Universidade Federal da Paraíba

Centro de Ciências e Tecnologia

Coordenação de Pós-graduação em Informática

Uma abordagem híbrida de aquisição de conhecimento empregando RBC e Aprendizagem Indutiva: Aplicação ao domínio de Help Desk

Álvaro Vinícius de Souza Coêlho

Campina Grande – Paraíba

Fevereiro de 2001

Universidade Federal da Paraíba

Centro de Ciências e Tecnologia

Coordenação de Pós-graduação em Informática

Dissertação submetida à Coordenação de Pós-Graduação em Informática do Centro de Ciências e Tecnologia da Universidade Federal da Paraíba como requisito parcial para a obtenção do grau de Mestre em Ciências (MSc)

Orientador: Prof. Dr. Edilson Ferneda

Co-Orientador: Prof. Dr. Marcelo Alves Barros

Campina Grande – Paraíba

Fevereiro de 2001

Ficha Catalográfica

COÊLHO, Álvaro Vinicíus de Souza

C672A

Uma Abordagem Híbrida de Aquisição de Conhecimento Empregando RBC e Aprendizagem Indutiva Aplicação ao Domínio de Help Desk.

Dissertação (Mestrado) – UFPB/CCT/COPIN, Campina Grande, dezembro de 2000.

95p. Il.

Orientadores: Elilson Ferneda
Marcelo Alves de Barros

1. Inteligência Artificial,
2. Raciocínio Baseado em Casos,
3. Data Mining,
4. Help Desk,
5. Banco de Dados.

CDU: 007.52

UMA ABORDAGEM HÍBRIDA DE AQUISIÇÃO DE CONHECIMENTO EMPREGANDO RBC E APRENDIZAGEM INDUTIVA. APLICAÇÃO AO DOMÍNIO DE HELP DESK

ÁLVARO VINÍCIUS DE SOUZA COÊLHO

DISSERTAÇÃO APROVADA EM 15.12.2000

PROF. EDILSON FERNEDA, Dr.
Orientador

PROF. MARCELO ALVES DE BARROS , Dr.
Orientador

PROF. MARCUS COSTA SAMPAIO, Dr.
Examinador

PROF. AGENOR DE SOUZA MARTINS, D.Sc
Examinador

CAMPINA GRANDE – PB

Para Vânia. Por tantos motivos que citá-los
reclamaria uma outra dissertação.

Agradecimentos

Ao Criador, pelos motivos de praxe além de outros de ordem pessoal.

Aos meus pais pelo amor que permitiu a eclosão do Zigoto.

Aos amigos de modo geral pois é deles que vem a alegria e o apoio necessário para que se vença caminhadas cada vez mais longas.

Aos professores Edilson Ferneda e Agenor Martins pela confiança quando erros passados não a inspiravam e pelos valiosos conselhos de última hora, respectivamente.

A todos os demais que eventualmente sintam-se esquecidos, pelo que deixei de mencionar.

Sumário

Capítulo 1
Introdução

1.1 Atender e Encantar

A competitividade globalizada e o sensível aumento do nível de exigência dos consumidores impõem às empresas, como condição *sine qua non* de sua própria sobrevivência, estar próximo das necessidades e dos anseios de seus clientes, prontas para atendê-los e encantá-los. Atendê-lo é dar a ele o pronto cumprimento de todas as suas vontades manifestas; é fornecer um produto ou serviço de boa qualidade, dentro do padrão esperado e permanecer em vigília para que eventuais problemas ou insatisfações oriundas de produtos ou serviços possam ser rapidamente sanados.

O encantamento constitui escopo mais difícil, trazendo em seu conceito a necessidade de previsão[1] de seus desejos ainda não conhecidos e estar provendo suas necessidades antes mesmo da manifestação destas. O cliente se encanta quando, ao solicitar um produto ou serviço, este não só atenda todas as suas expectativas mas também as supere de maneira agradável e surpreendente. As tecnologias computacionais objeto do presente trabalho estão colocadas em ambos os contextos de relacionamento com o cliente: atender e encantar.

[1] Existem inúmeras estratégias de pesquisa de mercado que permitem estabelecer uma previsão das vontades dos clientes. Neste trabalho será introduzida a abordagem que usa a base de dados de um sistema de Help Desk para isso.

1.2 Objetivos

De modo geral, pode-se afirmar que o objetivo geral deste trabalho é prover ao setor de Help Desk de uma organização um conjunto de ferramentas que o permitam atender e encantar os clintes. Mas a questão que imediatamente acode esta afirmação é: Como? Ou seja, Como um Help Desk atende, e como ele encanta?

Responder a estas duas indagações conduz com mais precisão para os objetivos do trabalho: O atendimento de um cliente, por um sistema Help Desk, é alcançado se o sistema é capaz de ouvir deste a descrição que ele faz de um problema que esteja enfrentando e apresenta rapidamente uma solução para o seu problema. Para encantá-lo, o Help Desk não atua diretamente. Mas pode prover informações preciosas aos setores estratégicos (Marketing, Planejamento, etc.) que lhes permitam antecipar tendências ou preferências ainda não manifestas.

Para a primeira solução, a literatura apontou uma abordagem por Raciocínio Baseado em Casos. Esta permitirá que o sistema busque, por semelhança, no conjunto de casos armazenados, aqueles que mais se aproximem da descrição atual do usuário.

A segunda solução é original: A idéia é usar a imensa fonte de informações que um sistema Help Desk possui para prover um conjunto de conhecimentos que possam ser capitalizados pelos setores competentes.

Surge ainda uma terceira opção que é, a partir do que é registrado nos atendimentos, obter-se informações para que o próprio sistema Help Desk seja monitorado e corrigido por especialistas.

As duas últimas soluções são alcançadas através de métodos indutivos. Como a terceira solução usa um método indutivo para a melhoria da acurácia e dos resultados de um método indutivo, fica assim justificada o hibridismo.

1.3 Contribuições

Para que o atendimento seja acessível aos mais diversos tipos de clientes, é necessário que a estes seja disponibilizada uma interface de acordo com suas preferências e seu nível de conhecimento técnico do problema. Aqui, ao contrário das abordagens mais comuns, é apresentada duas formas diferentes de interface com o cliente: Pelo Casamento de Padrão e Pelo Atributo[2].

Na primeira, vai-se utilizar uma estratégia muito comum em sistemas de ajuda *online* de ferramentas para ambente gráfico: O emprego de palavras-chave.

Na segunda esbarra-se numa característica do domínio, e uma adaptação do padrão da tecnologia proposta é apresentado: A implementação da similaridade como uma característica não mais do resgate, e sim do atributo do caso.

Para os métodos indutivos existe uma variedade de aplicações muito grande, e o ideal é apresentar ao usuário um método que lho permita identificar origens de dados diferentes de domínios diferentes para que possam ser extraídas toda a sorte de informações que interessem aos setores estratégicos. Para isso, e baseado num produto existente de mercado, foi montado um algoritmo que implementa métodos de Associação para regras de Classificação[3].

1.4 Estruturação do trabalho

O Capítulo 1 (este) faz uma introdução do trabalho, apresentando os objetivos, as contribuições e uma estruturação geral de todo o trabalho.

O capítulo 2 apresenta uma introdução do problema em estudo, o Help Desk, e mostra que as possibilidades que surgem a partir do emprego de técnicas computacionais de aprendizado são relevantes. Isto justifica a segunda parte deste, que apresenta uma visão resumida da teoria da Inteligência Artificial aplicada à aprendizagem.

[2] Estes métodos de recuperação são apresentados com mais detalhes no Capítulo 3, seção 3.12.1
[3] Associação e classificação são tipos de conhecimento indutivo, como pode ser visto no Capítulo 4, seção 4.5

O capítulo 3 apresentará um resumo do conhecimento teórico sobre RBC, a primeira das duas técnicas de aprendizagem empregadas neste trabalho. Isso situará o leitor no contexto dos potenciais daquela tecnologia.

O capítulo 4 forma a segunda parte da revisão bibliográfica do trabalho, que é o estudo sobre a aprendizagem indutiva. Faz-se um levantamento do processo como um todo, e mostra-se os tipos de conhecimento que podem ser extraídos, bem como suas aplicações. Um destaque especial é dado às Regras de Classificação, que serão o conhecimento a ser buscado no âmbito de Help Desk, e às Regras de Associação, por um motivo que será explicado no capítulo seguinte.

No capítulo 5 é mostrada efetivamente os métodos disponíveis e as necessidades encontradas no domínio deste trabalho, o Help Desk. De posse dessas duas informações, ainda neste capítulo, é mostrada uma alternativa de abordagem do problema com o uso do RBC,

Ainda no capítulo 5 faz um levantamento das características desejadas para que a aprendizagem indutiva tenha suas potencialidades realmente exploradas no domínio em estudo, e em seguida apresenta a solução aqui proposta, no tocante à aprendizagem indutiva: uma abordagem com métodos para regras de associação aplicados a regras de classificação.

O Capítulo 6 é um estudo de caso, mostrando todas as abordagens em uso, o hibridismo Regras-Casos, e uma aplicação do método indutivo proposto.

Finalmente o capítulo 7 dá uma visão geral e crítica deste trabalho, e apresenta as contribuições outorgadas além dos horizontes que se descortinam, e que poderão ser alcançados por outrem.

1.8 Conclusões

Este capítulo apresentou o trabalho que será mostrado nas páginas subseqüentes. Para isso, foi dada uma visão do domínio do problema e dos objetivos que deverão ser alcançados – para o atendimento e para o encantamento de clientes. Com efeito, um sistema de Help Desk pode ser particularmente útil na segunda tarefa se cumprir satisfatoriamente a primeira e for auxiliado por métodos computacionais adequados.

Os objetivos, então, são seguidos das contribuições que estão aqui abrigadas: Tanto na implementação de métodos de RBC quanto numa abordagem híbrida que usa métodos indutivos como provedor de infoirmações úteis ao RBC e ainda na apresentação de um método algorítmico que usa a extração de regras de associação para classificação.

O próximo capítulo é uma visão resumida de Help Desk e a apresentação da teoria computacional que será útil no escopo deste: A Aprendizagem Automática.

Capítulo 2

Help Desk & Aquisição de Conhecimento

2.1 Introdução

Este capítulo apresenta o domínio deste trabalhjo, o Help Desk, e mostra de que forma uma abordagem computacional com aquisição automática de conhecimento pode de fato melhorar a performance e a acurácia dos resultados do sistema bem como ainda permitir que haja uma aferição contínua do mesmo e ainda que este seja usado como suporte na geração de conhecimentos que possam ser úteis se capitalizados em setores estratégicos.

Inicialmente é apresentado o Help Desk com todas as suas características, e em seguida uma visão da abordagem mais atual, que emprega em diversos níveis de atendimento o sistema. Em seguida é mostrado um modelo que já vêm sendo desenvolvido, e que oferece uma grande quantidade de potenciais espaços a serem pesquisados e melhorados.

De fato, são apresentadas as possibilidades que existem a partir do emprego da aquisição de conhecimento – notadamente o Dedutivo e o Indutivo[4] – para efetivamente prover as características necessárias e assim se alcançar os objetivos desejados.

Finalmente, é mostrada a tecnologia de aquisição automática de conhecimento a partir da aprendizagem automática até a estruturação formal dessa abordagem – estas que precisamente serão as ferramentas que se empregará neste trabalho.

2.2 O que é um Help Desk

O termo Help Desk é relativamente novo na indústria da Computação. Apesar de haver uma infinidade de definições diversas (e na seção 2.3 será apresentada uma delas) pode-se entender o

Help Desk como um canal ou um lugar onde deve dirigir-se as pessoas que estejam com problemas ou necessitem de algum auxílio para operar um (ou mais) computador a fim de concluir uma tarefa com sucesso.

Existem uma infinidade de termos associados (sinônuimos) como por exemplo Call desk, Call center, TAC (Tecnical assistance centre – Centro de assistência técnica), Support centre (Centro de suporte), Service bureau (Bureau de serviços), Hot line, etc. Na prática, porém, como será visto, a literatura especializada apresenta pequenas distinções entre cada um desses termos.

2.3 Help Desk como solução para atender e encantar

Segundo Gorgônio [Gorgônio, 1999], as Centrais de Atendimento ao Consumidor, que por esse motivo cada vez mais são incluídas nos planos de Marketing das empresas, são ditas *Help Desk* quando designam serviços de suporte em que o objeto em questão é um produto ou serviço na área de Informática (software ou hardware) ou que utilizam tecnologia computacional para provê-lo.

Um bom serviço de Help Desk precisa dar ao cliente a solução imediata ou no mais curto espaço de tempo possível para seu problema. Isso é sua missão imediata, e cumpri-la satisfaz os requisitos de atendimento ao cliente, conforme exposto acima.

Ultimamente, porém, um sistema Help Desk tem sido concebido como um poderoso armazém de dados de onde podem ser extraídas informações que sejam do interesse dos setores estratégicos de marketing da empresa, principalmente no planejamento da qualidade dos produtos e na adequação destes às verdadeiras necessidades, dificuldades e insatisfações dos clientes – e neste caso o Help Desk torna-se uma ferramenta que, mais que atender, servirá de forte provedor de subsídio necessário para que se possa encantar o cliente.

2.4 Help Desk: Possibilidades da abordagem moderna

De um modo geral, os sistemas de Help Desk funcionam basicamente como as Centrais de Atendimento ao Consumidor. Normalmente é provido um número de telefone, provavelmente gratuito, e a partir de um contato telefônico um *expert* treinado dá o suporte necessário, buscando atender e, se possível, encantar o consumidor.

---

[4] Dedutivo e Indutivo são paradigmas de aprendizagem automática, como pode ser visto na seção 2.7.3

Modernamente as *Home Pages* das empresas também provêem o mesmo serviço via formulários ou e-mail. Isso permite um grande aumento de produtividade pois o atendimento *off line* permite a formação de um catálogo de problemas semelhantes e a especialização do atendimento para casos específicos. Além disso, pode-se perceber as eventuais semelhanças entre os problemas e tratá-los com mais rapidez e confiabilidade.

Em grandes organizações que possuam ponto de acesso pela Internet o Help Desk pode ser provido mais rapidamente e de maneira mais interativa, estabelecendo um diálogo *on line* com o cliente, ao mesmo tempo em que catálogos e bases de dados podem ser acessadas na busca de problemas semelhantes já resolvidos. Para isso, porém, toda uma estrutura de apoio deverá existir, uma malha interna que permeia a Web no ponto de interface com o cliente. Tipicamente uma Intranet.

Com um vultoso catálogo de problemas e dúvidas dos clientes, o Help Desk pode também ser usado como importante suporte no planejamento evolutivo dos produtos ou serviços. Ele provê informações valiosas que guiarão para uma contínua melhoria de qualidade, em direção ao horizonte inalcançável da plena satisfação do cliente.

2.5 Tratamento automatizado do Help Desk

Albuquerque [Albuquerque, 2000] propõe um modelo básico automatizado para o planejamento coerente e fundamentado da qualidade a partir de sistemas Help Desk.

O cliente acessa, provavelmente através de uma rede (possivelmente a própria Internet), um módulo automático de Atendimento Help Desk. Este módulo dialoga com ele, guiado pelo Sistema de Help Desk, e tenta apresentar uma solução para o seu problema baseando-se num Banco de Casos armazenados e buscando algum que possua semelhança conveniente.

Nesse processo, três resultados podem ser encontrados:

a) É encontrado um caso suficientemente semelhante e sua solução é apresentada ao cliente que, assim, consegue resolver o seu problema.

b) A solução encontrada não satisfaz o cliente.

c) Não é encontrada nenhuma solução.

As duas últimas alternativas forçam a intervenção do especialista (ou de uma equipe) para encontrar e apresentar ao cliente a solução mais adequada. Em qualquer dos três casos a solução satisfatória

final é armazenada no banco de casos para futura consulta, o que diminui progressivamente a probabilidade de aparecerem problemas que o sistema não seja capaz de resolver.

2.6 Problemas em discussão no Help Desk: Aquisição de Conhecimento

O modelo proposto por Albuquerque possui três qualidades importantes: a primeira é que, sendo o atendimento feito pelo Sistema Automatizado, pode-se encontrar a solução com custos menores de tempo e mão de obra especializada. A segunda é que, como a quantidade de problemas resolvidos pelo sistema de maneira automática tende a ser crescente, pode-se dimensionar a equipe especializada de maneira mais confortável pois esta deverá ser progressivamente menos requisitada. A última, embora não menos importante, vai de encontro a um problema levantado em [Thomas, 1996] a que o autor se referiu como *Career Path*: O fato de que em geral é difícil para as organizações em geral possuir um número elevado de bons *experts* em Help Desk pois eles são muito facilmente atraídos para outras áreas; no modelo proposto, há uma necessidade muito menor de pessoal altamente qualificado.

Há, porém, no modelo, uma característica potencialmente importante que poderá ser explorada de maneira a melhorar ainda mais o processo.

A partir de uma Base de Casos, e com o emprego de uma técnica adequada de Aquisição Indutiva de Conhecimento, pode-se encontrar informações "ocultas" na vastidão dos dados registrados. Este tipo de aquisição de conhecimentos em bases de dados é denominada *Data Mining* (DM)[5] e, neste trabalho, será utilizada de maneira a prover duas características importantes a mais.

A primeira é a possibilidade de extrair conhecimentos diretamente acionáveis no processo de promoção da qualidade. Regras gerais sobre os problemas mais freqüentes, as dúvidas mais comuns, o que é fácil e difícil, as potencialidades exploradas e inexploradas. Neste aspecto, pode-se buscar conhecimentos que relacionem clientes a produtos (que perfil de cliente efetivamente tem utilizado quais produtos), clientes a problemas e produtos a problemas. Qualquer das possibilidades proverá ao departamento de Marketing importantes subsídios para o planejamento promocional da qualidade dos produtos ou serviços disponíveis, adequando-os às reais necessidades de cada um dos seus clientes.

[5] Daqui para a frente será usado o termo Data Mining bem como Knowledge Discovery em virtude de, no caso, tratar-se de KDD – Knowledge Discovery in Databases, o que torna os dois termos equivalentes para este trabalho.

A segunda é prover aos administradores do próprio serviço informações a respeito da condução geral do processo, de forma que eventuais problemas ou falhas no atendimento possam ser percebidas e sanadas. Aqui a fonte de informações, portanto, não serão informações catalogadas ou gerais, mas as que efetivamente estejam sendo utilizadas pelo módulo de atendimento, previsto e mostrado em [Gorgônio, 1999].

A Fig. 2.1 ilustra o modelo com a possibilidade da utilização de um processo de Data Mining para a geração de uma Base de Conhecimento. Aqui o Help Desk passa a ser guiado, no diálogo com o cliente, pelos casos tratados até que possa ser encontrada alguma resposta satisfatória. Caso não seja possível encontrar nenhuma, ou a(s) resposta(s) encontrada(s) não satisfaça(m), será necessária a intervenção de um especialista.

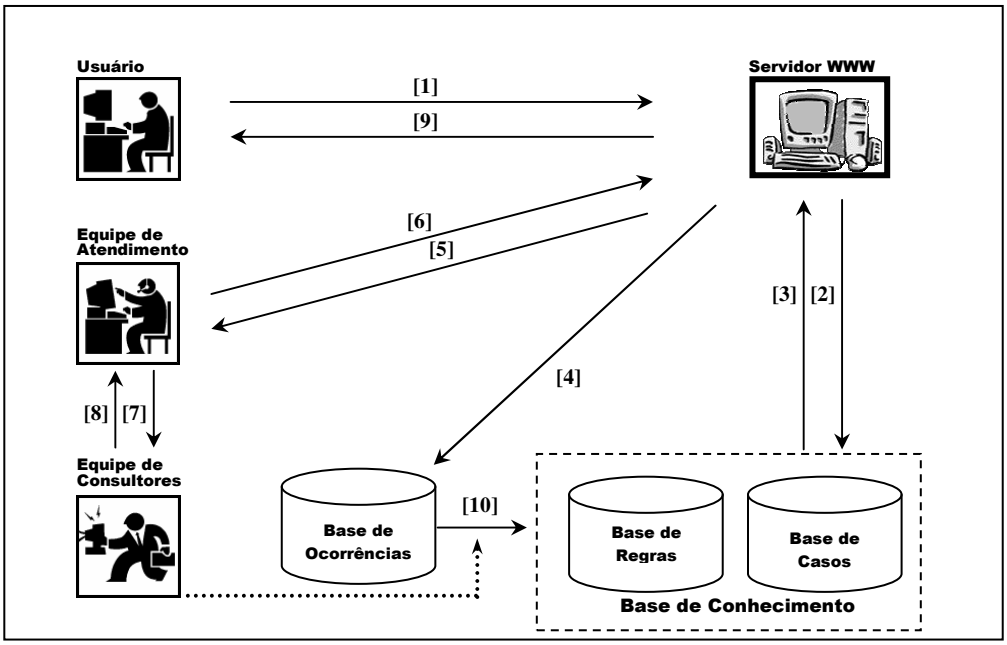

Figura 2.1: O modelo de um sistema do tipo Help Desk apoiado em RBC [Gorgônio, 1999]

A dissertação de F. L. Gorgônio [Gorgônio, 1999] é particularmente interessante porque explora uma arquitetura de *help desk* baseada em casos e regras, apoiada em suporte de Internet. O autor se refere a este modelo com o nome de *Web Help Desk*. A arquitetura, como se vê, compreende uma Base de Ocorrências, uma Base de Conhecimento formada tanto por casos como por regras, um servidor WWW, uma Equipe de Atendimento e uma Equipe de Consultores no domínio de aplicação. A idéia central na arquitetura é fazer com que somente problemas de extrema

complexidade no domínio cheguem às mãos dos consultores e sejam, por seu turno, *aprendidos* pelo sistema a fim de que, em caso de nova ocorrência, a equipe não seja mais utilizada.

Ao necessitar de suporte, o usuário se conecta à Internet e acessa o suporte **[1]** através do servidor WWW da empresa. Em uma primeira instância, o sistema irá tentar responder diretamente à indagação do usuário através de busca à base de casos **[2]**, que soluciona problemas que ocorrem com maior freqüência **[3]**. Quando o sistema não consegue uma resposta direta para o problema do usuário, isto significa que, numa primeira instância, não existe qualquer caso similar na respectiva base capaz de responder satisfatoriamente à indagação feita. Só então o problema é repassado para a Equipe de Atendimento **[5]**. Ao resolver o problema, uma resposta é enviada **[6,9]** ao usuário. Só se houver nova falha, quando a equipe de atendimento não puder resolver o problema, será acionada a Equipe de Consultores **[7,8]**. Todo o histórico de resolução de problemas é armazenado na Base de Ocorrências. Cabe ainda à Equipe de Consultores a responsabilidade de criação e validação de novos casos a povoarem a Base de Casos[6] **[10]** de modo que se verifique a constante aprendizagem do sistema. As regras poderão ser usadas na Promoção na Qualidade, bem como na melhoria do próprio serviço de Help Desk conforme será visto adiante.

Qualquer que seja o caminho adotado, seja pela Base de Conhecimentos seja pelo especialista, a solução deverá ser encontrada. Depois disso, ela será igualmente armazenada no Banco de Casos para uso posterior como fonte na busca pelas Regras ou como pesquisa direta por semelhança.

Com isso justifica-se um estudo sobre as possibilidades de se aplicar a aprendizagem automática aos sistemas de Help Desk de forma a explorar todas essas potencialidades. Neste trabalho, por conseguinte, é mostrado um levantamento das necessidades encontradas de modo geral para as duas abordagens – RBC e Extração de Regras – e é apresentado um modelo de um sistema utilizando de maneira eficiente estas duas técnicas de aquisição de conhecimento.

2.7 Aquisição de Conhecimento

A capacidade de adquirir conhecimento automaticamente é objeto de recentes esforços multidisciplinares de pesquisadores em áreas como Computação, Pedagogia e Psicologia. Mas, enquanto os dois últimos tratam das propriedades do conhecimento, e das características que os seres humanos e os animais possuem em decorrência da sua capacidade de aprender, os Cientistas

[6] Toda a base de casos será projetada de forma a atender o mais completamente possível aos casos que ocorrerem com mais freqüência, bem como representarão também os que forem potencialmente mais importantes, conforme será mostrado nos capítulos 3 e 5.

de Computação cuidam mais do processo em si, na tentativa de compreendê-lo ou pelo menos simulá-lo com acerto satisfatório.

2.7.1 Aprendizagem

O objetivo é dotar uma máquina da prerrogativa que os humanos temos de aprender, ou, como quer Marvin Minsky, "*fazer mudanças úteis em nossa mente*" [Minsky, 1975].

Para outros autores importantes o conceito de aprendizagem muda um pouco, como:

Herbert Simon: "*representa mudanças no sistema, que é adaptativo no sentido de realizar, de uma próxima vez, a mesma tarefa ou tarefas similares de uma forma mais eficiente e eficaz*" [Simon, 1983].

Jaime Carbonell: "*o resultado produzido por um processo em que se adquire a habilidade de executar novas tarefas que não podiam ser executadas antes, ou então executar melhor antigas tarefas*" [Carbonell, 1989].

e Ryszard Michalski: "*a construção ou modificação da representação do que está sendo experimentado*" [Michalski, 1986].

É interessante notar, conforme Mongiovi [Mongiovi, 1995] as diferentes tendências dos autores: Enquanto Minsky é preocupado com o "humano", Simon e Carbonell tentam mostrar a eficiência, e Michalski se preocupa com a forma como a aprendizagem se reflete no sistema.

Ainda segundo Mongiovi [Mongiovi, 1995], "*o termo aprendizagem não deve ser visto como um termo de significado único e fossilizado, e sim como um termo de semântica flutuante cuja interpretação é função do contexto*".

Ou seja, em cada abordagem que se faz necessário tratar com o termo aprendizagem, seu significado será compreensível de acordo com o contexto em que estará imerso.

2.7.2 Técnicas de Inferência

A processo de aprendizagem começa tendo como base algum conhecimento preexistente, possivelmente obtido através de algum método (automático ou não), e o uso de um mecanismo de *inferência*, que permite derivar daí algum conhecimento novo.

Do ponto de vista puramente filosófico, segundo Marcel Holsheimer, et. all. [Holsheimer, 1994], duas diferentes técnicas de inferência podem ser distinguidas. Aqui, para efeito de melhor documentação, será mostrada uma terceira forma que Holsheimer não apresentou embora seja de não menos relevância:

Dedução é a técnica que permite inferir um conhecimento que seja uma *conseqüência lógica* de outro. Classicamente é representado por

.

que é a formalização, em lógica das proposições, do *Modus Ponens*.

A dedução, portanto, não é passiva de erro **desde que** tanto a premissa quanto o processo dedutivo (a implicação lógica) estejam corretos. Um exemplo clássico de dedução é o período: *Sócrates é um homem. Todos os Homens são Mortais. Logo* (dedução)*, Sócrates é mortal*. A dedução possui duas características importantes: é uma inferência do *geral* para o *particular* e também uma inferência do *antecedente* para o *conseqüente*.

Indução é a técnica de inferir informações que são *generalizadas* de outras. A indução, portanto, se opõe à dedução ao tentar inferir do *particular* para o *geral*. Observa-se exemplos da realidade e tenta-se gerar um conjunto de regras que preveja e explique os fenômenos futuros.

Formalmente, em lógica das proposições, a indução seria representada por:

$$\frac{x, y}{x \rightarrow y}$$

Outra importante diferença para a dedução é que, **ainda que** o processo esteja correto e os fatos também, o resultado final é um conhecimento passivo de erro.

Por exemplo, em:

Asa de Águia é baiano e faz música medíocre.
É o Tchan é baiano e faz música medíocre.
Harmonia do Samba é baiano e faz música medíocre.

$\therefore$

Todo Mundo que é baiano faz música medíocre

o processo indutivo resultou em erro, partindo de exemplos corretos e sendo executado com perfeição. Por infelicidade não foram apresentados ao sistema cognitivo[7] exemplos como Caetano Veloso e Dorival Caimmy.

Mas a indução é, assim mesmo, um mecanismo bastante útil de generalização pois pode se aproximar muito da verdade, principalmente se os exemplos fornecidos garantirem uma amostragem significativa do domínio.

Abdução é a técnica que permite derivar os fatos que geraram um estado atual percebido. É a técnica classicamente utilizada pelos heróis de Agatha Christie e Conan Doyle em suas aventuras policiais, e que foi erroneamente chamada de dedução por estes autores. A abdução pode ser exemplificada no período: "*Todos os que foram à segunda guerra são soldados reformados. Logo, aquele soldado reformado foi à segunda guerra*". Ou, como disse o segundo autor, "*eliminando-se as possibilidades aquilo que restar, por improvável que seja, precisa ser a verdade*" [Doyle, 1986].

Apesar de ser também passiva de erro, a abdução pode ser, conforme ilustra Mongiovi em [Mongiovi, 1995], a mais provável inferência entre as muitas possíveis.

Em lógica das proposições a abdução poderia ser representada por:

$$\frac{x \rightarrow y, y}{x}$$

[7] Sistema Cognitivo é a denominação geral que se dá aos humanos e outras criaturas com capacidade de aprender (possivelmente máquinas) que são objetos do processo de Aprendizagem Indutiva.

