ProbLog
Programação Lógica Probabilística: Um Estudo e Aplicação

Lucas Silva Arenstein

Instituto de Matemática e Estatística

Universidade de São Paulo

Orientador: Prof. Dr. Denis Deratani Mauá

Trabalho de Conclusão de Curso Para a Obtenção do Título de Bacharel em Matemática Aplicada e Computacional

São Paulo, 2018

ProbLog
Programação Lógica Probabilística:
Um Estudo e Aplicação

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Lucas Silva Arenstein e aprovada pela comissão julgadora.

Banca Examinadora:

Prof. Dr. Denis Deratani Mauá (Orientador) - IME/USP
Prof. Dr. Luís Gustavo Esteves - IME/USP
Prof. Dr. Glauber De Bona - POLI/USP

São Paulo, dezembro de 2018

Agradecimentos

Ao Professor Denis Mauá, pelas incontáveis contribuições neste trabalho. Agradeço pela paciência e por todas as conversas e valiosas orientações. Agradeço também pela sua constante preocupação na minha formação acadêmica e o estímulo a minha criatividade.

Aos meus queridos pais Dirce e João por terem me apoiado, mesmo com as minhas bagunças que provavelmente já saíram do controle, e também pelo afeto e conselhos.

A minha querida Verônica por todos os momentos felizes que já passamos juntos, as nossas conversas, por todo o seu carinho, e também por me aguentar mesmo quando eu estou chato e com fome.

A todos os meus familiares, irmãos e sobrinhos por todos os sorrisos e brincadeiras que me proporcionaram.

Ao amigo Zebu que com a sua companhia consegue deixar até o bandejão um pouco mais gostoso.

E a todos os meus Professores que de um modo ou de outro contribuíram para a minha formação.

Resumo

Arenstein, L. S. **ProbLog - Programação Lógica Probabilística: Um Estudo e Aplicação**. 2018. Trabalho de Conclusão de Curso - Instituto de Matemática e Estatística, Universidade de São Paulo.

Uma das questões atuais da inteligência artificial é a construção de modelos que combinem o conhecimento lógico relacional, consigam lidar com incertezas probabilísticas e também tenham a capacidade de aprendizagem de parâmetros por meio de tarefas de inferências. A área de programação lógica probabilística surgiu desse interesse, e o *ProbLog* é uma linguagem desse paradigma de programação que tem tido um grande destaque.

Podemos entender o *ProbLog* como uma extensão do *Prolog* que é uma linguagem de programação lógica, onde a ideia central, é a definição de dois tipos de relações: os fatos (objetos de um domínio) e as regras (leis sobre este domínio). A partir dessas, realizamos perguntas onde o programa nos retorna uma resposta booleana.

O grande diferencial do *ProbLog* é que podemos anotar uma medida de incerteza aos fatos de um programa lógico. Deste modo, é possível construir modelos complexos que consigam lidar ao mesmo tempo com fatos e regras determinísticas e também variáveis probabilísticas. Além disso, diversas tarefas de inferência e aprendizado de parâmetros foram implementadas nesta linguagem.

Inicialmente, neste trabalho, apresentaremos a teoria da programação lógica e os aspectos relevantes para nós do *Prolog*. Em seguida, estudamos o *ProbLog*, sua sintaxe, semântica, e exemplos de problemas modelados nesta linguagem. A nossa aplicação será a construção de um modelo lógico probabilístico inédito para a recomendação de filmes em que os parâmetros são aprendidos a partir do gosto cinematográfico de um indivíduo. Finalizamos este trabalho com um estudo de caso investigando como este nosso modelo funcionaria na prática com dados reais obtidos de um questionário.

Palavras-chave: Programação Lógica Probabilística, ProbLog, Programação Lógica, Prolog, Inferência, Inteligência Artificial, Recomendador de Filmes.

Abstract

Arenstein, L. S. **ProbLog - Probabilistic Logic Programming: A Study and Application**. 2018. Undergradute Thesis - Institute of Mathematics and Statistics, University of Sao Paulo.

A key question in artificial intelligence is the construction of models that combine reasoning with relational logical knowledge, probabilistic uncertainties, and also have the ability to learn parameters through inference tasks. The area of probabilistic logic programming arose out of this interests. *ProbLog* is a programming language of this paradigm that has had a great impact on the field.

We understand *ProbLog* as an extension of *Prolog* which is a logical programming language where the central idea is the definition of two types of relations: Facts (objects of a given domain) and Rules (laws on this domain). From these relations we can make queries and the program returns a Boolean answer.

