很困难,而一些被人们提出的、不很复杂的,且能够比较容易开发SDS的方法其理解能力则受到了限制。在介绍并比较了以上所提到的一些方法之后,一个基于文本的自然语言的SDS工具包(被称为SDS Lite)将会被详细地介绍。我们的系统处理中文对话并使不仅是专家而且是普通人都很容易学习和开发他们自己的SDS。

关键词 口语对话系统,SDS Lite,上下文无关增强文法,语义抽取

## A New Implementation Approach of Grammar Generation Tool for Text based SDS

LIU Zhi Bo[1] Michael Brasser[2] ZHENG Fang[1] XU Ming Xing[1]
(Center for Speech Technology, State Key Laboratory of Intelligence Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing 100084)[1]
(Beijing dEar Technologies Co, Ltd.)[2]

**Abstract** Spoken Dialogue Systems (SDS) are now more and more used in real life. However, current powerful systems for dealing with English dialogue are usually difficult for lay people to create, while some less complicated methods proposed to create SDS easily are with less powerful understanding capability. After introducing and comparing the approaches mentioned above, a natural language text interface SDS toolkit, named SDS Lite, is described in detail. Our system deals with Chinese dialogue and makes it very easy for not only experts, but also lay people to learn and create their own SDS.

**Keywords** Spoken dialogue systems, SDS lite, Enhanced context free grammar, Semantic extraction

## 1 引言

口语对话系统(Spoken Dialogue Systems, SDS)现在已经越来越多地应用于实际生活之中,尤其是在受限领域的信息查询中。现在已经有了一些由CMU, MIT和其他大学与组织开发的很成熟很强大的SDS[1,2]。总体上讲,他们的系统处理英文,而我们的系统,被称为SDS Lite,处理中文。

基于SDS广泛的使用性,我们认为让普通人能够开发他们自己的系统是非常重要的。总体上,与由专家开发的系统比起来这样的系统必定会受到更多的限制。我们的系统就是这样一个系统,其目标很简单:开发一个有用的SDS工具包,它可以被对编程、自然语言理解或语音处理知之甚少的非专家人士所使用。必须澄清的一点是,SDS Lite是一个SDS的开发包,而不是具体的SDS应用,这种关系就如同Microsoft Visual C++ 6.0是一个C++程序的开发工具,其目的是开发出不同的C++应用一样。

对于一个SDS来讲,其核心是语法分析模块,因此SDS Lite要开发的SDS也必须具备很强的语法分析能力。在SDS Lite中,其语法生成工具被称为GrammarTool,专门用来生成SDS中的语法分析模块。GrammarTool是SDS Lite的非常重要的组织部分,具有很强的在受限领域中制取语法规则的能力。在这样的领域中通常有很多不同的和独特的表达方式(与独特的语法规则和关键词相联系)频繁地出现。因此为了开发一个SDS,提前提取出领域相关的语法规则和关键词(假定这是一个基于知识规则的系统)或拥有一个庞大的语料库(假定这是一个基于统计的系统)是很重要的。在这篇论文中,由于为新的领域采集语料很困难,因此假定使用基于知识规则的系统。

GrammarTool基于增强型上下文无关文法[3]的一个子集,它现在只支持这个文法的5种规则类型(苛刻型,跳跃型,无序型,长程型,交叉型)中最常用的跳跃型规则。其他规则类型,例如无序型规则,将在未来的工作中进行考虑。

## 2 相关工作

出于类似的目的,一些类似的工具包已经被开发出来了。下面将会介绍其中的两个,Phoenix和Application Generation工具包。

Phoenix分析器可以被用来开发简单但鲁棒的SDS应用。其基本的语义单位是帧,Phoenix将每个输入语句解析成一些帧。开发者必须为每个应用手动定义所使用的帧及语

刘智博 硕士研究生,主要研究兴趣为语言理解、应答生成、对话管理等;**Michael Brasser** 美国南加州大学计算机系硕士,主要研究兴趣为语言理解、分词等;郑 方 教授,研究生导师,主要研究兴趣为语言识别、语言理解、声纹识别等;徐明星 副教授,研究生导师,主要研究兴趣为声学模型建模、拒识、语言理解等。

