Uma base de conhecimentos sobre vinhos em Prolog

Matheus Teixeira Fernandes
João Paulo Pizani Flor
José João Junior

7 de Outubro de 2010

1 Introdução

Uma das questões mais importantes que concerne ao estudo da Inteligência Artificial é como se representar o conhecimento. Qualquer programa que almeje agir ou raciocinar de maneira semelhante à humana deve possuir um formato adequado para representar o conhecimento que possui. Esse formato adequado, combinado com bons algoritmos de busca, é que determinam a eficiência do agente inteligente.

Uma forma de representação de conhecimento muito utilizada em IA é a Lógica. A Lógica já é há muito tempo utilizada (de fato, desde a Grécia Antiga) para modelar o processo de raciocínio dos seres humanos, e seus fundamentos e suas leis já foram profundamente estudados e evoluídos. Várias tipos de lógica são utilizadas em IA, dentre as quais podem-se destacar:

Lógica clássica É a lógica mais antiga e difundida. Criada por filósofos gregos (Aristóteles, Platão, entre outros), ela representa o que normalmente se quer dizer quando se fala em "lógica".

Lógica não-monotônica É um tipo de lógica onde a adição de novas sentenças a um ambiente pode invalidar as verdades que existiam até então. É uma boa lógica para representar, por exemplo, o conhecimento científico.

Lógica difusa Conhecida em inglês como "fuzzy logic"[1]. Na lógica difusa, o valor verdade de uma proposição está localizado em um intervalo real entre 0 (falso) e 1 (verdadeiro). Sendo assim, ela se adequa a representar situações de incerteza e/ou conhecimento parcial.

Este trabalho constitui-se, especificamente, na descrição de uma base de conhecimentos sobre a harmonização entre vinhos e refeições, utilizando a linguagem Prolog. A linguagem Prolog foi inventada na França em 1970, por um grupo de pesquisadores liderados por Alain Colmerauer. A linguagem de Prolog é uma linguagem *declarativa*, projetada para representar fatos e regras da lógica clássica de primeira ordem.

Utilizando o sistema inteligente desenvolvido neste trabalho, o usuário irá entrar com detalhes sobre sua refeição como, por exemplo, o tipo de prato (carne, peixe, ave), o tipo de molho e quais são seus gostos pessoais de doçura e cor do vinho. O sistema irá então fornecer como resposta uma lista de vinhos, cada um com um *nível de adequação* à refeição, sendo que tal nível é um valor entre 0 e 1.

2 A linguagem Prolog

A linguagem Prolog é uma linguagem de programação em lógica, onde a tarefa de resolver um problema está dividida entre o programador e o motor de inferência da lingaugem.

O programador Prolog deve expressar os fatos e regras do domínio de aplicação utilizando-se de lógica. Mais especificamente, ele deve expressar o conhecimento na forma de *Cláusulas de Horn*[**?**]. Cada cláusula de Horn é uma disjunção de vários literais, em que existe no máximo um literal positivo (não-negado). Um exemplo de cláusula de Horn é a seguinte:

$$\neg p \neg q \vee \cdots \vee \neg t \vee u$$

Nesse caso, o literal positivo da cláusula é u. Uma cláusula de Horn também pode sempre ser interpretada como uma implicação. O exemplo acima dá origem à seguinte implicação:

$$(p \wedge q \wedge \cdots \wedge t) \rightarrow u$$

A interpretação da cláusula acima nos diz que para provar o consequente da implicação, devemos provar todos os antecedentes (E lógico). Na sintaxe própria da linguagem Prolog, uma implicação igual à acima é representada da seguinte forma:

```
u :- p, q, ..., t
```

Ou seja, u é consequência de p, q, t, etc. Para se mostrar que u é verdade, primeiro deve-se demonstrar a validade de cada um de seus antecedentes. Os predicados Prolog podem ser aplicados sobre átomos ou sobre variáveis. Todas as variáveis em Prolog são quantificadas universalmente, e o escopo de uma variável se restringe à cláusula onde ela se encontra. Por exemplo, a regra:

```
progenitor(X,Y) :- mae(X,Y); pai(X,Y).
```

quer dizer "X é progenitor de Y se X é mãe de Y ou X é pai de Y, PARA TODO X e PARA TODO Y". Um outro fato bastante notório da linguagem Prolog é que ela trabalha com um modelo de mundo fechado. Isso quer dizer que caso o interpretador Prolog não consiga provar a veracidade de uma sentença, ele irá considerá-la falsa. Para o Prolog, tudo que não decorre dos conhecimentos na base é falso.

