Sistemas de Raciocínio Baseado em Casos

Diego L. Ervatti[1], Patrick P. Andrade[1], Rafael Z. de Sá Silva[1], Erlon Pinheiro[1]

[1]Centro Universitário Vila Velha - (UVV)
Curso de Ciência da Computação
Rua Comissário José Dantas de melo, 21
Boa Vista - Vila Velha - ES - Brasil
CEP.: 29102-770
31/05/2010

{diegoervatti,patrickpeyneau,rafael_zacche}@hotmail.com, erlon@uvv.br

Abstract. *Artificial Intelligence (AI) is a research area that seeks a computational artifice that simulates the human capacity to solve problems. Different methods are employed to simulate the way humans think, from a wide variety is included the case-based reasoning, which tries to imitate the way humans recall past experiences to solve a given problem. This paper describes in details what is a CBR system, its application, and presents some key techniques used in building systems that use this technology. Some tools that aid in the construction of CBR systems are described. In the end it is proposed a development of an e-commerce SRBC.*

Keywords: artificial intelligence, case-based reasoning, reasoning in artificial intelligence.

Resumo. *A Inteligência artificial (IA) é uma área de pesquisa que busca por artifícios computacionais que simulem a capacidade humana de resolver problemas. Diferentes métodos são empregados para simular a forma como humanos pensam, dentre uma ampla variedade está o Raciocínio Baseado em Casos, que tenta imitar a forma como humanos relembram experiências passadas para resolver um dado problema. O presente artigo descreve pormenorizadamente o que é um sistema de raciocínio baseado em casos, suas aplicações, além de apresentar algumas das principais técnicas adotadas na construção de sistemas que utilizam esta tecnologia. Por fim com base nas ferramentas que serão apresentadas, propõe-se o desenvolvimento de um sistema de vendas online baseado em casos.*

Palavras chaves: inteligência artificial, raciocínio baseado em casos, raciocínio em inteligência artificial.

1. Introdução

Raciocínio baseado em casos (RBC)[1] é uma abordagem para solução de problemas e para o aprendizado com base em experiência passada [Wangenheim 2003]. Em síntese pode-se afirmar que o RBC resolve um novo problema a partir de soluções de problemas análogos que foram bem sucedidas no passado. A solução similar encontrada é adaptada para uma melhor representação e retida para futuras referências.

No enfoque do RBC, os problemas são conhecidos como casos, que são armazenados em uma base de conhecimento ou base de casos. RBC foi influênciado por diversas áreas, tais como, ciências cognitivas, sistemas baseados em conhecimento, aprendizado de máquina, bases de dados, recuperação de informações, redes neurais, reconhecimento de padrões, incerteza e estatística.

Na seção 2 apresenta-se um breve histórico sobre RBC, as estruturas comumente utilizadas, além do ciclo de vida do RBC. Na seção 3 alguns dos métodos para representação de casos são descritos, os tipos comumente utilizados, suas vantagens e desvantagens. A seção 4 é dedicada as medidas de similaridade, assim como sua representação matemática. As fases do ciclo de vida do RBC, recuperação de casos, reutilização de casos, revisão de casos e retenção de casos são apresentadas nas seções 5, 6, 7 e 8 respectivamente. Na seção 9 propõe-se um projeto que utiliza o raciocínio baseado em casos.

2. Raciocínio baseado em casos

Entre as habilidades inteligentes está a habilidade para armazenar e recuperar eficientemente grande quantidade de informação, além da capacidade de adaptar ou modificar um conteúdo [Rezende 2005].

Os primeiros aspectos do RBC foram infundidos nos trabalhos de Schank e Abelson sobre memória dinâmica e modelo cognitivo. O RBC originou-se em modelos cognitivos da solução de problemas e aprendizado de novas situações, no geral, seria gravar essas situações em memória como se fossem roteiros, ou seja, algumas coisas ocorrerão sempre conforme o esperado.

CYRUS foi o primeiro sistema de RBC, desenvolvido por Janet Kolodner na *Yale University*, seu enfoque era solução de problemas diplomáticos. Após esse sistema, diversos outros surgiram como: Mediator, Persuader, Chef, Julia, Hypo, Protos, Casey, Pantdex, dentre outros, cada um com enfoque específico.

O objetivo do RBC é tentar resolver um problema com base em experiências passadas. Conforme [Luger 2004], os sistemas RBC compartilham uma estrutura comum. Para cada novo problema:

- Recuperam o caso mais similar na base de casos;
- Reutilizam este caso para resolver o problema;
- Revisam a solução indicada;
- Retém a experiência representando o caso atual para referências futuras.

[1] Do inglês, *Case-Based Reasoning (CBR)*.

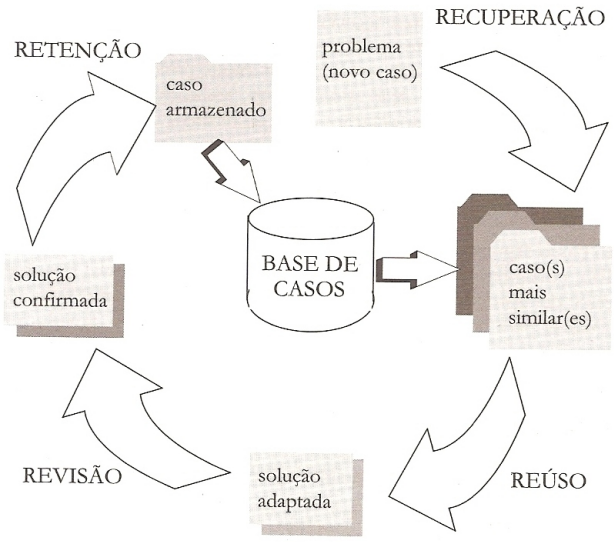

Figura 1. Ciclo de vida do raciocínio baseado em casos [Wangenheim 2003].

3. Representação de casos

Todo conhecimento que envolve sistemas de RBC são armazenados sob a forma de casos, casos estes que representam um conhecimento específico de experiências ou episódios concretos.

Não existe um conteúdo exato do que será representado por um caso, seu conteúdo é totalmente dependente da aplicação. O que há de comum entre eles é que todos tem como objetivo representar uma situação ocorrida que poderá futuramente ser relembrada, adaptada e aplicada a uma nova solução.

Exemplos de como casos podem conter diferentes conteúdos:

Comércio eletrônico: lista das características do produto (por exemplo, marca: fiat, modelo: siena, cor: prata, preço: R$ 30.000).

Comércio eletrônico: lista das características do produto (por exemplo, nome: mesa, tipo: madeira, cor: marfim).

3.1. Representação de casos

Várias estratégias podem ser adotadas para representação de casos, tais como: representação atributo-valor, orientação a objetos, redes semânticas e árvores k-d.Existem outras formas de representação de casos, mas somente as citadas serão abordadas no presente artigo.

O Ponto mais difícil na solução de problemas baseado em casos, independentemente da estrutura de dados selecionada para a representação dos casos, é a escolha das características relevantes para a indexação e a recuperação de casos [Luger 2004], que é apresentado na seção 5.

“A representação do caso significa codificar o conhecimento contido nos casos mediante uma ampla gama de formalismos representacionais ou linguagens desenvolvidas no âmbito da IA, permitindo armazená-lo em uma forma simples, concisa, completa e

clara ”[Russell 2004].

A metodologia a ser adotada para representar um caso dependerá do contexto ao qual o RBC está sendo aplicado e como os dados armazenados serão alterados.

3.2. Representação atributo-valor

A representação atributo valor é a forma mais simples de representação de casos, na qual um item é representado por um par atributo-valor [Wangenheim 2003].

Esta representação pode ser vista na tabela 1.

Atributos	**Valor**
Descrição	“Carro,siena, prata”
Ano	2010
Cor	Prata
Marca	Fiat
Modelo	Siena
Preço	R$ 35.125,00

Tabela 1. Exemplo de uma representação atributo-valor.