A abdução se opõe à dedução pelo fato de partir de um conhecimento conseqüente para tentar inferir um conhecimento conseqüente.

A abdução e a indução, portanto, opõem-se à dedução por duas características diferentes: a dedução é a inferência do *geral* para o *particular* e a indução é do *particular* para o *geral*. A dedução é a inferência do *antecedente* para o *conseqüente* e a abdução é a inferência do *conseqüente* para o *antecedente*.

2.7.3 Paradigmas do Processo de Aprendizagem

Do ponto de vista da Ciência da Computação, a aprendizagem se dá por alguns possíveis *paradigmas*, que pressupõem maneiras específicas de abordar o problema. Os paradigmas mais conhecidos são:

a) Paradigma Conexionista: São as redes neurais. Funcionam tentando copiar o método de funcionamento do cérebro humano.

b) Paradigma Genético: Utilizam-se dos chamados *Algoritmos Genéticos*, que se baseiam na teoria da evolução natural de Sir Charles Darwin.

c) Paradigma Analítico: Também chamado de paradigma dedutivo, pressupõe a aprendizagem a partir de substancial conhecimento preliminar (Background Knowledge) e poucos exemplos.

- Métodos Baseados em Casos: Também ditos Baseados em Analogia[8], tentam transferir conceitos de uma *base* para uma *meta*.

- Métodos Baseados em Explanações: Utilizam um tipo de aprendizagem onde a generalização deve ser justificada por explicações.

e) Paradigma Indutivo: Busca encontrar, em um conjunto de observações, ou exemplos, algum padrão ou descrição geral para aquele domínio.

[8] A moderna literatura faz uma diferenciação entre Raciocínio por Analogia e Raciocínio Baseado em Casos, conforme é ilustrado no Capítulo 3.

Destes, conforme exposto, serão de particular importância para este trabalho os paradigmas Analítico (RBC) e Dedutivo.

2.8 Conclusões

Um sistema de Help Desk, definido como serviços de suporte em que o objeto em questão é um produto ou serviço na área de Informática (software ou hardware) ou que utilizam tecnologia computacional para provê-lo, pode ser um importante aliado dos setores estratégicos de Marketing, fornecendo às organizações um importante subsídio para disputar mercado.

Sendo uma importante reserva de informações que podem ser aproveitadas no próprio atendimento dos clientes, na melhoria da qualidade dos produtos e serviços e para o diagnóstico de problemas no próprio sistema, pode-se munir o Help Desk de ferramentas que permitam à organização extrair informações desse tipo e fazer uso delas a seu favor.

Particularmente as abordagens de aprendizagem automática por Raciocínio Baseado em Casos e por Métodos Indutivos podem permitir que o sistema, por um lado, tenha um módulo de atendimento automático e, por outro, proveja toda essa sorte de informações aos setores estratégicos.

Capítulo 3

Raciocínio Baseado em Casos

3.1 Introdução

Neste capítulo será apresentada uma visão geral do Raciocínio Baseado em Casos (RBC), esta que será uma das técnicas utilizadas no decorrer deste trabalho.

Inicialmente é apresentado o RBC de maneira conceitual e intuitiva mostrando-se também uma idéia geral de todo o processo.

Como elemento fundamental da tecnologia, o Caso é mostrado com destaque para a modelagem, etapa onde devem ser estabelecidas suas características relevantes. Depois o espaço de casos, onde um sistema baseado em RBC poderá encontrar soluções para problemas futuros.

Sendo um método computacional, o processo de RBC é mostrado em seu algoritmo geral, chamado "Algoritmo dos 4-R" e, em seguida, discute-se as formas de se estruturar os casos e os métodos mais comuns de indexação de casos já que é essa a característica que delineará o funcionamento do resgate, que é o propósito principal dessa tecnologia no escopo deste trabalho.

A aprendizagem em RBC é feita a partir da retenção dos novos casos, e uma visão geral deste processo é apresentada, seguida de uma descrição dos métodos de recuperação e de adaptação dos casos a fim de adequá-los ao problema atual – outra etapa crucial para este estudo.

Finalmente, uma conclusão de tudo o que foi apresentado no capítulo, além de uma breve apresentação do próximo.

3.2 O que é RBC

Por Raciocínio Baseado em Casos, entende-se um processo de resolução de problemas cuja principal característica é a reutilização de casos passados e resolvidos como insumo básico para a resolução de novos problemas semelhantes. O RBC é um método do Paradigma Analítico [Mongiovi, 1995], e difere do Paradigma Indutivo por alguns aspectos; O mais interessante de ser notado é que, para os métodos indutivos, o sistema terá que gerar, **a partir dos exemplos**, uma base de conhecimento sobre a qual o raciocinador (humano ou automático) atua. Em RBC, o raciocinador atua **diretamente sobre os exemplos**, não havendo nenhum processo de extração de conhecimentos a partir deles; ou seja, em RBC os casos representam o próprio conhecimento. Isto é apresentado por alguns autores como a grande vantagem dessa abordagem sobre as demais (por exemplo os métodos indutivos) já que os custos de processamento e de geração da base de conhecimento para sistemas desse tipo é irrelevante se comparado às redes neurais ou os algoritmos indutivos.

3.3 Idéia central

Conforme Martins [Martins, 2000], o problema central do RBC constitui na manipulação dos casos. Mas como o próprio autor defende em seguida, "a manipulação computacional de casos é um problema tão multi-facetado e complexo que se desdobra em pelo menos cinco outros subproblemas".

Basicamente o RBC, então, é executado a partir da resolução de cinco subprocessos principais:

(i) *Indexação*, que é a identificação dos atributos que um problema já resolvido (aqui registrado como caso) precisa ter para que possa ser de fato relevante na solução de novos problemas subseqüentes.

(ii) *Resgate*, que é o retorno, para o usuário, de todos os casos já armazenados na memória mas levando-se em consideração, num primeiro passo, as descrições/restrições do novo problema de entrada (é o que se chama de apresentação) e em seguida aproximando o mais estreitamente possível estas descrições dos casos que estão sendo resgatados.

(iii) *Seleção*, que é a escolha, dentre os diversos casos que foram resgatados, daquele(s) que de fato apresente(m) a solução para o problema a ser resolvido. O melhor caso é selecionado então para adaptação.

(iv) *Modificação/Avaliação*, que podem ser vistos unicamente como subprocessos da *Adaptação* de casos. A *modificação* é a alteração criteriosa de partes contidas em sua descrição com o

objetivo de se alcançar um alvo enquanto a *avaliação* consiste em checar a viabilidade dessa nova descrição do caso.

(v) *Retenção*, que é o registro da solução correta (depois de ter sido resgatada, selecionada, modificada e avaliada) na base de casos de forma que, no futuro, fique previsto o emprego desta mesma solução – um processo também conhecido por *Aprendizagem por Experiência* [Martins, 2000].

3.4 Conceito de caso computacional

Posto que o problema central do RBC consiste na manipulação de casos, segue-se naturalmente a necessidade de se definir formalmente um caso.

Martins [Martins, 2000] emprestará sua definição muito bem ao escopo deste trabalho:

> "Caso é uma porção de conhecimento contextualizado que represente quer uma experiência passada, quer uma experiência hipotética[9]".

Resumidamente, portanto, um caso pode ser visto como a descrição de um problema unida à sua solução.

Opcionalmente pode-se agregar um terceiro elemento, que traga luz aos resultados obtidos a partir da aplicação da solução sugerida no caso a um determinado problema enfrentado no domínio da aplicação. Um caso, nessa abordagem, será a descrição do problema, mais a solução e mais o resultado obtido.

3.5 Modelagem de Casos

Na prática, os casos que estão armazenados em uma ferramenta de RBC (como CBR Express, ESTEEM, KATE, REMIND, S_3-CASE) [Martins, 2000] podem ser compostos também dos tipos complexos de dados que são usualmente encontrados em aplicações e bancos de dados modernos além dos componentes naturais que descrevem o problema, a solução e os resultados. Os casos podem, então, incluir:

(i) *Rótulo* ou *Título*, que o identificará como um dos casos disponíveis na base de casos.

---

[9] Na verdade esta definição é um pouco mais abrangente, estendendo-se aos conceitos de *true cases* (os casos concretos) e *abstract cases* (os casos hipotéticos) e segue de uma referência que o autor usa em seu trabalho.

(ii) *Indagações*, que ajudarão um usuário a aceitar ou não um caso proposto.

(iii) *Textos em Linguagem Natural*, que descreverão o caso em estudo

(iv) *Ponteiros*, apontando para um caso interessante ou relacionado, ou mesmo para uma outra fonte de conhecimento extra-casos[10].

(v) *Multimídia*, que poderão vir associados aos casos, ou mesmo descrever sua solução com recursos audiovisuais.

Todos os componentes de um caso costumam ser descritos na forma de um conjunto de pares <atributo, valor> e, nesse caso, o objeto ou a situação (problema ou evento) representado pelo caso será idealmente representado através das suas características mais essenciais[11].

3.6 Espaço de casos

Para se ilustrar o funcionamento básico da tecnologia de RBC, a Fig. 3.1. mostra a relação entre o espaço de problemas e o espaço de solução.

Nessa ilustração, tanto o *problema* quanto sua *solução* habitam seus respectivos espaços. Um novo problema a ser resolvido é colocado no espaço do problema via uma operação de *apresentação*. Uma operação de *resgate* vai identificar um caso como sendo a descrição mais semelhante ao problema que pôde ser encontrada (operação **R** da Fig. 3.1). O sucesso nesta operação implica que necessariamente uma solução também foi localizada. Se necessário, *adaptações* podem ser feitas (operação **A** da Fig. 3.1.) a fim de se propor uma nova solução.

Este modelo, por conseguinte, assume que há uma relação de um para um mapeando os espaços de problema e solução, e uma função que leve cada problema à sua solução específica. Logicamente, para cada novo problema que surge no espaço de problemas, uma nova solução surgirá no espaço correspondente.

Matematicamente teríamos o funcionamento do processo de RBC como uma função tal como segue:

```
A(F(R(Problema Novo)))
```

---

[10] Qualquer tipo de conhecimento armazenado pode ser apontado por um caso, por exemplo uma regra ou um conjunto de regras.

[11] Na indexação baseada em atributos, que será mostrada na seção 3.10 este processo será melhor descrito.

onde R é uma função de resgate, F uma função de mapeamento Problema→Solução e A uma função de adaptação de uma solução já armazenada em outra que seja útil para um problema novo.

A Fig. 3.1 porém não contempla na totalidade o conjunto de operações necessárias para a manipulação dos casos, apresentando apenas uma visão sintética do processo.

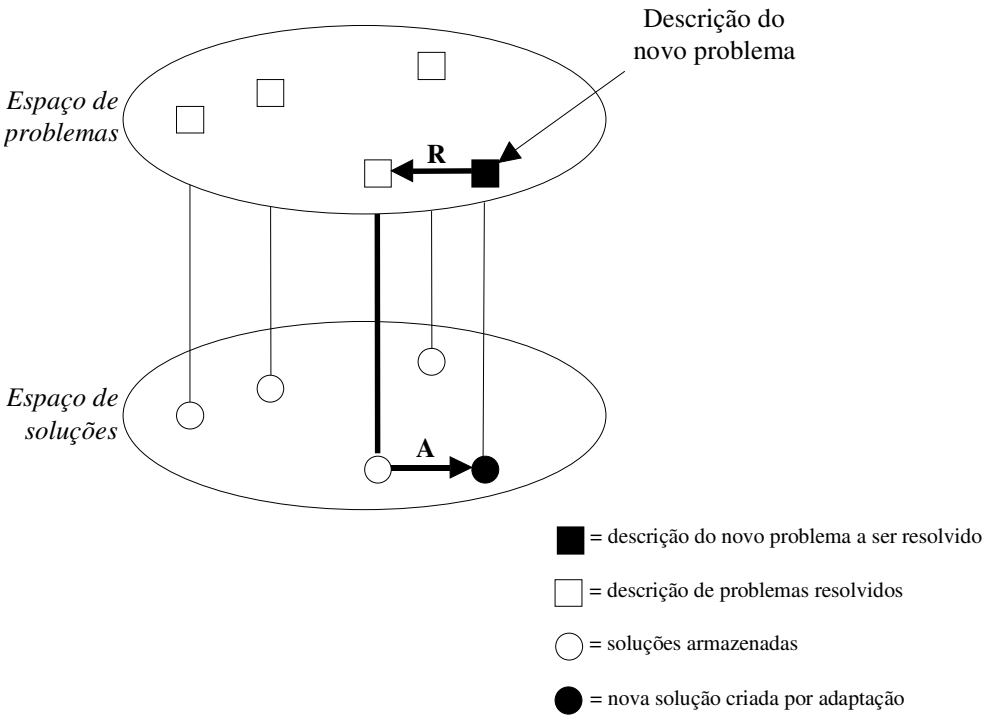

Fig. 3.1. Espaços de Problema e de solução

3.7 Algoritmo geral

De um modo geral, as diferentes tarefas envolvidas no processo de RBC – como manipular casos em uma base de casos já existente – podem ser expressas no que se chama de "Algoritmo dos 4-R":

RACIOCÍNIO BASEADO EM CASOS

Início
Obter as especificações do novo problema
Identificar atributos de indexação
Resgatar um conjunto de casos que atendam aos atributos
Repete
Seleciona um caso
Modifica o caso
Avalia solução
Até que a solução seja satisfatória
Fim

O algoritmo trata de quatro etapas básicas: [Martins, 2000]

(i) *Resgate* de casos armazenados, aqueles que sejam os mais semelhantes ao caso do usuário.

(ii) *Reutilização* desses casos para se tentar resolver um novo problema.

(iii) *Revisão* (*Avaliação*) da solução contida em casos prévios (se necessário).

(iv) *Retenção* da nova solução, a fim de se incrementar a base de conhecimentos para o futuro.

É interessante observar que, segundo alguns autores, para o bem do desempenho ou em respeito à complexidade de alguns domínios, algumas destas etapas podem ser agrupadas, desmembradas ou mesmo suprimidas.

3.8 Indexação: Problema fundamental do RBC

Um sistema de RBC é precisamente tão bom quanto os casos que ele pode recuperar, ou seja, sua habilidade em encontrar e selecionar os casos relevantes para um certo problema de forma rápida e precisa. O problema é escolher as características que devem ser atribuídas como índices aos casos a serem inseridos na base de casos, de forma que eles possam ser recuperados apropriadamente. É a isso que se chama *Indexação*.

As características deverão identificar em que circunstâncias um caso passado possui uma lição útil a ensinar [Kolodner, 1993]. O problema da indexação, por conseguinte, é exatamente organizar a base de casos de forma que a recuperação seja o mais eficiente e precisa possível.

Segundo C. Martin [Martin, 1989], é interessante que os sistemas de RBC sejam capazes de manipular representações complexas e estruturadas de conhecimento. Esta característica pode ser

mais facilmente garantida se por trás destes sistemas houver, como nos poderosos gerenciadores de bancos de dados modernos, um ambiente que permita a manipulação de estruturas desse tipo.

Ainda segundo J. Kolodner [Kolodner, 1993], os índices deverão também ser distintivos, ou seja, através destes o sistema deverá facilmente encontrar o caso adequado no momento correto.

3.9 Estruturação

Os índices que identificam um caso devem ser estruturados de acordo com o tipo de representação do conhecimento que serão necessários (vetores, regras heurísticas, frames, redes semânticas, linguagem orientada a objeto) [Martins, 2000].

As representações dos casos são ditas pouco estruturadas ou bem estruturadas se os casos contiverem respectivamente uma menor ou maior diversidade de tipos de dados de acordo com o que a aplicação venha a requerer.

Com efeito, classes de atributos diferentes, e preferencialmente de tipos de dados diferentes aumentam a discrepância (e consequentemente as possibilidades de distinção) dos diversos casos entre si, fato que por si só justifica a afirmação acima [Martins, 2000].

3.10 Métodos de indexação

O termo *método de indexação*, segundo [Martins, 2000], na verdade diz respeito às diretrizes de indexação. Segundo [Watson, 1997], a lista de diretrizes que servirão para classificá-los é a seguinte:

(i) Indexação por atributos e dimensões, que é a aplicação de princípios para a seleção de atributos que sejam mais específicos ou mais genéricos (dimensões). Segundo Watson, um exemplo de indexação por atributo é o sistema MEDIATOR, que tenta mediar questões de conflitos de terra. Os criadores tentam indexar os casos registrados segundo o tipo e as funções dos objetos em litígio, bem como as relações entre os litigantes.

(ii) Indexação baseada em diferenças, tenta encontrar e utilizar as diferenças entre as situações. Por esse motivo é desejável que os índices sejam as características que melhor diferenciam um caso de outro. Watson cita o sistema CYRUS[12] como exemplo.

---

[12] CYRUS (Computerized Yale Retrieval and Update System) é um sistema de Janet Kolodner, que trata de eventos sobre a vida de Cyrus Vance enquanto este era Secretário de Estado nos EUA.

(iii) Indexação baseada em similaridades/explicação são métodos de generalização que, segundo Watson, podem levar à produção de um conjunto de índices para casos com características compartilhadas. A idéia é usar a aprendizagem baseada em explicação (*explanation-based learning*) para extrair uma combinação de características (atributos) que levem em conta a generalização de uma explicação.

(iv) Indexação por aprendizagem indutiva, que é a utilização de métodos de aprendizagem indutiva para identificar e selecionar índices que sejam melhor prognosticadores para casos no paradigma RBC[13].

3.11 Novos casos para a base

Uma vez tendo sido *projetados*, os casos então precisarão ser organizados na memória a fim de que possam ser recuperados oportunamente.

De um modo geral, o armazenamento dos casos é feito associando-se cada atributo dele a uma lista de casos já armazenados de forma que a recuperação possa ser feita de maneira eficiente. O algoritmo abaixo [Martins, 2000] ilustra essa operação com mais clareza:

ACRESCENTA CASO

Início
Acrescenta o rótulo do caso na base de casos;
Para cada par <Atributo$_i$, valor$_i$> constante no caso
Obter a lista de rótulos de casos associada com o atributo$_i$ na lista de indexação;
Se a lista de rótulos de casos estiver vazia então
Acrescentar Atributo$_i$ na lista de indexação;
Criar a lista contendo o rótulo do caso;
Associar esta nova lista ao Atributo$_i$;
Senão
Acrescenta o rótulo do caso na lista de rótulos de casos
Fim

Para cada par <Atributo$_i$, valor$_i$> de um novo caso projetado, uma lista de indexação é pesquisada tendo em vista achar uma lista de rótulos de casos associados com o atributo$_i$. Um novo rótulo de caso será então acrescido à base se o atributo$_i$ já existir na lista; caso contrário este atributo será acrescido a ela [Martins, 2000].

[13] A Indexação Baseada em Explicação e a Indexação Baseada em Aprendizagem indutiva diferem entre si basicamente pela maneira com que algum tipo de conhecimento externo é utilizado a fim de se indexar os casos. A primeira usa conhecimento humano e a segunda indução automática.

De um modo geral pode-se armazenar os casos na memória de duas maneiras diferentes:

(i) Armazenamento Seqüencial, também conhecido por *Flat Library*, pressupõe o armazenamento dos casos em estruturas planas, de acesso seqüencial como em *Flat Files* ou *bases de dados relacionais*. Esses métodos, por se utilizarem de tecnologias já consagradas como Bancos de Dados, são de aplicação mais voltada para a Indústria. Em [Kolodner, 1993] é apresentado um método diferente mas ainda em estudo e de interesse acadêmico.

(ii) Armazenamento Hierárquico, um modelo que é útil, idealmente, para bases que contenham uma quantidade extremamente vasta de casos. A hierarquia de casos, segundo Martins [Martins, 2000], permite que, em dado momento, somente um subconjunto potencialmente útil desses casos necessitem ser alojados na memória. Kolodner, em [Kolodner, 1993] cita que há duas formas mais comuns de armazenagem hierárquica:

- Redes de características compartilhadas, conhecidas por *Shared-Feature Network*,
- Árvores de discriminação.

Ambas as abordagens geram agrupamentos (*clusters*) de casos similares, cuidando também de gerar discriminações entre estes casos em relação a um certo atributo previamente definido. Mas enquanto o primeiro método privilegia o *clustering*, o segundo foca mais a discriminação [Martins, 2000].

3.12 Recuperação de casos

Posto que o Raciocínio Baseado em Casos é um paradigma que busca solucionar problemas a partir de experiências anteriores, é de crucial importância que ele possua, entre seus componentes, um algoritmo de recuperação capaz de trazer da base de casos um caso, ou um conjunto de casos que tenham similaridade e sejam úteis na busca pela solução de uma situação nova.

Segundo Kolodner [Kolodner, 1993], "Um caso útil é aquele que é similar à nova situação em dimensões que ajudam o raciocinador (sistema) a realizar suas tarefas ou a alcançar suas metas".

De um modo geral pode-se ver a recuperação de casos como um problema de busca, embora com proporções consideráveis, similar a que ocorre nos bancos de dados[14] mas com uma diferença

[14] Com efeito, a partir das modernas tecnologias de pesquisa dos bancos de dados modernos, os algoritmos de recuperação de casos tiveram melhorias significativas em suas performances

principal: Nos bancos de dados há a necessidade de se exigir casamentos exatos, ao passo que os algoritmos de recuperação em RBC devem fazer a busca baseando-se em casamentos parciais – em casos práticos os casamentos exatos são muito raros.

Naturalmente, os algoritmos de recuperação estão intimamente ligados às estruturas de dados às quais eles se aplicam. Logo, a qualidade de um algoritmo é um paralelo com a qualidade da respectiva estrutura [Kolodner, 1993].

Martins [Martins, 2000] apresenta um algoritmo geral de resgate de casos de uma base cuja lista de pares <atributo, valor> casa (parcial ou exatamente) com a descrição do problema.

RESGATA CASO

Início

Para cada par <atributo$_j$, valor$_j$> *no novo problema*
Encontrar a lisa de rótulos associados com o atributo$_j$ da lista de indexação
Se a lista de rótulos não for vazia então
Para cada caso$_i$ da lista de rótulos de casos
Obter o valor$_i$ do atributo$_i$ no caso$_i$;
Se valor$_j$ casa com valor$_i$ então
Se caso$_i$ não estiver em casos-resgatados então
Acrescentar o caso$_i$ em casos-resgatados;

Fim

Logo, para cada atributo em um novo problema, se os valores desse mesmo atributo em algum dos casos se case com ele, esse caso será recuperado.

3.12.1 Métodos de resgate de casos

Martins descreve, ainda, os principais métodos de recuperação de casos, que são:

(i) *Casamento de padrão*. Este mecanismo, também conhecido por *Pattern Matching Retrieval*, envolve a seleção de dados a partir de uma dada função de casamento de caracteres (strings). Sua principal qualidade é a flexibilidade, já que a busca é feita de maneira seqüencial (pouca eficiência) e quase toda combinação de caracteres pode ser feita. O problema com este método é que buscas desse tipo não costumam oferecer muita orientação ao usuário – pode-se facilmente encontrar especificações e resultados não apropriados. Por outro lado, ainda que se especifique a entrada corretamente buscando casamentos exatos, o método se mostra pouco robusto pois pode recuperar muitos casos inadequados ao problema de entrada junto com os resgates mais válidos.

(ii) *Recuperação baseada em índices*. O uso dos índices como método de recuperação de casos é mais adequado do que o simples casamento de strings porque ele possui a vantagem de reduzir o custo da busca, conforme mostra Brown em [Brown, 1995]. Nesse método, os atributos (ou índices) dos casos determinam o funcionamento do resgate, ou seja, os caminhos de acesso aos casos são definidos rigidamente no momento da criação dos índices.

(iii) *Recuperação via algoritmos seqüenciais*. Se os casos estiverem armazenados em estruturas simples como listas, vetores ou arquivos, pode-se usar formas igualmente simples de recuperação através dos algoritmos seqüenciais. J. Kolodner apresenta em [Kolodner, 1993] algoritmos de busca seqüenciais, que fazem uma varredura de todos os casos selecionando os que se casem melhor com o caso-alvo (o problema). As ressalvas feitas com métodos desse tipo é que podem existir diferentes níveis de importância entre atributos de casos distintos que não são levados em consideração.

(iv) *Recuperação via algoritmos paralelos*. Para se obter um incremento na capacidade de recuperação de casos, uma alternativa interessante são os algoritmos paralelos. A vantagem óbvia é que pode-se consultar ao mesmo tempo todos, ou uma quantidade significativa de casos de uma só vez, como mostra J. Kolodner em [Kolodner, 1991].

(v) *Recuperação baseada em similaridades*. O princípio desse tipo de recuperação é o uso de *métricas de similaridade*, que serão aplicadas aos casos, a fim de se classificar os casos que obtenham os resultados mais altos, segundo um certo critério. O grande problema com esse método é estabelecer a métrica, que deverá ser baseada em dados estatísticos e algumas informações semânticas que permitam divisar valores como distâncias[15] entre os atributos além de outros. Esse método, porém, pode minimizar o tratamento das deficiências na entrada, como ruídos e omissões.

(vi) *Recuperação via algoritmos indutivos*. Na seção 3.11, foram mostradas as formas de armazenamento de casos, entre as quais o armazenamento hierárquico, que pode ser por *árvores de discriminação* ou por *rede de características compartilhadas*. O RBC empresta os algoritmos baseados em indução computacional para fazer a recuperação de casos armazenados sob essa estrutura [Martins, 2000]. Os algoritmos indutivos, nesse caso, usam árvores de decisão que são construídas através de exemplos (dados) preexistentes. Para o

[15] Assumindo-se cada atributo como uma dimensão, e tendo seus valores discretizados e ordenados (informação semântica) é possível mapear o caso num espaço cartesiano de N dimensões, N o número de atributos.

RBC, esses dados corresponderão a casos que serão objeto de manipulação. Posto que, conforme Kolodner, em [Kolodner, 1993]:

> "Se é possível colocar juntos os casos similares uns aos outros, de modo que se distinga os grupos de casos que melhor se casem com a situação representada por um novo caso-alvo, então somente os itens – de um grupo em particular – precisam ser considerados no momento da busca pelos melhores casamentos".

Segundo Martins [Martins, 2000] o algoritmo mais empregado em RBC para essa abordagem é o ID3.

(vii) *Recuperação baseada em Banco de Dados*. Estando os casos armazenados em um Banco de Dados, ou sendo de alguma forma por ele endereçados, é possível se fazer consultas complexas às bases de casos a fim de se maximizar as potencialidades de casamento entre os atributos do problema atual e os que podem ser encontrados nos casos passados. Com efeito, essas consultas podem usar casamentos alternativos com o emprego dos meta-caracteres. Há que ressaltar, porém, que as linguagens de manipulação de dados em Bancos de Dados (como SQL) são extremamente rígidas, o que dificulta o seu uso direto por usuários não especialistas, bem como dificulta o tratamento de ruídos e/ou omissões das informações oriundas do prompt do sistema (a descrição do problema atual).

3.12.2 Exemplos

1. Resgate de casos

   O algoritmo de busca em árvores seria, segundo Kolodner [Kolodner, 1993], o seguinte:

BUSCA EM ÁRVORES

```
Início
    Fazer N = Raiz;
    Repita até que N seja um caso:
        Em N questionar algo relacionado ao caso de
            entrada;
        Fazer N = nó detentor da melhor resposta;
    Retornar N;
Fim
```

2. Uso de banco de dados para potencializar as consultas

   Em SQL por exemplo, o caracter porcento "%" pode substituir uma cadeia finita de caracteres na consulta, potencializando as alternativas.

SQL é uma linguagem de manipulação e consulta de dados, desenvolvida no laboratório de pesquisa da IBM em San Jose (hoje o Centro de Pesquisa Almadem) no início dos anos 70 [Korth, 1991]. SQL permite que as consultas sejam mais flexíveis, como segue:

```
select cd_caso
from tb_caso
where ds_caso like '%impressora%'
```

Que resgataria todos os casos de uma base que tenham, em qualquer lugar de sua descrição, a palavra "impressora".

3.13 Adaptação de Casos

De um modo geral, a probabilidade de haver, entre os casos recuperados, algum que se case exatamente com o atual é muito pequena.

Posto que essa situação ótima é rara, na imensa maioria dos casos o sistema terá que fazer algum tipo de *adaptação* do(s) casos(s) recuperado(s) a fim de adequá-lo(s) ao problema. Segundo Kolodner [Kolodner, 1993] existem três alternativas para resolver o problema:

- Adaptação automática de casos
- Adaptação manual de casos
- Não adaptação de casos

A adaptação dos casos pode se dar manualmente, com o usuário interferindo no processo diretamente, ou automatizada com o uso de heurísticas, regras e outras informações de caráter semântico que porventura sejam utilizáveis.

Os métodos de adaptação de casos, segundo Kolodner [Kolodner, 1993] podem ser agrupados segundo as seguintes classes: *métodos de substituição* e *métodos de transformação*.