O motor de busca da linguagem Prolog usa como regra de inferência para processar as consultas a *resolução*. A técnica de prova por resolução é um tipo específico de redução ao absurdo, ou seja, assume-se que o objetivo a ser provado é falso e então, caso isso leve a uma contradição, o objetivo é verdadeiro.

3 Estratégias adotadas

Na implementação do trabalho, os predicados melhor_cor, melhor_docura, docura_escolhida, cor_escolhida e vinho foram implementadas como *predicados estáticos*. Cada um dos predicados tinha uma ou mais regras, e cada regra tinha um fator de certeza associado, que era calculado a partir das certezas dos antecedentes, assim como levando em conta um fator de certeza inerente à própria regra:

$$FC(cabeca) = min(FC(ant1), FC(ant2), \cdots, FC(antN)) * FC(regra)$$

Já os predicados

- vitela
- peru
- molho
- prato
- tipo_molho
- docura
- cor

foram implementados como *predicados dinâmicos*. Os fatos referentes a esses predicados compoem as informações sobre a refeição do usuário, e são inseridos na base de conhecimento à medida que as perguntas são feitas ao usuário.

O uso de impurezas da linguagem Prolog foi necessário, sobretudo, para realizar as tarefas de entrada/saída, isto é, toda a parte de interação com o usuário. Os predicados *write*, *assert*, *retract*, *nl*, entre outros, foram extensivamente utilizados no trabalho e são todos IMPUROS. Eles produzem efeitos colaterais quando o motor Prolog passa por eles durante uma busca.

Além disso, a impureza *cut* foi também utilizada na implementação do predicado min. Isso para que o interpretador, após conseguir provar que um argumento é menor que o outro, não continue tentando mostrar que ele é maior ou igual. A impureza fail também foi usada no arquivo de testes (tests.pl) para implementar o procedimento *retract_all*. O fail foi utilizado para forçar o backtracking por parte do interpretador Prolog, fazendo assim com que não apenas a primeira sentença relativa a um predicado fosse removida, mas todas.

4 Completude da base

Pode-se dizer que a base de conhecimento fornecida é COMPLETA. Ela oferece uma resposta de vinho recomendado para cada possível combinação de entradas fornecidas pelo usuário.

Isso pois, na base, toda combinação de antecedentes está contemplada com ao menos uma regra que a contenha. Algumas possuem um fator de certeza mais baixo na recomendação, o que irá gerar recomendações com menor certeza, porém sempre haverá ao menos uma recomendação.

5 Instruções para execução

Para executar o trabalho é necessário ter instalado na máquina do usuário o interpretador SWI-Prolog. Ele pode ser obtido facilmente no site dos desenvolvedores, através do seguinte endereço: `http://www.swi-prolog.org/`. Para instalar o SWI-Prolog no Ubuntu não é necessário visitar o site dos desenvolvedores, pois o interpretador é um pacote nos repositórios padrão. Para baixar e instalar no Ubuntu Linux, basta usar o comando abaixo:

```
sudo aptitude install swi-prolog
```

O trabalho encontra-se dividido nos seguintes arquivos:

sommelier É o arquivo principal, um script shell, o qual ao ser executado irá carregar todos os demais módulos do trabalho e esperar pelo usuário para iniciar a consulta.

sommelier_rules.pl O arquivo contendo as regras da base de conhecimento.

tests.pl Contém alguns casos de teste, para uma verificação rápida da base.

Para executar uma consulta, deve-se executar o arquivo sommelier em um terminal, da seguinte maneira:

```
./sommelier
```

O interpretador Prolog irá carregar os demais arquivos necessários e aguardar pela entrada do usuário. Para iniciar uma consulta no sistema de recomendação de vinhos deve-se digitar "main" no prompt. O sistema irá então fazer todas as perguntas necessárias sobre a refeição do usuário, e ao final das perguntas irá exibir na tela a(s) sua(s) recomendação(ões).

Referências

[1] Fuzzy logic. http://en.wikipedia.org/w/index.php?title=Fuzzy_logic&oldid=389287196. Link permanente p/ a revisão referenciada.