说明文档——利用prolog实现电影助手

一、选题说明

选题背景

我原来想做旅游助手，感觉比较实用，而且因为现在POI推荐挺火，找数据集应该不是很难。可惜找到的数据集都是英文地点，测试友好度就不行（比如我想做一个猜你所想的功能，要怎么测啊）。转做电影助手，找到的整理好的数据集也是英文的，像是movielen的1M和10M数据集。但是，后来我找到豆瓣电影排行榜250。如下：

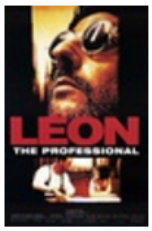
1 

肖申克的救赎 / The Shawshank Redemption / 月黑高飞(港) / 刺激1995(台) [可播放]

导演: 弗兰克·德拉邦特 Frank Darabont 主演: 蒂姆·罗宾斯 Tim Robbins /...
1994 / 美国 / 犯罪 剧情

★★★★★ 9.6 747718人评价

❝ 希望让人自由。 ❞

2 

这个杀手不太冷 / Léon / 杀手莱昂 / 终极追杀令(台) [可播放]

导演: 吕克·贝松 Luc Besson 主演: 让·雷诺 Jean Reno / 娜塔丽·波特曼 ...
1994 / 法国 / 剧情 动作 犯罪

★★★★★ 9.4 718753人评价

❝ 怪蜀黍和小萝莉不得不说的故事。 ❞

数据来源

根据以上叙述，我的数据都来自豆瓣了。原来设想用API来请求的，但是现在不能用了。改用爬虫获取。现在我拥有250个条目（为经典电影），还了解到豆瓣电影也提供开放API，后期可以扩展近期热门电影，即将上映电影。这样的话就可以有一个较好的知识库。

1. 首先在排行榜上爬取电影的基本信息。
2. 然后用获取到电影名，再通过豆瓣搜索功能，定为到这部电影，再获取它的简介。
3. 数据模板：

```
电影（名称，导演，主演，年代，国家，类型[list]，评分，热度，短评）
电影（名称，简介）
```

功能设想

1. 猜你所想：这个就和猜动物的功能是一样的，用户心中拟定一部电影，程序通过一系列问题，猜出这部电影。
2. 荐你所想：用户有自己的喜好，但是又不能明确表述。程序通过问用户一些列问题来引导用户并分析用户所

想，然后做出推荐。这对于选择综合症的用户来说是个不错的选择。
3. 热门推荐：简单但是效果良好的推荐方法就是分析一下所有电影，然后给出热门好评电影列表。
4. 电影查询：用户可能已经知道了一些电影。想要更进一步的了解电影信息。此时程序就可以从知识库中找到这些电影的信息然后展示给用户。
5. 电影搜索：大多数情况下用户想要查一些电影，但是可能不是清除地知道电影名称。这时候用户可根据已有地信息来搜索。比如这是张国荣的电影或者这是一部美国电影等等。

二、功能实现

猜你所想

这个是本软件的核心功能，也是最为复杂的功能。总体逻辑是通过问一些问一些问题，筛选出符合要求的电影列表，然后再问一些问题，直到电影列表收敛到只有一部或者没有电影输出结果。

简要逻辑如下：

1. 确定电影制作国家和地区。这个采取的策略如下：

```
% ask for area
hypoth_area('中国大陆'):-
    asia,
    verify('它是内地电影吗？').% ask for whether it is true or not.
%omit some rules here
hypoth_area('美国') :-
    verify('它是美国电影吗？').
hypoth_area('德国') :-
    europe,
    verify('它是德国电影吗？').
%omit some rules here
hypora_area('others')

asia :-
    verify('它是亚洲电影吗？').
europe :-
    verify('它是欧洲电影吗？').

% ask questions
ask(Question) :-
    write(Question),
    write("y or n:"),
    read(Response),
    ( (Response == y ; Response == yes)
      ->
       assert(yes(Question)) ;
       assert(no(Question)), fail).

:- dynamic yes/1,no/1.

% verify the question
verify(S) :-
   (
    yes(S) -> true ;
    (no(S) -> fail; ask(S))%no -> stop and back trace
    ).
```