A diferença entre as duas é que a primeira pressupõe a substituição de valores ou conjuntos de valores (objetos, atributos, etc.) por outros que se aproximem mais do caso atual enquanto a segunda tenta apenas transformar, de algum modo, porções de uma solução passada para se ajustar a uma nova situação [Martins, 2000].

Para domínios bem conhecidos e compreendidos, e preferencialmente bem comportados (que não possuam uma variedade muito grande de casos), a adaptação de casos é uma técnica cuja

implementação é relativamente simples e possui resultados interessantes. Em outros, porém, isso pode ser extremamente complicado e levar a resultados errôneos com muita freqüência. Para domínios desse tipo alguns autores propõem uma abordagem simplificada do algoritmo dos 4-R em que a adaptação é simplesmente descartada. Essa denominação é denominada *retrieve and propose* (resgata e propõe) e corresponde, em Inteligência Artificial, a uma estratégia de agentes inteligentes para a implementação física de bases de conhecimento conhecida por *store and fetch* conforme descrito em [Russel, 1995].

3.14 Conclusões

Neste capítulo foi apresentada uma visão geral, para que se possa ter uma compreensão do Raciocínio Baseado em Casos (RBC), já que esta será uma das técnicas utilizadas no decorrer deste trabalho pois, como será visto, é a partir de estratégias desta natureza que se poderá estabelecer, com um usuário, um diálogo que conduza-o rapidamente à solução do seu problema.

O RBC foi apresentado de maneira conceitual e intuitiva, sendo desvendado aos poucos desde seu elemento fundamental – o Caso – até o processo geral, ilustrado pelo Algoritmo dos 4-R, além de uma descrição mais detalhada de cada etapa dos processos componentes da tecnologia de RBC. Isso permitiu que se divisasse, entre tantas, características que serão aproveitadas no decorrer deste trabalho, como as facilidades providas pelo uso de bancos de dados, as estratégias de indexação, pois duas delas – a indexação pelo atributo e a indexação por casamento de padrão – serão de interesse também, e os métodos de resgate, e as estratégias de adaptação, entre as quais o *retrieve and propose*, aplicável a domínios complexos, e que também serão úteis nesta abordagem.

O capítulo seguinte, então, apresentará de maneira semelhante o paradigma indutivo, mostrando as aplicações, as características, os tipos de conhecimento, etc.

Capítulo 4

Regras – Conhecimento do paradigma indutivo

4.1 Introdução

Tratou-se até o presente do papel do RBC para efeito de Help Desk. A partir desta parte explana-se o papel das regras indutivas tendo em vista a construção de fato de um modelo híbrido para Help Desk.

Este capítulo mostra uma visão geral de outra técnica que será utilizada neste trabalho: A aquisição de conhecimento indutivo, e mais precisamente a extração de regras por indução.

Inicialmente é mostrado o Paradigma Indutivo, a partir de uma idéia geral, donde se define seu insumo básico para funcionar: o Conjunto de Treinamento ou *Trainning Set*.

Em seguida é mostrada uma visão geral do processo de aquisição de conhecimento indutivo automaticamente, com uma visão resumida de cada sub-processo envolvido.

Os tipos de conhecimento indutivos que podem ser úteis são apresentados, e com mais detalhamento os dois que serão particularmente interessantes para este trabalho; as **regras de classificação**, que são mostradas como árvores de decisão e lista de regras - cada uma dessas estruturas com suas vantagens e desvantagens, além de uma breve discussão do problema semântico e as **regras de associação**, que são discutidas e mostradas formalmente, além de ressaltada a importância de duas variáveis estatísticas bem particulares e também importantes para esta abordagem: confiança e suporte.

4.2 Paradigma Indutivo

O Paradigma Indutivo é a tentativa de fornecer à máquina uma coleção (ou amostra) de exemplos do domínio a ser explorado e, a partir destes, desejavelmente, o sistema ser capaz de descobrir

conhecimentos novos que sejam válidos com respeito à amostragem dada. O processo poderá se dar de duas maneiras: *Supervisionada* ou *Não Supervisionada*, caso exista ou não a presença de um usuário especialista (possivelmente humano) do domínio para guiar o processo[16].

4.3 Idéia Geral

Dado um Espaço de Busca S, o Sistema tentará encontrar, dentro dele, as descrições, associações ou padrões gerais que estejam respaldados em S, ou seja, que são verdadeiros *dentro do universo de S*. Este espaço é chamado de *Trainning Set* ou *Conjunto de Treinamento*.

O Sistema deverá executar uma busca iterativa no Trainning Set para extrair o conhecimento desejado. No caso de Aprendizado Supervisionado, essa busca terá o auxílio de algum especialista externo que cuidará para que a amostra seja de boa qualidade, e que o sistema não siga por caminhos que o leve a conclusões inválidas ou irrelevantes.

Durante a busca, o Sistema produz hipóteses que são verificadas através de uma *função de qualidade*, conforme ilustrado na Fig. 4.1. Esta função se baseia em técnicas estatísticas e retorna um *veredicto* que definirá se a hipótese formulada será aceita ou rejeitada.

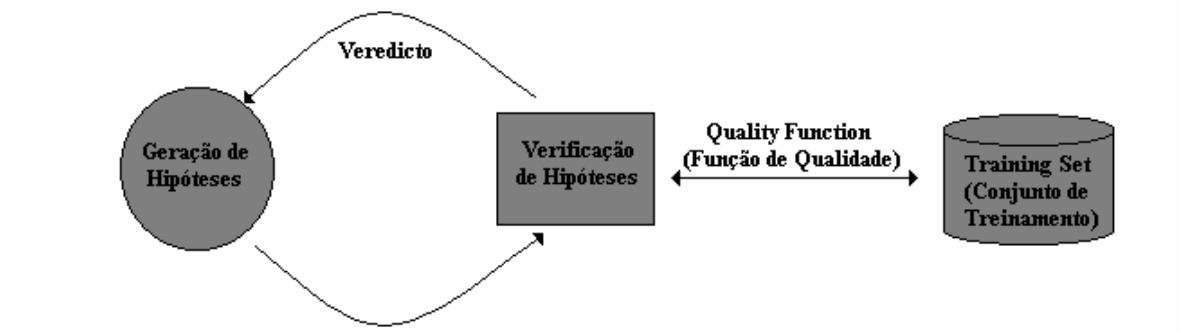

Fig. 4.1. O processo de Aprendizagem Indutiva

4.3.1 O Conjunto de Treinamento

O Conjunto de exemplos que é apresentado ao Sistema Cognitivo constitui o Trainning Set ou Conjunto de Treinamento. O Trainning Set deverá conter elementos que guiem o aprendiz (Sistema Cognitivo) para um resultado o mais acurado possível.

[16] O Protocolo de aprendizagem MOSCA prevê a figura do Mestre que será, em casos práticos, o Supervisor desse tipo de aprendizagem.

Isso em si representa um grande problema: Em muitos casos não é possível selecionar os exemplos de maneira suficientemente representativa ou então as relações (de onde se extrairão as regras) que existem nos exemplos não correspondem ao que se encontra no mundo real.

Isso é uma característica natural da Inferência Indutiva: Uma amostra mal formada leva a conclusões errôneas ou inexatas.

4.4 Aquisição Automática de Conhecimento Indutivo

A partir do Conjunto de Treinamento, o Sistema poderá encontrar por si só os conhecimentos que estão ali encerrados.

Para isso ele usa um algoritmo de busca[17] e, possivelmente, a supervisão de um Mestre[18]. Seus resultados são um conjunto de Regras, Padrões ou qualquer outro tipo de conhecimento que possa ser útil e seja previamente desconhecido.

Via de regra, entretanto, essa busca constitui um problema de complexidade exponencial já que potencialmente todas as combinações possíveis de relação entre os dados deverão ser testadas. Isso é minimizado com o uso de diversos tipos de Heurísticas que podem melhorar o desempenho sem perda substancial de acurácia.

É interessante notar, porém, que o processo na prática não se dá de maneira absolutamente ideal em nenhum caso. Muitos problemas são encontrados no armazenamento e no pré-processamento dos dados que são usados para compor o Trainning Set e no próprio conhecimento gerado.

Segundo Carvalho [Carvalho, 1998], "*o processo é iterativo, cada ciclo da iteração envolvendo várias etapas seqüenciais*". A Fig. 4.2. dá uma idéia do processo como um todo:

---

[17] Os algoritmos variam de acordo com o tipo de conhecimento a ser extraído, conforme será mostrado na seção 4.5.

[18] Mestre é um agente formal do Protocolo MOSCA. Em casos gerais diz-se simplesmente Especialista ou Humano Especialista

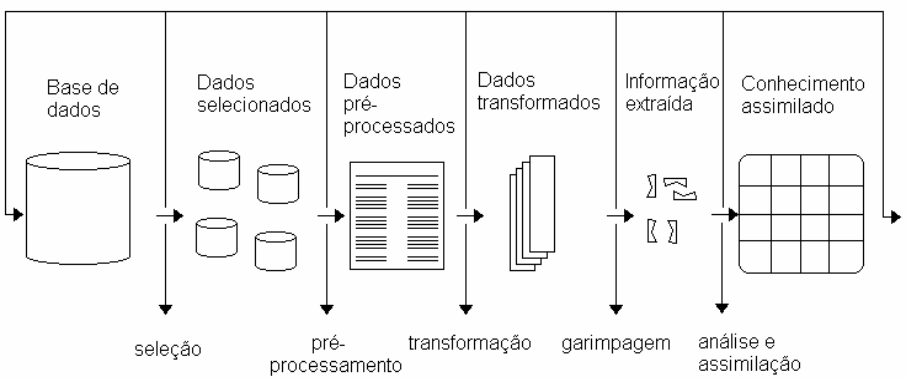

Fig. 4.2. Etapas do processo de Aquisição de Conhecimento Indutivo

Ainda segundo Carvalho, as três primeiras etapas constituem o que se chama *preparação dos dados*. A *garimpagem*[19] é a etapa principal, onde de fato são gerados os conhecimentos. Uma descrição resumida de cada etapa segue abaixo:

1) Preparação dos Dados

i) Seleção de Dados: É a identificação das origens dos dados que comporão o Trainning Set e a seleção conveniente desses dados.

ii) Pré-processamento dos Dados: Naturalmente as fontes de dados existentes não estão em conformidade para serem utilizados na etapa de garimpagem. Além disso, é sempre provável que hajam inconsistências e/ou dificuldades de integração de dados com origens diferentes.

iii) Transformação dos Dados: existem vários métodos e algoritmos de garimpagem de dados, cada um deles necessitando de uma entrada com formato específico. Para que os dados sejam passíveis de utilização por esses métodos eles precisarão ser transformados para se apresentarem de acordo.

2) *Garimpagem de Dados*: É a aplicação prática de um ou mais algoritmos de garimpagem (extração) de dados. Mais adiante serão mostrados alguns diferentes tipos de algoritmos disponíveis de acordo com o tipo de dados e com o tipo de conhecimento a ser extraído.

3) *Análise e Assimilação de Resultados*: Aqui deve-se apresentar os resultados em confronto com as seguintes questões: O conhecimento é relevante e acionável?[20] Caso a resposta não seja

[19] A Garimpagem é o processo em si de extração do conhecimento.

[20] Relevante é o conhecimento que seja, de fato, importante para o domínio estudado. Acionável é o conhecimento que poderá permitir tirar dele alguma vantagem.

satisfatória (e normalmente a princípio não é) é necessário, conforme ilustra a Fig. 4.2, se retomar o processo a partir de alguma etapa anterior.

4.5 Tipos de Conhecimento Indutivo

Os métodos e as estratégias de extração de conhecimento indutivo precisam ser determinados, conforme Chen[21] em [Chen, 1996], de acordo com uma série de características como o tipo de disposição dos dados de origem (o tipo de banco de dados), o tipo de técnica e o tipo de conhecimento a ser extraído entre outras.

Os tipos de conhecimento a serem encontrados e as técnicas e métodos a serem empregados estão diretamente inter-relacionados.

De um modo geral, ainda segundo Chen [Chen, 1996], pode-se extrair os seguintes conhecimentos de maneira indutiva:

- Regras de Associação, que são um conjunto de regras de associação fortes na forma de "de "A_1 ^ . . . $A_m \Rightarrow B_1$ ^ ... B_n" Onde A_i (Para $i \in \{1, ..., m\}$) e B_j (Para $j \in \{1, ..., n\}$) são conjuntos de valores atributivos de conjuntos de dados relevantes no banco de dados. Por exemplo, pode-se encontrar a partir de um grande conjunto de dados de transações uma regra de associação como se um cliente compra (uma marca de) leite ele/ela normalmente compra (outra marca de) pão na mesma transação.
- Ferramentas de Generalização de dados, que no mercado apresentam alguns nomes alternativos como *On Line Analitical Processing* (OLAP), Bancos de Dados Multi-Dimensionais, Data Cubes, Abstração de Dados, Generalização, Sumarização, Caracterização, etc. A generalização de dados apresenta as características gerais ou uma visão sumarizada de alto nível sobre um conjunto de dados especificados pelo usuário em um banco de dados. Por exemplo, as características gerais de staffs técnicos em uma companhia podem ser descritos como um conjunto de regras características ou um conjunto de tabelas generalizadas de sumários. Todavia, é sempre desejável apresentar visões generalizadas dos dados em múltiplos níveis de abstração (drill up e drill down).
- Outro importante tipo de conhecimento indutivo é a habilidade de executar classificação em uma enorme quantidade de dados. Essa tarefa é referida como extração de regras de

[21] Chen, na verdade, apresenta estas características direcionadas para Data Mining que é, conforme será mostrado adiante, um caso específico de aquisição de conhecimento indutivo.

classificação. *Classificação de Dados* é classificar um conjunto de dados baseado em seus valores em certos atributos. Por exemplo, é desejável para um vendedor de carros classificar seus clientes de acordo com suas preferências de carros de forma que o pessoal de vendas saiba antecipadamente como abordá-los, e os catálogos de novos modelos podem ser enviados diretamente para esses clientes cujas características identificadas vão aumentar as oportunidades de negócio

- Outro tipo de conhecimento é conhecido como *data clustering*. Basicamente, data clustering é agrupar um conjunto de dados (sem um atributo de classe predefinido), baseado no princípio conceitual de agrupamento: *maximizar as similaridades intraclasse e minimizar a similaridade interclasse*. Por exemplo, um conjunto de objetos de commodity pode ser primeiro agrupado em um conjunto de classes e então um conjunto de regras pode ser derivado baseando-se em tal classificação. Alguns agrupamentos podem facilitar a *formação taxonômica* que é a organização de observações numa hierarquia de classes que agrupam eventos similares juntos.

- Dados temporais ou dados espaciais-temporais constituem uma grande porção dos dados armazenados em computadores. Exemplos desse tipo de Banco de Dados incluem: Banco de Dados Financeiro para Índice de Preço de Estoque, Bancos de Dados Médicos e Bancos de Dados Multimídia para citar alguns. A procura por padrões similares em bancos de dados temporais ou espaciais-temporais é essencial em muitas operações de Aquisição de Conhecimento a fim de descobrir-se e prever-se o risco, casualidade e tendências associadas a um padrão específico. Pesquisas típicas nesse tipo de banco de dados incluem a identificação de companhias com padrões de crescimento similares, produtos com padrões de vendas similares, estoques com movimento de preços similares, imagens com padrões de repetição similares, características geológicas, poluição ambiental ou padrões astronômicos. Essas pesquisas invariavelmente exigem que se encontre algo similar, e não exato.

- Em um ambiente que provê informações distribuídas, documentos ou objetos são usualmente ligados para facilitar acesso interativo (os Links). O entendimento dos padrões de acesso dos usuários em cada ambiente não vai ajudar apenas a melhorar o projeto do sistema mas também viabilizar uma melhor decisão de marketing. Capturar os padrões de acesso dos usuários é chamado *mining path traversal patterns*. Observa-se, todavia, que posto que os usuários estão andando através da informação e usando serviços para pesquisar as informações desejadas, alguns objetos são visíveis por causa de suas localizações mais do que pelo seu conteúdo, mostrando a grande diferença entre o problema de padrão transversal e outros que são principalmente baseados em transações de clientes.

Como pode ser rapidamente percebido, são quase infinitas as aplicações de onde se pode extrair conhecimento indutivo. No escopo deste trabalho será de particular importância o estudo da extração de *Regras de Classificação* e *Regras de Associação*.

4.5.1 Regras de Classificação

Classificação de Dados é o processo que encontra as propriedades comuns a uma série de objetos numa *amostra* e os classifica em diferentes *classes* de acordo com um modelo de classificação. Em geral, a determinação de uma classe se dá pela discrepância de um ou mais atributos que são eleitos como discriminantes.

Para que possa ser construído um modelo de classificação, é necessário um conjunto *E* de dados de exemplo (o Conjunto de Treinamento) em que cada elemento consiste do mesmo conjunto de atributos (ou características) dos elementos do ambiente para o qual este será utilizado.

As descrições de classes serão usadas para classificar os dados futuros ou para desenvolver-se uma descrição melhor de cada classe real. As aplicações típicas de classificação de dados incluem diagnósticos médicos (na UFPb o Oftalmo e o Ginecol são exemplos), previsão de performance e marketing seletivo entre outras.

Além disso, a classificação de dados está sendo substancialmente usada em Estatística, Aprendizagem Automática, Redes Neurais e Sistemas Especialistas em geral.

4.5.1.1 Classificação baseada em Árvores de Decisão

As regras discriminantes podem ser armazenadas em árvores de decisão, onde cada nó representa um determinado atributo (ou característica) e cada ramo um valor (ou um conjunto de valores) que ele pode assumir; os nós folhas (que não possuem ramos a partir deles) representarão as classes.

Essa abordagem é muito interessante porque permite uma rápida avaliação das características que identifiquem uma determinada classe além de requerer poucos recursos computacionais para si (armazenamento e caminhamento top-down na árvore).

4.5.1.2 Problemas da Árvore de Decisão

A primeira dificuldade que existe na geração das árvores de decisão é a determinação dos atributos que ficarão na raiz e nos nós de menor profundidade. Com efeito, mesmo que cubra satisfatoriamente todos os exemplos apresentados, há a possibilidade de uma árvore ser pouco

eficiente porque vai exigir uma quantidade muito grande de atributos para identificar qualquer classe.

A Fig. 4.3. ilustra, num exemplo hipotético, uma árvore de decisão que supostamente cobre as descrições das classes C_1, C_2 e C_3 em função dos atributos A_1, A_2 e A_3 respectivamente.

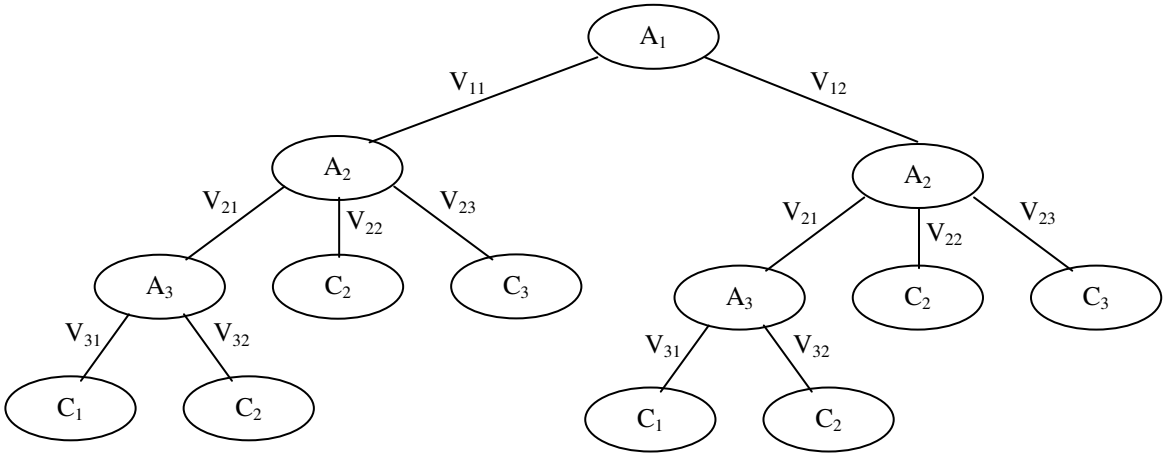

Fig. 4.3. Uma Árvore de Decisão Hipotética

Observa-se que, neste exemplo, a determinação de uma classe necessita do conhecimento de pelo menos dois dos atributos, mas essa necessidade não é absolutamente imprescindível.

Uma árvore alternativa, conforme mostra a Fig. 4.4. permite que se identifique as mesmas classes e mostra que, na realidade, o atributo A_1 é irrelevante para a determinação.

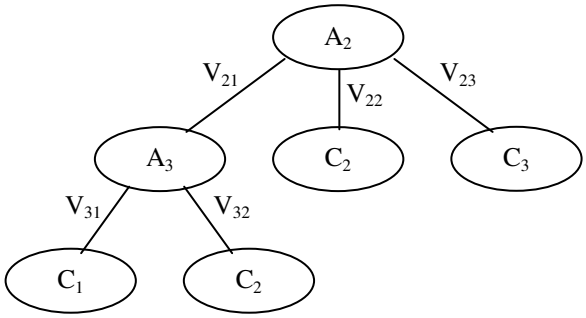

Fig. 4.4. Uma árvore mais eficiente para o mesmo domínio

As árvores supostamente cobrem o mesmo conjunto de dados, mas há uma óbvia diferença de eficiência entre as duas.

Para que se encontre árvores mais eficientes, é necessário que se use uma boa *função de avaliação*, que escolherá o atributo mais adequado para ficar nos nós mais acima de forma que a determinação das classes possa se dar de maneira eficiente e rápida.

Essa função utiliza algumas técnicas estatísticas como Entropia, Quiquadrado, Estatística G, Índice GINI e Medida proporcional de ganho [Mongiovi, 1995].

A segunda dificuldade que as árvores de decisão apresentam é o que se chama de *questão sintática*.

A partir das árvores de decisão pode-se facilmente encontrar as regras de classificação. Por exemplo, na Fig. 4.4, tem-se as regras:

Se ($A_2 = V_{21}$) e ($A_3 = V_{31}$) *Então* $\mathbf{C_1}$.
Se ($A_2 = V_{21}$) e ($A_3 = V_{32}$) *Então* $\mathbf{C_2}$.
Se ($A_2 = V_{22}$) *Então* $\mathbf{C_2}$.
Se ($A_2 = V_{23}$) *Então* $\mathbf{C_3}$.

Mas um determinado conjunto de regras nem sempre pode ser visto como uma árvore de decisão. Os conjuntos de regras que tenham pelo menos duas regras cujos atributos sejam totalmente diferentes não podem ser tornados em árvores de decisão. Por exemplo, em

Se ($A_2 = V_{21}$) e ($A_3 = V_{32}$) *Então* $\mathbf{C_2}$.
Se ($A_1 = V_{12}$) *Então* $\mathbf{C_1}$.

não é possível encontrar uma árvores que represente estas regas de classificação[22].

4.5.1.3 Classificação baseada em Listas de Regras

A solução baseada em listas de regras de classificação é largamente utilizada pois evita o problema sintático. Os métodos são semelhantes, embora a geração se dê em um conjunto de pares (Atributo, Valor), ligados a uma determinada classe – enquanto a geração de árvores considera apenas os atributos isoladamente.

Naturalmente as preocupações com a escolha do atributo mais adequado para ser considerado em cada regra, através das funções de avaliação, permanecem praticamente inalteradas já que é igualmente desejável se ter um conjunto de regras o menor e o mais simples possível.

---

[22] Na verdade existem técnicas que permitem a construção de árvores a partir de regras, mas em casos como estes são geradas árvores extremamente grandes e pouco eficientes

Para o escopo deste trabalho serão tratadas Regras de Classificação. Porém, como será mostrado no Capítulo 5, o estudo das regras de associação e dos métodos de extração deste tipo de conhecimento será de particular interesse para este trabalho.

4.5.1.4 O Problema Semântico

Mongiovi em [Mongiovi, 1995] apresenta com clareza o problema semântico com um exemplo real:

O sistema Ginecol, de auxílio ao diagnóstico ginecológico apresentou resultados bastante satisfatórios mas o conjunto de exemplos não era totalmente perfeito, e surgiram algumas regras cuja validade prática era absolutamente nula, como por exemplo: "Se Idade = Criança Então Vaginite".

Mais que uma imperfeição nos exemplos apresentados, o problema suscitou o questionamento da semântica.

Os métodos de classificação precisam, em muitos casos, ser dotados de algum tipo de supervisão para que possam automaticamente descartar as regras inválidas ou inúteis ainda que respaldadas nos exemplos a ele apresentados. Opcionalmente pode-se acrescentar aos dados alguma informação de caráter semântico a fim melhorar os resultados.

Mongiovi apresenta algumas estratégias para que possa ser utilizada a relevância semântica nesses casos. Maiores detalhes podem ser encontrados em [Mongiovi, 1995].

4.5.2 Regras de Associação

Dado um banco de dados de Transações de Vendas, é desejável descobrir as associações importantes entre itens de tal sorte que a presença de um conjunto não vazio (mas possivelmente unitário) numa transação implica na presença de outro conjunto, inteiramente disjunto ao primeiro, e não vazio (embora também possivelmente unitário) de itens na mesma transação.

Embora tipicamente apresentado para o domínio de vendas a varejo por transação, o conceito pode ser utilizado para diversos tipos diferentes de aplicação como resultados de experimentos físicos ou químicos, dados e eventos meteorológicos, procedimentos solicitados por clientes ou pacientes, etc.

4.5.2.1 Formalização

Dado $I = \{i_1, i_2, ..., i_m\}$ um conjunto de literais chamados itens[23]. Dado D um conjunto de transações, em que cada transação T é um conjunto de itens não repetidos tais que $T \subseteq I$. É interessante ressaltar que a quantidade do item comprada em uma transação não é considerada, significando que cada item é uma variável binária indicando se o item foi ou não comprado. Cada transação é associada a um identificador único, chamado TID. Dado X um conjunto de itens, uma regra de associação é uma implicação na forma $X \Rightarrow Y$, X chamado *antecedente* e Y o *conseqüente*, onde $X \subset I$, $Y \subset I$ e $X \cap Y = \varnothing$ [Chen, 1996].

4.5.2.2 Confiança e Suporte

A regra $X \Rightarrow Y$ garante o conjunto de transações D com *confiança c* se c% de transações em D que contém X também contém Y. A regra $X \Rightarrow Y$ possui *suporte s* no conjunto de transações d se s% das transações em D contém $X \cup Y$.

A confiança possui semântica ligada à força que a implicação possui, ou seja, qual o percentual de vezes que ela apresenta-se com acerto. Regras com confiança baixa não são idealmente utilizáveis pois tendem a representar exceções.

O suporte indica a freqüência de vezes que a regra poderá ser verificada no conjunto de transações, ou seja, em quantas transações ela é encontrada. Regras com suporte muito baixo também não são boas, pois a probabilidade de ser usada é muito pequena.

Dado X e Y conjuntos de itens e Pr(X) a probabilidade de todos os itens de X estarem contidos numa transação, então o suporte de $X \Rightarrow Y$ = Pr($X \cup Y$) [Agrawal, 1995].

A confiança de $X \Rightarrow Y$ = Pr($Y \mid X$), ou seja, a probabilidade de se ter Y dado que já se tem X. [Agrawal, 1995].

Matematicamente tem-se que: para uma regra $X \Rightarrow Y$, sendo X e Y conjuntos disjuntos de itens,

$$s(X \Rightarrow Y) = Pr(X \cup Y).$$

Ou, generalizando, s(X) = Pr(X). Por outro lado,

---

[23] O conjunto de literais é, na verdade, uma lista de chaves identificadoras de um determinado item em um nível taxonômico qualquer. Pode significar um item específico (leite da marca A) ou uma classe generalizada (leite).

$$c(X \Rightarrow Y) = Pr(X \mid Y) = Pr(X \cup Y) / Pr(X) = s(X \cup Y) / s(X).$$

Ou seja, a confiança de uma regra é igual ao suporte de todos os itens contidos na regra (antecedente e conseqüente) dividido pelo suporte do antecedente da regra.

Esse resultado será particularmente útil na próxima seção onde será apresentado o modelo matemático que permitirá a extração das regras de associação em um conjunto de transações dados o suporte e a confiança mínimos requeridos.

4.6 Métodos disponíveis para a Extração de Regras

Para este trabalho, serão de particular interesse os métodos de extração de regras. Aqui, tratar-se-á de regras de Classificação. Mas os métodos de extração de Regras de Associação também serão mostrados, e isso será justificado na seção 4.5.2.

4.6.1 Métodos de Extração de regras de Classificação

Já foram desenvolvidos diversos métodos algorítmicos para a extração de regras de classificação: de um modo geral eles se dividem em dois grandes grupos; os que usam árvores de decisão, e os que usam listas de regras de classificação. O uso de relevância semântica, mostrado na seção 4.5.1.4, é um estudo recente que não será escopo deste trabalho. Basicamente existem duas abordagens básicas que serão aqui apresentadas, a partir da qual derivou-se todos os métodos modernos que se conhece hoje: ID3, que pertence à família que se convencionou chamar de Algoritmos Tididt [Mongiovi, 1995] e PRISM.

