Universidade de São Paulo
Instituto de Matemática e Estatística
MAC 5739 - Inteligência Artificial

Exercício Programa 2: Prolog

Autor:

Walter Perez Urcia

São Paulo

Outubro 2015

Resumo

Akinator é um gênio virtual que é capaz de adivinhar a personagem em que o usuário está pensando, seja ela real ou fictícia, através de perguntas sobre suas características (`http://www.akinator.com`). Neste exercício usaremos Prolog para modelar uma base de conhecimento, e implementar uma mini-versão do jogo Akinator. Além disso, serão comparados varias ordens das características e das regras das personagens em termos do número medio de perguntas.

1 Descrição dos personagens

Para este exercício foram considerados às personagens do anime Pokemon, especificamente aquelas da primeira temporada: 151 em total [1]. O conjunto de dados tem 7 características (ou atributos) que serão descritos a continuação:

- Type: O tipo dos poderes que o Pokemon usa
- Color: O cor principal da personagem
- Size: Escala do tamanho padrão da personagem
- Weight: Uma escala do peso da personagem
- Has_evolution: Diz se o pokemon tem evolução a outro pokemon
- Is_evolution: Diz se o pokemon evoluiu de outro
- Is_starter: Diz se o pokemon é um dos principais nos jogos do anime

A tabela 1 mostra cada um das características das personagens e seus possíveis valores.

Tabela 1: Características e possíveis valores

Type	poison (15.07%)	water (14.61%)	normal (10.05%)	flying (9.13%)
	grass (6.39%)	ground (6.39%)	psychic (6.39%)	fire (5.48%)
	bug (5.48%)	rock (5.02%)	electric (4.11%)	fighting (3.65%)
	ice (2.28%)	fairy (2.28%)	ghost (1.37%)	dragon (1.37&)
	steel (0.91%)			
Color	brown (23.18%)	blue (16.56%)	purple (15.23%)	yellow (12.58%)
	red (10.60%)	pink (7.95%)	green (5.96%)	gray (5.30%)
	white (1.99%)	black (0.66%)		
Size	small (62.25%)	medium (33.11%)	big (4.64%)	
Weight	light (66.23%)	medium (31.79%)	heavy (1.99%)	
Has_evolution	true (46.36%)	false (53.64%)		
Is_evolution	true (47.68%)	false (52.32%)		
Is_starter	true (1.99%)	false (98.01%)		

[1] Todos os dados foram obtidos do site oficial: http://www.pokemon.com/es/pokedex/

2 Regras

Além das características que descrevem cada personagem, a base de conhecimento precisa de regras para poder inferir qual é a personagem que o usuário está pensando. Nesta seção serão explicadas as regras colocadas na base de conhecimento.

2.1 Regras de descrição

Para que o programa consiga inferir o personagem, tem que existir fatos diferentes na base de conhecimento para cada um. Então para cada característica o jeito de representar um fato é da seguinte forma:

- V_type
- V_color
- V_size
- V_weight

Onde V é um dos possíveis valores para cada atributo. Além disso, para as características binarias (true,false) só vão existir se fossem verdadeiros para algum personagem.
Por exemplo, a primeira personagem se representa

Exemplo 1: Bulbasaur

```
1  bulbasaur :- verify( grass_type ) ,
2               verify( poison_type ) ,
3               verify( green_color ) ,
4               verify( small_size ) ,
5               verify( light_weight ) ,
6               verify( has_evolution ) ,
7               verify( is_starter ) , ! .
```

Por último, a função $verify$ é usada para perguntar ao usuário se aquele fato é verdade para a personagem que ele está pensando.

2.2 Regras de exclusão

Se só consideramos as regras de descrição anteriores, o programa teria que fazer muitas perguntas para encontrar qual é a personagem do usuário, mas muitos valores das características de uma personagem são exclusivos. Por exemplo, uma personagem não pode ter tamanho pequeno e tamanho grande ao mesmo tempo, do mesmo jeito, não pode ter como cor principal verde e azul ao mesmo tempo (cada personagem tem só um cor principal).
Então foram adicionadas regras de exclusão em Prolog que tem a seguinte forma:

$$no(V_k) : -yes(V_2); yes(V_3); \ldots; yes(V_{k-1}); yes(V_{k+1}); \ldots; yes(V_n).$$

onde V_i é um dos possíveis valores de uma característica. O que aquela regra diz é que para algum valor k de uma característica, se algum dos fatos para os outros valores é verdadeiro, então não pode ser verdadeiro o valor k também.

Por exemplo, para a característica Size temos [2]:

```
1       no( small_size ) :- yes( big_size ) , !.
2       no( small_size ) :- yes( medium_size ) , !.
3       no( big_size ) :- yes( small_size ) , !.
4       no( big_size ) :- yes( medium_size ) , !.
5       no( medium_size ) :- yes( small_size ) , !.
6       no( medium_size ) :- yes( big_size ) , !.
```

Um caso especial de estas regras é feito para a característica Type porque uma personagem pode ter mais de uma e então não todas as possíveis combinações de pares geram regras de exclusão. No exemplo 1, a personagem é do tipo grass e poison e então na base de conhecimento não temos uma regra de exclusão para aquele par de valores.