法规则，再将包含语法规则的文件编译，利用编译后生成的文件来配置成一个 SDS。

在介绍 Application Generation 工具包之前，先简单介绍一下统一语音接口（Universal Speech Interface，USI）项目[4, 5]。在开发 SDS 的时候，由于输入输出过程以及开发过程没有标准化，以至于开发过程中的经验及数据得不到充分的共享，从而每开发一个 SDS 都非常费时、费力。在这方面，大家一致认为影响 SDS 广泛应用的一个非常关键的问题就是移植性问题。基于这个问题，CMU 的一些学者就提出一种将用户与 SDS 交互时的输入输出进行标准化的方法，即统一语音接口（Universal Speech Interface，USI），它致力于通过使用半结构化的交互方式来创建并测试感官上很统一的语音接口。USI 的优点包括：在使用上，用户的使用技巧与经验具有可传递性；半结构化的交互方式提高了语音识别的精确度并减少信息收集量；当给定了一些规格说明之后，可以以 USI 为标准，进行新的 SDS 的自动生成。

可以看出 USI 还只是强调输入输出的标准化，这些学者在 USI 的标准上，又对 SDS 的开发过程进行了标准化的工作，这就是 Application Generation 工具包，它的输入输出是满足 USI 标准的，同时又从其使用者，即 SDS 开发者的角度对 SDS 开发过程进行规范与简化，最终让 SDS 开发者能够在网页上对自己要开发的 SDS 的各种特性进行定制，然后生成一个 XML 文件，并基于 USI 的标准生成一个 SDS。它允许对编程了解不多的普通人在短时间内用一个特定的挑选出来的数据库来创建并使用功能非常强大的 SDS 应用。

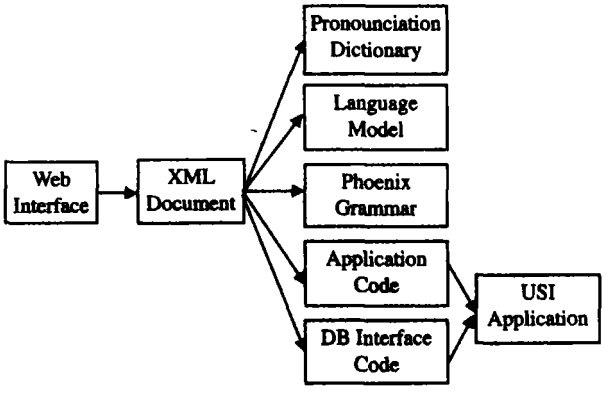


图 1　Application Generation 工具包应用产生过程

Application Generation 工具包使人们更容易地开发出他们自己的 SDS。然而开发者只能在网页上输入一些数据库表的列名及其他短语。因此，为了提高可用性，此工具包的能力受到了限制。

SDS Lite 提供了一个折衷的解决方案：对于开发者来说学习和使用我们的工具包没有 Phoenix 那样复杂，而同时又比 Application Generation 工具包功能强大（也因此稍微有些复杂）。

## 3　SDS Lite 的总体结构

SDS Lite 由四部分组成：KeywordTool、GrammarTool、服务器端、客户端。SDS Lite 的使用者，即 SDS 的开发者，需要定制一些完成查询操作所必需的信息，包括关键词和语法的定制，这两项定制工作可分别通过 SDS Lite 中的 KeywordTool 和 GrammarTool 来完成。

使用 KeywordTool 定制关键词的目的在于将查询所在领域的领域相关的关键词分成几组，并分别用抽象的组名来代表，同时一个关键词还可以添加多个同义词。比如以 NBA 查询为例，在此领域内，经常出现的关键词包括球员名，球队名等，则可以考虑添加两类新的关键词 player 和 team。在 player 中加入尽可能全的 NBA 著名球员的名字，在 team 中也要把所有 NBA 的参赛队的队名作为关键词加进去。当然对于同一个球员或球队可能会有不同的表述方式，可以考虑将一种常用的说法作为关键词，其他的表达方式作为其同义词。在添加关键词时，KeywordTool 支持从文本文件和各种不同类型的数据库中批量导入关键词。