4.6.1.1 O ID3

O primeiro método a ser considerado, que extrai regras de classificação em uma estrutura de árvore de decisão, é o ID3.

A Fig. 4.5. mostra um exemplo extraído de [Mongiovi, 1995] onde alguns atributos são considerados para se tentar prever o risco de obesidade de um determinado grupo de pessoas.

O Algoritmo ID3 começa pela escolha, através de uma função de avaliação, do atributo mais adequado para ficar na raiz.

No	Come	Vegetariano	Idade	Diabetes	Classe
1	Pouco	Sim	Velho	Sim	Magro
2	Médio	Sim	Velho	Não	Magro
3	Muito	Não	Velho	Sim	Gordo
4	Pouco	Não	Velho	Não	Magro
5	Médio	Não	Jovem	Sim	Gordo
6	Pouco	Sim	Jovem	Não	Magro
7	Muito	Não	Velho	Não	Gordo
8	Médio	Não	Jovem	Sim	Gordo

Fig. 4.5. Um conjunto de Exemplos para Classificação

Para este exemplo, usa-se o cálculo da entropia, que é dado por

$$H(a) = \sum_{i=1}^{n_v} \frac{n_{v_i}}{} \bullet H(a = v_i)$$

$$H(a = v_i) = \sum_{j=1}^{m} - p(E_j / v_i) \bullet \log_2 (p(E_j / v_i))$$

conquanto pareça complicado o cálculo desta variável estatística, na verdade é resultado de operações computacionalmente simples de implementar embora com considerável custo de processamento e seu estudo detalhado não será escopo deste trabalho. Maiores detalhes sobre o cálculo da entropia podem ser encontrados em [Mongiovi, 1995]. Deve-se escolher o atributo de menor entropia entre todos.

O atributo escolhido é {Come}, que ficará na raiz da árvore.

A partir deste atributo, o algoritmo ID3 já consegue divisar a raiz e três sub-árvores, conforme mostra a Fig. 4.6. e registra que em cada sub-árvore só serão considerados o subconjunto dos dados originais formados pelas chaves selecionadas.

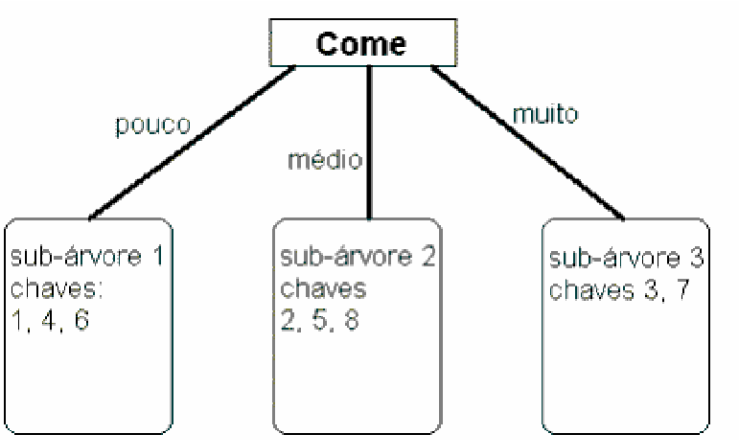

Fig. 4.6. Determinada a raiz da árvore de decisão

A seleção das chaves que comporão cada sub-árvore é feita de acordo com o valor do atributo considerado no nível superior. No caso, as chaves 1, 4 e 6 que serão consideradas na sub-árvore 1 são precisamente as que atendem ao requisito {Come}=”pouco”, que gera esta sub-árvore. O mesmo se aplica às demais sub-árvores.

O algoritmo segue, então, processando recursivamente cada sub-árvore encontrada. Para a sub-árvore 1, é criado o conjunto de exemplos ilustrado na Fig. 4.7, que são os exemplos que comporão o processamento desta sub-árvore. Além disso, naturalmente, o atributo {Come}, que já está na raiz, é desconsiderado.

No	Vegetariano	Idade	Diabetes	Classe
1	Sim	Velho	Sim	Magro
4	Não	Velho	Não	Magro
6	Sim	Jovem	Não	Magro

Fig. 4.7. O conjunto de exemplos utilizado na sub-árvore 1

Para este conjunto, já que todas as classes são idênticas, a execução é encerrada e substitui-se a sub-árvore por um nó-folha de classe, com o valor da classe encontrada (no caso, “Magro”).

A sub-árvore 2 é processada de maneira semelhante. Seu conjunto de exemplos é o que está na Fig. 4.8.

Neste caso, o critério de parada (a classe ser a mesma em todos os exemplos) ainda não foi alcançado. O processo, então, segue da mesma maneira: o cálculo das entropias e a seleção do atributo conveniente.

Há então uma coincidência nas entropias dos três atributos e, por um critério qualquer (uma outra função de avaliação), escolhe-se um deles. No caso, {Idade}.

A Fig. 4.9 ilustra como está a árvore de decisão neste instante do processamento.

No	Vegetariano	Idade	Diabetes	Classe
2	Sim	Velho	Não	Magro
5	Não	Jovem	Sim	Gordo
8	Não	Jovem	Sim	Gordo

Fig. 4.8. O conjunto de exemplos utilizado na sub-árvore 2

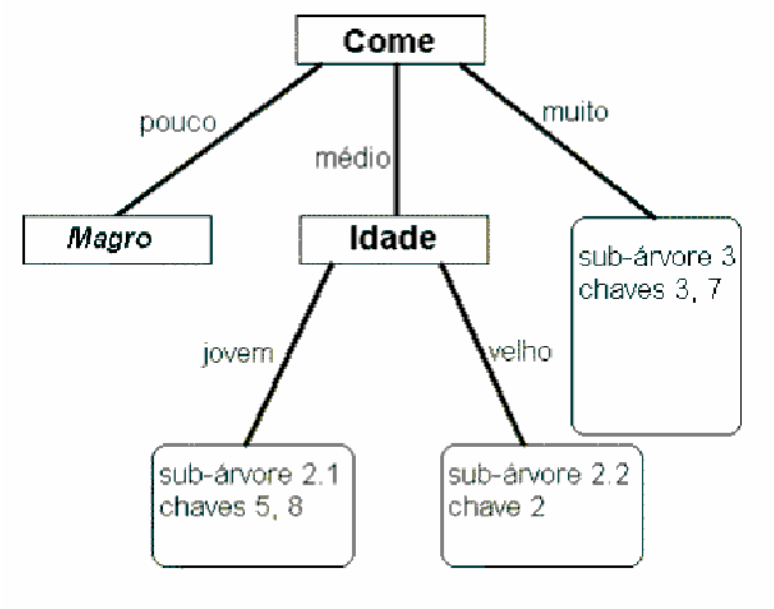

Fig. 4.9. Árvore em formação após a determinação da raiz da sub-árvore 2

A sub-árvore 2.1 terá um processamento semelhante ao da sub-árvore 1. Seu conjunto de exemplos será formado pelas chaves 5 e 8, desprezados os atributos {Come} e {Idade}, e em ambos os exemplos haverá somente uma classe que aparecerá assim no nó-folha que ocupará seu lugar.

O processamento da sub-árvore 2.2 é irrelevante, pois como só há um exemplo, garante-se que a condição de parada é alcançada por antecedência. Em seu lugar surge um nó folha com a classe específica de seu único exemplo (“Magro”).

Assim encerra-se o processamento da sub-árvore 2.

A sub-árvore 3, na qual serão consideradas somente as chaves 3 e 7 também atende a condição de parada: Em ambos os exemplos há somente uma classe, no caso, “Gordo”.

A árvore final é a apresentada na Fig. 4.10, que corresponde à árvore que cobre as descrições de todos os exemplos apresentados.

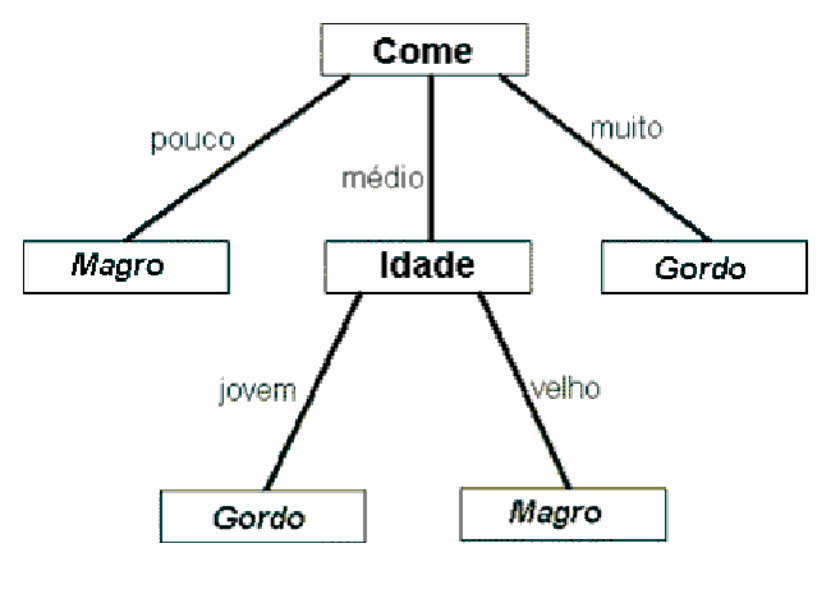

Fig. 4.10. A árvore de decisão final.

4.6.1.2 O PRISM

Conforme mostrado na seção 4.5.1.2, o armazenamento de regras em árvores de decisão, no que pese ter custo computacional extremamente baixo, possui diversas características indesejáveis. Uma abordagem interessante, partindo do próprio ID3, foi apresentada por Cendrowska [Cendrowska, 1988], que gera as regras em uma lista, e não numa árvore de decisão, evitando assim o problema sintático; o PRISM.

Para este algoritmo, a função de avaliação não será a entropia, e sim uma variável [Cendrowska, 1988] dada por

$$P(E_j|C)=\log_2(p(E_j|C)/P(E_j))$$

uma grandeza cujos detalhes e explicações mais profundas podem ser encontradas em [Mongiovi, 1995] e não são escopo deste trabalho. Esta função, porém, não seleciona apenas um atributo, mas um par (Atributo, Valor) que será utilizado na geração das regras.

O algoritmo trabalha em sucessivos passos, um para cada valor de classe. Em cada passo é selecionada uma condição e, para cada condição, um conjunto auxiliar de exemplos que será usado

para gerar uma nova condição (o conjunto original não é alterado neste instante: somente depois que forem sendo encontradas as regras).

A geração de novas condições a partir dos conjuntos auxiliares e de novos conjuntos auxiliares a partir das condições é repetida até que exista apenas uma classe no conjunto, semelhantemente ao ID3.

A Regra é, então, estabelecida pela conjunção de todas as condições encontradas, na forma C_1 e C_2 e ... e C_n.

Depois de encontrada uma regra, o conjunto original é modificado, com a exclusão dos casos cobertos por ela. O processamento de uma classe se encerra quando no conjunto original não restam mais exemplos dela. E o PRISM pára quando não restam mais valores de classe para serem processados.

No exemplo dado haverá, portanto, um processamento para a classe “Magro” e outro para a classe “Gordo”.

O primeiro processamento é iniciado e seleciona-se, pelo critério descrito acima, o atributo/valor {Come}=”Pouco”. Esta será a primeira condição da primeira regra da primeira classe.

Gera-se um conjunto auxiliar de exemplos de maneira semelhante à do ID3, excluindo-se o atributo {Come} que já foi considerado, e mantendo-se especificamente os exemplos que satisfazem à condição {Come}=”Pouco”.

A Fig. 4.7 mostra o conjunto auxiliar considerado e, já que há uma única classe encontrada, o PRISM encerra o processamento desta regra. A semelhança do processamento com relação ao ID3 é natural e previsível, decorrente da proximidade entre as duas abordagens.

R1: Se {Come} = “Pouco” então Magro.

O processamento segue na busca de novas regras, considerando-se o conjunto original subtraído das chaves que já estão cobertas pela regra recém criada.

A Fig. 4.11 ilustra o conjunto que está sendo considerado neste momento.

No	Come	Vegetariano	Idade	Diabetes	Classe
2	Médio	Sim	Velho	Não	Magro
3	Muito	Não	Velho	Sim	Gordo
5	Médio	Não	Jovem	Sim	Gordo
7	Muito	Não	Velho	Não	Gordo
8	Médio	Não	Jovem	Sim	Gordo

Fig. 4.11. O conjunto de exemplos depois de gerada a primeira regra

A geração da regra seguinte se dá de maneira semelhante: Escolhe-se outro par atributo/valor pelo mesmo critério. No caso, {Vegetariano}=”Sim”.

O subconjunto considerado para este caso é composto apenas da chave 2. Logo, a condição de parada (uma única classe) está satisfeita. O PRISM registra, então, a segunda regra encontrada:

R2: Se {Vegetariano} = “Sim” então Magro.

O conjunto de exemplos é novamente alterado com a exclusão das chaves que são cobertas pela regra recém criada (apenas a chave 2) e o processamento da classe “Magro” se encerra porque não há mais exemplos dessa classe.

Para o segundo processamento, o da classe “Gordo”, o conjunto é restaurado como no início, de acordo com o que é ilustrado na Fig. 4.5.

A escolha do par atributo/valor dessa vez recai em {Come} = “Muito”. Os exemplos desse atributo são os casos 3 e 7, que possuem uma única classe. O PRISM encontra mais uma regra então:

R3: Se {Come} = “Muito” então Gordo.

A alteração do conjunto de exemplos novamente se dá, com a exclusão das chaves que foram cobertas pela regra criada. A Fig. 4.12. mostra o conjunto considerado.

No	Come	Vegetariano	Idade	Diabetes	Classe
1	Pouco	Sim	Velho	Sim	Magro
2	Médio	Sim	Velho	Não	Magro
4	Pouco	Não	Velho	Não	Magro
5	Médio	Não	Jovem	Sim	Gordo
6	Pouco	Sim	Jovem	Não	Magro
8	Médio	Não	Jovem	Sim	Gordo

Fig. 4.12. o conjunto de exemplos depois de gerada a regra R3

Processa-se a regra seguinte do mesmo modo. Escolhido o par {Come}=”Médio” como condição, mas dessa vez não se consegue uma única classe. São selecionados as chaves 2, 5 e 8 e gera-se um conjunto auxiliar conforme o da Fig. 4.8.

Para este caso, então, é necessário a seleção de uma nova condição para que a regra se complete.

O processamento é o mesmo: Seleciona-se o par atributo/valor mais conveniente neste conjunto auxiliar. No caso, {Vegetariano} = “Não”.

Ainda no conjunto que está sendo considerado neste instante (Fig. 4.8), a exclusão dos que atendem a esta condição deixa somente a chave 2 que, obviamente, é um caso de única classe. O PRISM encerra aqui a busca pela 4ª regra:

R4: Se {Come} = “Médio” e {Vegetariano} = “Não” então Gordo.

O conjunto de exemplos é modificado, sendo excluídos os casos cobertos pela regra criada. Restam as chaves 1, 2, 4 e 6 que não possuem nenhum exemplo da classe “Gordo” que está sendo processada neste momento.

Como a classe “Magro” já foi processada, a execução é encerrada.

As regras encontradas, então:

R1: Se {Come} = “Pouco” então Magro.
R2: Se {Vegetariano} = “Sim” então Magro.
R3: Se {Come} = “Muito” então Gordo.
R4: Se {Come} = “Médio” e {Vegetariano} = “Não” então Gordo.

O PRISM e o ID3 são algoritmos clássicos de extração de regras de classificação. Eles possuem uma abordagem muito parecida, ambos partindo de uma função estatística que aumente a

probabilidade de ser selecionado um atributo conveniente para iniciar cada regra e gerando subconjuntos que são sucessivamente podados até que uma única classe seja isolada a fim de se construir uma regra ou parte dela. Algumas alterações do ID3, como o C4.5 e o C5 permitem uma grande flexibilidade nas árvores de decisão encontradas, e por isso mesmo possuem uma ampla aceitação no mercado. Os algoritmos ID3 e PRISM, em pseudocódigo podem ser encontrados no Anexo II.

4.6.2 Métodos de Extração de Regras de Associação

Neste trabalho o interesse está voltado para a extração de Regras de Classificação. É, porém, relevante mostrar aqui os métodos de extração de Regras de Associação porque estas usam um conjunto de métodos que possuem um nível de performance considerável em relação aos métodos tradicionais de Classificação[24] e podem ser adaptados, com mais algum ganho de esforço computacional, para extrair regras de classificação.

O processo de extração de regras de associação a partir de um conjunto de transações é relativamente complexo. O universo de combinações possíveis, dado um conjunto de N itens poderá potencialmente alcançar a ordem de 2^N o que é um valor impossível de ser computado quando por exemplo N toma a ordem de grandeza de 10^3 itens disponíveis, situação possível de ser encontrada em casos reais.

4.6.2.1 O Modelo Matemático

Conforme mostrado na seção 4.5.2.2, $c(X \Rightarrow Y) = s(X \cup Y) / s(X)$, ou seja, a confiança de uma regra é dada pelo suporte da união do antecedente com o conseqüente, dividido pelo suporte do antecedente.

Mas se $s(X) = Pr(X)$ e $s(X \cup Y) = Pr(X \cup Y)$ temos que $s(X \cup Y) \leq s(X)$ pois todo item i que pertença a X necessariamente pertence a $X \cup Y$ mas a recíproca não é verdadeira.

Com efeito, para um determinado conjunto de itens L que tenha suporte igual a s(L), qualquer conjunto L' tal que $L' \subset L$ terá suporte $s(L') \geq s(L)$ pelo mesmo motivo.

---

[24] Com efeito, é possível se encontrar as regras de associação com apenas duas varridas no conjunto de exemplos, conforme será mostrado na seção 4.6.2.3

Definição 1: Conjunto Freqüente de Itens (ou simplesmente Conjunto Freqüente) é um conjunto X $\subseteq$ I tal que s(X) é maior ou igual a um valor mínimo preestabelecido chamado Suporte Mínimo. Por analogia, os Conjuntos Infreqüentes ficam também definidos.

Teorema 1: Dado L um conjunto freqüente de Itens, qualquer subconjunto de L também o será.

Prova: Tem-se que dado L' $\subseteq$ L, s(L') $\geq$ s(L). Se L é um conjunto freqüente, s(L) $\geq$ Suporte Mínimo e, por transitividade, s(L') também o é.

Teorema 2: Extraindo-se todos os conjuntos freqüentes e seus suportes de um conjunto de transações é possível encontrar-se todas as regras e suas respectivas confianças usando-os como único subsídio sem a necessidade de nenhuma nova consulta às transações originais.

Prova: Na seção anterior foi mostrado que $c(X \Rightarrow Y) = s(X \cup Y) / s(X)$. Mas, de acordo com o Teorema 1, se $X \cup Y$ é um conjunto freqüente, X também o será. Logo, todos os antecedentes possíveis (que são subconjuntos de $X \cup Y$) estarão listados como conjuntos freqüentes.

O Teorema 2 indica que em se encontrando os conjuntos freqüentes de itens, a extração das regras com suas respectivas confianças para verificação de quais delas superam a confiança mínima estabelecida é trivial. Isso mostra que a complexidade reside, de fato, na geração dos conjuntos freqüentes.

O Teorema 1 indica que para se encontrar conjuntos freqüentes de itens de tamanho M é necessário encontrar os conjuntos de tamanho 1 até M-1 pois estes serão os seus únicos componentes possíveis.

O processo, então, deverá partir pela busca dos conjuntos freqüentes de itens de tamanho 1, chamados 1-conjuntos freqüentes de itens. A partir destes, combinando-os 1 a 1 e excluindo as combinações que não atingirem suporte mínimo, encontra-se os conjuntos de tamanho 2 (2-conjuntos freqüentes de itens ou 2-frequent itemsets).

Os restantes podem ser obtidos pela combinação de qualquer n-frequent itemsets com os 1-*frequent itemsets* sucessivamente[25] até que não seja gerado nenhum conjunto de tamanho n.

[25] Na verdade, a combinação com os 1- conjuntos freqüentes é uma abordagem simplificada, mas potencialmente ruim. Em termos práticos demanda menor esforço computacional combinar os (n-1)-conjuntos freqüentes que, unidos, formem um conjunto de tamanho n.

Como pode ser facilmente percebido, esse modelo, baseado no Teorema 1, reduz substancialmente o universo de pesquisa pelos conjuntos freqüentes.

Encontrados os conjuntos freqüentes e seus respectivos suportes, a tarefa torna-se tão somente a de comparar, dentre as combinações possíveis dos conjuntos freqüentes, quais superam a confiança mínima estabelecida[26] e assim gerar as regras.

4.6.2.2 O Apriori

Partindo-se das premissas estabelecidas acima, Agrawal et all. [Agrawal, 1994] apresenta um algoritmo básico que as cumpre na busca pelos conjuntos freqüentes, exatamente como foi mostrado. Ele busca, então, encontrar primeiro os conjuntos freqüentes com menor quantidade de itens e, a partir deles, usando o Teorema 1, tentar encontrar os seguintes partindo dos que foram encontrados a priori (daí o nome do algoritmo).

Considerando-se o exemplo de banco de dados de transação mostrado na Fig. 4.13, o Apriori fará uma série de iterações (leituras) e construirá, a cada uma delas, um conjunto de candidatos a conjuntos freqüentes, contando o número de ocorrências de cada conjunto candidato e então determinará os frequent itemsets (conjuntos freqüentes) pela eliminação dos que não atingem o suporte mínimo predeterminado.

TID	Itens
100	A C D
200	B C E
300	A B C E
400	B E

Fig. 4.13. Exemplo de Banco de Dados

Na primeira iteração, o Apriori simplesmente encontra o número de ocorrências de cada item. O conjunto de C_1 candidatos a 1-conjuntos é assim determinado e, excluídos os que não alcancem suporte mínimo, tem-se o L_1, os 1-conjuntos freqüentes como mostra a Fig. 4.14.

[26] Isso pode ser verificado com a simples divisão do suporte do conjunto inteiro pelo suporte do antecedente considerado.

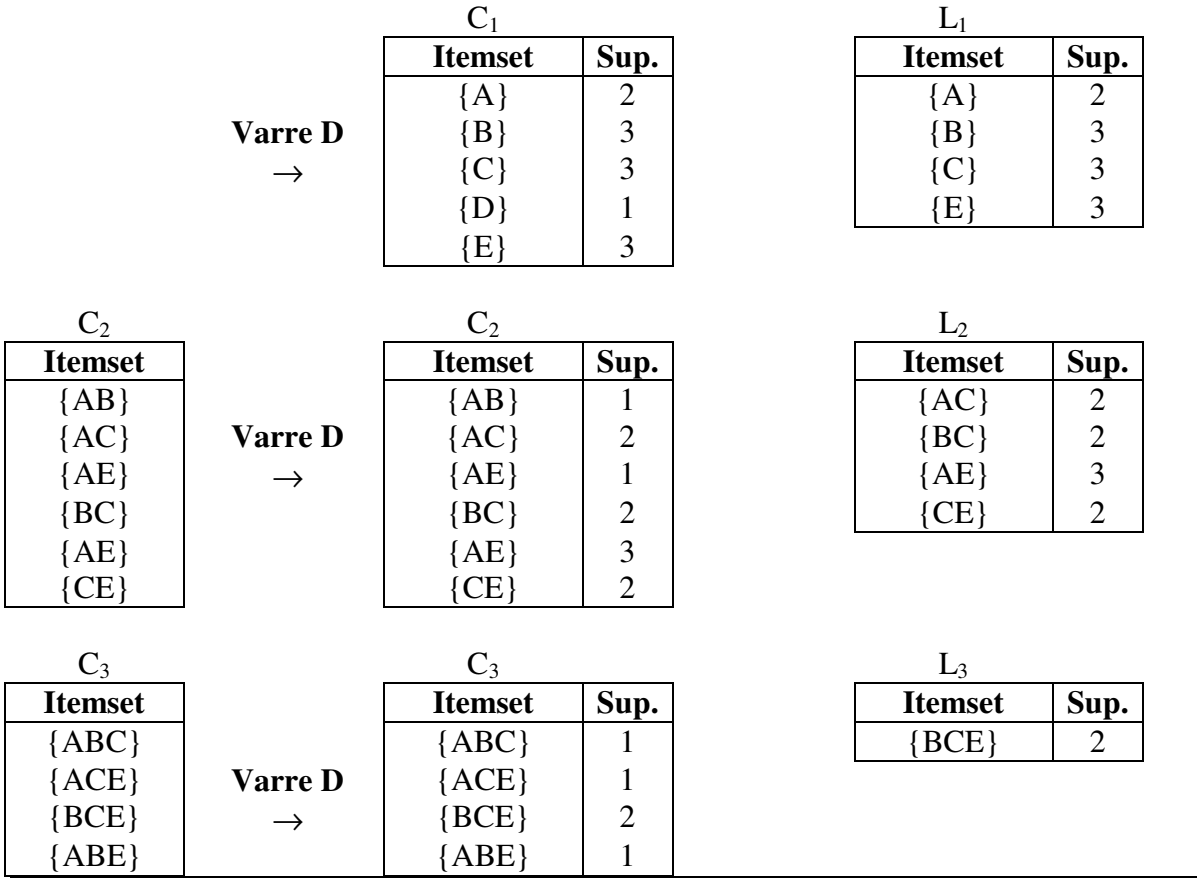

Fig. 4.14. Geração de Itemsets candidatos e Frequent Itemsets

O 2-conjuntos L_2 é obtido usando-se o produto cartesiano L_1 X L_1[27] para gerar os candidatos C_2 e depois suprimir os que não possuam suporte mínimo.

O conjunto de candidatos C_3 é gerado a partir de L_2 concatenando-os aos elementos de L_1 e obtendo conjuntos com 3 elementos. De C_3 extrai-se L_3 também pela observação de seus suportes.

O Apriori repete a operação com L_3 (concatenando-o com L_1) e gerando C_4 e L_4. A operação se repete até que para um determinado número de itens não seja encontrado nenhum conjunto freqüente.

Existem duas variantes do Apriori, na tentativa de otimizar a utilização de recursos de memória: o Apriori-TID e o Híbrido. Descrições destes métodos podem ser encontradas em [Srikant, 1995],

4.6.2.3 O Partition

As sucessivas leituras no banco de dados, principalmente em se tratando de uma quantidade muito

grande de informações, é um elemento crítico do desempenho dos algoritmos de extração dos conjuntos freqüentes.

Savasere et all. [Savasere, 1995] propõe um algoritmo chamado Partition, que faz efetivamente duas únicas leituras no banco de dados em disco. Isso será de particular interesse no escopo deste trabalho pois, em um ambiente Web, é sempre provável que os dados estejam distribuídos geograficamente o que aumenta consideravelmente o custo de cada varrida sobre todos eles. Além disso, como será visto, o algoritmo trabalha particionando os dados e, no caso distribuído, isso ganha uma conveniência ainda maior pois pode-se alojar partições ou conjuntos de partições em nós distintos, reduzindo-se assim drasticamente os custos de comunicação.

O Partition subdivide o banco em partições com tamanho máximo suficiente para ser alojada na memória[28].

Lê-se uma partição por vez e, para esta partição, segundo algum algoritmo de extração de conjuntos freqüentes (o Apriori por exemplo), encontra-se todos os conjuntos locais a ela. É importante ressaltar que, como os dados estão dispostos na memória, o número de vezes que eles são varridos é irrelevante[29].

Esse processo se repete em todas as partições subsequentes, gerando-se um conjunto de frequent itemsets locais L*. Assim, depois de lida a última partição, foi feita exatamente uma leitura do banco de dados inteiro.

É interessante observar que, seja L o conjunto de frequent itemsets do banco de dados, necessariamente tem-se que $L \subseteq L^*$. Isso porque, se um dado conjunto X for freqüente em todo o banco de dados, necessariamente ele será freqüente em pelo menos uma partição.[30]

Há, então, apenas a necessidade de se fazer uma nova leitura do banco de dados para que seja calculado o suporte efetivo de cada frequent itemset local e, suprimindo os que não alcancem

---

27 Nesse produto, naturalmente, são desconsiderados os pares repetidos

28 Por memória, aqui, pode-se entender um pool de máquinas que compartilhem entre si suas memórias num canal de comunicação local de altíssima velocidade.

29 Na verdade não é desprezível, embora a velocidade de acesso de dados na memória seja extremamente mais alta que em disco.

30 Isso é um fato que, para possuir rigor matemático, só se dará se a aproximação do cálculo de M, o número mínimo de ocorrências necessário para alcançar o suporte mínimo, for tomado por seu piso – o maior inteiro menor que M. M é calculado pela multiplicação direta do número de transações considerado pelo Suporte Mínimo requerido.

suporte mínimo em relação ao conjunto inteiro, finalmente se estabelecer o verdadeiro conjunto de frequent itemsets.

Os experimentos feitos mostraram que, em casos reais, o Partition supera o Apriori quando o conjunto de transações é muito grande e/ou bem distribuído (sem concentrações locais). Mais detalhes sobre os resultados comparativos podem ser encontrados em [Savasere, 1995].