这种设计思路来源于cmu网站上关于猜动物实现的教程（7个动物）。其基本上就是一个个假设匹配，使用动态插入语句，问过的问题被记为事实。理论上可以将所有电影按这种思路实现，但是由于数据量大，人工设计这种格式工程量大，而且不易扩展，若是增加电影，原分类可能不能解决，又要重新分类。而且规则类型单一，规则数计算也有危险。所以分而治之，部分相对不会变的属性，采用这种策略，例如国家地区、时间、名字长度。

2. 确定时间范围，方法同上。这层筛选后，会检查是否收敛，不收敛再继续。
3. 确定名字长度范围，同上。（即使加入新的电影，以上这些属性区间也不会改变，所以加电影时，不需修改规则。）
4. 这时候收敛的差不多了，可以用电影类型来询问。一开始电影类型有几十种，不适合问，但现在电影类型就很少了。统计剩下的电影类型（即使后期加电影，引入新类型，也会被统计到）。然后随机选一个来问一下（暂时没有择优选取），筛选出剩下的电影列表，检查是否收敛。然后再重新统计，去掉问过的属性，再筛选，检查是否收敛。重复直至类型都问过，还有多部电影，跳到下一步。具体如下：

```
%use the type of the movie to distiguish
type_simply(All,Types,Checked,Lefts):-
    nth0(0,Types,Type),
    (
        (
            verify('它是'-Type-'片吗？'),undo,
            select_type(Type,All,[],Left)
        );
        (
            delete_type(Type,All,[],Left)
        )
        ),
    (
        (
            check_lens(Left),
            Lefts = Left%exit the recursion
        );
        (
            allType(Left,[],Newtypes),
            append(Checked,[Type],New),
            delete_checked(Newtypes,New,Lefttypes),

            length(Lefttypes,Len),
            ((
                Len > 0,
                type_simply(Left,Lefttypes,New,Lefts)
            );
            (
                Lefts = Left% exit the recursion
            ))
        )
    ).
```

5. 然后根据导演和主演来筛选方法同上。
6. 尚存问题：
    1. 如果两部在数据集内的电影除名称外都相同，就区分不出，此时采取打出剩余列表的方法。不采用选一个打出来的方法。
    2. 如果用户恰好想的是数据集不存在的电影，则可能收敛到一部相近电影，或者收敛到没有这部电影。

荐你所想

这个功能针对于用户想看电影，但又不是电影专家，对电影不是很了解。此时程序当通过一些问题，来分析用户可能喜欢的电影。注意用户是不会选电影，也不能问过于专业的术语，比如导演风格，电影类型啊。我们要将问题转化，这也是难点所在。比如询问观影者是男生还是女生，获得此信息后就可以推荐性别偏向的电影。

然后考察实现逻辑。猜动物是个筛选过程，但是推荐则不然，这是一个打分过程。因为用户可能对科技不是很感兴趣，但是有一部电影又满足了用户的其它需求，任然是要推给用户的。所以采用打分逻辑可以解决这一问题。

简要逻辑如下：

1. 询问性别，性别是一个好问题，它一般很影响用户口味。如果是男生，那么动作类，冒险类电影就要加一点分。如果是女生爱情片，韩剧就要加一点分。

2. 询问场景,如下：

```
%场景处理，获得场景
get_scene(Scene):-
    write('请选择如下场景之一：'),nl,
    write('1. 和家人一起看.'),nl,
    write('2. 和朋友一起看.'),nl,
    (
        (
            boy,
            write('3. 和女朋友一起看.'),nl
        );
        write('3. 和男朋友一起看.'),nl
    ),
    write('4. 自己一个人看.'),nl,
    read(Res),
    (
        (
            int(Res),
            Scene = Res
        );
        get_scene(Scene)
    ).
```

用户到底是和哪些人看电影也是很影响的，比如一家人看电影，那家庭剧，动画就要加分；情侣看电影，爱情剧就要加分；一个人看的话，就要把那些虐狗剧减去一些分；好友一起看就没有太大特色。