使用 GrammarTool 定制语法用于指定查询句子的句子结构，由一组或多组句子组成，每组句子表示一类查询，同一组中的各个句子为表示同样的查询的多种不同的表达方式。关于 GrammarTool 的结构和具体的使用方法，后面会详细地加以介绍。

在使用 KeywordTool 和 GrammarTool 这两个工具定制好了需要的信息之后，就可以把生成的配置文件放在服务器端的特定路径下，并由远端的客户端根据服务器端的 IP 地址和端口进行访问，并完成查询操作。用户通过不同的网络向客户端发出具体的查询语句，客户端负责向服务器查询请求并从服务器取得查询结果，再将结果整理成适应的形式和内容反馈给用户。本系统可处理中英文混合输入，多字、漏字、带错别字的中文输入，对于拼音和拼音首字母的输入还可进行智能匹配。软件系统以 service 的方式提供 http 服务，可开机自动启动；每次交互服务可在不到 1 秒内完成；系统日志功能可自动记录所有交互情况。软件附带的工具可用于系统配置文件的生成、服务功能的配置和系统服务的管理，使用方便，无需编程。

由于客户端和服务器之间采用了 SOAP 接口，因此服务程序和客户端程序可以在同一台计算机中，也可以在不同的网络节点上。实际服务系统也可能采取多个客户端，分别处理不同的用户群。客户端与服务器端构成的服务的系统结构如图 2 所示。

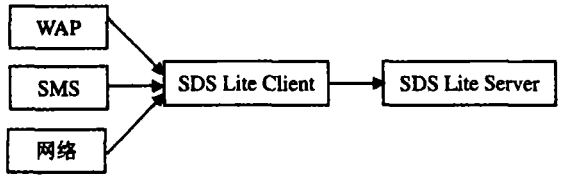


图 2　客户端与服务器端构成的服务的系统结构

上面曾经提到，语法分析模块是 SDS 的核心，下面将详细介绍 SDS Lite 的语法生成工具 GrammarTool。

## 4　GrammarTool 的结构

这里将“主题”定义如下：

主题是一个集合的名字，这个集合包括了一些可能的输入模式（或称为“伪语句”），并且按这些模式填入的语句将使系统给出同样的回答。例如：

[player_stats]

我想了解一下〈player〉的统计资料

请说明一下〈player〉的统计结果

〈player〉的统计情况有吗

这 3 个中文语句模式表达了类似的意思（它们都在询问一个 NBA 球员的统计信息），它们组成了一个主题，称为

“player_stats”。

一个领域内通常有多个主题。以上面的“NBA 信息”领域为例，相关的主题可能包括某个球队或球员最近的新闻，两队最近一场比赛的结果等等。每一个主题中要想表达同样的意思，通常有好几种表达文法。以查询一个球员的统计信息为例，用户可能会问：

我想了解一下〈player〉的统计资料

请说明一下〈Player〉的统计结果

注意，有些关键词被认为是该领域内的基础知识。我们称这些词为“用户定义的关键词”，它们是由手工来收集的（通常是从一个已经存在的数据库中导入）。在这个例子中，“〈player〉”表示 NBA 中一个球员的名字，比如“Karl Malone”或者“姚明”。这些词是提前在 SDS Lite 的 KeywordTool 中定义的。

具有相似含义的语句可以放在同一个主题中。对每个主题来说，语法规则和额外的“起帮助作用的”关键词被提取出来。每个语法规则属于其所在的主题，而关键词则全都属于这个领域。

此外，系统还提供了一个文件，它包括许多常用词，称为“预定义关键词”。GrammarTool 所自动抽取出来的“起帮助作用的”关键词是不在上面所提到的两类关键词之中的词。

## 5 使用 GrammarTool

要使用 GrammarTool，使用者需要给每个主题命名。然后这个类名之后应该是一系列句子，且满足当使用这个系统进行查询时这些句子会产生相同的应答。再次以 NBA 信息查询为例，可以定义以下内容：