Para conseguir um bom uso da memória, é interessante que o Partition gere o menor número de falsos conjuntos freqüentes possível. Quanto mais o conjunto L* supere L, maior será a possibilidade de ser necessário o alojamento de parte dele em disco, prejudicando o desempenho do algoritmo.

A probabilidade de se gerar falsos candidatos diminui com o número de partições feitas. Isso, porém, aumenta o tamanho de cada partição. A Fig. 4.15. mostra os resultados obtidos em experimentos variando o número de partições em uma mesma amostra.

Número de partições	Tamanho do maior conjunto local de *frequent itemset*	Tamanho médio dos conjuntos locais de *frequent itemset*	Tamanho dos candidatos globais a *frequent itemset*
2	91	89.0	93
4	100	82.5	108
10	149	109.1	170
20	273	211.9	381
30	463	344.1	673

Fig. 4.15. Comportamento do Partition em relação ao número de partições [Savasere, 1995]

A distribuição dos dados também modifica as características dos conjuntos locais. Uma distribuição uniforme tende a distribuir com relativa eqüidade os conjuntos freqüentes em todas as partições e isso diminui o número de falsos candidatos.

Num conjunto de transações comerciais, a promoção de alguns produtos, uma estação de escassez ou oferta excessiva de algum item ou outra variável qualquer pode determinar que, durante um período, a distribuição sofra uma distorção significativa. Caso o banco seja lido cronologicamente, as partições que abrigarem os dados referentes a este período certamente apresentarão muitos candidatos falsos. Isso sugere que seja feita uma leitura aleatória, para que a probabilidade de ocorrer discrepâncias dessa natureza seja reduzida.

A Fig. 4.16 ilustra uma comparação entre os desempenhos fazendo-se uma leitura seqüencial e outra aleatória. Os resultados experimentais mostram que o Partition possui desempenho muito melhor se os dados forem apresentados numa distribuição mais uniforme.

Número de partições	Leitura de blocos			
	Seqüencial		Aleatória	
	Soma de todos os conjuntos locais de *frequent itemset*	Tamanho dos candidatos globais a *frequent itemset*	Soma de todos os conjuntos locais de *frequent itemset*	Tamanho dos candidatos globais a *frequent itemset*
5	3961	3831	69	26
10	9281	6166	150	39
20	22871	8228	439	87
30	36311	9598	760	121

Fig. 4.16. Comportamento do Partition em relação à distribuição dos dados [Savasere, 1995]

O Partition possui, portanto, algumas variáveis que precisam ser corretamente sintonizadas para que ele apresente seu desempenho máximo. É possível, também, combiná-lo com alguns algoritmos e algumas idéias a fim de maximizar seu desempenho em todos os tipos de casos.

Toivonen [Toivonen, 1996] ainda apresenta um algoritmo que utiliza-se de uma amostragem do banco de dados para gerar um conjunto conveniente de potenciais candidatos a conjuntos freqüentes e com isso consegue uma alta probabilidade de varrer apenas uma vez o banco de dados, sendo que nos casos em que essa única varrida não se verifica, uma segunda apenas será necessária conforme o algoritmo Partition. Maiores detalhes sobre esse método podem ser encontrados em [Toivonen, 1996]

4.7 Conclusões

Este capítulo deu uma visão geral dos processos de aquisição de conhecimento indutivo, apresentando o conceito de Conjunto de Treinamento e dando uma visão geral do processo como um todo, de sua iteratividade e suas características de feedback a partir de algum sub-processo para outro que esteja localizado mais atrás a fim de se melhorar os resultados. Isso servirá para ilustrar a necessidade de os métodos de extração de conhecimento apresentarem aos usuários a possibilidade de restabelecer os parâmetros sob o qual ele deverá ser executado para que a tarefa possa ser refeita, se for o caso, conforme será explorado neste trabalho.

Mostrou-se diversos tipos de conhecimento que podem ser extraídos por métodos automáticos, e a utilidade de cada um deles; além disso destacou-se as Regras de Classificação e de Associação, que serão de particular interesse aqui. A primeira tratará efetivamente do tipo de conhecimento a ser explorado e a segunda será usada numa maneira alternativa de abordagem.

As regras de classificação podem ser armazenadas em estruturas de dados do tipo Árvore de Decisão, que são melhores de serem tratadas computacionalmente embora tragam agregadas a si uma falha no tocante à sintaxe, ou em Listas de Regras, que se eximem do problema sintático embora tenham custo de manutenção e extração significativamente mais alto.

Em seguida mostrou-se rapidamente o problema semântico, que apesar de transcender o escopo deste trabalho, é importante característica a ser considerada pelos engenheiros e facilitadores do conhecimento.

As Regras de Associação foram mostradas com base num formalismo matemático, suportado na teoria dos conjuntos, e terão sua importância justificada a posteriori. Mostrou-se também as variáveis estatísticas de Confiança, que exprime a "força" de uma regra, ou seja, a probabilidade de ela se verificar de fato, dadas as condições de que ela necessita, e Suporte que exprime "a freqüência" da regra, ou seja, a probabilidade daquela regra ser de fato um conhecimento potencialmente útil, não uma chance isolada ou uma situação atípica perdida num emaranhado de dados armazenados.

As regras, neste trabalho, serão de Classificação. Um levantamento de dois métodos principais, o ID3 e o PRISM, é feito, mostrando ilustrativamente como estes geram as regras a partir dos dados e apresentando suas abordagens por árvore de decisão e lista de regras respectivamente.

Em seguida são mostrados, a partir de um modelo matemático, métodos de extração de regras de associação – particularmente o Apriori, que também é ilustrado, e o Partition, uma versão melhorada daquele. Esta abordagem é interessante para este trabalho porque, será usado um método de extração de regras de associação para a extração de regras de classificação.

O próximo capítulo vai mostrar, a partir dos métodos aqui apresentados, um que atenda às características desejáveis no escopo deste trabalho, e este será o ponto que justificará o estudo das regras de associação.

É nesse capítulo que será, também, mostrada a abordagem adotada em RBC, que implementa dois métodos de resgate distintos num mesmo produto e sugere que haja um hibridismo entre o RBC

(método dedutivo) e a extração de Regras (método indutivo) em que esta trabalhará na extração de conhecimento que será útil e aplicável na melhoria e no monitoramento do funcionamento deste.

Capítulo 5
Aplicando RBC e Indução automática ao Help Desk

5.1 Introdução

Este capítulo é composto de duas partes distintas, que se completam numa integração parcial em que a abordagem por aprendizagem indutiva serve como método de aferição da abordagem por casos.

Inicialmente é mostrado como a tecnologia de RBC é aplicada efetivamente ao Help Desk, a fim de melhorar a eficiência do diálogo com o cliente, permitindo que seu problema seja solucionado rapidamente e que o sistema consiga reter os novos problemas resolvidos, a fim de efetivar a aprendizagem automática.

A primeira parte, que vai até a seção 5.5, mostra um levantamento das tarefas e das responsabilidades que caberá à implementação do RBC atender. É mostrada uma visão do que seria o RBC em Help Desk, depois mostra-se os objetivos e as características desejáveis que essa abordagem deverá prover ao usuário, e uma visão de alguns métodos disponíveis, particularmente duas ferramentas disponíveis no mercado, haja vista apresentarem, cada uma delas, abordagens interessantes para o que será apresentado neste trabalho.

A segunda parte, que segue a partir da seção 5.6, apresenta os mecanismos e os métodos de implementação da solução desejada, apresentando duas alternativas de interface distintas para que o usuário tenha mais flexibilidade em suas consultas, mostrando para cada uma delas uma estrutura de dados eficiente – baseada em bancos de dados relacionais e SQL – bem como a maneira como funcionarão os métodos para resgate e inclusão de novos casos em cada uma delas.

Em seguida é apresentada a aplicação efetiva da extração de conhecimento indutivo, com o uso de uma abordagem de extração de regras de classificação por método de regras de associação, no escopo deste trabalho.

Inicialmente são mostrados os motivos que forçam a necessidade de se extrair regras de dados de um ambiente Help Desk. Mostra-se os objetivos que deverão ser alcançados, entre os quais o suporte tático ao método de apoio ao usuário por RBC, e as características que o método escolhido deverá apresentar – particularmente em eficiência e flexibilidade – para atender a contento.

Finalmente é mostrado o método proposto nesse trabalho, que utiliza uma abordagem para regras de associação modificada – e com alguns ganhos – a fim de se adequar à busca por regras de classificação. É mostrada uma estrutura de dados adicional que será usada além de uma descrição do método proposto de forma algorítmica.

No final uma conclusão faz um balanço de tudo que foi mostrado, encerrando a apresentação do conteúdo deste trabalho de dissertação.

5.2 RBC para Help Desk

Segundo Aha [Aha, 1997], "[o RBC] captura o conhecimento em forma de casos melhor que as regras, e (o conhecimento) pode ser estendido incrementalmente". Com efeito, no RBC a busca por soluções armazenadas é feita a partir de crescentes restrições e pode ser reduzida a sucessivas "podas" em um conjunto de dados, ao invés de uma busca exaustiva numa árvore de decisão ou num conjunto de regras.

5.3 Objetivos do RBC para Help Desk

Aplicações de Help Desk usando RBC não são novidade. De fato, a literatura aponta uma série de sistemas dessa natureza [Aha, 1997], sendo que a principal vantagem reside no fato de, essencialmente, a busca de uma solução com o uso de uma abordagem baseada em casos ser extremamente eficiente.

Neste trabalho, o uso do RBC se dá efetivamente na seleção, entre um conjunto de casos armazenados, daqueles que possam de fato solucionar, ou ajudar a solucionar um problema atual de um usuário.

De modo geral, o conceito é restringir, a base de casos àqueles casos que possam ser potencialmente úteis na busca da solução (Resgate, conforme mostrado no Capítulo 3, seção 3.12.1). Essa operação de RBC, apropriadamente, pode ser feita com o uso de tecnologias de Bancos de Dados tais como Índices, Tabelas Hash além de uma série de outras técnicas de performance já comprovada em Bancos de Dados.

Por outro lado, o uso de regras (particularmente indutivas, extraída a partir dos problemas armazenados) poderia ser apresentado como uma solução alternativa, já que traria uma informação (e uma justificativa) mais racional para o usuário do ponto de vista lingüistico. Com efeito, idealmente as regras apresentariam ao usuário uma explicação com "Se ... então ... " que seria mais convincente e mais esclarecedora.

Mas a busca em árvores de decisão pode ter custo extremamente alto, principalmente se for considerado que os custos para disponibilização de determinados atributos podem ser proibitivos (ou, em ocasiões especiais, tais atributos podem ser simplesmente desconhecidos), e isso inviabilizaria todo o processo pois seria impossível verificá-lo em regras que necessitassem desse valor para serem consideradas.

Por outro lado, uma varredura exaustiva nas árvores de decisão a partir de um conjunto aleatório de valores de atributo – como é, potencialmente, a característica de um problema de entrada apresentado pelo usuário já que não se pode saber previamente quais atributos ele vai de fato mostrar – é um problema de alta complexidade computacional já que possivelmente todos os caminhos precisarão ser visitados para se verificar cada atributo.

É assim que, apesar de ser menos elucidativo, o RBC se torna uma tecnologia mais aplicável ao contexto de Help Desk no tocante ao suporte pela busca da solução de um problema.

5.4 Características desejáveis da abordagem RBC

O usuário, desejavelmente, poderá tentar encontrar a solução do seu problema a parir de duas formas diferentes:

i) partir de palavras-chave ou

ii)pelo estabelecimento de parâmetros (índices) do seu problema atual.

As palavras-chave poderão deixar o usuário mais "próximo" da descrição pessoal que ele faz do problema. De fato, se ele puder descrever o problema atual e o sistema capturar, a partir de sua descrição, informações que aproximem seu problema de algum dos casos armazenados, a busca pode se tornar extremamente simplificada, principalmente em se tratando de usuários que não tenham familiaridade com termos e expressões técnicas.

Por outro lado a busca textual assim procedida pode levar à recuperação de um número muito grande de casos ruins, que em nada contribuam para a solução [Martins, 2000].

Na prática, os problemas que podem ser encontrados residem no fato de que os termos técnicos que descrevem um problema do ponto de vista dos especialistas serem potencialmente diferentes dos que podem ser usados pelo usuário. Esta discrepância pode se tornar ainda mais relevante se for levado em consideração as diferenças regionais e culturais, e naturalmente a expressão oral ou escrita entre os diversos tipos de usuários que o sistema poderá atender.

Por outro lado, a probabilidade de alguns conjuntos de palavras aparecerem em uma grande quantidade de problemas diversos é alta, diminuindo assim a chance de serem recuperados apenas os casos relevantes ao contexto atual do usuário.

Neste aspecto, o resgate baseado em índices se mostra mais acurado. Se o usuário apontar alguns atributos identificadores de seu problema, e se esses atributos puderem ser endereçados em casos armazenados, ainda que parcialmente ou por aproximação, a recuperação deverá ter um custo inferior, e a probabilidade de serem resgatados casos irrelevantes ou espúrios diminui significativamente [Brown, 1995].

Outra vantagem do resgate baseado em índices é que o usuário não precisa conhecer de fato o valor de nenhum atributo em especial, já que qualquer índice identificado já permitirá ao sistema restringir os casos armazenados. Isto é particularmente útil devido ao fato de os usuários muitas vezes desconhecerem algumas informações ou a maneira correta de expressá-las.

Para viabilizar e potencializar as vantagens desse fato, o sistema precisará prover uma interface que permita ao usuário selecionar que atributo deseja quantificar, e a partir desta entrada, ser-lhe-ão apresentadas todas as possíveis descrições que porventura cubram o problema que está sendo exposto. Um modelo interessante de interfaces desse tipo é apresentado em [Aha, 1997].

5.5 Métodos e ferramentas RBC para Help Desk

A abordagem por Raciocínio Baseado em Casos, segundo [Martins, 2000] pode variar dentro do algoritmo dos 4R em qualquer um dos seus diferentes processos:

- Indexação – por atributos, baseada em diferenças, baseada em similaridades, por aprendizagem indutiva
- Armazenamento – seqüencial ou hierárquico (redes compartilhadas e árvores de discriminação)
- Recuperação – por casamento de padrão (textual), baseado em índices, via algoritmos seqüenciais, via algoritmos paralelos, via algoritmos indutivos, baseada em banco de dados, baseada em similaridades
- Adaptação – que pode ser automática, manual ou simplesmente não ser feita

Uma descrição mais detalhada de cada um desses métodos pode ser encontrada no Capítulo 3.

É importante ressaltar que a escolha de um determinado método para a execução de uma tarefa implicará nas opções disponíveis para as tarefas restantes. Neste trabalho, por exemplo, serão de particular interesse a recuperação baseada em atributos e por casamento de padrão. Isto implicará em restrições correspondentes nos métodos de armazenamento e indexação.

Dois sistemas de Help Desk existentes e documentados serão de particular interesse na ilustração de métodos úteis e interessantes para esta implementação do RBC em Help Desk: o Clire™ e o Help!CPR™.

5.5.1 O Clire™: Indexação pelo atributo

O Clire é um sistema já desenvolvido e apresentado em [Aha, 1997] que tem a vantagem de apresentar uma interface ao usuário na qual ele pode selecionar um atributo qualquer, de acordo com o conhecimento que ele possui sobre o próprio problema, e estabelecer um valor para ele. O Clire automaticamente atualiza uma lista de casos selecionados que possam de fato possuir relação com o problema em curso até que o usuário alcance uma solução satisfatória – ainda que carecendo de modificações.

5.5.2 O Help!CPR™: Indexação textual

O Help!CPR por sua vez tem a vantagem de apresentar ao usuário uma série de perguntas, às quais ele vai respondendo seqüencialmente. Isso é indexação por atributos, porém as respostas do usuário devem ser feitas em linguagem natural (inglês para o caso) e o sistema tentará, a partir de um

casamento de padrões textuais, recuperar os casos que se aproximem da descrição feita pelo usuário, a fim de encontrar um que seja satisfatório – mais uma vez ainda que haja necessidade de adaptações.

Os dois sistemas apresentam abordagens que vêm se consagrando, no tocante à recuperação de casos em aplicações para Help Desk. Para este trabalho, contudo, algumas funcionalidades a mais devem ser consideradas, com vistas à geração de uma base de casos o menor e o mais eficiente possível, sem nenhum prejuízo para a exatidão das soluções apresentadas.

5.5.3 Outras funcionalidades a serem consideradas

Assim, serão consideradas duas maneiras diferentes de se introduzir casos no sistema, de acordo com a classificação mostrada em [Martins, 2000]: *casos reais* e *casos abstratos*.

Os primeiros deverão ser extraídos a partir das ocorrências registradas. Obviamente, uma análise do especialista será necessária para que uma ocorrência possa de fato ser considerada um caso[31].

Uma ocorrência, para ser "promovida", deverá possuir duas características: *Originalidade* e *Relevância*.

A primeira pressupõe que aquele exemplo considerado abriga a solução de um problema cujas características ainda não estão inteiramente refletidas na base de casos vigente – pois nesse caso corre-se o risco de introduzir na base de casos uma redundância ou, o que é pior, uma contradição, coisa que, conforme segue, também deve ser evitada.

A segunda faz referência à potencial utilidade do caso que está sendo considerado. Se a ocorrência for irrelevante estatisticamente, ou representar algum tipo de anomalia (contradição com algum caso armazenado ou um fato isolado que fuja ao aceitável naquele domínio), esta ocorrência evidentemente não deve ser considerada.

Os casos abstratos devem ser criados pelo especialista a fim de cobrir os problemas que potencialmente serão encontrados pelos usuários, mas que ainda não tenham ocorrido de fato, e por esse motivo não se encontram na base de ocorrências.

Tanto para os casos abstratos quanto para os casos reais há ainda que se considerar o fato de que um conjunto possivelmente diferente de atributos deve representar cada um deles (casos bem

[31] Isso poderá ser parcialmente implementado por um Sistema Especialista no futuro

estruturados) e que, para cada atributo, o casamento poderá ocorrer de forma exata ou parcial. Em cada atributo de caso, portanto, deverá existir informação suficiente para determinar o tipo de casamento desejado.

Neste trabalho, então, é feita uma estrutura viável de manipulação de casos para atender a todas as características mostradas na seção 5.4.

5.6 Estrutura de dados: associando RBC a Bancos de Dados

Os bancos de dados são potenciais aliados no processo de RBC pois fornecem, em função de suas vantagens e de sua própria natureza, uma boa base para o armazenamento dos dados de sistemas deste tipo bem como uma infra-estrutura de eficiência comprovada para as operações de resgate [Martins, 2000].

Especifica-se neste ponto, que estruturas de RBC a serem utilizadas estarão alojadas em tabelas de bancos de dados relacionais. Isso proverá facilidades nas operações de indexação, armazenamento e resgate com o uso das facilidades providas pela linguagem padrão SQL[32], que foi apresentada na seção 3.12.2.

5.6.1 Manipulação de casos por casamento de padrão

A Fig. 3.1 apresenta o modelo, em base relacional, do esquema ideal aqui proposto para se permitir a manipulação de casos através do casamento padrão (via recuperação textual).

O conceito é que, para cada caso, o especialista estabelecerá uma série de perguntas às quais o usuário responderá em linguagem usual. O casamento entre a resposta do usuário e o resgate provido pelo sistema se dará por palavras-chave, e os casos que possuírem ao menos uma palavra-chave citada pelo usuário será trazido para ele. A apresentação do resgate se dará ordenadamente de acordo com o número de casamentos encontrados, aumentando assim a probabilidade de surgirem rapidamente os casos pertinentes ao problema.

Uma tabela de sinônimos será utilizada, e deverá ser preenchida pelo especialista no instante em que ele estabelece as palavras-chave de uma pergunta, a fim de minimizar as diferentes formas de expressão, que surgem em função dos fatores mostrados na seção 5.4. Esta tabela de sinônimos permitirá que, ainda que a resposta dada pelo usuário não contenha nenhuma das palavras-chaves

[32] Neste trabalho será usada uma versão de SQL para o SGBD Oracle

preestabelecidas para aquela pergunta daquele caso, possa ser elucidativa por conter algum dos sinônimos associados e assim o caso será selecionado sem nenhuma restrição.

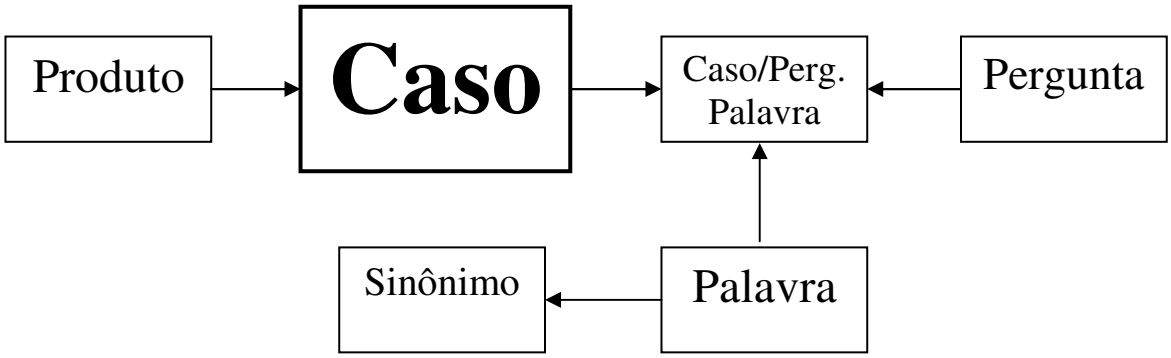

Fig. 5.1 Diagrama Relacional do modelo de manipulação por casamento padrão

Um script de criação dessas tabelas em linguagem SQL (Oracle), bem como dos demais objetos de banco de dados associados a elas é apresentado no Anexo I.

Para essa abordagem, os casos não podem ser trazidos diretamente da base de ocorrências (casos reais), haja vista a óbvia ausência de estruturas de pergunta/resposta em meras ocorrências armazenadas. É, porém, facultada a inclusão de casos manualmente (casos abstratos) ou a utilização de algum caso que já esteja identificado, desde que acrescentando-se a ele toda a estrutura de pergunta/resposta mostrada acima.

Segundo o modelo proposto, a tabela de Casos registrará os dados referentes ao caso em si, como Rótulo e Descrição. Cada caso poderá possuir uma série de perguntas a serem respondidas e uma lista de palavras-chave associadas (relacionamento Caso/Perg./Palavra).

Uma busca, então, deverá trazer, para a pergunta selecionada e respondida pelo usuário, os casos (no relacionamento Caso/Perg./Palavra) que possuam na sua lista de palavras-chave pelo menos uma que tenha sido escrita por ele, ou um dos seus correspondentes sinônimos.

Assim, a cada pergunta, pode-se identificar uma lista de casos que a “atendam”, e ainda ordená-la de acordo com o número de casamentos encontrados, aumentando a probabilidade de mostrar primeiro os casos de maior relevância conforme mostrado acima.

A cada pergunta, naturalmente, uma nova lista de casos será obtida, que é unida à já existente, gerando uma outra que deverá ser reordenada de acordo com novo número de casamento encontrados para cada caso.

Cada palavra-chave, por sua vez, pode possuir uma lista de sinônimos associados, conforme exemplificado na Fig. 5.2, permitindo ao especialista a prevenção (o mais ampla possível) de expressões, abreviações ou palavras similares que podem ser usadas pelo usuário no lugar da expressão propriamente prevista por ele.

Isso será particularmente útil para os casos reais que foram identificados, a fim de que eles possam ser utilizados também desse modo (já que, conforme foi visto, não possuem agregados a si esta estrutura de perguntas, palavras e sinônimos).

Tabela de Palavras-Chave

...	**...**
20	Funciona
...	**...**

Tabela de Sinônimos

...	...	...
...	20	Liga
...	20	Inicia
...	20	Trabalha
...	...	...

Fig. 5.2 Exemplo de Hash entre uma palavra-chave e seus sinônimos

5.6.2 Manipulação de casos por casamento de atributo

Martins comenta em [Martins, 2000] a vantagem básica da utilização do resgate baseado em índices em relação ao casamento padrão pois reduz o custo na busca dos casos. O modelo apresentado na seção anterior é na verdade uma tentativa de reduzir a busca por recuperação textual a uma busca por índices. As palavras-chave nada mais são que valores pontuais pelos quais o sistema procura no instante do resgate.

A Fig. 5.4 lustra o esquema relacional que será usado para fazer a recuperação efetivamente baseada em atributo. Pelo modelo, cada caso possui uma lista particular de atributos (o que

permitirá a representação bem estruturada, de acordo com [Martins, 2000]) e associado a cada um deles uma lista de valores discretos.

A discretização dos valores dos atributos é necessária para viabilizar o casamento parcial (coisa que para a recuperação por casamento padrão não pode ser feita) já que, uma vez estabelecidos e ordenados todos os possíveis valores que um atributo pode possuir, um caso-alvo pode ter um valor de atributo facilmente identificado na lista e sua distância[33] em relação ao caso que está sendo considerado também será facilmente calculada.

Cada valor possível de um atributo é formado de uma chave para ordenação e do próprio valor a ser considerado. Também é identificado, para cada atributo, o *tipo* de dados associado. Basicamente Caracter, Data e Números.

Para permitir o casamento parcial, foi criada uma variação, que estabelece a distância máxima que o valor, no caso alvo, poderá estar em relação ao caso a ser comparado. Isso modifica ligeiramente o padrão da teoria de RBC que estabelece, normalmente, que as métricas de similaridades (como a distância considerada) sejam um atributo da operação de resgate. Aqui, em virtude da enorme variação da natureza dos atributos foi necessário se estabelecer os parâmetros para a métrica de similaridade (a distância máxima tolerada) como uma característica intrínseca ao próprio atributo.

Na Fig. 5.3 está ilustrado um exemplo de casos utilizando dois atributos, Sistema Operacional e Velocidade (Clock) de processador. Note que, para casos diversos, as distâncias máximas permitidas para o Clock são distintas (já que provavelmente trata-se de problemas com características diferentes) bem como apenas um deles utiliza o atributo Sistema Operacional.

[33] Distância Cartesiana.

Tabela Atributo/Valor/Caso

Caso	Atr.	Valor	Dist.
2013	25	412	0
2013	29	721	1
3020	29	723	0

Tabela Atributo

...	...
25	Sist. Operacional
...	...
29	Clock da Máquina
...	...

Valores dos Atributos

Ord.	Cod.	Atr.	Descrição
1	410	25	Windows NT
2	411	25	Windows 95/92/2000
3	412	25	Unix
4	413	25	OS/2
...	...	...	
1	719	29	< 100
2	720	29	Entre 100 e 200
3	721	29	Entre 201 e 400
4	722	29	Entre 401 e 700
5	723	29	> 700

Fig. 5.3 Um exemplo de casamento parcial e total

Isso vai permitir que, ainda que não tenha o valor exatamente igual ao que está sendo considerado no momento, um caso possa ser selecionado em função da proximidade que ele possui com o que esta sendo analisado.

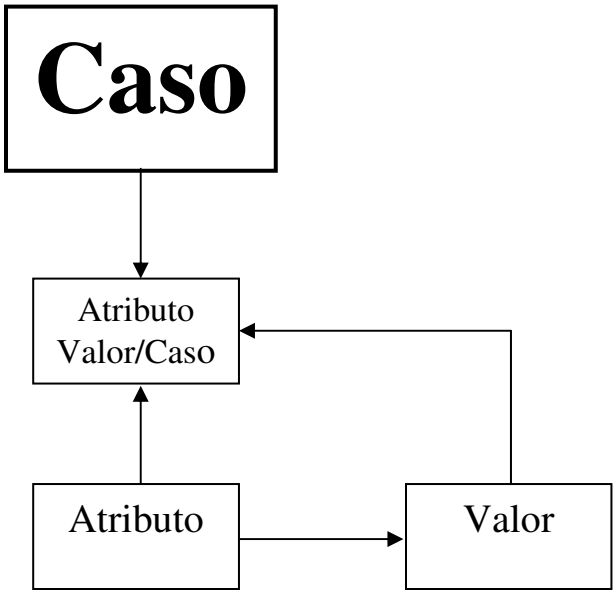

Fig. 5.4 Diagrama Relacional do modelo de manipulação por casamento de atributo.

Um script SQL que cria essa estrutura num banco de dados (Oracle) , bem como dos demais objetos de banco de dados associados a elas também pode ser encontrado no Anexo I

5.7 Retenção de novos casos

Uma vez estabelecida a estrutura de armazenamento dos casos, no caso da recuperação por casamento padrão bem como na recuperação pelo atributo, cabe estabelecer o método pelo qual serão apresentados, para o sistema, os novos casos.

Basicamente, em RBC, duas são as maneiras pela qual os novos casos podem ser inseridos no sistema:

1. Por inclusão direta, que são os casos que precisam existir a princípio, a fim de que o sistema possa operar com casos já existentes. Basicamente os casos incluídos diretamente ainda podem ser classificados em Reais e Abstratos, conforme mostrado na seção 5.5.3.
2. Por aprendizagem, ou retenção. Numa situação real, se um caso é modificado/adaptado e sua solução atende de fato a um problema atual, o sistema deverá naturalmente retê-lo a fim de estabelecer o que de fato se conhece por “Aprendizagem por Experiência”.