3. 询问用户口味。用户口味设置如下：

```
% 你现在心情好吗
% 喜欢听音乐吗
% 你喜欢刺激吗
% 胆子大吗
% 是技术控吗
% 是怀旧派吗？
% 喜欢推理吗？
% 是美剧党吗
```

用户心情若是不好，喜剧片就要加一些分。其它也是要按照同样的方式处理。加分或减分。详见代码。

4. 所有问题都会被问完，问完之后每部电影都有自己的分数。然后还需要考虑电影基本质量，然后排序，取出前几位输出作为推荐。如果数据量很大，问题很多，在过程中也可以丢掉靠后电影，以加快性能。

热门推荐

以上功能，虽然很个性化，也更好的捕捉到用户口味。但是操作复杂，过程繁琐。有一个简单有有效的方法就是热门高分推荐，也是各大平台真正使用的方法。

实现热门推荐，就要考虑电影哪些属性可以反映这是一部热门好电影。考虑一下自然是评分和参与人数。这些数据可以从豆瓣上获取。

简要逻辑如下：

1. 为每一部电影这和上述属性做出数字评分。

2. 排序，然后输出指定数量的热门好电影。

电影查询

这是最常见的功能了，用户知道了某部电影，想要了解其详细信息，就可以使用此功能获取电影详细信息。

简要逻辑如下：

1. 用prolog来实现匹配，匹配到输出电影的详细信息。这规则没有难度，主要是电影详细信息收集的问题。

电影搜索

大多数情况下用户想要查一些电影，但是可能不是清除地知道电影名称。这时候用户可根据已有的信息来搜索。比如这是张国荣的电影或者这是一部美国电影等等。

简要逻辑如下：

1. 对于电影的所有属性，都用关键字去匹配，这里利用prolog回溯打出所有符合条件的电影
2. 由于搜索内容较多，只输出简略信息。详细信息可用电影查询来获取。

三、使用说明

所需环境

开发是使用swi-prolog，需要中文支持，中文支持的编码为：UTF8+BOM。如果遇到问题可以查看文件的编码是否被修改。

启动程序

使用swi-prolog加载**load.pl**文件即可。

程序助手

启动程序后，输入**hp**即可查看帮助信息。对应内容如下：

1. hp: 打开帮助信息。
2. guess: 开始猜你所想的小游戏，你在心中拟定一部电影，然后程序通过一系列问题来猜出这部电影。
3. guide: 打开荐你所想的功能，不要纠结不会选电影，该看什么电影了，打开它就会引导你，并给出推荐列表。
4. hot: 显示当前热门电影，默认为3部。
5. hot(Num): 显示当前热门电影，Num 指定要显示的电影数目。
6. show(Name): 查询电影名称为Name的电影信息，Name必须是电影名称，展示详细信息。
7. search(Key): 显示与Key匹配的电影列表，Key可为导演，主演，电影名，类型，评分等信息，返回匹配列表。

四、其它开发问题及经验

1. prolog原子不能以数字开头，一般语言都有的，不要忘了。
2. prolog的异常捕捉。
3. swi-prolog windows平台下自带的gui debug工具的使用，好工具，配合eclipse PDT插件使用也很搭。
4. 一般问题都可以在**stackoverflow**上找到解答。
5. 中文支持编码格式：UTF8 + BOM. 这样之后注释还是有一些小问题，但可以跑了。
6. prolog的逻辑和java，C++不同，不推荐使用分号，前期很多代码都大量使用或并混合逻辑（if else逻辑），难读，代码复杂。这虽扣紧了消解原理，但没有很好利用回溯功能。建议将或的逻辑尽量拆成两条规则，prolog有回溯机制，这样使得代码简洁易读，易维护。比如prolog递归，应将终止条件写成一条独立的rule。

7. 注意使用prolog module。
8. 猜电影时，问题要有动态性，不能问所有的问题，要控制收敛，对于不同电影问题不能都一样。
9. 推荐电影时，不能问过于专业的问题。