A representação de casos atributo-valor pode ser definida basicamente por tipos como: booleano, data, símbolo (ordenado, não-ordenado ou taxonomia), número e string.

Existem inúmeras vantagens na utilização de par atributo-valor, tais como:

- Representação simples e fácil de programar;
- Fácil implementação de medidas de similaridade;
- Fácil armazenamento, podendo ser armazenado em SGBD[2] relacionais;
- Recuperação eficiente.

Porém apresenta desvantagens como:

- Não é capaz de representar informações estruturais;
- Não é capaz de representar informações relacionais.

É recomendável que pares atributo-valor sejam usados somente para tarefas de diagnóstico que tem que lidar com grandes bases de casos.

3.3. Representação orientada a objetos

Nesta representação os casos são vistos como objetos[3], que por sua vez são agrupados por tipos similares formando uma classe[4].

Essa representação permite a modelagem de relacionamento entre diferentes tipos de objetos, tais como: relações taxonômicas, relações composicionais e relações especiais.

[2]Sistema de gerenciamento de banco de dados.
[3]Instância de uma classe.
[4]Estrutura que abstrai um conjunto de objetos com características similares.

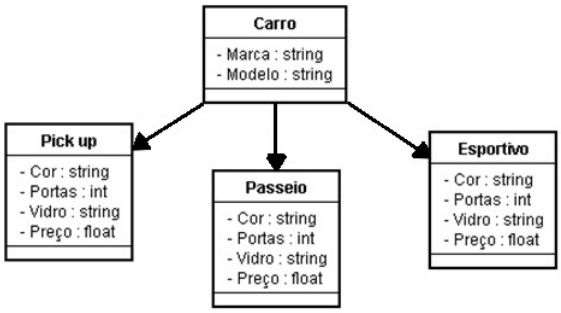

Figura 2. Representação orientada a objetos.

Algumas vantagens dessa representação:

- Representação estruturada e natural de casos;
- Representação direta de informações estruturais e relacionais;
- Armazenamento compacto.

Porém possui as seguintes desvantagens:

- Cálculo de similaridade complexo;
- Recuperação de casos complexa.

3.4. Representação utilizando redes semânticas

É uma forma de representação do conhecimento determinada como um grafo direcionado no qual os vértices representam unidades conceituais, e as arestas representam relacionamentos entre essas unidades [Real 2003].

O tipo de rede semântica utilizada em RBC denomina-se Rede de Recuperação de Casos (RRC)[5]. Esta é uma forma eficiente e flexível de recuperação de casos, capaz de entender termos vagos e ambíguos, além da eficiente manipulação de bases de casos de tamanho razoável [Wangenheim 2003].

De acordo com [Wangenheim 2003] o conceito fundamental das RRC são as Entidades de Informação (EI). Um caso consiste de um conjunto de EI, e a base de casos em uma rede com nodos para as EI observadas no domínio e nodos adicionais denotando os casos particulares.

3.5. Representação utilizando árvore k-d

Árvore k-d é uma árvore de pesquisa binária k-dimensional onde cada nível desta árvore se ramifica baseando-se no discriminador, que por sua vez, tem como base uma medida estatística denominada amplitude interquartil[6] [Goetze 2005]. Sua divisão é feita alternadamente utilizando as coordenadas x, y de acordo com a profundidade da árvore [Heredia, Iochpe, Comba, 2003], como pode ser observado na figura 3.

[5]Do inglês, Case Retrieval Nets.

[6]É dada pela diferença entre o quartil superior e o quartil inferior.

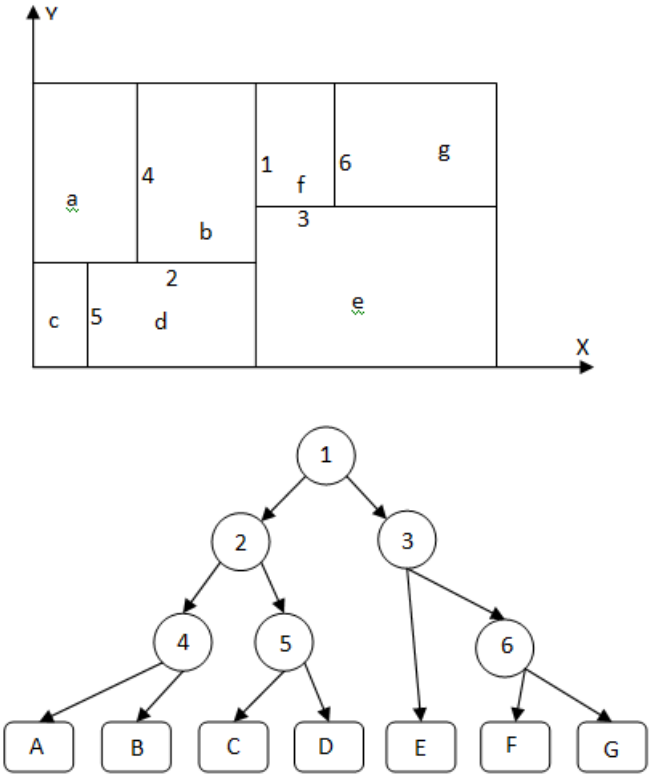

Figura 3. Espaço de busca bidimensional hipotético e a árvore k-d de busca correspondente.

A construção da árvore k-d é feita por uma função recursiva que verifica a possibilidade de continuar a divisão dos dados para adicionar novas ramificações.

Essa é a estrutura de dados comumente utilizada por ser aplicável a qualquer tipo de sistema, onde se queira fazer recuperação de chaves secundárias ou multichaves, visando a criação de estruturas de índices sobre base de casos que contenham grandes quantidades de dados. Porém, possui a desvantagem de gerar árvores de grande profundidade e sua inserção balanceada é extremamente custosa.

4. Similaridade

O objetivo do RBC é recuperar casos de sua base que sejam o mais similar possível ao problema a ser resolvido. Os casos recuperados não precisam ser necessariamente idênticos a situação atual, porém quanto maior o nível de similaridade melhor será a solução encontrada.

Casos podem ser considerados similares se eles forem úteis para a solução do caso atual e seu nível de similaridade varia entre 0 e 1, onde 0 é a dissimilaridade total e 1 a coincidência absoluta. Apresenta-se nesta seção vários métodos heurísticos para determinar o caso de maior utilidade.

4.1. Modelos de similaridade global simétrica

A similaridade global simétrica é calculada entre objetos com base nos seguintes modelos: similaridade como distância geométrica, *nearest neighbour*, distância euclidiana, distância euclidiana ponderada e métrica do quarteirão (distância de *Manhattan*).

4.1.1. Distância geométrica

Consiste em determinar o menor valor de similaridade para encontrar o vizinho mais próximo, por exemplo, considere os três casos, chamados A, B e C, onde o caso A se refere ao caso atual.

Nessa medida calcula-se a distância de x e y para A. A distância x e y em relação a A para o caso B é de 5 e 1, respectivamente, enquanto que a distância x e y entre os casos A e C é 5 e 2, respectivamente.

Portanto:

Eixo	**A**	**B**	**C**
X	-	5	5
Y	-	1	2

Tabela 2. Representação da distância geométrica.

- A distância de A ao caso B: $d_1 = x_1 + y_1 = 5 + 1 = 6$;
- A distância de A ao caso C: $d_2 = x_2 + y_2 = 5 + 2 = 7$.

Conclui-se então que o valor mais próximo de A é o caso B.

4.1.2. *Nearest neighbour* ponderado [Watson 1999]

Semelhante a distância geométrica, porém utiliza o conceito de peso para cada atributo. Por exemplo, podemos considerar que o modelo de um produto possui um peso maior do que a cor do produto.

Portanto temos:

Eixo	**A**	**B**	**C**	**Modelo**	**Cor**
X	-	5	5	1	1
Y	-	1	2	1	0

Tabela 3. Representação *nearest neighbour* ponderado.