Os Casos Reais serão trazidos da Base de Ocorrências, conforme mostrado na seção 5.6.1, e precisarão ser “preparados” para serem úteis na recuperação por casamento padrão ou por atributo. O sistema provê duas interfaces para viabilizar isso:

No casamento padrão, cada ocorrência que for selecionada deverá ser acrescida de uma lista de perguntas, que podem ser criadas naquele momento ou reutilizadas pelo especialista a partir de outras já existentes, desde que ele refaça a lista de palavras-chave e sinônimos associados. Isso garantirá haver para cada uma das perguntas associadas aos casos suas respectivas palavras-chave e seus respectivos sinônimos. Isto será tarefa do especialista, que estabelecerá assim o diálogo básico que esse módulo do sistema poderá ter com o usuário.

No casamento por atributo, é possível uma seleção automática de uma série de ocorrências, sob a supervisão mas não necessariamente a intervenção direta do especialista. Este, porém, poderá, depois de feita a seleção, modificar alguns parâmetros de relevância em cada caso, como a discretização dos valores dos atributos e a variação que será tolerada nos atributos daquele caso para viabilizar o casamento parcial.

Os casos abstratos serão tratados de maneira semelhante: O especialista vai manualmente estabelecer uma a uma todas as perguntas e os atributos que julgar necessário, e o sistema imediatamente passará a manter o caso disponível na base para futuras buscas.

A retenção dos casos será feita de maneira simplificada. Conforme sugerido em [Martins, 2000], domínios extremamente complicados ou pouco constantes (como é particularmente o caso do Help Desk) não se mostram convenientes para se aplicar métodos automáticos de adaptação.

Neste trabalho, conforme já foi mostrado no capítulo 2, a eventual incapacidade do sistema de resolver satisfatoriamente o problema do usuário implicará na intervenção de uma equipe de especialistas. A essa equipe cabe trazer a solução mais adequada ao caso, e utilizar-se dos mecanismos de inclusão manual para prover ao sistema a retenção daquele novo problema.

5.8 Recuperação de casos

A tarefa de principal utilidade em RBC é precisamente a Recuperação, ou o Resgate dos casos mais interessantes ao problema atual.

O uso de Bancos de Dados para manter os registros de casos vai prover algumas características que serão de particular interesse para este trabalho: velocidade e precisão.

Com efeito, o uso de um Banco de Dados Relacional e de consultas em SQL garantem que os métodos de resgate aqui propostos de fato funcionarão com a eficiência e a velocidade que os modernos bancos de dados provêem.

5.8.1 Proposta de recuperação textual

Para a recuperação textual, o sistema deverá interagir com o usuário a fim de obter dele respostas sucessivas para as perguntas que ele desejar (ou souber) responder.

Cada resposta trará, da base de casos, uma coleção de exemplos que, idealmente, possuam uma probabilidade alta de atender ao seu problema. Um processo de interseção entre os exemplos trazidos de cada pergunta, somados ao número de casamentos encontrados (o número de palavras-chave) e classificados de acordo permitirá que a lista seja atualizada corretamente.

A recuperação, portanto, se dará de acordo com o seguinte algoritmo:

RECUPERA_TEXTUAL(CASO_SATISFATÓRIO)

Início
Lista_Candidatos := ∪(*casos da base*)
Caso_Satisfatorio := Falso;
Le_produto(:*produto*)
Repita
Separa_Palavras(*Lista*, :*Pergunta*)
Seleciona_Casos(*Casos_Selecionados*)
Lista_Candidatos := *Lista_Candidatos* ∩ *Casos_Selecionados*
Testa_Caso(*Caso_Satisfatorio*)
Até *Caso_Satisfatório* ou *Lista_Candidatos* = ∅
Fim

onde *Lista_Candidatos* e *Casos_Selecionados* são variáveis que armazenarão coleções (possivelmente) distintas de casos; *Separa_Palavras* é um procedimento que permite o usuário selecionar e responder a uma pergunta e separa as palavras que ele usou na sua resposta, mantendo em *:pergunta* o apontador para a pergunta e na tabela Lista o conjunto de palavras encontrado; *Seleciona_Casos* é um procedimento que faz uma pesquisa, em função da pergunta selecionada e da resposta oferecida, dos casos na base de casos, levando em consideração a quantidade de casamentos a fim de se classificar o resultado de acordo com a probabilidade de acerto. A consulta que atende a esse quesito, em SQL, será exatamente conforme a Fig. 5.5:

```
Select cd_caso, count(cd_caso)
From tb_caso_perg_palavra
Where cd_pegunta = :Pergunta and
      (tb_caso_perg_palavra.cd_palavra in
      (select cd_palavra from tb_palavra where ds_palavra in
            (select ds_palavra from Lista))
      or
      tb_caso_perg_palavra.cd_palavra in
      (select tb_sinonimo.cd_palavra
       from tb_sinonimo
       where tb_sinonimo.ds_sinonimo in
             (select ds_palavra from Lista)))
group by cd_caso;
```

Fig. 5.5 Consulta em base de casos em SQL para resgate textual

Testa_Caso atualiza *Caso_Satisfatório* para Verdadeiro se foi encontrada uma solução que atenda, e exclui os que não atendam ao problema, de acordo com o usuário.

No final do procedimento o Especialista será acionado caso não seja possível encontrar uma solução satisfatória para o usuário (de acordo com o conteúdo da variável *Caso_Satisfatório*).

Os casos que vão sendo selecionados podem ser apresentados ao usuário em componentes do tipo *ListBox*, que permitirão que ele selecione algum para ver a solução sugerida e depois informe se a solução foi satisfatória ou não. Cada caso "reprovado" pelo usuário é imediatamente excluído da lista e cada nova pergunta respondida por ele também servirá para excluir mais uma quantidade de candidatos.

Antes de encerrado o procedimento, a tabela auxiliar (*Lista*) é excluída (drop table).

5.8.2 Proposta de recuperação pelo atributo

No caso da recuperação pelo atributo, é mostrada ao usuário uma lista de atributos disponíveis, e o usuário pode selecionar qualquer um deles e estabelecer, de acordo com seu problema atual, o valor que ele possui. O Sistema vai recuperar, a um comando (o clicar de um botão) do usuário, todos os casos armazenados que satisfaçam as condições estabelecidas pelo usuário. Cada alteração no conjunto de atributos/valores feita pelo usuário resultará numa coleção diferente de casos recuperados.

A recuperação segue, portanto, de acordo com o seguinte algoritmo:

```
RECUPERA_ATRIBUTO (CASO_SATISFATORIO)
Início
    Caso_Satisfatorio := Falso;
    Repita
        Le_Problema(AtributosAtuais)
        Seleciona_Casos(Casos_Selecionados)
        Testa_Caso(Caso_Satisfatorio)
    Até Caso_Satisfatório ou Casos_Selecionados = ∅
Fim
```

onde *Casos_Selecionados* é uma variável que armazenará coleções de casos; *Le_Problema* é um procedimento que apresenta os atributos para o usuário e cobra dele a seleção de algum(s) com seu(s) respectivo(s) valor(es) armazenando tudo isso na tabela *AtributosAtuais* (a lista de valores possíveis para um atributo é conhecida pelo sistema, de acordo com o modelo mostrado na seção 5.6.2); *Seleciona_Casos* traz uma lista de casos que atendam, completa ou parcialmente os requisitos do problema que o usuário apresentou. A consulta que satisfaz essa pesquisa fica, em SQL, exatamente como mostra a Fig. 5.6:

```
Select cd_caso
From tb_atributo_valor_caso v, AtributosAtuais a
Where v.cd_caso not in (select cd_caso from casos_reprovados) and
      (v.cd_atributo = a.cd_atributo and
  v.vl_classificacao between
      a.vl_classificacao – v.vl_distancia_maxima and
              a.vl_classificacao + v.vl_distancia_maxima);
```

Fig. 5.6 Consulta em SQL para resgate por atributo

Testa_Caso atualiza *Caso_Satisfatório* para Verdadeiro se foi encontrada uma solução, e exclui os que não atenderem de acordo com o usuário, mantendo uma lista temporária destes casos numa tabela auxiliar.

No final do procedimento, o Especialista será acionado, caso não seja possível encontrar uma solução satisfatória para o usuário (de acordo com o conteúdo da variável *Caso_Satisfatório* ou se CasosSelecionados for igual a ∅).

Os casos que vão sendo selecionados podem ser apresentados ao usuário em variáveis do tipo *ListBox*, que permitirão que ele selecione algum para ver a solução sugerida e depois informe se a solução foi satisfatória ou não. Cada caso "reprovado" pelo usuário registrado na tabela auxiliar (casos_reprovados), não será mais resgatado numa futura consulta.

Antes de encerrado o procedimento, a tabela auxiliar (casos_reprovados) é excluída (drop table).

5.9 Extração de Regras para Help Desk

Conforme foi mostrado anteriormente na seção 5.3, o RBC é, em termos de performance, a técnica mais adequada para um sistema automatizado de Help Desk buscar a solução de um problema atual do usuário pois faz apenas seleções sucessivas em um conjunto de dados, o que possui performance significativamente melhor que a busca em árvores de decisão.

Para outras abordagens, notavelmente a extração de informações a partir das ocorrências de um sistema de Help Desk, seja para gerência estratégica dele próprio, seja no levantamento de informações capitalizáveis na promoção da qualidade, o RBC não se aplica idealmente. Neste caso, a extração de regras que apresentem regularidades e generalizações encontradas nos dados podem ser de maior utilidade pois comunicam uma informação mais inteligível do ponto de vista lógico, da forma SE ... ENTÃO ... e sendo respaldadas pela amostra considerada de ocorrências ou casos registrados.

Existe uma infinidade de aplicações que justificam os métodos automáticos de extração de conhecimento indutivo. Cada uma delas possui características específicas e cada caso a ser levantado carece de um estudo localizado.

Neste trabalho, para as duas abordagens, configurou-se mais adequada a extração de regras de classificação, que relacionam atributos de uma amostra, mostrando generalizações, padrões ou regularidades entre eles. Todavia, por questões de flexibilidade e performance, a implementação será a partir de métodos de extração de regras de associação conforme será mostrado na seção 5.13.

5.10 Objetivos e Características desejáveis da extração de regras

São dois os objetivos para os quais as regras extraídas a partir das ocorrências de atendimento num sistema Help Desk se prestarão:

(i) Para gerência estratégica do próprio sistema.

As regularidades que podem ser encontradas a partir das ocorrências ou dos casos podem servir como elemento de suporte à tomada de decisão na administração do próprio serviço de Help Desk.

A idéia é encontrar, dentro dos diversos atributos considerados, informações (regras) que sejam previamente desconhecidas, e que sejam úteis para a melhoria da qualidade do serviço.

Estas informações a serem encontradas, na verdade, correspondem precisamente ao que é desejável de ser obtido como resultado da execução de um sistema de extração de conhecimento, conforme mostrado no Capítulo 4, seção 4.4.

Para este caso, contudo, serão particularmente importantes as grandezas estatísticas de Confiança e Suporte, que poderão indicar o nível de relevância das informações encontradas em comparação com outras, bem como eventuais anomalias (do ponto de vista semântico, de acordo com o conhecimento do especialista) que denunciem um funcionamento indesejado ou inesperado.

Por outro lado, é interessante notar a necessidade de se verificar regras conflitantes ou discrepantes, que poderão mostrar falhas (operacionais ou táticas) ou, o mais interessante do ponto de vista da aprendizagem automática, a insuficiência ou irrelevância de alguns dos atributos considerados – caso em que todo ou parte do processo de aquisição automática de conhecimento deve ser revisado. Aqui, as grandezas Confiança e Suporte também serão de particular interesse para que se possa eventualmente estabelecer um limiar mínimo de tolerância a situações dessa natureza.

Para este último caso, portanto, o sistema deverá prover uma interface capaz de permitir que, uma vez incluído, excluído ou alterado um atributo da amostra, novas generalizações possam ser feitas sem maiores complicações para o usuário.

(ii) Para capitalização de conhecimento estratégico na promoção da qualidade.

O contato com clientes, a partir do qual são passadas por este informações ligadas ao uso e funcionamento de um produto ou serviço são elementos fundamentais para se extrair conhecimento útil na promoção da qualidade.

A tarefa, então, é encontrar, dentro das ocorrências ou dos casos[34], generalizações que demonstrem tendências ou padrões úteis na melhoria da qualidade do produto/serviço, e que, consoante com o que está no Capítulo 4, seção 4.4, sejam relevantes e acionáveis além de previamente desconhecidas.

De modo geral, pode-se encontrar relações entre os seguintes conjuntos de dados:

1. Perfil do Cliente X Produto Utilizado. Conquanto os registros de vendas possam prover uma relação fiel dos padrões de aquisição de produtos por partes dos clientes, aqui a relação pode ser estabelecida com mais fidelidade para se traçar o perfil do cliente em contraposição aos produtos que ele de fato utiliza – principalmente no caso de produtos adquiridos em "pacotes" juntamente com outros. Útil, portanto, na formulação das estratégias de marketing e segmentação de mercado.
2. Perfil do Cliente X Problema Encontrado. A partir dos registros de Help Desk, informações que relacionem os tipos de clientes aos problemas que normalmente são encontrados por eles pode permitir que o atendimento seja feito mais objetivamente e que o modo de apresentar o produto e suas características relevantes do ponto de vista do usuário se adeqüe mais ao que é esperado por ele. Neste caso, a utilidade é precisamente na adequação do produto aos potenciais usuários.
3. Produto Utilizado X Problema Encontrado. Aqui as informações podem trazer um diagnóstico muito aproximado do que de fato pode e precisa ser modificado/acrescentado em cada produto do ponto de vista da apresentação e da funcionalidade. Padrões que relacionem produtos a problemas permitem que se identifiquem falhas nos setores táticos ou operacionais a fim de se corrigir imperfeições ou defeitos, além de sugerir aperfeiçoamentos interessantes principalmente se as informações do cliente também forem levadas em consideração.

Além dessas relações, é importante que o sistema permita uma liberdade de escolha entre os atributos considerados, a fim de que as possibilidades de relacionamento entre as informações sejam o mais flexível possível. De fato, da mesma forma que no caso anterior, é interessante que seja viabilizada uma interface que permita que, mesmo em se incluindo, alterando ou excluindo algum atributo a generalização possa ser feita sem maiores complicações para o usuário.

5.11 Como gerar as Regras

A fim de cumprir as necessidades especificadas para a extração de regras, conforme a seção 5.10, este trabalho aponta para uma solução que seja o mais possível versátil. Isso permitirá que o usuário possa estabelecer, em momentos distintos, tanto a fonte dos dados (ocorrências ou casos[35]) quanto os atributos-causa e os atributos-alvo das regras, que também podem ser encontrados tanto nas ocorrências quanto nos casos. Isso proverá uma maior liberdade para que possam ser encontradas relações entre as mais diversas características dos dados conforme preconizado na seção 5.10.

5.12 Estrutura de dados auxiliar

Os algoritmos de extração de regras, idealmente, operam sobre uma "tabela normalizada", ou seja, uma visão completa de todos os dados do domínio. O modelo ideal, portanto, será uma tabela contendo nas colunas informações relativas aos atributos considerados e nas linhas os exemplos (casos ou ocorrências) que serão pesquisados.

O método que adotado neste trabalho necessita de uma informação adicional a respeito dos dados. Esta estrutura será denominada Tabela de Valores de Atributos.

Essa tabela mostrará, para cada atributo a ser considerado na pesquisa, uma lista de seus diferentes valores possíveis. No caso da pesquisa por regras a partir dos casos, de acordo com o modelo mostrado na seção 5.6.2, as informações necessárias para a construção dessa tabela já estarão disponíveis e podem ser obtidas trivialmente a partir de uma consulta simples na tabela Valor (e sua relação com a tabela Atributo).

Para a pesquisa na base de ocorrências, contudo, o processo não poderá ser tão trivial. Isso forçará uma intervenção parcial do especialista, a fim de que se possa estabelecer, para os atributos

---

[34] Para este caso específico, os casos – principalmente os casos abstratos – serão menos relevantes que as ocorrências propriamente exceto em diagnósticos sobre o próprio Sistema de Help Desk

[35] Os casos, para a extração de regras, serão utilizados apenas sob o ponto de vista do resgate baseado em atributo

contínuos, faixas de valores que permitam uma discretização desses dados. Isso poderá ser feito mediante uma caixa de diálogo, em que a lista de atributos contínuos é mostrada e o especialista define as faixas a serem consideradas (um intervalo) e o sistema poderá facilmente verificar a sintaxe disso, ou seja, se as faixas de fato cobrem todos os valores possíveis ou se deixam alguma "lacuna". Os atributos contínuos terão seus diferentes valores possíveis estabelecidos no decorrer da própria extração das regras, à medida em que o algoritmo vá encontrando seus diferentes valores.

5.13 Método proposto de extração de regras

Considere-se a Tabela de Valores de Atributos, cuja estrutura é ilustrada na Fig. 5.7. Pode-se fazer um mapeamento entre os valores dos atributos dos exemplos de forma a apresentá-los como uma lista de itens em transações, onde cada par atributo/valor eqüivalendo a um elemento da tabela e, por conseguinte, podendo ser tratado como um item independente.

Atributo 1	Atributo 2	Atributo 3	Atributo 4	...	Atributo alvo (Classe)
a_{11}	a_{21}	a_{31}	a_{41}	...	c_1
a_{12}	a_{22}	a_{32}	a_{42}	...	c_2
⋮	⋮	⋮	⋮	⋮	⋮
a_{1n1}	a_{2n2}	a_{3n3}	a_{4n4}	...	c_n

Fig. 5.7 Forma geral da Tabela de Valores de Atributos

A Tabela de Valores de Atributos mostra uma lista de atributos e, para cada um deles, todos os possíveis valores que ele pode receber. Isso garante que o conjunto de diferentes "itens" (pares atributo/valor) que existem no exemplo poderão ser conhecidos a qualquer tempo. Um (ou mais) dos atributos devem ser identificados como *atributo-alvo*, ou *classe*, que será usado pelo algoritmo para encontrar as regras – e estas serão precisamente aquelas que os incluam como conseqüentes. Assim, durante a execução, o algoritmo não precisará processar nenhuma regra que não inclua somente atributos/alvo do lado direito, o que proverá um ganho computacional conforme será mostrado.

Com o auxílio desta tabela é, portanto, possível a utilização de um algoritmo de extração de regras de associação para extrair regras de classificação. Particularmente para esse trabalho será utilizado o algoritmo Apriori (mas no caso distribuído pode-se aplicar o algoritmo Partition sem problemas), com duas vantagens sobre os métodos tradicionais de classificação:

1. Trata trivialmente os "buracos", ou seja, os atributos não informados. Serão apenas transações com um número de itens inferior ao total de atributos disponíveis.

2. Pode executar, conforme mostrado no capítulo 4, seção 4.6.2.3, um máximo de duas varridas na base de dados (ou potencialmente apenas uma conforme mostrado em [Toivonen, 1996]).

Esta concepção também apresenta uma funcionalidade superior ao Apriori tradicional (aplicado às regras de Associação) por outros dois motivos:

1. Não precisa considerar todas as combinações possíveis como candidatos a grandes conjuntos, já que não há como haver combinação, em uma mesma lista de itens, de grupos que, na tabela de Valores de Atributos, sejam referentes ao mesmo atributo[36].
2. Partindo dos conjuntos freqüentes, na tentativa de se encontrar as regras, só precisará considerar os conjuntos de itens que contenham de fato atributos-classe. Os que não os contenham, na prática, não serão capazes de gerar regras que sejam do interesse especificado pelo usuário.

O uso de métodos de extração de regras de associação para se encontrar regras de classificação é uma novidade que tem sido o diferencial de alguns produtos no mercado, como por exemplo o Aira Data Mining™ (www.hycones.com.br).

5.14 Descrição do processo de extração de regras

O primeiro passo que deverá ser executado é a discretização dos atributos contínuos. É nesse ponto que as regras a partir de casos ou de ocorrências serão tratados a fim de se chegar a um conjunto de exemplos consistente e a uma Tabela de Valores de Atributos válida.

Conforme visto na seção 5.6.2, para os casos armazenados, os atributos e seus possíveis valores já estão registrados e podem ser encontrados sem dificuldades. Caberá ao usuário apenas identificar, entre os atributos listados, quais deles serão usados como classe nas regras – doravante referidos como Atributos-Alvo ou Atributos-Classe.

As ocorrências carecerão de um processo um pouco mais elaborado, que consistirá de uma seleção dos atributos contínuos para que o usuário estabeleça patamares ou limiares de discretização e uma varrida nos dados para os atributos discretos (que será feita ao mesmo tempo em que se encontra os 1-conjuntos freqüentes) a fim de identificá-los.

Estabelecida essa tabela de Valores de Atributos, o algoritmo segue de maneira exatamente igual ao Apriori, que pode ser encontrado no Anexo 2 (ou o Partition), fazendo varridas sucessivas (possivelmente na memória) dos dados com algumas diferenças:

Na geração de candidatos, há algumas modificações a fim de não se considerar as combinações que incluam itens que, na tabela, pertençam ao mesmo atributo. Com isso ganha-se muitas combinações a menos para testar.

Além disso, para cada candidato gerado, seu suporte, juntamente com o suporte de todos os "pseudo-candidatos" formados pela concatenação dele com cada um dos valores dos atributos-classe – e não serão muitos pois a quantidade de valores diferentes para uma única classe não é significativamente elevada – , seja calculado.

Caso o resultado seja superior ao suporte mínimo estabelecido, busca-se entre os pseudo-candidatos algum(s) que possua(m) confiança superior à confiança mínima estabelecida para ser incluído no conjunto R das regras, caso em que o candidato é descartado, pois já foi incluído em uma regra. Se não houver nenhum, há a inclusão apenas do candidato no conjunto L de conjuntos freqüentes. Assim, para cada candidato, ou ele vai para a lista dos conjuntos freqüentes, a fim de se buscar alguma combinação futura que o inclua em alguma regra ou ele é incluído como regra, sendo esquivado do conjunto L. Isso garantirá duas coisas:

a) que serão geradas as regras mais simples possíveis já que o tamanho dos conjuntos cresce a cada passo começando com combinações pequenas e

b) que o crescimento do conjunto L, de onde surgirão os candidatos, será bem menor já que muitos candidatos não serão incluídos ali. Isso trará um ganho particularmente significativo já que a quantidade de candidatos de um dado tamanho K gerados são, em ordem de grandeza, função exponencial do número de conjuntos freqüentes de tamanho k-1 encontrados no passo anterior já que isso é feito através de combinação.

Para este trabalho, porém, é interessante notar que, em se tratando de uma base de dados disponibilizada em ambiente Web com recursos distribuídos, é necessário que seja possível descrever o processo para bancos de dados deste tipo.

Para o caso distribuído – em que deve-se aplicar uma versão modificada deste algoritmo para adequá-lo ao Partition – deve-se ter, em cada base de dados local, uma ou mais partições completas a fim de minimizar a necessidade de comunicação entre os nós e, por conseguinte, reduzir-se o tráfego nos canais de comunicação de dados. O algoritmo deverá exportar os conjuntos L de

[36] Não há como haver, num mesmo exemplo, mais de um valor associado ao mesmo atributo.

conjuntos freqüentes locais e R de regras locais para que se possa construir os candidatos globais L* e R*.

Assim, ao se combinar cada elemento de L* com os valores dos atributos-classe e unir isso a R*, ter-se-á o conjunto total de todas as regras candidatas globais que poderão ser aferidas numa segunda lida nos dados, de maneira semelhante ao preconizado em [Savasere, 1995], a fim de se identificar efetivamente as que são válidas com respeito aos dados globais..

O algoritmo proposto a partir do algoritmo Apriori é mostrado como segue:

C_APRIORI(**Entrada:** Banco de transações D, SupMin E ConfMin; **Saída:** Conjuntos freqüentes de *itemset* L, Tabela de Valores de Atributos A e Conjunto de regras R) /*Apriori para Classificação*/

```
Início
    Discretiza(A)
    Gera_L1(L1, R, A)
    K=2
    enquanto Lk-1 ≠ ∅ faça
        Apriori-Gen(Lk-1, R)
    k++
    L = ∪k Lk.
Fim
```

APRIORI-GEN (L_K, R)

```
Início
    para cada P ∈ Lk-1 faça
        para cada Q ∈ Lk-1, Q > P, Atrib(Q) ≠ Atrib(P) faça
            se |P ∪ Q| = k e P ∪ Q ∉ Lk então
                D_Conjunto (P ∪ Q)
Fim
```

D_CONJUNTO(C)

```
Início
    Aux = ∅
    para cada E ∈ A.Classe faça
        insere C ∪ {E} em Aux
    para cada t ∈ D faça
        se t ⊃ C então
            C.sup ++
            seja Z ∈ Aux, Z ⊂ t
                Z.sup ++
    se C.sup > SupMin então
       Início
            Inseriu = Falso
            para cada Z ∈ Aux faça
                se Z.sup ÷ C.sup ≥ ConfMin então
```

```
                    Início
                        Insere Z em R
                        Inseriu = Verdadeiro
                    Fim
                se não (Inseriu) então
                    Insere C em L_k
            Fim
        Fim
```

Aqui *Discretiza* é um procedimento que insere na tabela de valores de atributos as faixas dos atributos contínuos, a fim de que eles possam ser tratados pelo algoritmo.

Gera_L1 é um procedimento que faz uma leitura nos dados e estabelece o seu 1-conjuntos freqüentes e as primeiras regras (que ficam registradas em R, o conjunto de regras) formadas pelos 1- candidatos a conjuntos freqüentes, combinados a atributos-classe e que satisfaçam a confiança mínima (esses candidatos, por serem incluídos em R, não são incluídos como conjuntos freqüentes). Também a tabela de valores de atributo é construída nesse processo, conforme mostrado anteriormente).

Atrib é uma função que retorna o conjunto de atributos que estão envolvidos em um determinado conjunto de itens (que nesta abordagem são o mapeamento, pela tabela de valores de atributos, dos pares atributo/valor).

Outro ponto interessante desse método, herança de seu inspirador, o Apriori, é a possibilidade do usuário estabelecer, a cada vez que o executa, valores diferentes para as variáveis de Confiança e Suporte, a fim de ajustar a acurácia das regras levantadas de acordo com sua necessidade. Isso poderá permitir a busca por anomalias, cuja ocorrência, embora pequena seja porém significativa (suporte muito pequeno, mas confiança alta ou vice-versa). Além de, evidentemente, selecionar um conjunto diferente de atributos a serem considerados.

5.15 Conclusões

O Capítulo 5 mostra que, de fato, num sistema de atendimento automático de Help Desk, a busca de uma solução baseada na história registros semelhantes armazenados deve ser cumprida com o uso da tecnologia de Raciocínio Baseado em Casos, e apresenta as características que esse sistema precisará prover para que o usuário tenha uma interface amigável e eficiente, e que o guie com rapidez em direção à solução de um problema atual.

O resgate dos casos armazenados poderá ser feito por casamento de padrões textuais, baseando-se em palavras-chave, de acordo com um dos modelos de interface proposto e aceito no mercado, e pelos valores dos atributos, consoante com a outra vertente dominante das ferramentas disponíveis.

O resgate textual permitirá que o usuário possa tentar responder a uma série de perguntas sobre o caso, oriundas do próprio sistema, e a partir dessas respostas o sistema se aproxime de uma solução que satisfaça o seu problema. Notar que nessa modelagem o especialista terá que acoplar a cada caso do sistema um conjunto de perguntas ou indagações incluindo palavras-chave e seus respectivos sinônimos, conjunto esse que será utilizado em associação com a resposta do usuário a fim de que este possa interagir com o sistema no processo de seleção de casos.

A recuperação pelo atributo terá uma acurácia melhor, pois guia mais rapidamente para um conjunto restrito de casos, porém o usuário provavelmente necessitará de um conhecimento um pouco mais profundo de termos e expressões técnicas para utilizá-la.

A recuperação textual não admite a inclusão direta de casos reais, pois as ocorrências registradas não possuem uma estrutura de perguntas e respostas associada a elas. É facultado ao especialista, porém, adicionar a cada caso uma estrutura desse tipo, sejam eles oriundos de ocorrências (casos reais) ou criados (casos abstratos). Ambas as soluções estão apresentadas em um modelo de banco de dados relacional (Oracle), o que garante que a eficiência e a precisão comprovada da linguagem SQL possa ser usada a favor do desempenho dos métodos de resgate.

Outrossim, o conhecimento que pode ser extraído, em forma de regras, de maneira indutiva a partir de informações úteis a Help Desk pode servir a muitos propósitos, tanto no suporte ao sistema em si (formando um modelo híbrido) na busca de eventuais falhas ou anomalias, quanto na melhoria das relações com o cliente e a busca pela promoção da qualidade. Estas informações poderão ser úteis na gerência do próprio sistema de Help Desk e nos setores estratégicos respectivamente.