The great advantage of *ProbLog* is that we can add a measure of uncertainty to the facts of a logic program so that it is possible to construct complex models that can deal with both deterministic facts and rules and also probabilistic variables. In addition, several inference and parameter learning tasks have been implemented in this language.

Initially in this work we will present the main concepts of the theory of logic programming language, and also an introduction to the relevant aspects of *Prolog*. Next, we will introduce *ProbLog*, its syntax, semantics and examples of problems modeled in this language. Our application will be the construction of a logical probabilistic model for the recommendation of films in which the parameters are learned by the cinematographic preference of an individual. Until today, it has not been yet proposed how to create this model, and this will be one of our contributions. We conclude this thesis with a case study investigating how our model of the theoretical recommendation system would work in practice with real data obtained from a questionnaire.

Keywords: Probabilistic Logic Programming, ProbLog, Logic Programming, Prolog, Inference, Artificial Inteligence, Recommendation System.

Sumário

1. Introdução

"11:15, restate my assumptions: 1. Mathematics is the language of nature. 2. Everything around us can be represented and understood through numbers. 3. If you graph these numbers, patterns emerge. Therefore: There are patterns everywhere in nature."

– Max Cohen, Pi 1998

Atualmente um dos desafios da inteligência artificial é a construção de modelos que consigam combinar três áreas do conhecimento matemático e da ciência da computação, que até então vem sendo desenvolvidos independentemente. A representação do conhecimento lógico relacional é a primeira delas, mas queremos também que este modelo saiba lidar com incerteza probabilísticas. O último elemento que estamos interessado que este nosso modelo tenha é a capacidade de aprender e ajustar os parâmetros lógicos e probabilísticos por meio de tarefas de inferências. A nova área de programação lógica probabilística trabalha com estas três áreas do conhecimento de um modo unificado. O *ProbLog* que é uma linguagem desse paradigma de programação tem obtido um grande destaque na atualidade. Diversos artigos sobre esta linguagem foram aceitos nas principais entidades de inteligência artificial. A programação lógica probabilística vai ser o tema principal de estudo deste trabalho.

No início da década de 70, houve o surgimento de um novo paradigma de programação, a programação lógica que é baseada fortemente na lógica matemática.

A ideia fundamental deste tipo de programação é utilizar o computador para inferir conclusões a partir de um conjunto de sentenças lógicas. Um programa lógico é um conjunto dessas sentenças que no caso são fórmulas lógicas.

Tenham o argumento a seguir como exemplo para a próxima explicação:

Sócrates é um homem.
Todo homem é mortal.
Logo, Sócrates é mortal

Este argumento é composto de três sentenças lógicas. Por meio da lógica de predicados, conseguimos programar as duas primeiras sentenças como fórmulas lógicas de primeira ordem, de um modo que o programa consiga inferir a última delas. Iniciaremos este trabalho com um estudo teórico da programação lógica de primeira ordem.

O *Prolog*[1] é uma linguagem de programação lógica que obteve o maior destaque entre as linguagens desta categoria. Conceitualmente programar em *Prolog* consiste em definir relações que podem ser de dois tipos: *fatos* e *regras*, e a partir desses podemos fazer *perguntas* para o *Prolog* que, por meio de uma inferência lógica, nos retorna uma resposta. Abaixo, temos a representação em *Prolog* do argumento visto acima

```
%Fato:
homem(sócrates).
%Regra:
mortal(X) :- homem(X).
%Pergunta:
?- mortal(X) - Resposta: sócrates
```

Antes de iniciarmos o estudo do *ProbLog* nesse trabalho, desenvolvemos um capítulo com as características relevantes do *Prolog* que são pertinentes para o aprendizado do *ProbLog*.

[1]Foi desenvolvido pelos cientistas da computação Alain Colmerauer e Philippe Roussel em 1972 na Universidade de Marselha - França

Com a teoria da programação lógica e a linguagem *Prolog*, conseguimos modelar e fazer inferências sobre o conhecimento lógico de primeira ordem. No entanto, queremos construir um modelo que também saiba representar incertezas probabilísticas. Podemos interpretar o *ProbLog* como uma extensão do *Prolog* em que os nossos *fatos* podem ser expressos com uma medida de incerteza, chamados de *fatos probabilísticos*. Com ele é possível modelar distribuições complexas de probabilidade em que conseguimos trabalhar com variáveis determinísticas e não determinísticas simultaneamente.

Considere que o Sócrates resolveu entrar em um jogo de roleta russa. Conseguimos descrever esse jogo com a seguinte *regra*: Um indivíduo X morre se ele estiver jogando e atirar o revolver quando o tambor estiver com uma bala. E com o *fato probabilístico*: Temos 1 bala entre as 6 posições possíveis do tambor do revólver.