- A distância de A ao caso B: $d_1 = x_1 * p_1 + y_1 * p_2 = 5 * 1 + 1 * 1 = 6$;
- A distância de A ao caso C: $d_2 = x_2 * p_1 + y_2 * p_2 = 5 * 1 + 2 * 0 = 5$.

Conclui-se então que o valor mais próximo de A é o caso C.

A fórmula pode ser generalizada para:

$$sim(Q, C) = \sum_{i=1}^{n} f(Qi, Ci)wi \quad (1)$$

- Q = problema atual;
- C = caso recuperado da base;
- N = número de casos da base;
- F = similaridade local(a princípio utilizaremos F=1);
- W = peso.

4.1.3. Distância euclidiana

Dado um problema Q e um caso C representados por índices $Q = (q_1, q_2, ..., q_n)$ e $C = (c_1, c_2, ..., c_n)$.

A distância euclidiana representa a real distância entre dois pontos em um espaço qualquer e é dada por:

$$d(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \quad (2)$$

4.1.4. Distância euclidiana ponderada

Inserindo pesos W, podemos estender a distância euclidiana para:

$$d(Q, C) = \sqrt{\sum_{i=1}^{n} w_i (q_i - c_i)^2} \quad (3)$$

4.1.5. Métrica do quarteirão (distância de *Manhattan*, distância pombalina ou distância de taxi)

É uma medida neutra, avalia todas as diferenças de forma idêntica.

$$d(Q, C) = \sum_{i=1}^{n} |qi - ci| \quad (4)$$

4.2. Medidas de similaridade local

Ao término do cálculo de similaridade global, pode haver a necessidade de um cálculo mais preciso que atue sobre os atributos, essa medida de similaridade é denominada similaridade local.

A medida de similaridade deve ser definida em relação ao tipo específico de um atributo, tais como: número, símbolo binário, símbolo (ordenado, não-ordenado, taxonômico), conjunto, intervalo, *string*, dentre outros.

Atributos	Situação atual	Caso 1	Valor similaridade
Descrição	Carro, siena, prata	Carro, siena	1
Ano	2010	2009	0.9
Cor	Prata	Preto	0.5
Marca	Fiat	Fiat	1
Modelo	Siena	Siena	1
Preço	R$ 35.125,00	R$ 32.540,00	0.8

Tabela 4. Tabela de demonstração de similaridade local.

4.2.1. Número

Pode ser expressa pelo módulo da diferença entre valores, será utilizado o preço do carro como exemplo: $|preço_1 - preço_2|$.

4.2.2. Função escada

Se o caso é totalmente útil ou totalmente inútil, então a similaridade local deve ser definida por meio de uma função escada.

- $F(p_1, p_2)$ = 0;
- $|p_1 - p_2| \geq$ S sendo 'S' o ponto do degrau.

O preço dos carros poderá ser considerado igual quando sua diferença for menor que R$ 350,00, e sua similaridade será 1, para valores maiores de R$ 350,00 será considerado dissimilar e assumir o valor de similaridade igual a 0.

4.2.3. Função linear

Adequada para a maioria dos tipos numéricos. A idéia é que a similaridade cresce com o decrécimo da distância entre os dois valores, ponderada pelo tamanho do intervalo assumido pelo domínio dos valores do atributo, [Wangenheim 2003].

$$\rho(p_i, p_j) = \begin{cases} 1 & p^i = p^j \\ 1 - \frac{|p_i - p_j|}{l_s - l_i} & \text{senão} \end{cases} \quad (5)$$

- l_s= limite superior;
- l_i = limite inferior.

4.2.4. Tipo escalar

As medidas de similaridade para tipos numéricos possuem um problema como: valores muito grandes têm o mesmo valor de similaridade que valores muito pequenos, por exemplo: a similaridade de 30.120 e 30.110 é igual à similaridade de 10 e 20, isso ocorre porque a distância é a mesma.

Ao atribuir valores diferentes de similaridade de acordo com o tamanho do valor, é possível adotar uma escala logarítmica onde a similaridade de valores pequenos será menor que a similaridade de valores grandes.

4.2.5. *Strings*

É aconselhável substituir strings por valores simbólicos, porém existem alguns enfoques para cálculo da similaridade entre strings:

Correspondência exata: as strings serão consideradas similares se e somente se forem escritas da mesma forma.

Correção ortográfica: grau de similaridade é expresso pegando-se o número de caracteres iguais dividido pelo número total de caracteres da maior string, por exemplo, ”writeln”e ”write”, o grau de similaridade será 5/7=0.7.

Contagem de palavras: nesse caso conta-se o número de palavras idênticas, por exemplo, ”carro siena vermelho quatro portas”e ”carro siena vermelho duas portas”, então o número de palavras idênticas é 4 e o número total de palavras da consulta é 5, logo a similaridade será 4/5= 0.8.

5. Recuperação de casos

Seu objetivo é encontrar um ou mais casos na base de conhecimento que seja similar ao caso atual, ou seja, que pode ser útil à situação atual. Para isso, é necessário casar a descrição do caso atual com os casos armazenados, baseando-se em uma medida de similaridade. Esta seção irá detalhar a estrutura geral da recuperação de casos.

O processo de recuperação pode ser formalmente descrito por um conjunto de subtrarefas que devem ser realizadas pelo sistema RBC, são estas: Assessoramento da situação, casamento e seleção.

O **assessoramento** da situação, mais complexa dentre as três subtarefas, utiliza conhecimento e pode exigir uma iteração proativa com o usuário. O objetivo é encontrar os casos potencialmente úteis a situação atual, logo, é necessário utilizar a situação atual como entrada para o processo de recuperação. Para isto, são utilizados índices(características) que são relevantes para encontrar o caso adequado chamados *descritores de entrada*.

Quando a primeira descrição da situação atual estiver incompleta, o assessoramento elabora índices relevantes para permitir a recuperação de casos adequados. Neste contexto, a seleção de testes que deverão ser feitos é muito importante para o ganho de informação potencial, aumentando as evidências associadas à situação corrente. Este procedimento é chamado de Levantamento de Sintomas Dirigido.

O **casamento** de casos consiste em associar a descrição do caso atual com um ou mais casos da base de casos de acordo com as medidas de similaridade.

Os algoritmos responsáveis por esta etapa da recuperação de casos podem ser caros computacionalmente por necessitar da combinação de busca e comparação de casos, no entanto existem deferentes técnicas de recuperação que podem ser aplicadas para tornar o processo mais ágil.

Qual técnica utilizar depende da estruturação da base de casos, índices e medidas de similaridade. A partir do conjunto de casos similares gerado pela etapa de casamento, é realizada a escolha de um caso, considerado o mais similar.

Dentre as características da recuperação de casos, estão a qualidade da solução gerada e a eficiência do processo de solução. Visando estes pontos, os conceitos de matemáticos de correção e completeza da solução são aplicados.

Um método é considerado **correto**, se uma relação de similaridade definida pelo método entre um caso e o problema atual também existe no conceito de similaridade desenvolvido para a aplicação.

Um método é considerado **completo**, se toda relação de similaridade representada no modelo do sistema também se encontra no resultado deste método de recuperação.

5.1. Técnicas de recuperação

Esta seção destaca as principais técnicas de recuperação de casos.

