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.

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

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

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%)		

- 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

```
1       bulbasaur :- verify( grass_type ) ,
2                    verify( posion_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 por aquele fato.

2.2 Regras de exclusão

3 Dificuldades

4 Experimentos e resultados

Nesta seção será detalhado o experimento feito usando os diferentes métodos de busca desenvolvidos. Além disso, os resultados serão mostrados para fazer algumas comparações.

4.1 Considerações geráis

4.2 Resultados

5 Conclusões

- Conc. 1

Por último, uma possível melhoria ao sistema poderia ser considerar a seguinte peça que vai ser jogada. Por outro lado, poderia encontra-se uma melhor heurística para a busca BestFS e A^* dado que aqueles são os métodos mais rápidos e que consomem menos memória. Da mesma forma, para a heurística da primeira etapa (aquela que encontra a melhor posição para a peça) poderiam ser feitos experimentos para obter a melhor combinação de valores dos fatores de importância nos termos da função.