A primeira rodada desta partida pode ser modelada no *ProbLog* como:

```
%Fato Lógico:
assistindo(pitagoras).
jogando(socrates).
%Fato Probabilístico:
1/6::tambor_revolver(bala); 5/6::tambor_revolver(vazio).
%Regra:
morre(X) :- jogando(X), tambor_revolver(bala).
%Pergunta:
query(morre(X)). %Resposta morre(socrates) = 1/6.
```

No capítulo *ProbLog* apresentamos a sintaxe, semântica e exemplos de programas que podem ser desenvolvidos, e o terceiro elemento do nosso modelo desejado que são as ferramentas de inferência e aprendizado do *ProbLog*.

Após o estudo do *ProbLog*, vamos começar o desenvolvimento da nossa aplicação que vai ser a construção de um recomendador de filmes nesta linguagem. Queremos construir um modelo de distribuição de probabilidade complexo que integre também *fatos* e *regras* lógicas. Nosso objetivo é que seja possível calcular a chance de um determinado indivíduo gostar de um filme, dado o conhecimento prévio

dos seus gostos cinematográficos. A motivação para a construção deste modelo é que até a presente data, ainda não foi proposto nenhum outro modo de como criar um recomendador de filmes (que é um problema bastante estudado) em uma linguagem lógica probabilística.

Finalizamos este trabalho com um estudo de caso de como este nosso recomendador de filmes desenvolvido no *ProbLog* funcionaria na prática, isto é, com informações obtidas a partir de um questionário sobre as preferências cinematográficas de uma entrevistada. Para esta tarefa, utilizando a linguagem de programação *R*, iremos desenvolver um algoritmo que consigam criar os *fatos* probabilísticos a partir do questionário realizado. Os parâmetros destes *fatos* serão aprendidos de acordo com os gostos da entrevistada. Como ainda não foi sugerido nenhum método, e não existe nenhuma referência de como transformar informações de um banco de dados para *fatos* na formatação aceita pelo *ProbLog*, uma parte significativa deste capítulo final vai ser destinada a essa tarefa. Visto a inexistência de algum outro trabalho que se preocupou com essa tarefa, esta é outra contribuição inédita deste trabalho em que esperamos que seja útil para o desenvolvimento da área de programação lógica probabilística. E por fim, realizaremos uma análise teórica e empírica dos resultados obtidos pelo nosso modelo lógico probabilístico de um recomendador de filmes.

2. Programação Lógica

Neste capítulo, apresentarem a programação lógica, que é um paradigma de computação baseado na lógica matemática. Para isso serão apresentados alguns dos conceitos elementares da lógica de primeira ordem, onde em cima deles vamos construir as cláusulas que são os elementos principais da programação lógica.

O *Prolog* é o exemplo de maior destaque dentre as linguagens de programação lógica e vai ser o tema do próximo capítulo, assim sendo, já vamos acostumando o leitor com a sua sintaxe. O *ProbLog* que é o objeto principal de estudo desta dissertação é uma linguagem lógica probabilística que entendemos como uma extensão da programação lógica, e por isso, a importância desses dois primeiros capítulos.

2.1 Fórmulas Lógicas

O foco desta seção é o estudo de fórmulas lógicas dentro do contexto da lógica de primeira ordem. Vamos começar com uma breve explicação do que é a lógica de primeira ordem e em seguida a partir de um exemplo vamos construir os elementos que fazem parte de uma fórmula lógica.

A lógica de primeira ordem é um sistema lógico que estende a lógica proposicional que é o tipos mais simples de sistema lógico. A lógica proposicional é composta apenas por 5 conectivos lógicos que veremos adiante. Já a lógica de primeira ordem além desses conectivos lógicos possui também 2 quantificadores lógicos, o quantificador universal e o existencial. Deste modo o alfabeto da lógica de primeira ordem é muito mais rico que este primeiro sistema como veremos nos exemplos abaixos.