5.1.1. Recuperação Seqüencial

Técnica mais simples, onde a medida de similaridade é calculada sequencialmente para todos os casos na base de casos. Permite a determinação dos 'm' casos mais similares. Vantagens:

- Comprovadamente completa e corret. Completa porque analisa todos os casos da base, e correta porque aplica diretamente e na íntegra o conceito de similaridade definido pelo sistema a cada caso;
- Implementação simples;
- Aceita utilização de medidas de similaridade arbitrárias;
- Permite consultas ad-hoc independentes do conceito de similaridade. Desvantagens:
- Performance fraca para bases de casos grandes;
- Esforço de recuperação constante, independente da complexidade e do número de casos a serem recuperados;
- Esforço de recuperação não pode ser reduzido ou limitado por conhecimento adicional.

```
TIPOS:
  1. Tipocaso = ...
  2. SimCaso = REGISTRO
  3.    case: TipoCaso;
  4.    similaridade: [0..1]
  5. FIM;
VARIAVEIS
ListaCasoSim: VETOR[1..m] DE SimCaso
CaseBase: ARRAY[1..n] DE TipoCaso (* base de casos *)
Consulta: TipoCaso
```

Tabela 5. Estrutura de dados (Wangenheim, 2003).

```
1. FUNÇÃO SelecRel(CaseBase, Consulta, m): ListaCasoSim
2. INÍCIO
3. ListaCasoSim[1..m].similaridade :=0
4. PARA 1:=1 TO n FAÇA
  i. SE sim(Consulta, CaseBase[i])¿ListaCasoSim[m].similaridade
  ii. ENTÃO insira CaseBase[i] em ListaCasoSim
5. RETORNE ListaCasoSim
6. FIM
```

Tabela 6. *Algoritmo de recuperação seqüencial (Wangenheim, 2003).*

5.1.2. Recuperação de dois níveis

Neste método, o princípio é reduzir ao máximo o número de comparações de casos por meio da aplicação de limitações adequadamente escolhidas à base de casos. Assim, são excluídos da comparação com a descrição do problema atual os casos para os quais se pode determinar, com certeza, que não serão úteis para a solução.

Isto significa que aqui serão aplicadas heurísticas para a redução do espaço de busca, para agilizar o processo de recuperação. Baseado no modelo MAC/FAC[7], o método de recuperação de dois níveis funciona por meio dos seguintes processos:

1. Pré-seleção de possíveis candidatos;
2. Ordenação dos candidatos de acordo com o conceito de similaridade.

No primeiro passo, denominado passo-MAC, candidatos potenciais à solução são determinados a partir do conjunto de todos os casos por meio de similaridades no nível mais baixo da descrição sintática dos casos.

No segundo passo, denominado passo-FAC, os melhores candidatos são escolhidos a partir do conjunto de casos pré-selecionados. Isto é realizado por meio de comparações mais complexas, com aplicação de análise estrutural dos casos ou mesmo de todo o conceito de similaridade.

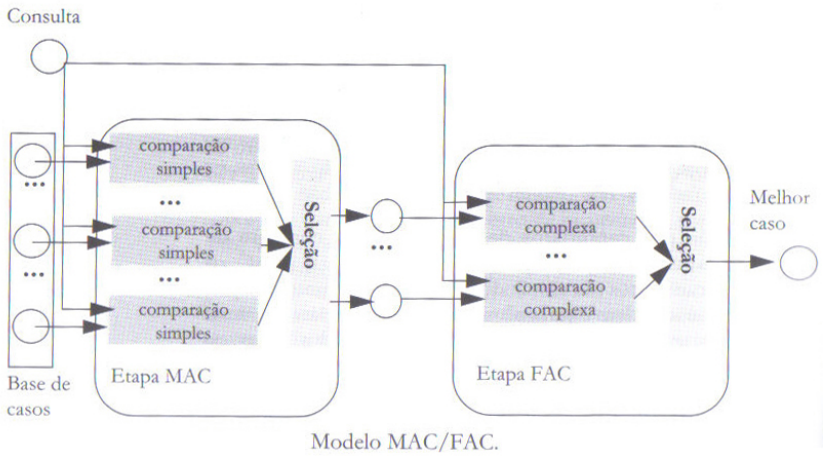

Figura 4. Modelo MAC/FAC (Wangenheim, 2003).

[7]Do inglês, *Many Are Called; Few Are Chosen*.

```
1. FUNÇÃO RestrigeBase(CaseBase, Consulta, m): ConjCasos
2. INÍCIO
3.   ...
4.   RETORNE(ConjCasos)
5. FIM

6. FUNÇÃO Selec2N(CaseBase, Consulta, m): ListaCasoSim
7. INÍCIO
8.  RETORNE (SelecSeq(RestringeBase(CaseBase,Consulta,m),Consulta,m))
9. FIM
```

Tabela 7. Algoritmo de recuperação de dois níveis (Wangenheim, 2003).

Este modelo pode ser facilmente mapeado em um banco de dados relacional. No primeiro passo, uma série de consultas (que representam o predicado SIM()) recupera o conjunto de candidatos potenciais. A medida de similaridade calculada por meio da aplicação do conceito SIM() sobre cada informação pode ser usada como chave para ordenação desta tabela.

A utilização deste método provê uma melhora de desempenho considerável, caso o número de candidatos potenciais seja consideravelmente menor do que o total de casos na base. A complexidade será equivalente a soma dos custos para aplicar a pré-seleção a todos os casos da base mais os custos da determinação seqüencial da medida de similaridade para os casos selecionados no primeiro passo.

Esta melhoria na performance pode ter um preço: a possibilidade de erros de recuperação. Estes possíveis erros são classificados em dois tipos:

- Erro α: um caso suficientemente similar à consulta não é selecionado no primeiro passo;
- Erro β: um caso que não é suficientemente similar à consulta é selecionado no primeiro passo como candidato potencial.

Desta forma, a eficiência deste método fica por conta dos erros α e β, o erro α influencia diretamente na eficiência do método, pois afeta a completeza do resultado. Caso o erro β, se apresente em grande número, o desempenho do algoritmo é afetado, pois casos irrelevantes aparecerão no conjunto de potenciais candidatos. Isto deve ser controlado pela escolha do predicado de pré-seleção SIM().

5.1.3. Determinação de SIM()

Existem vários enfoques para se determinar a SIM(), dentre eles:

- **Igualdade**: todos os atributos de um caso correspondem aos atributos do caso em questão;
- **Igualdade parcial**: pelo menos um atributo de um caso corresponde ao atributo do caso em questão;
- **Similaridade local**: todos ou k atributos de um caso são similares aos atributos do caso em questão;
- **Similaridade local parcial**: pelo menos um dos atributos de um caso é suficientemente similar a um atributo do caso em questão;

Vantagens:

- Possível vantagem de performance;
- Correção do método;
- Custo de recuperação é variável;
- Utilização de conhecimento;
- Consultas ad-hoc são possíveis.

Desvantagens:

- Possibilidade de erros;
- Ganho em performance não é garantido;
- Difícil definição do predicado SIM().

5.1.4. Recuperação Orientada a Índices

Este método é executado em duas fases, primeiro seleciona os índices adequados a serem utilizados, então efetua a recuperação baseada nestes índices:

1. **Pré-processamento**: todos os casos são analisados com base em critérios tomados a partir da consulta e uma estrutura de índices para o acesso é gerada.
2. **Recuperação**: os casos mais similares são determinados a partir da estrutura de índices gerada anteriormente.

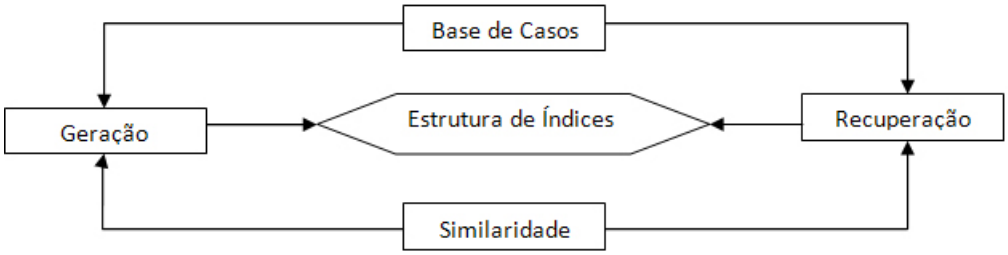

Figura 5. Princípio básico de métodos de recuperação orientados a índices.

A principal vantagem desse método é o ganho de performance devido à comparação de casos durante a recuperação. É necessário atingir um equilíbrio entre custos de recuperação de casos, complexidade de processamento e manutenção de uma estrutura complexa de índices.

Suas desvantagens são: custos de processamento elevados, o esforço de implementação e a não possibilidade de efetuar consultas ad-hoc.