[player_stats]

我想了解一下〈player〉的统计资料

请说明一下〈player〉的统计结果

〈player〉的统计情况有吗

[team_stats]

能介绍一下〈team〉吗

我想了解〈team〉的情况

〈team〉的介绍有吗

[team_news]

〈team〉

我要看〈team〉的新闻

有〈team〉的消息吗

与“〈player〉”类似，“〈team〉”代表着 NBA 球队的名字，并且所有这样的名字都应该用 KeywordTool 提前收集好。“player_stats”，“team_stats”和“team_news”是主题名，每个主题名后面都跟着 5、6 个特定主题下的伪语句。

根据上面的描述，用户应该遵循如下步骤：

（1）创建一个新的工程来代表一个新的领域，并用 KeywordTool 创建一个关键词文件，其中包含了这个领域内的基本知识，如 NBA 球员和球队的名字。

（2）定义各个主题，并为每个主题提供一些训练用的例句。

GrammarTool 会处理所有的主题名及语句然后提取并保存领域相关的语法规则及关键词。在这个训练过程之后，SDS Lite 的语法分析器就可以分析并理解这个领域内的大部分查询语句了。

SDS Lite 平台并不是一个全能的对话管理器。它不能在所有条件下被用于所有场合。要使用这个系统，需要明确以下几个假设：

（1）查询只是在一个领域内的，即每个系统只是领域相关的。

（2）当前情况下查询只能是单回合的。当开发者输入了足够的主题和例句，然后训练过程结束后，当用户使用 SDS Lite 进行查询的时候，查询结果会立即返回给用户，或者完全没有结果返回。GrammarTool 还不能进行多回合交互，这个问题在将来的工作中会得到考虑。

（3）系统是由用户主导的，即系统不会主动与用户展开交流。

SDS Lite 不能被用于所有地方，主要的原因是它目前只支持单回合交互而且不能记住在当前对话之前用户所提供的信息。此外，对查询语句的语法和语义上的支持也不像复杂的 SDS 那样强大。然后，GrammarTool 可以只使用有限的一些训练语句就很好地处理单回合领域相关的查询。

因此这个系统应当被用于相对简单的查询情况之下，如歌曲与歌手的查询，新闻、股票、天气信息。然而，GrammarTool 不适用于相对复杂的查询。例如，预订车票、旅馆，或其他应该应用多回合交互的领域。

## 6 GrammarTool 的算法设计

GrammarTool 的主要功能是从样例语句中抽取语法规则。与 Application Generation 工具包中在网页上输入简单的单字词不同，使用 GrammarTool 的开发者应该遵循上面提供的指导来输入训练语句并定义领域中的主题。当然，与 Phoenix 中手工书写规则相比，使用 GrammarTool 要简单得多。

在解释 GrammarTool 的工作原理之前，首先需要介绍一下整个 SDS 系统中三种不同类型的关键词：

（1）预定义关键词：这些是人们常用的各种词汇、词组及其同义词，如“你好”，“我想问一下”，“非常”，“什么时候”，等等，它们已经被收集好并存储在一个文本文件中，现在共收集了 134 类预定义关键词，共约 500 词。

（2）用户定义关键词：这些就是用户在 KeywordTool 中定义的关键词。

（3）自动抽取关键词：这些是由 GrammarTool 在训练时从训练语句中抽取出来的关键词，GrammarTool 也会将其存储起来在该特定领域内使用。

在 GrammarTool 中，一个语句的“语义”包括：

（1）领域中的一个主题

（2）槽及其内容。槽代表着关键词类而槽的内容则是关键词本身。

要想知道语义的这两个方面是如何被抽取的，对语句如何被分组进行解释是很重要的。

同一个主题中的语句将被放在一个组中，然后所有在预定义关键词表中出现的词将被特定的预定义关键词类的名字所代替。例如，如果预定义关键词文件中包括下面的关键词类：

[these]

这些

则任何语句中的“这些”这个词将被“〈these〉”所代替。在这个替换以及类似的替换之后，具有相同结构的语句将被放在同一个子组中。对于具有“相同结构”的语句，即它们包