Quando precisamos expressar relações no mundo real utilizamos sentenças declarativas, como:

(1) “Toda mãe ama os seus filhos”

(2) “Márcia é a mãe de José que é o seu filho”

Utilizando as sentenças acimas é possível chegar em novas conclusões. Sabendo que (1) e (2) são verdades é possível concluir que:

(3) “Márcia ama José”

Analisando o resultado, vemos que as sentenças (1) e (2) descrevem um *universo* de pessoas (Márcia, José) e também de *relações* (“...ama...”, “...é a mãe...”), que podem ser verdadeiras ou não entre as pessoas. Neste exemplo, temos a ideia principal de uma Programação Lógica que é descrever relações sobre objetos ou um domínio e utilizar um sistema de programação para conseguir chegar em alguma conclusão, no nosso caso (3). Para um computador conseguir lidar com sentenças declarativas como (1) e (2), temos que nos preocupar com a sua sintaxe, e mais importante que isso, com o conjunto de regras lógicas que permitam ao programa fazer inferências e chegar em conclusões do mesmo modo que chegamos em (3).

As nossas sentenças relacionam *indivíduos* em um *universo* e as *relações* entre eles, então o nosso ponto de partida, é que o alfabeto da nossa linguagem deve conter:

- Símbolos para denotar os *indivíduos* (Márcia e José) que são chamados de *Constantes*.
- Símbolos para denotar as *relações* (Mãe, ama, filho de) que são chamado de *Predicados*

Todo *Predicado* tem um número n natural associado chamado de *aridade*. A aridade de um *predicado* é o número n de elementos que compõem as *n-tuplas* ordenadas pertencentes à este predicado. Na sentença (3), temos um predicado com aridade = 2 (n-dupla), pois a relação “ama” denota um conjunto com dois elementos. Com a definição deste alfabeto, podemos expressar a sentença “Márcia ama José” como uma fórmula *ama(márcia,josé)*.

Agora, suponha que queremos criar sentenças declarativas mais genéricas, que possam se referir a todo o nosso universo, como por exemplo:

(4) “Para quaisquer que sejam os indivíduos X e Y, se X é uma mãe e Y é o seu filho, então X ama Y.”

Para conseguirmos expressar essa sentença na linguagem lógica, precisamos de símbolos lógicos. Abaixo, temos uma tabela com os conectivos ou operadores lógicos e em seguida uma tabela com os quantificadores lógicos. Vale lembrar que com estes 5 conectivos lógicos temos os símbolos da lógica de predicados.

Conectivos Lógicos

Símbolo	**Ler Como**	**Interpretação**
$\wedge$	Conjunção	“E”
$\vee$	Disjunção	“Ou”
$\neg$	Negação	“Não”
$\Longrightarrow$	Implica	“Se A Então B”
$\Longleftrightarrow$	Equivalência Lógica	“A implica B e B implica A”

Quantificadores Lógicos

Símbolo	**Ler Como**	**Interpretação**
$\forall$	Quantificador Universal	“Para Todo (Para cada)”
$\exists$	Quantificador Existencial	“Existe (Pelo menos 1)”

Esses são os 7 operadores da Lógica de Primeira Ordem.

Escrevendo a sentença (4) como uma expressão lógica, temos:

$$\forall X(\forall Y((mãe(X) \wedge filho\_de(Y, X)) \implies ama(X, Y)))$$

Ao utilizarmos os símbolos X e Y, estamos utilizando os chamados *Símbolos Variáveis*.

Para ilustrar a utilização dos símbolos: Conjunção, Negação, Equivalência Lógica e o Quantificador Existencial consideramos as sentenças:

(5) “Márcia tem um filho chamado José ou João e eu sei que o Carlos não é o filho dela”

$\exists X\, filho\_de(X, marcia) \wedge (filho\_de(josé, marcia) \vee filho\_de(joão, marcia))$
$\wedge (\neg filho\_de(carlos, marcia))$

(6) "Se X é Mulher e X deu à Luz, X é uma Mãe" é equivalente a:
"Se X é uma mãe, X é Mulher e X deu à Luz"

$$(mulher(X) \wedge deu\_a\_luz(X)) \iff mae(X)$$

O último símbolo que falta apresentarmos são os *functores* ou *funcionais* que representam uma função sobre um domínio. Todo funcional tem uma aridade que determina o número de argumentos da função. Se quisermos definir uma relação da família Pereira composta pela mãe Márcia o pai Joaquim os filhos José e Paulo e o cachorro Rex, podemos construir um funcional família de aridade 4, um funcional filhos de aridade 2 e um funcional cachorro de aridade 1. Construindo essa relação como um termo composto temos:

$$familia(marcia, joaquim, filhos(jose, paulo), cachorro(rex))$$

Com os exemplos e conceitos vistos nesta seção, introduzimos os elementos que compõem uma fórmula lógica. Utilizando a formatação do *Prolog*, o alfabeto da lógica de primeira ordem é composto por símbolos de:

- ***Constantes***, expressos por números ou alfanuméricos de primeira letra minúscula (x, pedro, 7).
- ***Variáveis***, expressas por alfanuméricos de primeira letra maiúscula (X, Y_1, Y_2).
- ***Predicados***, expressos por alfanuméricos de primeira letra minúscula e com uma aridade ≥ 0.
- ***Funcionais***, expressos por alfanuméricos de primeira letra minúscula e com uma aridade > 0.
- ***Conectivos Lógicos***: $\wedge$ E, $\vee$ Ou, $\neg$ Não, $\implies$ Implicação, $\iff$ Equivalência.