1. Desça a árvore até uma folha
2. Calcule a similaridade para os casos na folha
3. Determine término
4. Determine mais candidatos
a. SE repositorios intersectam ENTAO
b. Pesquise em ramos alternativos
5. Retorne a p.3
7. Pare se não há repositórios que intersectam

Tabela 8. Princípio da recuperação em árvores k-d (Wangenheim, 2003).

1. INÍCIO
2. SE K é folha ENTÃO
3. PARA cada caso F de K FAÇA
4. SE sim(Q,F) $>$ scq[m] ENTÃO
5. Insira F em scq
6. SENÃO (* nodo interno *)
7. SE Ai é o atributo e v_i o valor que rotulam K
8. SE Q[A_i] $\leq v_i$ ENTÃO
9. recupere($K_{\leq}$)
10. SE Teste-EIL é verdade ENTÃO
11. recupere($K_{>}$)
12. SENÃO
13. recupere($K_{>}$)
14. SE Teste-EIL é verdade ENTÃO
15. recupere($K_{\leq}$)
16. SE Teste-EDL é verdade ENTÃO
17. Termine recuperação com scq
18. FIM

Tabela 9. ALGORITMO recupere (K: árvore k-d) (Wangenheim, 2003).

5.1.5. Recuperação com árvores k-d

Na recuperação com árvores k-d, o espaço de busca é estruturado com base em sua densidade observada, a recuperação então, é feita nesta estrutura.

Principais vantagens:

- Recuperação extremamente eficiente;
- Esforço de recuperação depende do número *m* de casos que se deseja recuperar;
- Extensão incremental se mais casos se tornarem disponíveis;
- Armazenamento dos casos em uma base de dados fácil de ser realizado.

Desvantagens:

- Esforço maior na criação da estrutura de índices da árvore k-d.

5.2. Redes de Recuperação de Casos

Os principais passos do processo de recuperação são:

1. Ativação das Eis dadas por meio do caso de recuperação;
2. Prorrogação dessa ativação de acordo com a similaridade por meio de toda a rede de EI até que nodos de casos sejam alcançados;
3. Coleta da ativação adquirida nos nodos de caso associados, a qual reflete a similaridade à consulta.

Formalmente, pode-se definir uma rede de recuperação de casos básica como descrito abaixo:

N=[E, C, S, F, T) onde: . E é um conjunto finito de nodos EI; . C é um conjunto finito de nodos de casos; . S É uma medida de similaridade S: E x E $\rightarrow$ R que descreve a similaridade local S (e', e") entre duas EIs e', e"; . F é uma função de relevância F: E x C $\rightarrow$ R que descreve a relevância (peso) F(e,c) da EI e para o nodo de caso c; . T é o conjunto de funções de propagação t_n: $R^E \rightarrow$ R para cada nodo n ϵ E $\cup$ C.

Tabela 10. Rede de recuperação de casos (Wangenheim, 2003).

Vantagens:

- Representação de medidas de similaridade compostas;
- Capacidade de manipular consultas incompletas;
- Medidas de similaridade sensíveis a contexto;
- Recuperação eficiente;
- Recuperação completa;
- Recuperação e representação flexíveis de casos.

Desvantagens:

- Construção inicial requer custos computacionais altos;
- Sensível a inconsistências nos dados armazenados;
- Falta de nodos para atributos numéricos e muitos passos de propagação são necessários se o grau de conectividade é alto e com muitas EI semelhantes.

5.3. Qual o melhor método?

O método que melhor irá se adaptar a uma aplicação dependerá dos seguintes aspectos:

- Representação de casos: casos se apresentam em uma estrutura fixa, pré-definida, ou sem formato e pode variar de acordo com a situação do sistema;
- Estrutura de Base de casos: tamanho desta base, do conhecimento a ser aplicado ou a eficiência que será necessária;
- Medida de similaridade: locais, globais ou até mesmo os dois tipos, dependendo da situação.

6. Reutilização de casos

A partir de um caso armazenado na base, sua solução e os detalhes de como o mesmo foi realizado são objetos que podem ser analisados para a solução de um novo problema no sistema. O simples fato de se utilizar um caso armazenado caracteriza a reutilização em sistemas de RBC.

O processo de reutilização se dá através da adaptação de soluções encontradas de casos anteriores, que podem ser utilizadas na solução de novos casos, porém, espera-se que esta adaptação não seja necessária, e sim, que o sistema possa encontrar uma solução em potencial para o problema apresentado.

Conforme [Wangenheim 2003], ao invés de se adaptar um caso recuperado, pode-se optar pela inclusão de uma grande quantidade de casos na base, possibilitando uma solução garantida a todo problema apresentado no sistema. A principal dificuldade enfrentada no processo de adaptação está na definição de como realizá-la. A adaptação pode ocorrer de forma automática, porém, deve-se levar em consideração os seguintes detalhes: as diferenças entre os casos (atual e o passado) e que detalhes do caso passado podem ser utilizados no caso atual.

Várias técnicas já foram pesquisadas e desenvolvidas para o processo automatizado de adaptação. Conforme [Wangenheim 2003], as estratégias de adaptações são as seguintes:

- Adaptação nula;
- Adaptação transformacional;
  - Adaptação substitucional;
  - Adaptação estrutural;
- Adaptação gerativa/derivacional;
  - Inicialização;
  - *One-shot replay*;
  - *Replay* entrelaçado;
- Adaptação composicional;
- Adaptação hierárquica.

Em meio a essas técnicas, a questão é como utilizar a solução de um caso armazenado para o caso proposto. A solução se daria através da transferência completa ou parcial do caso similar, ou utilizando fragmentos dos casos candidatos, gerando assim uma solução com o conjunto de informações obtidas.

7. Revisão de casos

Nesta seção é abordada as questões de aprendizado do sistema. Quando um caso não correto é selecionado, há a necessidade de armazenar esta situação a fim de evitar possíveis erros no futuro. Da mesma forma, quando a solução gerada for considerada correta, o sistema deve "aprender" com o sucesso e reter o caso. A revisão de casos consiste em duas tarefas:

- Avaliar a solução e fazer a retenção, no caso de sucesso;
- Caso contrário, reparar a solução com base no conhecimento específico ou informações do usuário.

Primeiramente, a revisão se concentra em encontrar falhas nas soluções apresentadas, considerando o resultado da aplicação destas soluções através de monitoração automática ou interação com o usuário. Em seguida, as falhas encontradas são reparadas ou explicadas.

Quando uma explicação é dada para uma falha, significa que a solução em questão não gera os resultados esperados, esta explicação então, é utilizada para modificar a solução ou a forma como o sistema chegou à solução de maneira que o caso obtenha sucesso e a falha não ocorra futuramente.

Quando uma falha é reparada, é recomendado que este reparo seja feito em possíveis falhas similares, nos casos armazenados na base de casos.

"Um bom sistema RBC deve ser projetado de forma a oferecer uma interface para captação de *feedback* sobre o resultado da aplicação da solução fornecida"[Wangenheim 2003].

8. Retenção de novos casos

De acordo com [Toniazzo 2005] é importante reter o conhecimento toda vez que um novo problema for solucionado, estendendo assim a base de casos.

A retenção automatizada de um novo caso em RBC pode ser mais complexa do que apenas inserir um novo caso de sucesso, também se faz necessário utilizar técnicas de Aprendizado de Máquina. Nesta seção será abordado algumas filosofias de retenção de casos e as técnicas utilizadas em cada uma.

8.1. Tipos de retenção

[Wangenheim 2003] Apresenta três tipos de retenção conhecidos em um sistema de RBC:

- **Sem retenção de casos:** Geralmente aplicado onde se tem um domínio satisfatório do conhecimento necessário para a estruturação da aplicação. Neste caso, a retenção de novos registros não se faz necessário, pois o mesmo não contribuirá com a otimização na performance da aplicação;
- **Retenção de soluções de problemas:** Este tipo de retenção é a que caracteriza de forma específica o aprendizado no RBC. Quando solucionado um problema, o conhecimento a experiência é armazenada para ser utilizada como um auxiliar nos novos problemas;

- **Retenção de Documentos:** Aqui a retenção ocorre separada ao processo de solução de um problema. Sempre que disponibilizado um novo conhecimento no sistema, seja através de documentos, descrições, ou qualquer outro tipo, o processo de retenção é ativado.