含完全一样的用户定义和预定义关键词，一样的顺序，一样的位置，例如：

xxx〈关键词 A〉xxx〈关键词 B〉xxx

这里“xxx”代表一个或多个既非用户定义也非预定义的词。如果几个语句有相同的结构，它们将被一起放在一个子组中并且 GrammarTool 会将未定义的部分抽取出来（包括两个关键词之间，从句子开头到一个关键词或从一个关键词到句子结尾，这些部分可能包含一个或多个字）并将它们放进一个新的已被命名的关键词类中，称为“生成关键词类”。例如，假设同一个主题中有以下两个语句：

语句 1：我想问一下〈player〉的消息

语句 2：我想知道一下〈player〉的新闻

并假设有如下的预定义关键词：

[wen_yi_xia]

我想问一下

我想知道一下

[de]

的

经过预定义关键词替换后，两个句子将变成：

〈wen_yi_xia〉〈player〉〈de〉消息

〈wen_yi_xia〉〈player〉〈de〉新闻

显然，这两个句子具有相同的结构并将被放入同一个子组中。然后 GrammarTool 将生成一个新的关键词类：

[categoryM_keyword_N]

消息

新闻

这里 M 和 N 是非负整数。

最后，这个子组将生成一条语法规则：

〈wen_yi_xia〉〈player〉〈de〉〈categoryM_keyword_N〉

## 7 测试及结果

起初，我们的测试数据包括一些语气助词如“啊”，这些词在口语中应用非常普遍，但用户不太可能将它们输入到计算机中。因此在包含了一些这样词的领域 NUM_AGE 上进行了一些初始测试之后，我们决定手工删除这些词，下面所列出的也是这样做之后的测试结果。

测试的步骤如下：

（1）确定某一特定领域，并选取该领域内的一些语句作为数据集；

（2）使用 KeywordTool 从数据集的语句中收集领域相关的关键词；

（3）从数据集中选取一部分语句，将其转化成伪语句后，作为训练集，其他语句作为测试集；

（4）在服务器端配置好 SDS 应用程序，并在客户端输入测试语句进行测试。如果服务器中的语法分析器利用 GrammarTool 在训练后所产生的语法规则能够解析测试语句的语法结构，它会将一个成功的信号反馈给客户端，从而我们得知这次测试成功。经过多次这样的测试，我们可以得到一个准确率。

三个领域及相关的语句被选择出来进行训练和测试。

领域 NUM_AGE，有一个主题[current_age]和 30 条语句，询问一个的当前年龄；

领域 TIME_DAY，有一个主题[which_day]和 30 条语句，询问日期；

领域 NUM_SPEED，有一个主题[speed]和 50 条语句，询问一个人或物体的运动速度。

对每个领域来说，使用了 3 种方法来挑选训练语句：

（1）使用 5 条语句作为训练语句，剩下的所有语句用来测试；

（2）使用 10 条语句作为训练语句，剩下的所有语句用来测试；

（3）使用一半语句作为训练语句，剩下的另一半语句用来测试。

当从语句集中挑选语句的时候，使用了两种策略。以包含 30 条语句的领域 NUM_AGE 为例，要从中挑选 5 条语句的话：

（1）顺序法。每次挑选序号为 5i+1，5i+2，5i+3，5i+4，5i+5（0<=i<=9）的语句作为测试集，将剩下的 25 句作为测试集，因此会产生 6 个测试结果。

（2）人工法。人工挑选 5 条“有代表性”的语句来尽可能多地覆盖到其他语句。这时将产生 1 个测试结果。

我们相信“人工法”是对 GrammarTool 在实际中应用情况的一个相对真实的模拟，因为当输入测试语句的时候，开发者会倾向于想出不同的表达方式，而我们语句集中相近的句子有着类似的结构，因此在使用“顺序法”的时候也就被一起选进了训练集。根据上文的解释可以看出，在同一个主题下这样的句子不会产生出多样性的语法规则，而这削弱了我们系统的处理能力。