- ***Quantificadores Lógicos***: $\forall$ Para Todo, $\exists$ Existe.
- ***Auxiliares***: Como parênteses ( ) e vírgulas " ".

2.2 Cláusulas Definidas Positivas

A ideia da programação lógica é a de utilizar o computador para inferir conclusões a partir de um conjunto de sentenças. Um programa lógico é um conjunto dessas sentenças que são compostas por fórmulas lógicas (as mesmas apresentadas na seção anterior) finitas. Nesta seção, vamos introduzir e definir as cláusulas definidas positivas que é o principal elemento dos programas definidos positivos dos quais o *Prolog* faz parte.

Existem dois tipos de cláusulas definidas positivas utilizadas para representar sentenças declarativas:

- ***Fatos***: Utilizados para mostrar que existe uma relação entre objetos.
  (1) José é o filho da Márcia. (2) Jaime é o filho do José.
- ***Regras***: Utilizados para mostrar que uma relação entre objetos é verdade quando alguma outra relação também é verdade.
  (3) O neto de uma pessoa é o filho do filho dessa pessoa.

Utilizando a sintaxe do *Prolog* para descrever (1) e (2), temos as chamadas fórmulas atômicas:

$$filho(jose, marcia)$$

$$filho(jaime, jose)$$

Cada um dos 3 elementos é chamado de átomo. O átomo filho é a *Cabeça* (*Head*) da fórmula e os átomos (jose, marcia e jaime) são chamados de *Corpo* (*Body*) da fórmula.

Escrevendo a sentença (3) como uma expressão lógica temos:

$$\forall X \forall Y (neto(X, Y) \Longleftarrow \exists Z (filho(X, Z) \wedge filho(Z, Y))) \qquad (3.1)$$

Na notação do *Prolog* :

$$neto(X, Y) : -filho(X, Z), filho(Z, Y)$$

Agora, já estamos aptos a entender a definição de Cláusulas e o caso especial que é o enfoque desta seção:

Definição de Cláusula: A cláusula é uma fórmula do tipo: $\forall(L_1 \vee \cdots \vee L_n)$ onde cada L_i é uma fórmula atômica (um literal positivo) ou a negação de uma fórmula atômica (um literal negativo).

Estamos interessados no caso especial de cláusula que é a **cláusula definida positiva** que contém exatamente **um** literal positivo que é uma fórmula do tipo:

$$\forall(A_0 \vee \neg A_1 \vee \cdots \vee \neg A_n)$$

Ou a notação mais usual:

$$A_0 \Longleftarrow A_1 \wedge \cdots \wedge A_n \qquad \text{(Para n} \geq 0),$$

onde $A_0, \ldots, A_n$ são átomos.

A sentença (3.1) é um exemplo de uma cláusula definida positiva.

Todas as fórmulas lógicas que podem ser expressas deste modo são chamadas de *Cláusulas Definidas Positivas*. Fatos são cláusulas em que $n = 0$. Neste caso, temos apenas A_0 e omite-se a seta, as Regras são cláusulas em que $n > 0$.

Finalizamos esta seção com a seguinte definição:

Definição de Programas Definidos Positivos: Um programa definido positivo é um conjunto finito de cláusulas definidas positivas.

2.3 Funcionamento de Programas Definidos Positivos

Nas seções anteriores, vimos que a construção de um programa definido positivo consiste na criação de cláusulas definidas positivas. Agora, estamos interessados no seu funcionamento. Podemos interpretar a programação lógica como uma modelagem que é realizada por meio de fatos e regras e o intuito deste processo é poder fazer perguntas (*queries*) para o programa sobre o nosso modelo.

Quando fazemos uma pergunta estamos na realidade criando um *objetivo definido positivo* que é uma fórmula lógica do tipo:

$$\forall(\neg(A_1, \ldots A_n))$$

Usualmente escrita como:

$$\Longleftarrow A_1, \ldots A_n$$

Onde os $A_i's$ são fórmulas atômicas chamadas de *sub-objetivos*.