8.2. Fases do processo de retenção

O processo de retenção pode ser refinado em três fases: extração de conhecimento, indexação de casos e integração na base de casos.

8.2.1. Extração do conhecimento

É a aquisição das informações necessárias para a estruturação de uma solução na tomada de decisão. Esse novo conhecimento adquirido será integrado a um caso que já exista na base, assim como, pode ser construído um novo caso [Toniazzo 2005]. As fontes para novas experiências podem ser:

- **Para sistemas de retenção de solução problemas:** A solução do problema, estrutura do caminho de solução, representação do conhecimento usado, histórico de adaptação de casos, protocolos de solução gerados pelo sistema, explicações e justificativas;
- **Para sistemas de retenção de documentos:** Documentos, manuais técnicos, descrições de produtos, base de dados online com resumos de conhecimento em uma área.

Com base nas informações adquiridas, um novo caso poderá ser integrado em um caso existente ou um caso similar poderá ser generalizado para inserir a nova experiência. Para qualquer uma das situações citadas é preciso levantar o que deve ser utilizado como fonte de aprendizado e como estruturar a nova experiência.

8.2.2. Indexação

É a fase que decide qual a melhor forma de indexar os casos para futuras recuperações. Uma solução para este problema seria utilizar todas as entidades como índices.

8.2.3. Integração de casos

Esta fase realiza a inclusão, exclusão ou modificação de casos na base. A integração pode realizar um ajuste nos índices existentes e nos pesos dos mesmos, proporcionando o refinamento dos casos. O objetivo é que em consultas futuras a recuperação dos casos venha a ser feita em uma base de casos atualizada.

8.3. Aprendizado baseado em casos

RBC implica em uma forma de aprendizado por analogia, em que, por meio da transformação e extensão do conhecimento existente, uma tarefa ou problema similar é executado ou removido [Andrade 2008].

Algoritmos para aprendizados baseado em casos pertencem à classe de algoritmos de aprendizado de instâncias, os chamados Algoritmos IBL[8].

8.3.1. Algoritmos IBL

Conforme [Alves 2005] sempre que um sistema altera seu conhecimento, através da adição de novos padrões, ou outro tipo de alteração, podemos falar de um processo de aprendizado.

Algoritmos IBL aprendem a categorizar um conjunto de classes de objetos de forma incremental com base em exemplos de instâncias dessas categorias.

Segundo [Wangenheim 2003] existem três componentes presentes em todas as classes de algoritmos-IBL:

- **Função de Similaridade:** Computa a similaridade entre uma instância de treinamento i e as instâncias em uma dada descrição conceitual;
- **Função de Classificação:** Recebe o resultado da "Função de Similaridade"e provê uma classificação para i;
- **Atualizador do descritor conceitual:** Decide quais instâncias incluir na descrição conceitual.

As funções de similaridade e classificação têm o objetivo de determinar como o conjunto de instâncias salvas na descrição conceitual serão utilizadas para predizer valores para o atributo de categoria.

De acordo com [Wangenheim 2003], para supervisionar o desempenho de algoritmos de aprendizado usamos as seguintes dimensões:

- **Generalidade:** Representa as classes de conceito que podem ser aprendidos e descritos pelo algoritmo em questão;
- **Acurácia:** É a acurácia da classificação provida pela descrição conceitual;
- **Taxa de Aprendizado:** È a velocidade com a qual a acurácia classificatória aumenta durante o aprendizado;
- **Custos de Incorporação:** Custos que decorrem da atualização da descrição conceitual por meio da inclusão de uma instância única;
- **Requisitos de Armazenamento:** Tamanho da descrição conceitual, que em algoritmos-IBL é definida com o número de instâncias que necessitam ser salvas, para prover um desempenho classificatório adequado.

O algoritmo IBL1 é o mais simples algoritmo de aprendizado baseado em instâncias. A função de similaridade é a seguinte:

$$sim(Q, C) = -\sqrt{\sum_{i=1}^{n} f(Q, C)} \tag{6}$$

Onde os casos são descritos através de n entidades. Definimos $f(Qi, Ci) = (Qi - Ci)^2$ para entidades de valor numérico, e $f(Qi, Ci) = (Qi.\frac{1}{4}Ci)$ para entidades booleanas e de valor simbólico ou textual [Wangenheim 2003].

[8]Do inglês, *Instance-Based Learning*.

Entidades ausentes são consideradas como sendo maximamente diferentes do valor presente. Se ambos faltam, então vale $f(Qi, Ci) = 1$. De acordo com [Wangenheim 2003], o algoritmo IBL1 é idêntico ao algoritmo *nearest neighbour* discutido na seção 4, exceto que, pelo fato de normalizar as faixas das EI, é capaz de processar casos de forma incremental, e possui uma política simples para lidar com valores desconhecidos.

As funções de similaridade e de classificação de IBL1 provê uma descrição conceitual extensiva a partir do conjunto de casos salvos, mostrado na tabela 11.

DC: Descritor Conceitual
Inicialize $DC := 0$
PARA cada $C \in$ conjunto de treinamento FAÇA
PARA cada $C \in DC$ FAÇA
$sim(C) := sim(Q, C)$
$Cmax :=$ algum $C \in D$C com MAX $sim(C)$
SE $classe(Q)$ = $classe(Cmax)$ ENTÃO
classificação := correta
SENÃO classificação := incorreta
$DC := DC \in Q$
FIM

Tabela 11. Algoritmo IBL1

Existem duas extensões deste algoritmo que são respectivamente o algoritmo IBL2 e IBL3. A idéia básica do IBL2 é que não precisamos instanciar todos os novos casos para permitir uma boa descrição dos limites de um conceito. É necessário apenas utilizar um caso similar de sucesso e seus casos vizinhos similares de insucessos para se formular uma nova descrição da solução. IBL2 reduz drasticamente as necessidades de armazenamento.

O maior problema do IBL2 é o ruído, que rapidamente mortifica seu desempenho, podendo levar a uma acurácia bastante inferior à do IBL1. Isto ocorre porque IBL2 salva todos os exemplos de treinamento com ruído que classifica erroneamente e depois os utiliza.

IBL3 é uma extensão do IBL2 tolerante a ruídos que emprega uma estratégia de coleta de evidências de "esperar para ver"para então averiguar quais das instâncias salvas vão funcionar bem durante a classificação. Sua função de similaridade é idêntica ao IBL2. No entanto, IBL3 registra a freqüência com a qual um caso armazenado quando escolhido como o mais similar ao caso atual correspondeu ao valor do atributo-meta deste. A função de classificação e o algoritmo de atualização diferem do IBL3. Entre outras coisas, IBL3 mantém um registro de classificações, registrando o número de tentativas corretas e incorretas, associado a cada caso armazenado.

9. Domínio da aplicação

A tecnologia de RBC pode ser utilizada em vários domínios, como por exemplo: análise financeira, manutenção técnica, controle de processos, controle da qualidade, diagnóstico médico, sistemas de suporte a software, previsão, planejamento, projeto, classificação, interpretação de imagens, avaliação imobiliária, comércio eletrônico, suporte ao usuário e etc.

Cada domínio possui suas características particulares que influenciam fortemente na escolha da representação do conhecimento a ser utilizado [Wangenheim 2003].

9.1. Suporte à vendas

O objetivo desse domínio é oferecer auxílio à venda de produtos, estimando custos adequados ao perfil do cliente.

De acordo com [Nisanbayev, Ko, Abdulaev, Na, Lim 2009] e [Wangenheim 2003] o domínio suporte à vendas, pode ser definido como qualquer atividade de comunicação em forma eletrônica realizada para oferecer suporte à atividades comerciais. Podendo se inserir em diferentes momentos tais como: pré-venda, venda e pós-venda.