测试结果如表 1～3。

表 1 领域 NUM_AGE 的测试结果

| NUM_AGE | | 测试集大小 | | |
|---|---|---|---|---|
| | | 5 | 10 | 一半 |
| 顺序法 | 最大值 | 60.0% | 70.0% | 66.7% |
| | 最小值 | 4.0% | 15.0% | 26.7% |
| | 平均值 | 24.0% | 38.3% | 46.7% |
| 人工法 | | 60.0% | 70.0% | 86.7% |

表 2 领域 TIME_DAY 测试结果

| TIME_DAY | | 测试集大小 | | |
|---|---|---|---|---|
| | | 5 | 10 | 一半 |
| 顺序法 | 最大值 | 40.0% | 40.0% | 46.7% |
| | 最小值 | 4.0% | 25.0% | 40.0% |
| | 平均值 | 24.7% | 35.0% | 43.3% |
| 人工法 | | 40.0% | 60.0% | 80.0% |

表 3 领域 NUM_SPEED 测试结果

| NUM_SPEED | | 测试集大小 | | |
|---|---|---|---|---|
| | | 5 | 10 | 一半 |
| 顺序法 | 最大值 | 51.1% | 60.0% | 60.0% |
| | 最小值 | 0.0% | 20.0% | 48.0% |
| | 平均值 | 18.2% | 40.5% | 54.0% |
| 人工法 | | 55.6% | 62.5% | 96.0% |

可以看出，“人工法”比“顺序法”取得的结果要好，这与我们在测试之前所推断的一样。此外，为了得到一个性能较好的 SDS，对于每个主题，开发者只需要准备 10～20 条语句就可以，我们认为这样的要求对于普通人来讲是可以接受的。

将来的工作 我们认为 SDS Lite，尤其是 GrammarTool

有以下几方面可进行改进，包括：

(1) 支持无序型规则：在增强型上下文无关语法的5种语法规则类型中，无序型和跳跃是最常用的两种，现在还只支持跳跃型。

(2) 扩充预定义关键词：在预定义关键词文件中，将会加入更大量的同义词列表。

(3) 继续进行测试：SDS Lite的使用性和可用性应该被更正式地、更彻底地进行测试，如果可能的话，甚至可以提出一系列的测试原则。

(4) 支持简单的多回合对话：由SDS Lite生成的SDS当前还只能支持单回合地交互，不过我们的目标是在将来能够支持简单的多回合对话。

感谢　在SDS Lite的开发过程中使用了由哈尔滨工作大学信息检索实验室提供的语料及数据。此外，我们也要感谢SDS Lite团队中其他成员所做的贡献，尤其是清华大学的邬晓钧博士。

## 参 考 文 献

1 Seneff S. TINA: a natural language system for spoken language applications. Computational Linguistics, 1992, 18(1): 61～86
2 Ward W. The CMU air travel information service: Understanding spontaneous speech. In: Proceedings of the DARPA Speech and Natural Language Workshop, June 1990
3 燕鹏举. 对话系统中自然语言理解的研究:[博士学位论文]. 北京:清华大学计算机科学与技术系, 2002
4 Rosenfeld R, Zhu X J, Shriver S, et al. Towards a universal speech interface. In: Proceedings of International Conference on Speech Language Processing'00, Oct. 2000
5 Rosenfeld R, Olsen D, Rudnicky A, A universal human-machine speech interface: [Tech Rep]. CMU-CS-00-114. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Mar. 2000
6 Toth A, Harris T, Sanders J, et al. Towards every citizen's speech interface: An application generator for speech interfaces to databases. In: Proceedings of International Conference on Speech Language Processing, Denver, CO, 2002. 1497～1500

(上接第179页)

表2

| 项目类型 | 正常 | 违约 | 合计 |
|---|---|---|---|
| 训练集 | 873 | 267 | 1140 |
| 测试集 | 845 | 155 | 1000 |
| 合计 | 1718 | 422 | 2140 |