O significado lógico de um objetivo pode ser expresso como:

$$\neg\exists X_1 \ldots \exists X_n \ (A_1 \wedge \cdots \wedge A_n) \qquad (1)$$

Onde $X_1 \ldots X_n$ são todas as variáveis que pertencem a este objetivo.

Interpretamos o objetivo como uma pergunta existencial da forma (1) onde o sistema tenta negá-la construindo um contra exemplo para mostrar que a expressão é falsa, ou seja, o objetivo existe. Para isso, ele tenta encontrar termos $t_1, \ldots, t_n$ tais que a fórmula obtida de $(A_1 \wedge \cdots \wedge A_n)$ é verdadeira quando substituímos a variável X_i por t_i $(1 \leq i \leq n)$ em qualquer configuração possível do programa. Se isso for possível, temos que existe uma conjunção lógica dos sub-objetivos que permite chegar na conclusão (resposta) desejada da nossa pergunta.

2.3.1 Exemplos de Perguntas

Considerando os fatos (1) e (2) abaixo e a regra (3):

$$filho(jose, marcia)$$

$$filho(jaime, jose)$$

$$neto(X, Y) : -filho(X, Z), filho(Z, Y)$$

Veremos, na tabela abaixo, exemplos de perguntas, o equivalente do seu objetivo definido positivo, e a resposta:

Pergunta	**Objetivo**	**Resposta**
"José é um filho da Márcia?"	$\Longleftarrow filho(jose, marcia)$	Sim (pelo fato 1)
"Quem é o neto da Márcia?"	$\Longleftarrow neto(X, marcia)$	X=jaime (satisfaz (3))
"Quem é a avó do Jaime?"	$\Longleftarrow neto(jaime, X)$	X=marcia (satisfaz (3))

Último Comentário da Seção:

Neste trabalho, vamos sempre interpretar que um Programa Lógico é o mesmo que um Programa Definido Positivo baseado na Lógica de Primeira Ordem. Nas próximas seções, vamos utilizar apenas a primeira denominação. Note que existem outras classes de programas lógicos, mas estas não serão utilizadas neste trabalho, pois o *ProbLog* é baseado na Lógica de Primeira Ordem.

3. Uma Breve Introdução ao Prolog

Agora que já temos uma noção do que é a programação lógica, designamos esse capítulo para um estudo do *Prolog*. Inicialmente, apresentamos os três tipos de cláusulas existentes do *Prolog* e exemplificamos o uso delas em um programa simples. Mais adiante, mostramos a importância das regras recursivas que são possíveis de serem criadas nessa linguagem e, por fim, mostramos o funcionamento do *Prolog* para responder as nossas perguntas.

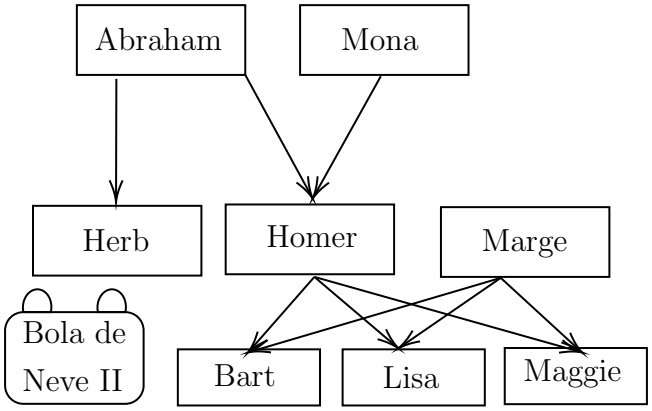
Caso o leitor esteja interessado em praticar o conhecimento adquirido, sugerimos um exercício na seção (3.4) que é parecido ao apresentado na primeira seção. Todos os programas foram desenvolvidos no site[1] que permite compilar e executar programas em *Prolog* gratuitamente no seu próprio navegador.

3.1 Prolog: Fatos, Regras, Perguntas e Simpsons

Conceitualmente, programar em *Prolog* consiste em definir relações e fazer perguntas sobre elas. Desenvolvemos o programa criando uma sequência de cláusulas, que podem ser de três tipos: **Fatos, Regras e Perguntas**. Uma relação pode ser feita por meio de **fatos**, que simplesmente mostram quais objetos satisfazem esta relação, ou por **regras** que definem algum tipo de conceito ou lei sobre o domínio para satisfazer as relações desejadas. Após definirmos esses dois tipos de cláusulas, podemos fazer **perguntas** como, por exemplo, se um determinado objeto (ou qual objeto) satisfaz ou não uma relação.