Para atender a isso tudo, um método de extração de regras em um sistema de Help Desk deve ser capaz de considerar tanto os casos que são usados no atendimento (a Base de Casos) quanto as ocorrências de atendimento (a Base de Ocorrências), e deve primar pela eficiência ao apresentar seus resultados ao mesmo tempo em que provê flexibilidade para permitir que sejam relacionados diversos tipos de informação.

Mostrou-se então a solução proposta, e a forma como os métodos de associação podem ser utilizados para a extração de regras de classificação com o uso de uma estrutura de dados auxiliar, a

Tabela de Valores de Atributos, fazendo uso de uma adaptação do Apriori, munindo-o de uma série de vantagens interessantes em termos de custo computacional e de facilidades para o usuário.

Este capítulo encerra a apresentação de uma concepção que mostra como potencializar as capacidades de um sistema de Help Desk fazendo uso de dois paradigmas distintos de aprendizagem, o Raciocínio Baseado em Casos e a Aprendizagem Indutiva.

O próximo capítulo é a apresentação do modelo todo sendo aplicado, mostrando uma base de casos sendo empregada em consultas pelo usuário a partir do resgate por palavras-chave e por recuperação pelo atributo bem como o acompanhamento da extração de informações pelo método indutivo C-Apriori aqui ilustrado.

Capítulo 6
Utilizando o Sistema

6.1 Introdução

Este capítulo é composto de duas partes distintas, em que a abordagem por aprendizagem indutiva serve como método de aferição da abordagem por casos.

A primeira parte é, por sua vez, subdividida em mais duas sub-partes: uma para apresentar o resgate pelo casamento de padrão textual, ilustrando uma base de casos e um modelo de pesquisa feito pelo usuário, e a segunda é, analogamente, uma ilustração de uma busca feita pelo casamento de atributo.

A segunda parte do capítulo é dedicada ao algoritmo C_Apriori, que foi apresentado no Capítulo 5, seção 5.14. Aqui é apresentada a visão híbrida que torna um sistema baseado em aprendizagem dedutiva (RBC) monitorado através de um outro, que se baseia num paradigma indutivo (o C_Apriori). É mostrado, também, todo o andamento de uma execução deste algoritmo para ilustrar sua eficiência.

6.2 Um panorama para recuperação pelo resgate textual

A recuperação textual permitirá que um usuário, a partir do diálogo com o sistema, possa selecionar uma ou mais perguntas para responder e a partir de suas respostas o sistema seja capaz de lhe apresentar um conjunto cada vez mais restritivo de casos, entre os quais deverá haver pelo menos um que o auxilie na solução do seu problema.

Assim, para efeito de ilustração, apenas um subconjunto dos casos que efetivamente serão pesquisados pelo cliente (os que façam referência a um mesmo produto) serão aqui mostrados.

6.2.1 Uma visão lógica

Aqui está uma visão esquemática da estrutura de casos que neste momento está sendo usada pela base:

Caso Nº 1: Diagnóstico: *não instalou corretamente o suporte à placa 3D* Solução*: Re-instalar suporte 3D*

Pergunta Nº 1: *Como é o processamento de vídeo?* Palavras-Chave (Sinônimos*): Lento (Devagar, Complicado, Pesado). Trava (Congela)*

Caso Nº 2: Diagnóstico: *corrupção pelo sistema operacional* Solução: *Re-instalar todo o produto*

Pergunta No 1: Como é sua rede elétrica? Palavras-Chave (Sinônimos): Falha (Falta, Instável, Ruim, Defeituosa)

Pergunta No 2: Como está o funcionamento do seu computador? Palavras-Chave (Sinônimos): Lento (Devagar), Trava (Congela), Reseta (Resetar, Reinicia, Reiniciar), Proteção (GPF, Protection)

Caso Nº 3: Diagnóstico: *corrompeu a instalação do módulo multi-usuário* Solução: *Re-instalar multi-usuário*

Pergunta No 1: O que não consegue fazer com o documento? Palavras-Chave (Sinônimos): Compartilhar (Compartilhado, Dividido, Sharing, Junto, Juntamente)

Pergunta No 2: O que acontece quando se cria um documento compartilhado? Palavras-Chave (Sinônimos): Proíbe (Erro, Trava, Impede, Congela, Mensagem)

Caso Nº 4: Diagnóstico: *instalou suporte a Placa 3D mas ela não foi encontrada* Solução: *Refazer instalação do suporte 3D*

Pergunta No 1: Como é o processamento de vídeo? Palavras-Chave (Sinônimos): Trava (Congela, Parado, Imóvel), Perde (Apaga, Limpa, Escurece)

Caso Nº 5: Diagnóstico: *corrompeu-se a configuração de rede* Solução: *Ré-configurar toda a máquina*

Pergunta No 1: Como está o acesso a outros computadores e recursos compartilhados da rede? Palavras-Chave (Sinônimos): Impede (Impossível, Inalcançável, Difícil, Falha)

Pergunta No 2: O que acontece quando se faz o Login? Palavras-Chave (Sinônimos): Erro (Perfil, Descarregar, Descarrega, Falha)

Caso Nº 6: Diagnóstico: *instalou suporte a Placa 3D mas ela não foi encontrada* Solução: *Refazer instalação do suporte 3D*

Pergunta No 1: Como é o processamento de vídeo? Palavras-Chave (Sinônimos): Trava (Congela, Parado, Imóvel), Perde (Apaga, Limpa, Escurece)

Caso Nº 7: Diagnóstico: *houve remoção ou substituição da Placa 3D mas a instalação permanece* Solução: *Refazer instalação do suporte 3D*

Pergunta No 1: Como é o processamento de vídeo? Palavras-Chave (Sinônimos): Trava (Congela, Parado, Imóvel), Perde (Apaga, Limpa, Escurece)

Pergunta No 2: Houve alguma coisa com a placa 3D? Palavras-Chave (Sinônimos): Removida (Retirada, Tirada, Subtraída), Trocada (Substituída, Upgrade, Atualização)

Caso Nº 8: Diagnóstico: *corrompeu a instalação do módulo multi-usuário* Solução: *Re-instalar multi-usuário*

Pergunta No 1: O que não consegue fazer com o documento? Palavras-Chave (Sinônimos): Compartilhar (Compartilhado, Dividido, Sharing, Junto, Juntamente)

Pergunta No 2: O que acontece quando se cria um documento compartilhado? Palavras-Chave (Sinônimos): Proíbe (Erro, Trava, Impede, Congela, Mensagem)

Caso Nº 9: Diagnóstico: *corrupção pelo sistema operacional* Solução: *Re-instalar todo o produto*

Pergunta No 1: Como é sua rede elétrica? Palavras-Chave (Sinônimos): Falha (Falta, Instável, Ruim, Defeituosa)

Pergunta No 2: Como está o funcionamento do seu computador? Palavras-Chave (Sinônimos): Lento (Devagar), Trava (Congela), Reseta (Resetar, Reinicia, Reiniciar), Proteção

Caso Nº 10: Diagnóstico: *corrompeu-se a configuração de rede* Solução: *Ré-configurar toda a máquina*

Pergunta No 1: Como está o acesso a outros computadores e recursos compartilhados da rede? Palavras-Chave (Sinônimos): Impede (Impossível, Inalcançável, Difícil, Falha)

Pergunta No 2: O que acontece quando se faz o Login? Palavras-Chave (Sinônimos): Erro (Perfil, Descarregar, Descarrega, Falha)

Com esse conjunto de casos, com suas respectivas perguntas que se acoplam das estruturas de palavras-chave e de sinônimos, será possível ilustrar uma busca, pelo usuário, da solução de um problema qualquer.

6.2.2 A visão relacional

Conquanto seja muito natural para que se possa apresentar concretamente a estrutura que compõe cada um dos casos, a forma acima não corresponde especificamente à verdade. Na prática, conforme mostrado no capítulo anterior, na seção 5.6.1, toda essa estrutura estará disposta em tabelas de bancos de dados relacionais. Haverá, portanto, 6 tabelas que comporão todos os casos, estabelecendo as ligações de pertinência das estruturas através de relacionamentos[37] conforme mostrado abaixo (uma descrição mais detalhada das definições em SQL dessas tabelas podem ser vistas no anexo I):

Tabela Tb_Caso

Cd_Caso	Cd_Produto	Ds_Caso	Ds_Solucao
1	3	não instalou corretamente o suporte à placa 3D	Re-instalar suporte 3D
2	3	corrupção pelo sistema operacional	Re-instalar todo o produto
3	3	corrompeu a instalação do módulo multi-usuário	Re-instalar multi-usuário
4	3	instalou suporte a Placa 3D mas ela não foi encontrada	Refazer instalação do suporte 3D
5	3	corrompeu-se a configuração de rede	Ré-configurar toda a máquina
6	3	instalou suporte a Placa 3D mas ela não foi encontrada	Refazer instalação do suporte 3D
7	3	houve remoção ou substituição da Placa 3D mas a instalação permanece	Refazer instalação do suporte 3D
8	3	corrompeu a instalação do módulo multi-usuário	Re-instalar multi-usuário
9	3	corrupção pelo sistema operacional	Re-instalar todo o produto
10	3	corrompeu-se a configuração de rede	Ré-configurar toda a máquina

Tabela Tb_Pergunta

Cd_Pergunta	Ds_Pergunta
1	Como é o processamento de vídeo?
2	Como é sua rede elétrica?
3	Como está o funcionamento do seu computador?
4	O que não consegue fazer com o documento?
5	O que acontece quando se cria um documento compartilhado?
6	Como está o acesso a outros computadores e recursos compartilhados da rede?
7	O que acontece quando se faz o Login?
8	Houve alguma coisa com a placa 3D?

[37] Relacionamentos em Bancos de Dados relacionais: Uma linha de uma tabela possui um valor de uma coluna (ou atributo) que aponta para o identificador de outra linha em outra tabela

Tabela Tb_Palavra

Cd_Palavra	Ds_Palavra
1	Lento
2	Trava
3	Falha
4	Lento
5	Reseta
6	Proteção
7	Compartilhar
8	Proíbe
9	Trava
10	Perde
11	Impede
12	Erro
13	Removida
14	Trocada
30	10	Escurece
31	11	Impossível
32	11	Inalcançável
33	11	Difícil
34	11	Falha
35	12	Perfil
36	12	Descarregar
37	12	Descarrega
38	12	Falha
39	13	Retirada
40	13	Tirada
41	13	Subtraída
42	14	Substituída
43	14	Upgrade
44	14	Atualização

Tabela Tb_Sinônimo

Cd_Sinônimo	Cd_Palavra	Ds_Sinônimo
1	1	Devagar
2	1	Complicado
3	1	Pesado
4	2	Congela
5	3	Falta
6	3	Instável
7	3	Ruim
8	3	Defeituosa
9	4	Devagar
10	5	Resetar
11	5	Reinicia
12	5	Reiniciar
13	6	GPF
14	6	Protection
15	7	Compartilhado
16	7	Dividido
17	7	Sharing
18	7	Junto
19	7	Juntamente
20	8	Erro
21	8	Trava
22	8	Impede
23	8	Congela
24	8	Mensagem
25	9	Congela
26	9	Parado
27	9	Imóvel
28	10	Apaga
29	10	Limpa

Tabela Tb_Caso_Perg_Palavra

Cd_Caso	Cd_Pergunta	Cd_Palavra
1	1	1
1	1	2
2	2	3
2	3	4
2	3	5
2	3	6
3	4	7
3	5	8
4	1	9
4	1	10
5	6	11
5	7	12
6	1	9
6	1	10
7	1	9
7	1	10
7	8	13
7	8	14
8	4	7
8	5	8
9	2	3
9	3	4
9	3	5
9	3	6
10	6	11
10	7	12

A esta estrutura será temporariamente acrescida uma tabela, *Lista*, que conterá em separado as palavras usadas pelo usuário em sua resposta.

6.2.3 Uma consulta pelo resgate textual

Um usuário, por hipótese, receberá em sua frente a lista de perguntas que de alguma forma se associem ao produto no qual ele vem enfrentando problemas. A tela de consulta é ilustrada na Fig. 6.1, em que ele selecionou a pergunta "Como é o processamento de vídeo?" e respondeu com a frase "Perde a imagem com freqüência".

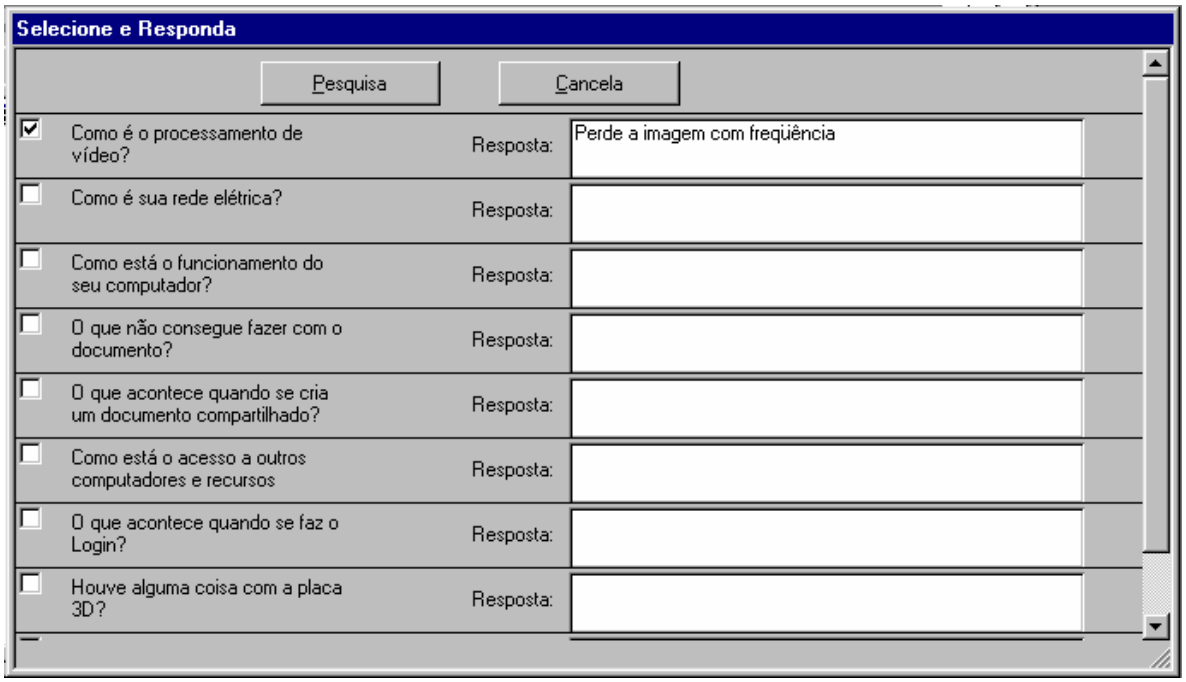

Fig. 6.1. Uma tela de consulta para resgate textual

Ao ser feita a consulta, as palavras "Perde", "Imagem" e "Freqüência" serão usadas na busca (em Tb_Palavra ou Tb_Sinônimo). Uma delas, "Perde", existe em Tb_Palavra. Seu código é 10. Em Tb_Caso_Perg_Palavra ela precisa aparecer associada à pergunta de código 1 para que o caso seja selecionado. Com efeito, ela aparece três vezes e em todas associadas à pergunta referida. São então selecionados (e apresentados ao usuário) os casos números 4, 6 e 7 (Caso Nº 4: Diagnóstico: *instalou suporte a Placa 3D mas ela não foi encontrada* Solução: *Refazer instalação do suporte 3D*, Caso Nº 6: Diagnóstico: *instalou suporte a Placa 3D mas ela não foi encontrada* Solução: *Refazer instalação do suporte 3D,* Caso Nº 7: Diagnóstico: *houve remoção ou substituição da Placa 3D mas a instalação permanece* Solução: *Refazer instalação do suporte 3D*) Cada um ocorrendo apenas uma vez na pesquisa.

O usuário então refaz sua pesquisa, e dessa vez responde à pergunta "Houve alguma coisa com a placa 3D?" dizendo "Eu fiz um Upgrade dela". Dentre as que usou na sua resposta, a única palavra

que aparece é “Upgrade”, em Tb_Sinônimo, associado à palavra de código 14. Essa palavra, então, é cruzada com os casos e com as perguntas (só serão selecionados os casos que contiverem essa palavra associada à pergunta de código 8, que foi a selecionada pelo usuário) e assim o caso número 7 é mais uma vez selecionado. Ele agora é apresentado à frente dos demais pois o número de coincidências associadas a ele é mais alto.

O usuário, então, hipoteticamente, dá-se por satisfeito. Tem um diagnóstico e uma solução para o seu problema (*houve remoção ou substituição da Placa 3D mas a instalação permanece: Refazer instalação do suporte 3D*) ou então os casos armazenados não resolvem o seu problema e este é um caso em que o sistema é omisso, e a solução terá que ser dada (e posteriormente armazenada) de outra forma (pela equipe de atendimento ou pelo especialista).

6.3 Um panorama para recuperação pelo resgate pelo atributo

O resgate pelo atributo é feito a partir de uma janela de consulta muito semelhante à que é mostrada na Fig. 6.1. Dessa vez o usuário poderá selecionar algum atributo (e portanto necessitará de um conhecimento técnico mais acurado) e estabelecer algum valor para ele de acordo com o que ele conhece de seu problema atual.

O sistema deverá, então, a partir dos atributos e dos valores que ele estabeleceu para eles, fazer o resgate, na base, dos casos que atendam parcial ou totalmente ao casamento. Para esta abordagem, conforme mostrado no Capítulo 5, as métricas de similaridade que serão consideradas estão implementadas dentro das características do atributo do caso. Esse fato é particularmente interessante para esta abordagem, conforme mostrado no mesmo capítulo.

6.3.1 Uma visão lógica

Tomando-se como ponto de partida os mesmos casos que foram ilustrados na apresentação do resgate pela recuperação textual, conforme mostrado na seção 6.2, pode-se ver o conjunto de casos a ser pesquisado como uma única e complexa tabela, conforme mostrada na página seguinte:

CasoID	Multiusuário	Conexão	Sist. Oper.	Placa 3D	Suporte 3D Instalado	Clock da Máquina	Espaço Livre em Disco	Alt. Config. Após Instal.	Teste Map. Unidade Ok	Num. Replies Servidor	Placa 3D Inst. a Pouco	Placa 3D Inst. ou Subst.	Máq. Trava com Freq.
1	Não(0)	Modem(0)	Linux(0)	Sim(0)	Sim(0)								
2	Não(0)	Modem(0)	Windows(0)	Não(0)		<=100 (1)	11 - 20 (1)						
3	Sim(0)	Modem(0)	MacOS(0)	Sim(0)				Sim(0)					
4	Sim(0)	Modem(0)	Linux(0)	Não(0)							Sim(0)		
5	Sim(0)	Rede Local(0)	Windows(0)	Sim(0)					Não(0)	4(1)			
6	Não(0)	Rede Local(0)	Linux(0)	Não(0)							Sim(0)		
7	Sim(0)	Modem(0)	MacOS(0)	Não(0)								Sim(0)	
8	Sim(0)	Rede Local(0)	Windows(0)	Sim(0)									Sim(0)
9	Não(0)	Modem(0)	Windows(0)	Não(0)		<=100 (1)	11 - 20 (1)						
10	Sim(0)	Rede Local(0)	Windows(0)	Sim(0)					Não(0)	4(1)			

Fig. 6.2. Uma visão lógica da disposição dos casos para recuperação pelo atributo

Nesta figura não estão claramente indicados, para cada atributo, quais são os possíveis valores que ele pode obter. Aqui são mostrados os valores e, entre parênteses, a distância máxima que aquele atributo deverá estar do seu correspondente na descrição que o usuário faz de seu problema atual. Assim, segue uma descrição de cada atributo disponível, bem como a lista ordenada de seus possíveis valores.

Atributo: *Multiusuário* Valores: 1. *Sim* 2. *Não*

Atributo: *Conexão* Valores: 1. *Modem* 2. *Rede Local*

Atributo: *Sist. Operacional* Valores: 1. *Windows* 2. *MacOS* 3. *Linux*

Atributo: *Placa 3D* Valores: 1. *Sim* 2. *Não*

Atributo: *Suporte 3D Instalado* Valores: 1. *Sim* 2. *Não*

Atributo: *Clock da Máquina (MHz)* Valores: 1. *<= 100* 2. *101 a 200* 3. *201 a 400* 4. *401 a 700* 5. *> 700*

Atributo: *Espaço Livre em Disco (GB)* Valores: 1. *<=10* 2. *11 a 20* 3. *21 a 40* 4. *41 a 80* 5. *> 80*

Atributo: *Alteração das Configurações Após Instalação* Valores: 1. *Sim* 2. *Não*

Atributo: *Teste de Mapeamento de Unidade de Rede Ok* Valores: 1. *Sim* 2. *Não*

Atributo: *Núm. de Replies do Servidor* Valores: 1. *0* 2. *1* 3. *2* 4. *3* 5. *4*

Atributo: *Placa 3D Instalada a pouco* Valores: 1. *Sim* 2. *Não*

Atributo: *Placa 3D Instalada ou Substituída* Valores: 1. *Sim* 2. *Não*

Atributo: *Máquina Trava com Freqüência* Valores: 1. *Sim* 2. *Não*

Com este conjunto de casos, com os atributos que lhes são pertinentes valorados, bem como as distâncias máximas permitidas (métrica de similaridade) igualmente estabelecidas mais a lista de valores possíveis que cada atributo pode assumir, é possível ilustrar-se uma pesquisa de um usuário por esta abordagem.

6.3.2 A visão relacional

Na prática, contudo, os casos estarão armazenados numa estrutura relacional, conforme mostrado no Capítulo 5, seção 5.6.2, e será a partir desse modelo que a consulta será processada computacionalmente. A seguir é mostrada esta disposição, de acordo com o modelo de criação das estruturas em SQL que está disponível no Anexo I. Como na seção 6.2.2 já foi mostrada, entre outras, a tabela de casos (Tb_Caso), está será aqui suprimida já que seu formato em absoluto muda. É importante ressaltar que a esta estrutura é acrescida, no instante da busca, duas tabelas temporárias: uma que conterá os atributos e os valores estabelecidos pelo usuário e outra que conterá uma lista de casos reprovados pelo usuárioque não devem ser mais selecionados durante a busca. Ambas as tabelas são destruídas depois de terminado o processo.

Tabela Tb_Atributo

Cd_Atributo	Nm_Atributo	Ds_Atributo
1	Multiusuário	O sistema está sendo usado em ambiente Multiusuário?
2	Conexão	Qual o tipo de conexão usado?
3	Sist. Operacional	Qual o Sistema Operacional?
4	Placa 3D	Existe Placa 3D na máquina?
5	Suporte 3D Instalado	O suporte, via Software, da placa está instalado?
6	Clock da Máquina	Qual a velocidade, em MHz, da máquina?
7	Espaço livre em Disco	Quanto há, em GB, de espaço disponível em disco?
8	Alt. Conf. Após Inst.	Houve alteração nas configurações da máquina depois de instalado?
9	Teste Map. Unid. OK	É possível se mapear com sucesso uma unidade ou recurso da rede?
10	Num. Replies Servidor	Usando um Ping, quantos replies são recebidos do servidor?
11	Placa 3D Inst. a pouco	Foi instalada a placa 3D há pouco?
12	Placa 3D Inst. / Subst.	Houve instalação ou substituição de placa 3D?
13	Máquina trava c/ freq.	A máquina trava/congela/pára com freqüência?

Tabela Tb_Valor

Cd_Valor	Cd_Atributo	Nu_Ordem	Ds_Valor
1	1	1	Sim
2	1	2	Não
3	2	1	Modem
4	2	2	Rede Local
5	3	1	Windows
6	3	2	MacOS
7	3	3	Linux
8	4	1	Sim
9	4	2	Não
10	5	1	Sim
11	5	2	Não
12	6	1	< 100
13	6	2	101 a 200
14	6	3	201 a 400
15	6	4	401 a 700
16	6	5	> 700
17	7	1	< 10
18	7	2	11 a 20
19	7	3	21 a 40
20	7	4	41 a 80
21	7	5	> 80
22	8	1	Sim
23	8	2	Não
24	9	1	Sim
25	9	2	Não
26	10	1	0
27	10	2	1
28	10	3	2
29	10	4	3
30	10	5	4
31	11	1	Sim
32	11	2	Não
33	12	1	Sim
34	12	2	Não
35	13	1	Sim
36	13	2	Não

Tabela Tb_Atributo_Valor_Caso

Cd_Atributo	Cd_Valor	Cd_Caso	Vl_Classif	Vl_Dist_Max
1	2	1	2	0
2	3	1	1	0
3	7	1	3	0
4	8	1	1	0
5	10	1	1	0
1	1	2	1	0
2	3	2	1	0
3	5	2	1	0
4	9	2	2	0
6	12	2	1	1
7	18	2	2	1
1	1	3	1	0
2	3	3	1	0
3	6	3	2	0
4	8	3	1	0
8	22	3	1	0
1	1	4	1	0
2	3	4	1	0
3	7	4	3	0
4	9	4	2	0
11	31	4	1	0
1	1	5	1	0
2	4	5	2	0
3	5	5	1	0
4	8	5	1	0
9	25	5	2	0

10	30	5	5	1
1	2	6	2	0
2	4	6	2	0
3	7	6	3	0
4	9	6	2	0
11	31	6	1	0
1	1	7	1	0
2	3	7	1	0
3	6	7	2	0
4	9	7	2	0
12	33	7	1	0
1	1	8	1	0
2	4	8	2	0
3	5	8	1	0
4	8	8	1	0
13	35	8	1	0
1	1	9	1	0
2	3	9	1	0
3	5	9	1	0
4	9	9	2	0
6	12	9	1	1
7	18	9	2	1
1	1	10	1	0
2	4	10	2	0
3	5	10	1	0
4	8	10	1	0
9	25	10	2	0
10	30	10	5	1

6.3.3 Uma consulta através do Resgate pelo Atributo

Aqui o usuário se depara com uma janela que lho apresenta a lista de atributos disponíveis para serem usados na pesquisa e solicita dele que estabeleça algum valor para algum deles. A janela de parâmetros é como a mostrada na Fig. 6.3. O usuário deve marcar os atributos que ele deseja valorar (de acordo com o que ele conhece do próprio problema) e em seguida disparar a consulta à base de casos. Por hipótese, o usuário estabeleceu os valores dos atributos *Sistema Operacional*, com o valor *Windows*, e *Conexão* com *Rede Local*.

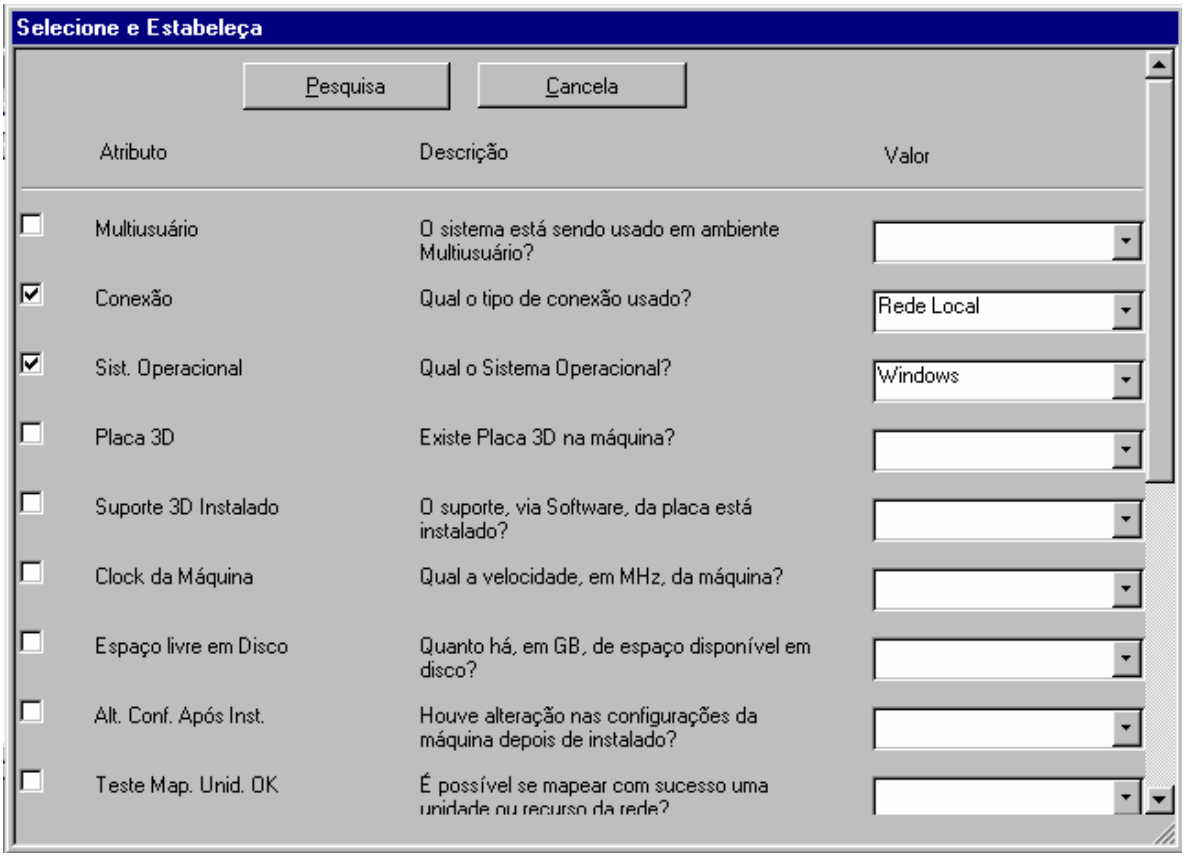

Fig. 6.3. Uma janela para estabelecer valores de atributos

A pesquisa é, então, disparada e o sistema retorna os casos que possuam os atributos selecionados com valor igual ou próximo o suficiente[38] ao que foi estabelecido. No exemplo, são retornados para o usuário os casos 5 (*corrompeu-se a configuração de rede*: *Ré-configurar toda a máquina*), 8 (*corrompeu a instalação do módulo multi-usuário*: *Re-instalar multi-usuário*) e 10 (*corrompeu-se a configuração de rede*: *Ré-configurar toda a máquina*) que são apresentados ao usuário. Este, por conhecimento pessoal, pode descartar alguns dos casos apresentados ou voltar à janela de consulta e estabelecer novos atributos. Por hipótese, o *número de replies do servidor* com valor *3*. Feita a pesquisa, apenas os casos 5 e 10 serão selecionados, e através de casamento parcial pois o valor do atributo *número de replies do servidor* para estes casos é esperado 4. Mas com a similaridade tolerada, pode-se aceitar um valor com distância (Vl_Distancia_Maxima) imediatamente anterior ou posterior, o que precisamente ocorre.