根据财务数据的一般性指标，这里选择表3所示的十个相关属性作为主要考察目标。

表3

| 代码 | 相关属性 | 代码 | 相关属性 |
|---|---|---|---|
| I1 | 流动资产合计/资产合计 | I6 | 营运资本/资产合计 |
| I2 | 销售收入/资产合计 | I7 | 应收帐款/负债合计 |
| I3 | 保留盈余/资产合计 | I8 | 权益利润率 |
| I4 | 税息前利润/资产合计 | I9 | 固定资产/资产合计 |
| I5 | 净利润/所有者权益 | I10 | 销售利税率 |

使用本文提出的新型规则推理网络(NRRN)对以上数据进行学习，首先学习训练集数据，提取有效决策规则，然后使用测试集进行风险预测。记录这两个过程中对数据学习的错误判断类型，作为和已有的人工神经网络技术[3](NNR)对比的指标，如表4所示。

表4　训练样本和测试样本的误判

| 模型 | 训练样本 | | | 测试样本 | | |
|---|---|---|---|---|---|---|
| | 第一类错误 | 第二类错误 | 总误判 | 第一类错误 | 第二类错误 | 总误判 |
| NNR | 55 (4.8%) | 23 (2.0%) | 78 (6.8%) | 112 (11.2%) | 43 (0.43%) | 155 (15.5%) |
| NRRN | 11 (0.96%) | 9 (0.79%) | 20 (1.75%) | 9 (0.9%) | 5 (0.5%) | 14 (1.4%) |

我们把错误判断分为两类：第一类错误与第二类错误。在统计学中，第一类错误称为“纳伪”，第二类错误称为“拒真”。这里，“纳伪”指我们把不能偿还贷款的企业误判为能偿还贷款的企业；“拒真”指把能够偿还贷款的企业误判为不能偿还贷款的企业。显然，“纳伪”比“拒真”严重得多。犯“拒真”错误至多是损失一笔利息收入，而犯“纳伪”错误则会造成贷款不能收回，形成呆帐。因此我们应尽量避免“纳伪”错误。

表4显示，对于训练样本的学习中，模型NNR纳伪值是NRRN的4倍多，拒真值是NRRN的2倍多，总误判比值接近4∶1。由此可以看出，NRRN学习精确度比NNR要高得多。在预测能力上，通过对测试集的学习，发现模型NNR的纳伪值是112，几乎是模型NRRN(9)的10倍，而拒真比值达到43∶5=8.6，总误判比值超过11∶1，可见模型NRRN的预测能力远远强于模型NNR。总之，无论从学习精确度还是从预测能力方面看，模型NRRN都是一种更加有效、实用的技术。

结论　本文讨论了对于商业银行业务数据处理中的信用风险问题，使用流图技术，结合Bo L.等的相关性分析方法，提出了新型规则推理网络概念和技术，并且首次把该技术应用到商业银行信用风险决策的实际中，通过实际银行业务数据的实证，显示该技术具有良好的有效性和实用性，能够有效提取和解释商业银行信用风险的决策规则。这里暂时不考虑流图的可视化问题，而作为今后研究工作的方向。

## 参 考 文 献

1 Baltensperger E. Credit rationing: issues and questions [J]. Journal of Money, Credit and Banking, 1978, 10: 170～183
2 王春峰，万海晖，张维. 基于神经网络技术的商业银行信用风险评估[J]. 系统工程理论与实践, 1999(9): 24～32
3 Rudy S, James Y L. Thong, An approach to generate rules from neural networks for regression problems [J]. European Journal of Operational Research, 2004, 155: 239～250
4 Pawlak Z. Flow graphs and data mining [J]. Transactions on Rough Sets III, LNCS 3400, 2005. 1～36
5 刘博，郑启伦，彭宏. 基于非线性相关发现的数据挖掘算法[J]. 计算机应用研究, 2007(2)
6 Bo L, Qilun Zh, Hong P. A Novel Arithmetic of Holistic Correlation Analysis [J]. In: The Proceeding of the Fourth International Conference on Machine Learning and Cybernetics, August 2005. 2145～2150
7 Bondy J A, Murty M S R. Graph Theory With Applications [M]. The Macmillan Press Ltd, 1976
8 Bankruptcy Creditors' Service Inc. http://bankrupt.com. [DB/OL]