[1] https://swish.swi-prolog.org/ - clicando no botão *Create a Program*

Nosso primeiro algoritmo vai representar a árvore genealógica dos personagens da série de tv norte-americana *Os Simpsons*. Abaixo, temos uma figura com os nomes dos personagens e as setas representam os descendentes deles.



Vou dividir o nosso algoritmo em duas partes: na primeira vamos criar os fatos sobre a família Simpsons e, em seguida, vamos desenvolver as regras de parentesco; já na segunda parte, vou sugerir exemplos de perguntas (*queries*) que são possíveis de fazer ao *Prolog* com as suas respectivas respostas.

Caso o leitor deseje executar o código abaixo, basta copiar e colar o programa no site `https://swish.swi-prolog.org/` clicando no botão *Create a Program*.

Programa 1: Família Simpsons - *Fatos e Regras.*

```
%Primeiro vamos definir alguns Fatos:
homem(abraham). homem(herb). homem(homer). homem(bart).
mulher(mona). mulher(marge). mulher(lisa). mulher(maggie).
gato(bola_de_neve_2).
%Lemos esses fatos abaixos como:
%Homer é casado com Marge.
casal(abraham,mona). casal(homer,marge).
pai(abraham,herb). pai(abraham,homer).
pai(homer,bart). pai(homer,lisa). pai(homer,maggie).

%Note que definimos apenas 4 tipos de fatos.

%Abaixo temos exemplos de Regras criadas para
```

```
%definirmos as relações familiares entre os membros.

%Regra 1:
%Vamos definir que um humano é um homem ou uma mulher.
humano(X) :- homem(X).
humano(X) :- mulher(X).

%Regra 2: X é mãe de Y Se: X é Casada com Z e Z é Pai de Y.
mae(X,Y) :- casal(Z,X), pai(Z,Y).

%Regra 3: Z é avô de X Se: Z é pai de Y e Y é Pai de X.
avô(Z,X) :- pai(Z,Y), pai(Y,X).

%Regra 4: avó é a esposa do avô.
avó(Z,X) :- casal(Y,Z),avô(Y,X).

%Regra 5: Diferentes
diferente(X,Y) :- (X\=Y).

%Regra 6:
%X é irmão de Y Se: X é homem e ambos tem o mesmo pai.
irmão(X,Y) :- homem(X), pai(Z,X), pai(Z,Y), diferente(X,Y).

%Regra 7: X é tio de Y Se: X é homem, X é irmão de Z e Z é pai de Y.
tio(X,Y) :- homem(X), irmão(X,Z), pai(Z,Y).
```

No Programa 1 acima, vimos como modelar o problema da árvore genealógica com **Fatos** e **Regras**. Uma prática importante do *Prolog* é a de tentar utilizar poucos fatos e investir na criação de regras. No nosso algoritmo, não criamos nenhum fato sobre a relação de mãe e avós, pois é possível contruir essas relações por meio dos fatos já existentes. Abaixo, temos a segunda parte do algoritmo onde realizamos perguntas (*queries*).

Programa 1: Família Simpsons - *Perguntas.*

```
%Perguntando se o gato é um humano.
?- humano(bola_de_neve_2) - Resposta: False.

%Perguntando quem são os filhos da Marge.
```

```
?- mãe(marge,X) - Resposta: bart, lisa, maggie.

%Perguntando quem é o avó da Lisa.
?- avó(X,bart) - Resposta: abraham

%Perguntando quem são os irmãos do Bart
?- irmão(bart,Y) - Resposta: lisa, meggie.

%Note que se a nossa Regra 6 não tivese a Regra 5 como
%conjunção a resposta para a pergunta acima também teria
%a resposta bart como irmão do bart.

%Exemplo de uma Querie com duas condições:
?- avô(Z,X),not(homem(X)) - Respostas: Z=mona e X=lisa ou X=meggie
```

3.2 Prolog: Recursão

Suponha que queremos criar uma regra a respeito dos antecessores de um membro de uma família. Recursivamente, definimos um antecessor como:*o seu pai, ou o pai do seu pai, ou o pai do pai do seu pai,...* e assim por diante.

Um (péssimo) jeito de programar essa regra seria:

Para qualquer X e Z:
(i) X é um antecessor de Z Se X é o pai de Z.
(ii) X é um antecessor de Z Se X é o pai de Y e Y é o pai de Z.
(iii) X é um antecessor de Z Se X é o pai de Y1 e Y1 é o pai de Y2 e Y2 é o pai de Z.

O problema dessa programação é que ela é muito extensa no sentindo que para n gerações da família vão ser necessárias n dessas regras. E também, sempre que adicionarmos mais uma geração a família é necessário adicionar mais um regra.