No período de pré-venda, o cliente manifesta interesse por um produto ou serviço, e obtém toda a informação. Na venda, um cliente e um vendedor negociam produtos ou serviços. A tarefa do vendedor é descobrir as necessidades do cliente e oferecer-lhe um produto ou serviço adequado.

Neste contexto, o SRBC tem como objetivo oferecer ao cliente serviços e facilidades melhores de seleção. Por exemplo, durante a situação de pré-venda, o SRBC pode ser aplicado para capturar características do cliente e oferecer um produto similar ao perfil do mesmo.

Até o momento foi apresentada a técnologia RBC, em seguida, será feita uma análise detalhada da aplicação, o projeto e a modelagem do sistema proposto.

9.2. Projeto proposto

O projeto tem como objetivo modelar e desenvolver um sistema de vendas *on-line*, utilizando a tecnologia RBC. Com base nas características do cliente, o sistema buscará na base de conhecimento por situações (casos) que julgue similares ao perfil desse cliente.

Utilizando-se o exemplo abaixo, onde um cliente gostaria de comprar um carro em um ambiente *Web*, e fornecendo seus dados e características o sistema irá sugerir o carro mais similar ao seu perfil:

Caso atual:

- Profissão: Engenheiro elétrico
- Idade: 25
- *H*obby: Esportes
- Classe Social: Alta

Caso recuperado:

- Cliente: Carlos Magno
- Profissão: Engenheiro de Automação
- Idade: 29
- *H*obby: Esportes
- Classe Social: Alta

Descrição da Solução:

- Carro Comprado: Audi TT
- Cor: Amarelo

- Valor: R$ 120.000
- Forma de Pagamento: 10 vezes no cheque

Logo, o sistema recomendará o carro do caso mais similar recuperado da base de conhecimento e apresentaria seus benefícios. Caso o cliente goste do produto, o sistema o questionará sobre a forma de pagamento mais adequado e interagindo com o mesmo, calculará o valor do desconto possível e assim concluirá a negociação.

9.3. Modelagem do sistema

Nesta seção apresenta-se o diagrama de casos do sistema proposto e descreve-se o principal caso de uso: consultar produto RBC.

9.3.1. Diagrama de casos de uso

Casos de uso envolvidos: consultar produto, **consultar produto RBC**, comprar produto, cadastrar cliente, cadastrar produto, cadastrar benefício dos produtos, cadastrar fornecedor, cadastrar funcionário e cadastrar promoção, como apresentado na figura 6.

Atores do sistema: cliente *Web*, funcionário e gerente.

O caso de uso que será interessante para este artigo é o consultar produto RBC, que estende do caso de uso comprar produto.

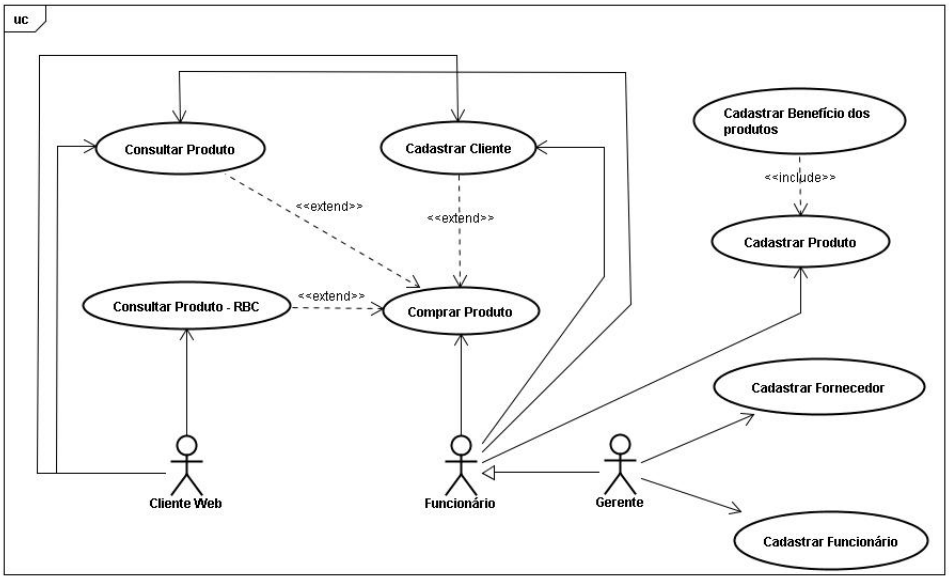

Figura 6. Diagrama de casos de uso do protótipo do sistema de vendas *on-line*.

9.3.2. Descrição do caso de uso comprar produto

Na tabela 12 apresenta-se a descrição do caso de uso comprar produto, que tem como fluxo alternativo consultar produto RBC e registrar pedido.

Projeto	Sistema e-commerce[9]
CSU01	Comprar Produto
Data	12/04/2010
Versão	1.0
Descrição	Descrição feita após a primeira validação de requisitos com o cliente.
Autor	Diego Lima Ervatti, Patrick Peyneau Andrade, Rafael Zacche de Sá Silva.
Sumário	Ator realiza a compra de um produto.
Ator Principal	Cliente *Web*
Pré-Condições	Cliente *Web* deverá estar cadastrado no sistema.
Risco: Alto	**Prioridade:** Alto
1.1. **Fluxo Principal: Comprar Produto** 1.1.1. Este caso de uso inicia quando o ator opta por comprar um novo produto **(♯A2)** 1.1.2. O ator insere seus os dados para pesquisar um produto adequado com seu perfil **(♯ A1)** 1.1.3. O ator lê as informações do produto 1.1.4. O ator solicita um pedido do produto 1.1.5. O sistema gera o código do Produto automaticamente 1.1.6. O sistema retorna uma mensagem de sucesso 1.1.7. O ator confirma a mensagem 1.1.8. O caso de uso se encerra.	
1.2. **Fluxos Alternativos** 1.2.1. **(♯A1) - Consultar Produto RBC** 1.2.1.1. O sistema faz a validação dos dados **(♯E1)** 1.2.1.2. O sistema aplica a similaridade e recupera o caso mais similar da base de casos 1.2.1.3. O sistema exibe o caso mais similar da base de casos com as informações do produto 1.2.1.4. O sistema retorna uma mensagem de sucesso na recuperação do caso 1.2.1.5. Este caso de uso se encerra.	
1.2.2. **(♯A2) - Cadastrar Cliente** 1.2.2.1. O sistema verifica se o ator já está cadastrado no sistema 1.2.2.2. Caso o ator não esteja cadastrado o sistema solicita os dados do ator **(♯E1)** 1.2.2.2. O ator informa seus dados 1.2.2.3. O sistema retorna uma mensagem de sucesso 1.2.2.4. O ator confirma a mensagem 1.2.2.5. Este caso de uso se encerra.	
1.3 **Fluxos Exceções** 1.3.1. **(♯E1) - Dados Obrigatórios** 1.3.1.1. Caso os dados obrigatórios não sejam fornecidos, é emitida uma mensagem de erro informando a falha e solicitando os dados 1.3.1.2. Retornar ao ponto de entrada de dados.	
Pós-Condições: Um compra de um produto foi realizada e um novo caso foi inserido ou alterado da base de casos com sucesso.	

Tabela 12. Descrição do caso de uso Comprar produto

9.3.3. Modelagem de casos

A modelagem de casos tem como objetivo aplicar o cálculo de similaridade global e local. Nesta subseção apresenta-se a modelagem de um sistema de vendas *o*n-line de carros.

Modelo de caso para classe carro

A modelagem de caso para a classe carro tem como atributos: ano, categoria, cor, marca, modelo, preço e potência, com suas respectivas descrições e tipos, representado na tabela 13.