Assim o usuário pode encontrar a solução mais próxima de seu problema com uma acurácia bem mais alta que na busca pelo casamento de padrão.

6.4 Uma aplicação da indução sobre RBC

[38] De acordo com a distância máxima permitida identificada na tabela Tb_Atributo_Valor_Caso

A visão híbrida das duas abordagens, RBC e Indução, fica justificada com a possibilidade de, a partir dos dados que são utilizados pelo sistema de Raciocínio por casos, o sistema possa, através de um método indutivo, encontrar e apresentar informações que sejam úteis e relevantes no monitoramento e na aferição dos resultados do próprio sistema servindo, assim, de *feed back* para que o sistema possa ser continuamente acompanhado, corrigido e melhorado por um (ou uma equipe de) especialista. A Fig. 6.4 dá uma idéia do processo:

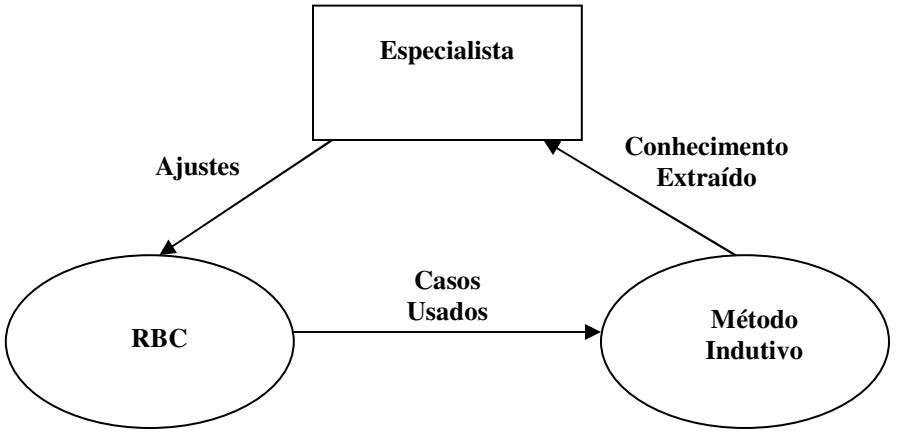

Fig. 6.4. Modelo de Feed Back entre RBC e Indução

Tomando como exemplo o mesmo conjunto de casos considerado até o presente, com um subconjunto finito de atributos e uma classe estabelecida, que é a definição do problema ser uma falha de instalação ou de corrupção, pode-se imaginar a execução do algoritmo C_Apriori proposto no capítulo anterior em busca do conjunto o mais amplo possível de regras para que se possa ter um diagnóstico do sistema RBC.

6.5 Um exemplo do C_Apriori

Em primeiro lugar, são geradas A, a tabela de valores de atributos, L_1, os grandes conjuntos de tamanho 1 e as primeiras regras, que são os conjuntos de tamanho 1 que, combinados aos atributos-classe, atingiram suporte e confiança mínimos.

Para este exemplo, considera-se o suporte mínimo igual a 0% (basta uma ocorrência) e a confiança mínima de 100%.

No	Multiusuário	Conexão	Sist. Oper.	Placa 3D	Loc. Problema
1	Não	Modem	Linux	Sim	Instalação
2	Não	Modem	Windows	Não	Instalação
3	Sim	Modem	MacOS	Sim	Corrupção
4	Sim	Modem	Linux	Não	Instalação
5	Sim	Rede Local	Windows	Sim	Corrupção
6	Não	Rede Local	Linux	Não	Instalação
7	Sim	Modem	MacOS	Não	Corrupção
8	Sim	Rede Local	Windows	Sim	Corrupção
9	Não	Modem	Windows	Não	Instalação
10	Sim	Rede Local	Windows	Sim	Corrupção

Fig. 6.5. Um conjunto de exemplos para classificação pelo algoritmo proposto

A tabela de valores de atributos seria, então, como segue (a notação serve apenas para facilitar a compreensão):

Multiusuário	Conexão	Sist. Oper.	Placa 3D	Loc. Problema
Sim (M1)	Modem (C1)	Linux (S1)	Sim (P1)	Instalação (L1)
Não (M2)	Rede Local (C2)	Windows (S2)	Não (P2)	Corrupção (L2)
		MacOS (S3)		

Fig. 6.6. Tabela de valores de atributos

A partir daí o algoritmo segue gerando as regras e os conjuntos freqüentes de maneira semelhante ao Apriori. Notar que os candidatos que foram gerados mas que não geraram conjuntos freqüentes nem regras também estão listados, para efeito de uma melhor compreensão, mas estes não são armazenados.

Como o objetivo dessa abordagem é dar ao usuário a descrição o mais ampla possível do domínio, o algoritmo vai, então, buscar exaustivamente por todas as regras que puderem ser encontradas, dados os parâmetros de funcionamento (confiança e suporte). Assim, no final da execução, as regras encontradas, e que podem ser "traduzidas" com o auxílio da tabela de valores de atributo são, respectivamente:

1. Se {Multiusuário} = “Não” Então Instalação
2. Se {Sistema Operacional} = “Linux” Então Instalação
3. Se {Sistema Operacional} = “MacOS” Então Corrupção
4. Se {Multiusuário} = “Sim” e {Conexão} = “Rede Local” Então Corrupção
5. Se {Multiusuário} = “Sim” e {Sistema Operacional} = “Windows” Então Corrupção
6. Se {Multiusuário} = “Sim” e {Placa 3D} = “Sim” Então Corrupção
7. Se {Conexão} = “Modem” e {Sistema Operacional} = “Windows” Então Instalação
8. Se {Conexão} = “Rede Local” e {Sistema Operacional} = “Windows” Então Corrupção
9. Se {Conexão} = “Modem” e {Placa 3D} = “Sim” Então Corrupção
10. Se {Conexão} = “Modem” e {Placa 3D} = “Não” Então Instalação
11. Se { Sistema Operacional } = “Windows” e { Placa 3D } = “Sim” Então Corrupção
12. Se { Sistema Operacional } = “Windows” e { Placa 3D } = “Não” Então Instalação
13. Se { Multiusuário } = “Sim” e {Conexão} = “Modem” e { Placa 3D } = “Sim” Então Corrupção

Candidatos	**Sup.**
M1	6
M2	4
C1	6
C2	4
S1	3
S2	5
S3	2
P1	5
P2	5

Varre D →

Conj. Freqüentes	**Sup.**
M1	6
C1	6
C2	4
S2	5
P1	5
P2	5

Regras	**Sup.**
M2 → L1	4
S1 → L1	3
S3 → L2	2

M1 C1	3
M1 C2	3
M1 S2	3
M1 P1	4
M1 P2	2
C1 S2	2
C1 P1	2
C1 P2	4
C2 S2	3
C2 P1	3
C2 P2	1
S2 P1	3
S2 P2	2

Varre D →

M1 C1	3
M1 P2	2
C1 P1	2
C1 P2	4

M1 C2 → L2	3
M1 S2 → L2	3
M1 P1 → L2	4
C1 S2 → L1	2
C2 S2 → L2	3
C2 P1 → L2	3
C2 P2 → L1	1
S2 P1 → L2	3
S2 P2 → L1	2

M1 C1 P2	2
M1 C1 P1	1

Varre D →

M1 C1 P2	2

M1 C1 P1 → L2	1

Fig. 6.5. Execução do C_Apriori

6.6 Conclusões

Este capítulo mostrou de forma ilustrativa o funcionamento de todo o sistema. Um conjunto hipotético de casos foi tomado como exemplo, e a partir desses foi apresentada cada uma das características que o sistema apresenta.

Inicialmente foi mostrada uma visão lógica da disposição dos dados para o resgate pelo casamento de padrão textual. Em seqüência, uma visão da disposição que esses dados terão dentro de um banco de dados relacional, tal como preconizado no capítulo anterior, para mostrar as estruturas que serão usadas na pesquisa.

Uma busca hipotética foi feita, mostrando a interface do usuário com o sistema e as respostas que ele apresenta de acordo com o que o usuário digitou, ou seja, a partir da descrição que ele mesmo faz do problema.

A segunda parte do capítulo é semelhante à primeira, porém aplicada à busca pelo valor do atributo. Da mesma forma, foi mostrada uma visão lógica dos dados e em seguida a sua real disposição no modelo relacional.

Da mesma forma, uma nova busca é feita, desta vez utilizando-se da interface de resgate pelo atributo, ilustrando como o usuário pode alcançar o resultado desejado.

A parte final é dedicada ao hibridismo, ilustrando a maneira pela qual um método indutivo pode ser aplicado ao RBC e assim extrair deste informações que sejam úteis e relevantes na manutenção do próprio sistema de raciocínio por casos. isso é feito ao mesmo tempo em que se apresenta uma execução do algoritmo proposto no Capítulo 5, o C_Apriori.

O próximo capítulo apresenta uma conclusão do que foi este trabalho, e mostra as contribuições e as perspectivas de trabalhos futuros que surgem a partir deste.

Capítulo 7
Conclusões e perspectivas de trabalhos futuros

A aplicação de métodos de aprendizagem em Sistemas de Help Desk provê um ganho significativo tanto em sua funcionalidade quanto na sua capacidade de prover informações capitalizáveis para as Organizações.

Abraçando áreas como Data Mining (Métodos de Associação e de Classificação e o relacionamento entre informações referentes a elementos distintos), Raciocínio Baseado em Casos (Recuperação pelo Atributo e por Casamento de Padrão), Banco de Dados (uma modelagem relacional consistente para que seja possível se apoiar na eficiência do SQL) e, muito claramente, na Inteligência Artificial, este trabalho trás uma proposta de hibridismo aplicável a Help Desk, utilizando-se de dois paradigmas diferentes e parcialmente integrados, para prover interfaces de atendimento ao usuário, verificação do funcionamento do sistema e geração de informações estratégicas a partir dos dados existentes num Sistema de Help Desk.

A concepção geral da aquisição de conhecimento aqui trabalhada engloba dois módulos: (i) um módulo que associa RBC a bancos de dados e (ii) um módulo que associa regras, data mining e Help Desk.

O primeiro utiliza a tecnologia de RBC apoiando-se em bancos de dados relacionais – notadamente SQL – para permitir que seja construída uma interface com o usuário que lhe proveja duas alternativas diferentes de acionamento do resgate, de acordo com a teoria básica do RBC: pelo Casamento de Padrões e pelo Atributo, respeitando-se e apresentando-se métodos para se contornar, bem evidentemente, as dificuldades específicas de cada uma delas: a inexatidão dos padrões textuais e o casamento parcial de atributos.

O segundo módulo, apoiado em Data Mining, permite que sejam extraídas, dos dados do Help Desk, informações que possam ser úteis de duas formas diferentes: para capitalização de conhecimento estratégico na relação dos produtos/serviços com os clientes e na gerência tática (ou

estratégica) do próprio sistema de atendimento. Para essa abordagem, foi utilizada um hibridismo, empregando métodos de regras de associação à extração de regras de classificação. Essa abordagem permite que o usuário do sistema tenha uma grande liberdade de ação a fim de fazer as extrações de informação de que necessite, relacionando toda sorte de informações disponíveis entre si além de prover ganhos de performance.

7.1 Contribuições

Este trabalho é o resultado de uma investigação multi-disciplinar, que levou em consideração áreas distintas da computação bem como algumas informações de caráter organizacional e administrativo. Entre as contribuições mais importantes, podem ser citadas:

- O levantamento bibliográfico, em separado, das teorias de RBC e de métodos indutivos de aquisição de conhecimento.

- A propositura de um método de resgate de casos que combina duas abordagens ou possibilidades diferentes (pelo atributo e pelo casamento de padrão) aplicado ao Help Desk.

- A propositura de resgate textual baseada em palavras-chave em hash com seus respectivos sinônimos. Esta abordagem é bastante empregada em sistemas de apoio ao usuário (não baseados em casos) como por exemplo o help online de muitas ferramentas mas ainda pouco explorada em RBC. Mostrou-se, aqui, uma maneira concreta de sua implantação.

- A propositura de resgate por atributos com casamento parcial, empregando o conceito de distância euclidiana entre valores de atributo. Neste trabalho, porém, este casamento foi abordado como uma característica intrínseca ao caso, e não ao método de busca, o que pode representar uma alternativa importante para uma infinidade de domínios, como por exemplo este que foi objeto do estudo.

- A disponibilização de um mecanismo para que um sistema suportado por RBC possa ter suas informações medidas e aferidas por um método indutivo (as regras extraídas), e assim se possa permitir que sejam diagnosticados problemas e/ou anomalias.

- A disponibilização de um mecanismo para que, a partir de dados do atendimento a clientes (como no Help Desk), possam ser extraídas informações úteis aos setores estratégicos.

- A exploração d uso de métodos de extração de regras de associação para regras de classificação. Conquanto isto não seja novidade, há ainda em seu âmago muitos espaços a serem explorados. Este trabalho apresenta uma contribuição a respeito pois consegue aliar as vantagens de métodos associativos, e particularmente de métodos aplicáveis ao caso distribuído, à extração de regras de classificação, dando uma visão bastante clara de onde estarão efetivamente os ganhos computacionais.

7.2 Perspectivas de trabalhos futuros

Para pesquisadores que possam se interessar em prosseguir neste caminho de pesquisa, as sugestões que imediatamente ocorrem deste trabalho são, dentre uma infinidade, estas que seguem:

- Criação de estruturas apontadoras de casos para regras e vice-versa. Isso permitirá que o sistema mostre-se mais consistente pois poderá apresentar uma justificativa inteligível para os casos (regras apontadas por eles) e uma efetiva garantia de consistência das regras (casos apontados por elas). Com isso, o sistema poderá ter a validade de suas informações mais facilmente verificadas.

- Na seleção de ocorrências que deverão originar casos computacionais, a tarefa poderá ser parcial ou totalmente automatizada com o uso de métodos estatísticos e, se for o caso, de um sistema especialista que execute esta tarefa.

- Exame de linguagem natural para Help Desk: O casamento de padrão na aplicação do Help Desk pode ter sua acurácia significativamente melhorada se for implementado o uso de um método de processamento de linguagem natural.

- Integrar, no momento do resgate, os métodos de recuperação pelo atributo e pelo casamento de padrão numa única busca interativa, dando ao usuário a possibilidade de juntar, a um só tempo, ambas as respostas dadas pelo sistema numa só.

- Os resultados da extração de conhecimento indutivo podem ser significativamente melhorados com o uso de informações de relevância semântica, como mostrado em [Mongiovi, 1995].

- Usar informações extraídas dos dados para se estabelecer os perfis do cliente e assim estabelecer mecanismos mais eficientes de atendimento.

- É possível que, usando-se os métodos de *sampling* [Toivonen, 1996], possa-se melhorar ainda mais a performance do método de extração de regras.

- A extração de regras a partir dos casos, com o uso de sua estrutura para a busca por casamento de padrão também é potencialmente útil na busca por inconsistências e/ou falhas no sistema.

- A discretização dos dados usados no algoritmo C_Apriori aqui proposto pode ser feita de maneira parcial ou totalmente automatizada, com o uso de algoritmos que empreguem métodos estatísticos.

- A geração e o uso de uma estrutura adicional – uma Tabela Sumarizada – que possuirá a lista de exemplos distintos disponíveis ao lado do número de vezes em que cada um deles aparece. Isto pode melhorar em muito a performance da extração de regras pois, em se utilizando esta abordagem de extração de regras de classificação por métodos de associação, a probabilidade de haverem exemplos coincidentes aumenta e muito.

- A integração total das duas abordagens, como já foi proposto na primeira perspectiva de trabalho futuro, pode ser suportada num ambiente formal de aprendizagem, garantido por um protocolo multi-agentes que permita a interferência humana, como o MOSCA, de maneira semelhante à que está mostrada em [Nóbrega, 1998].

Assim, esperando que o leitor deste trabalho tenha sido trazido ao mesmo patamar de entusiasmo científico que o autor, segue deste o sincero agradecimento pela atenção dispensada.

Referências Bibliográficas

[Agrawal, 1994]: Agrawal, R. & Srikant, R. "Fast Algorithms for Mining Association Rules", Proceedings of the 20th VLDB Conference Santiago, Chile, 1994

[Aha, 1997]: Aha, D. W. and Breslow, L. A., "Refining conversational Case Libraries", Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington DC, 1997.

[Albuquerque, 2000]: Albuquerque, A. R. R., "Proposta de um modelo de automação do planejamento da qualidade através da utilização de sistemas de Help Desk que empregam raciocínio baseado em casos", Dissertação de Mestrado, UFPb, COPIN, Novembro de 2000.

[Brown, 1995]: Brown, M.; Filer, N., "Beauty vs. The Beast: The Case Against Massively Parallel Retrieval", Progress in CBR. In: Lecture Notes in AI, 1020. First UK Workshop in CBR, pp. 42-58, Proceedings, Salford, UK, 1995.

[Carbonell, 1989]: Carbonell, J. G., "Introduction: Paradigms of Machine Lerrning", Artificial Intelligence, Vol. 40, 1989.

[Carvalho, 1998]: Carvalho, J., Sampaio, M. C., Mongiovi, G., "Utilizando Técnicas de Data Mining para o Reconhecimento de Caracteres Manuscritos"

[Cendrowska, 1988] Cendrowska, J., "PRISM: Na Algorithm for Inducing Modular Rules". Knowledge-Based Systems, vol. 1, Academic Press, 1988.

[Chen, 1996]: Chen, M-S., Han, J., Yu, P. S.. "Data Mining: An Overview from a Database Perspective", IEEE transactions on knowledges and data engineering, Vol 8, No 6, December 1996

[Doyle, 1986]: Doyle, C. "As aventuras de Scherlock Holmes", Círculo do Livro, São Paulo, 1986.

[Gorgônio, 1999]: Gorgônio, F. L., "Uma Arquitetura para Sistemas Inteligentes de Suporte ao Usuário", Dissertação de Mestrado, UFPb, COPIN, 1999.

[Holsheimer, 1994]: Holsheimer, M., Siebes, A., "Data Mining, the Search for Knowledge in Databases", Report CS-R9406 ISSN 0169-118X.

[Kolodner, 1991]: Kolodner, J.; Menachem, Y.J, "Case-Based Reasoning: An Overview." The Institute for the Learning Sciences, Northwestern University, Evanston (EUA), 1991.

[Kolodner, 1993]: Kolodner, J., "Case-Based Reasoning", San Mateo CA, Morgan Kaufmann Publishers, 1993.

[Korth, 1991] Korth, H. F., Silberschatz, A. "Database System Concepts – Second Edition", McGraw Hill, 1991

[Martin, 1989]: Martin C., "Indexing Complex Features", Procc. Case Based Reasoning Workshop, Holiday Inn, Pensacola Beach, Florida, May, 1989.

[Martins, 2000]: Martins, A. S., "Computação Baseada em Casos: Contribuições Metodológicas aos Modelos de Indexação, de Avaliação, de Ranking, e de Similaridade de Casos", Tese de Doutorado, UFPb, COPELE, 2000.

[Michalski, 1986]: Michalski, R. S., "Understanding the Nature of Learning: Issues and Research Directions", Machine Learning: An Artificial Intelligence Approach, Vol. 2, Morgan Kaufmann Publishers, Los Altos, CA, 1986.

[Minsky, 1975]: Minsky, M. A. "A framework for representing knowledge?" Psychology of Computer Vision. McGraw-Hill, N.Y., 1975.

[Mongiovi, 1995]: Mongiovi, G., "Uso de Relevância Semântica na melhoria da qualidade dos resultados gerados pelos métodos indutivos de aquisição de conhecimento a partir de exemplo", Tese de Doutorado, UFPb, COPELE, 1995.

[Nóbrega, 1998]: Nóbrega, G. M., "Especificação formal de um sistema de apoio à descoberta", Dissertação de Mestrado, UFPb, COPIN, 1998.

[Russel, 1995]: Russell, S.; Norvig, "P. Artificial Intelligence. A Modern Approach." Prentice-Hall, Inc., N.J, 1995.

[Savasere, 1995]: Savasere, A., Omiencinski E., Navathe S.. "An Efficient Algorithm for Mining Association Rules in Larges Databases", Proceedings of the 21st VLDB Conference Zurich, Swizerland, 1995

[Simon, 1983]: Simon, H. A. "Why should machine learn?" Machine Learning. An Artificial Intelligence Approach. Tioga Publishing Company, Palo Alto, CA, 1983.

[Srikant, 1995]: Srikant, R., Agrawal, R., "Mining Generalized Association Rules", Proceedings of the 21st VLDB Conference Zurich, Swizerland, 1995

[Thomas, 1996]: Thomas, A. H., Steele, R. M. : "Virtual Help Desk – Strategic Management Center", International Thomson Computer Press, EUA, 1996

[Toivonen, 1996]: Toivonen H.. "Sampling Large Databases for Association Rules", Proceedings of the 22nd VLDB Conference Mumbay (Bombay), India, 1996

[Watson, 1997]: Watson, I., "Applying Cased-Based Reasoning: Techniques for Enterprise Systems", Morgan Kaufmann Publishers, Inc. San Francisco, 1997.

Anexo I

Script SQL de criação dos objetos de banco de dados para o armazenamento de casos

```
/*********************************************************
*********************************************************/
/***********Tabelas para armazenar os casos**************/
/*********************************************************/
/*********************************************************/

/*********************************************************/
/***********Para busca por casamento padrão**************/
/*********************************************************/

create table tb_produto (
     cd_produto number
          constraint pk_prduto1 primary key,
     nm_produto varchar2(15) not null,
     ds_produto varchar2(50) not null
);

create sequence sq_produto;

create or replace trigger tg_produto_b
before insert
on tb_produto
begin
select sq_produto.nextval
into :new.cd_produto
from dual;
end;
/

create table tb_caso (
     cd_caso number
          constraint pk_caso1 primary key,
```

```
     cd_produto number
          constraint fk_caso2 referentes tb_produto,
     ds_caso varchar2(30) not null,
     ds_solucao varchar2 (30) not null
);

create sequence sq_caso;

create or replace trigger tg_caso_b
before insert
on tb_caso
begin
select sq_caso.nextval
into :new.cd_caso
from dual;
end;
/


create table tb_pergunta (
     cd_pergunta number
          constraint pk_pergunta1 primary key,
     ds_pergunta varchar2(50) not null
);

create sequence sq_pergunta;

create or replace trigger tg_pergunta_b
before insert
on tb_pergunta
begin
select sq_pergunta.nextval
into :new.cd_pergunta
from dual;
end;
/


create table tb_palavra (
     cd_palavra number
          constraint pk_palavra1 primary key,
     ds_palavra varchar2(30) not null
);

create sequence sq_palavra;

create or replace triger tg_palavra_b
```

```
before insert
on tb_palavra
begin
select sq_palavra.nextval
into :new.cd_palavra
from dual;
end;
/


create table tb_sinonimo (
     cd_sinonimo number
          constraint pk_sinonimo1 primary key,
     cd_palavra number
          constraint fk_sinonimo2 references tb_palavra,
     ds_sinonimo varchar2(30) not null
);

create sequence sq_sinonimo;

create or replace trigger tg_sinonimo_b
before insert
on tb_sinonimo
begin
select sq_sinonimo.nextval
into :new.sq_sinonimo
from dual;
end;
/

create table tb_caso_perg_palavra (
     cd_caso number
          constraint fk_caso_perg_palavra1 references
tb_caso,
     cd_pergunta number
          constraint fk_caso_perg_palavra2 references
tb_pergunta,
     cd_palavra number
          constraint fk_caso_perg_palavra3 references
tb_palavra,
     constraint pk_caso_perg_palavra4 prinary key
          (cd_caso, cd_pergunta, cd_palavra)
);

/*************************************************************
/
/********* Para busca pelo casamento de atributo************/
```

```
/*********************************************************
/

create table tb_atributo (
     cd_atributo number
          constraint pk_atributo1 primary key,
     nm_atributo varchar2(15) not null;
     ds_atributo varchar2(50),
);

create sequence sq_atributo;

create or replace trigger tg_atributo_b
before insert
on tb_atributo
begin
select sq_atributo.nextval
into :new.cd_atributo
from dual;
end;
/


create table tb_valor (
     cd_valor number
          constraint pk_valor1 primary key,
     cd_atributo number not null
          constraint fk_valor2 references (tb_atributo),
     nu_ordem number not null,
     ds_valor varchar2(15) not null,
     constraint un_valor3 unique (cd_atributo, nu_ordem)
);

create sequence sq_valor;

create or replace trigger tg_valor_b
before insert
on tb_valor
begin
select sq_valor.nextval
into :new.cd_valor
from dual;
end;
/


create table tb_atributo_valor_caso (
```

```
     cd_atributo number
          constraint fk_atributo_valor_caso1
               references tb_atributo,
     cd_valor number
          constraint fk_atributo_valor_caso2
               references tb_valor,
     cd_caso number
          constraint fk_atributo_valor_caso3
               references tb_caso,
     vl_classificacao number(1) not null,
     vl_distancia_maxima number(1) not null default 0,
     constraint pk_atributo_valor_caso4 primary key
          (cd_atributo, cd_valor, cd_caso),
     constraint un_atributo_valor_caso5 unique
          (cd_atributo, cd_valor, cd_caso, vl_classificacao)
);
```

Anexo II

Algoritmos clássicos de extração de regras de classificação e associação

Algoritmo ID3 {e demais da família Tididt}

Entrada: Um conjunto de treinamento D, uma condição de parada t(D) e uma função de avaliação aval(D, a)
Se todos os exemplos em D satisfazem t(D) então
  Retornar o valor da classe
Senão
  Para cada atributo a, calcular aval(D, a)
    Seja a_m o atributo que possui o melhor valor aval(D, a)

    Dividir D em subconjuntos com valores v_{m1}, ..., v_{mnm} do atributo a_m

    Para cada subconjunto D_k ($1 \leq k \leq nm$)
      ID3 (D_k,T(D_k), aval(D_k, a))

Algoritmo PRISM

Entrada: um conjunto de classes CT
Para cada classe E_j em CT
  Enquanto $\exists$ exemplo em CT com E_j **faça**
    Lista_Condições = $\varnothing$
    CT_Aux = CT
    Repete
      Seja C $\in$ CT_Aux | p(E_j|C) seja máximo
      Lista_condições += C
      CT_Aux = X | X $\in$ CT_Aux e X $\supset$ C
      Remover o atributo de C de CT_Aux
    Até Conjunto_Classes(CT_Aux) = E_j
  Nova_Regra = SE lista_condições ENTÃO E_j
  Remover de CT todos os exemplos cobertos por Nova_Regra
Restaurar CT original

Algoritmo Apriori

```
Entrada: Banco de transações D e SupMin;
Saída:     Grandes conjuntos de itemset (L)

L1 = {grande 1-itemsets};
k = 2;
Enquanto Lk-1 ≠ ∅ faça
   início
     Ck = apriori-gen(Lk-1);  { gera novos candidatos }
     Para toda transação t∈D faça
        início
          Ct = subconj(Ck, t);  {candidatos contidos em t}
          Para todo candidato c∈Ct faça
                c.suporte ++
        fim;
     Lk = {c∈Ck | c.suporte ≥ SupMin};
     k = k +1
   fim;
Retorna (L = ∪k Lk ).
```

Apriori-gen

```
Para cada P ∈ Lk-1
     Para cada Q ∈ Lk-1, Q > P
          Se |P∪Q| = k
               Insere (P∪Q) em Ck
```