3 Experimentos e resultados

Nesta seção serão detalhados os experimento feitos usando diferentes ordens das características e das regras de descrição. Além disso, os resultados serão mostrados para fazer algumas comparações em termos de desempenho.

3.1 Considerações geráis

Uma pequena descrição de cada experimento e o nome de cada um estão a continuação:

- **Default**: A ordem das regras de descrição é pela ordem que estão no site oficial (personagem 1 até 151) e a orden dos fatos das características é Type, Color, Size, Weight, Has_evolution, Is_evolution, Is_starter
- **Is_evolution**: As regras de descrição foram ordenadas pela característica Is_evolution, aqueles que tem aquela característica como falsa estarão no início. A ordem das características é igual ao anterior mas a característica Is_evolution mudou sua posição para o início.
- **Size**: Neste experimento, as regras de descrição foram ordenadas pela característica Size, estando primeiro as personagens de tamanho pequeno (small), depois mediano (medium) e por último (big). Por outro lado na ordem dos fatos, só mudou a posição de Size ao início.

[2] O operador lógico OR pode ser escrito como ";" em Prolog ou pode ser colocado implicitamente em varias linhas como o exemplo.

- **Best**: Baseado nas estatísticas da tabela 1, os fatos foram ordenados da seguinte forma: Has_evolution, Is_starter, Is_evolution, Weight, Size, Color, Type. E as regras de descrição foram ordenadas usando as distribuições de percentagens de cada característica, colocando no início aqueles que tem valores mais comuns e no final no caso contrário.

Os experimentos Is_evolution e Size foram feitos para encontrar se existe uma relação entre a distribuição dos valores de *UMA* característica e o número médio de perguntas que tem que ser feitas ao usuário. Por outro lado, o experimento Best busca a relação entre todas as distribuições de *TODAS* as características e o número médio de perguntas.

3.2 Resultados

A tabela 2 mostra os resultados depois de cada experimento para as 151 personagens do domínio onde #Conflictos é a quantidade de personagens que não conseguiu acertar o programa.

Tabela 2: Resultados dos experimentos

Experimento	#Conflictos	% personagens acertadas	Num. Médio de perguntas
Default	35	76.82	12.88 ± 3.28
Is_evolution	40	73.51	12.50 ± 3.42
Size	35	76.82	12.14 ± 2.88
Best	49	67.55	10.13 ± 2.97

Em cada um dos experimentos foram encontradas personagens que fazem conflictos e que não puderam ser ditas pelo programa porque tem as mesmas características que outras ou tem um subconjunto de fatos de outra. Verificando para cada experimento quais são as personagens que estão em conflicto e fazendo uma interseção foi obtida uma lista de personagens que sempre estão em conflicto de tamanho 33. Usando esta lista, aquelas personagens foram excluídas da base de conhecimento, ficando um domínio com 118 personagens e foram feitos experimentos novos com a mesma configuração explicada na seção 3.1. Os resultados para estos experimentos são mostrados na tabela 3.

Tabela 3: Resultados dos experimentos com 118 personagens

Experimento	#Conflictos	% personagens acertadas	Num. Médio de perguntas
Default	2	98.31	13.08 ± 3.41
Is_evolution	7	94.07	12.66 ± 3.48
Size	2	98.31	12.25 ± 2.94
Best	16	86.44	10.57 ± 3.00

Nesta vez, o número de personagens em conflicto é menor em todos os casos, mas para o que deveria ser a melhor ordem das regras ainda tem um número de conflictos maior que os outros experimentos apesar de ter um número médio de perguntas que os outros.

4 Conclusões

Com os resultados dos experimentos pode-se concluir que usando as 151 personagens e usando a melhor ordem para regras e fatos usando as distribuições (Experimento Best) pode ser obtida um número médio de perguntas menor que outros, mas é aquela que tem maior número de personagens em conflicto porque tem muitas personagens que tem subconjuntos de fatos de outros que estão depois delas. Por outro lado, se tiramos as personagens que estão em conflicto podemos obter melhores resultados, mas ainda assim o experimento Best tem muito mais conflictos que os outros. Os resultados mostram que não é sempre necessário ter a melhor ordem possível para as regras porque pode incluir muitos conflictos na base de conhecimento. Além disso, usando só uma característica se obtiveram resultados parecidos à ordem por default que está no site oficial do conjunto de dados como no casso do experimento Size.
Por último, as possíveis melhorias que poderiam ser feitas no programa seria adicionar mais características às personagens para não ter conflictos entre elas. Outra melhoria seria testar com todas as possíveis ordens das características e das regras de descrição na base de conhecimento para obter aquela que tenha o menor número médio de perguntas ao usuário.