	Tipo-base	Faixa
Ano	Data	2010/2011
Categoria	Símbolo	Econômico, esportivo, *pick-up*, clássico
Cor	Símbolo	Amarelo, preto, prata,...
Marca	Símbolo	Chevrolet
Modelo	Símbolo	Agile, astra, blazer, captiva, celta, classic
Preço	Real	[0,00, 200.000,00]
Potencia	Inteiro	[0 - 300]

Tabela 13. Definição de atributos e tipos para carro.

Existe uma similaridade local para o modelo de carro. Cada caso de venda possuirá os atributo para os modelos de carros, onde os valores de similaridade serão distribuídos conforme a tabela 14.

	Agile	Astra	Blazer	Captiva	Classic	Vectra GT
Agile	1.0	0.6	0.0	0.3	0.5	0.4
Astra	0.6	1.0	0.0	0.4	0.5	0.8
Blazer	0.0	0.0	1.0	0.2	0.0	0.0
Captiva	0.3	0.4	0.2	1.0	0.1	0.3
Classic	0.5	0.5	0.0	0.1	1.0	0.0
Vectra GT	0.4	0.8	0.0	0.3	0.0	1.0

Tabela 14. Similaridade local para modelo.

Modelo de caso para classe cliente

A modelagem de caso para a classe cliente tem como atributos: hooby, classe social,grau escolar, estilo de ser e estado civil, com suas respectivas descrições e tipos, representado na tabela 15.

	Tipo-base	Faixa
Hooby	Símbolo	Esportes radicais, colecionador, sair com a família
Classe social	Símbolo	Baixa, média, alta
Grau escolar	Símbolo	Fundamental, médio, superior
Estilo de ser	Símbolo	Simples, extravagante, formal, arrojado, espaçoso
Estado Civil	Símbolo	Solteiro, casado, separado, divorciado, viúvo

Tabela 15. Definição de atributos e tipos para cliente.

Existe uma similaridade local para a classe social de cada cliente. Cada caso de venda possuirá o atributo classe social do cliente, onde os valores de similaridade serão distribuídos conforme a tabela 16.

	Baixa	**Média**	**Alta**
Baixa	1.0	0.5	0.0
Média	0.5	1.0	0.5
Alta	0.0	0.5	1.0

Tabela 16. Similaridade local para Classe social.

9.4. Ferramentas para desenvolvimento de SRBC

Existem várias ferramentas (*shells*[10]) para desenvolvimento de SRBC no mercado. Dentre as quais destacam-se: CBR-Works, MyCBR[11] e JColibri[12].

A metodologia RBC pode ser aplicada em um problema de forma *ad-hoc*[13] ou a partir da especialização de infra-estruturas que englobem as diversas tarefas necessárias ao raciocínio.

O shell CBR-Works, desenvolvido pela TecInno/empolis, é um dos *shells* de RBC mais poderosos disponíveis, podendo ser utilizado para desenvolvimento de soluções inteligentes aplicados a diversos domínios e ambientes operacionais. Possui uma interface intuitiva e com poucos cliques é possível criar uma aplicação simples utilizando RBC. Sua grande desvantagem é ser um *shell* proprietário, possuindo um alto custo, o que para o desenvolvimento do projeto proposto no presente artigo se torna inviável.

MyCBR é um *shell* de código fonte aberto para o desenvolvimento de soluções que utilizam do RBC. Seu objetivo é ser fácil de usar e fornecer GUIs[14] poderosas para modelagem do sistema. Sua confortável interface gráfica permite que até mesmo os novatos em RBC criem rapidamente suas primeiras aplicações. No entanto, garante a flexibilidade suficiente para permitir que usuários experientes possam implementar aplicações avançadas de RBC.

Em particular, o *shell* jColibri desenvolvido pelo Grupo de Aplicações de Inteligência Artificial da Universidade Complutense de Madri (Espanha) e licenciado através da Licença Pública Geral. Esta apresenta *shells* para projetar RBC e APIs[15] para integrar a metodologia às aplicações.

O jColibri dispõe de funcionalidades para uso de ontologias na representação de casos e conta com uma biblioteca de métricas de similaridade para ontologias [Martins 2009].

O que torna o *shell* jColibri particularmente adequado à implementação do sistema proposto neste artigo, é que dentre os *shells* pesquisados, ele possui maior quanti-

[10] *Shells*, de maneira genérica, é um programa que faz a intermediação do contato entre o usuário e o computador.

[11] http://mycbr-project.net/

[12] http://gaia.fdi.ucm.es/projects/jcolibri/

[13] Termo usado em computação para resolver determinado problema ou realizar uma tarefa específica.

[14] Do inglês, *Graphical User Interface.*

[15] Do inglês, *Application Programming Interfaces.*

dade de material disponível para estudos e possui a integração de suas bibliotecas com a linguagem Java.

10. Conclusão

Neste trabalho foram discutidas as principais características que envolvem o estudo da tecnologia de RBC.

Sistemas de gerência de bancos de dados (SGBD) são largamente utilizados, para armazenamento e recuperação de informações em sistemas *e-commerce*. Estes sistemas são projetados para realizar consultas exatas na base de dados, em contrapartida os SRBC são projetados para realizar consultas inexatas.

Quando correspondências exatas não existem para uma nova situação um SGBD não consegue oferecer uma solução, porém um SRBC é capaz de encontrar um caso adequado.

Os objetivos propostos no artigo foram atingidos. Apresentamos o estado da arte na área de RBC e iniciamos a modelagem de uma aplicação RBC (sistema de vendas de carros *on-line*).

Como trabalhos futuros pretende-se desenvolver e testar o sistema de vendas de carro *on-line*. Utilizaremos a linguagem Java com o auxílio do *shell* jColibri. Assim, demonstraremos de forma prática o funcionamento de um SRBC e serão aplicados os conhecimentos adquiridos no decorrer do curso de Ciências da Computação (UVV).

Referências

[1] Alves, F. C. (19 de Abril de 2005). Aplicação de Técnicas de Mineração de Dados a uma Base de um Sistema Gerenciador de Informações para UTI. Dissertação de Mestrado , p. 112.

[2] Andrade, F. V. (19 de Novembro de 2008). Predição de Tempo de Execução de Tarefas em Grades Computacionais para Recursos não Dedicados , p. 107.

[3] Goetze, A. R. (11 de Abril de 2005). Unisinos. Acesso em 3 de Maio de 2010, disponível em Universidade do Vale do Rio dos Sinos: http://www.inf.unisinos.br/ ari/estrut/kdtree/kd-tree.html

[4] Heredia, L. R., Iochpe, C., e Comba, J. (2003). Explorando a multidimensionalidade da KD-Tree para suporte a temporalidade em dados espaciais vetoriais do tipo ponto. Instituto de Informática da UFRGS , p. 8.

[5] Luger, G. F. (2004). Inteligência artificial (4ª ed.). (P. Engel, Ed.) Porto Alegre: Bookmann.

[6] Martins, D. (Julho de 2009). Uma abordagem para recuperação de informações sensível ao contexto usando retroalimentação implícita de relevância. p. 107.

[7] Nisanbayev, Y., Ko, I. S., Abdulaev, S., Na, H., e Lim, D. (2009). E-commerce Applications of the Hybrid Reasoning Method. International Conference on New Trends in Information and Service Science ,p. 5.

[8] Real, R. (21 de Março de 2003). Redes semânticas . Acesso em 3 de Maio de 2010, disponível em http://www.inf.ufrgs.br/gppd/disc/cmp135/trabs/rodrigo/T1/html/index.html

[9] Rezende, S. O. (2005). Sistemas Inteligentes (1ª ed.). Barueri: Manole.

[10] Russell, S. J. (2004). Inteligência artificial (2 ed.). Rio de Janeiro: Norving.

[11] Toniazzo, L. H. (Junho de 2005). Módulo de Recuperação de Conhecimento para Auxílio na Tomada de Decisão em um Sistema de Ouvidoria. p. 66.

[12] Wangenheim. (2003). Raciocínio Baseado em Casos. Barueri: Manole.

[13] Watson, I. (1999). Case-based reasoning is a methodology not a technology. Elsevier , p. 6.