Um recurso muito importante e útil do *Prolog* é a **Recursão** A nossa regra antecessor vai ser composta em duas partes, a primeira parte é:

Para qualquer X e Z:
X é um antecessor de Z Se X é o pai de Z.

Já a segunda regra deve ser criada recursivamente, isto é:

Para qualquer X e Z existe um Y tal que:
(1) X é o pai de Y e
(2) Y é um antecessor de Z.

Em *Prolog* temos o seguinte código:

```
%Regra 1:
antecessor(X,Z) :- pai(X,Z).
%Regra 2:
antecessor(X,Z) :- pai(X,Y), antecessor(Y,Z).
```

Note que o ponto principal da formulação da segunda regra foi que usamos a própria definição de antecessor na regra de antecessor. Este é um exemplo de uma definição recursiva, um atributo muito importante na programação do *Prolog*. Abaixo, temos a definição de fatorial de um número e, em seguida, como escrever essa definição no *Prolog*.

$$n! = \begin{cases} 1, \ se \ n = 0 \\ \prod_{k=1}^{k=n} k, \ se \ n > 0, n \in \mathbb{N}^* \end{cases}$$

```
fatorial(0,Resultado) :-
    Resultado is 1.
fatorial(N,Resultado) :-
    N > 0,
    N1 is N-1,
    fatorial(N1,Resultado1),
    Resultado is Resultado1*N.

?- fatorial(5,X). - Resposta: X = 120
```

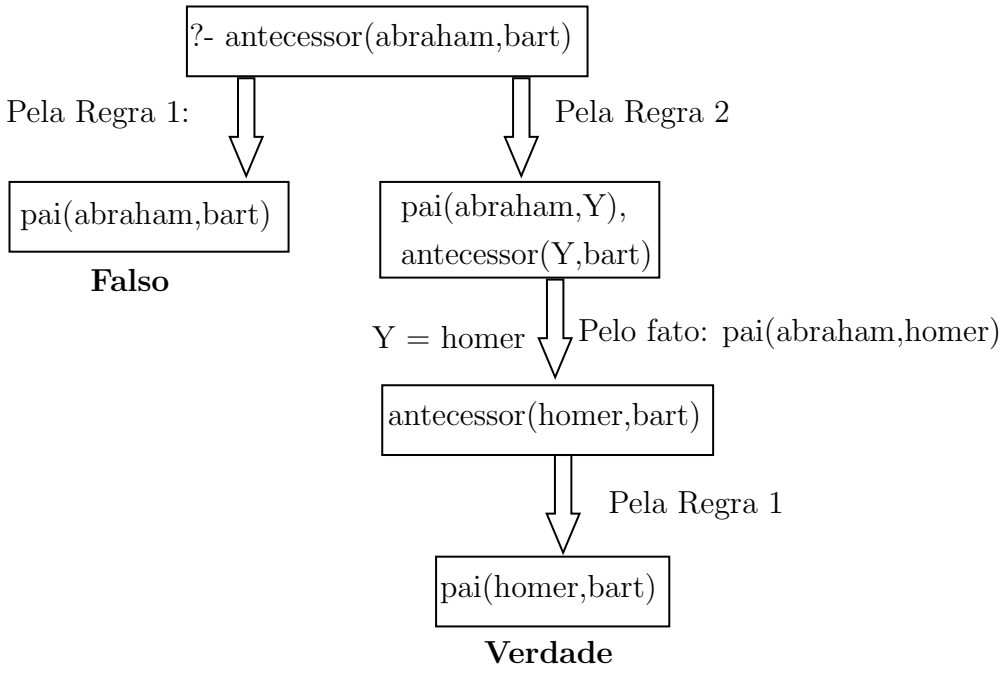
3.3 Como o Prolog Responde às Perguntas?

Uma *Pergunta* feita ao *Prolog* é uma sequência de um ou mais objetivos que ele verifica se são logicamente verdadeiros ou falsos, dado o conjunto de fatos e regras

do programa. Note que se a pergunta também contêm variáveis, o *Prolog* deve encontrar quais dessas variáveis satisfazem logicamente a nossa pergunta.

De um ponto de vista matemático, a interpretação de um programa feito no *Prolog* pode ser: O *Prolog* aceita um conjunto de fatos e regras como um conjunto de axiomas e a pergunta realizada pelo usuário é um teorema que o sistema vai tentar provar demonstrando se essa pergunta pode ser deduzida logicamente pelos axiomas.

Considerando que implementamos as duas regras de antecessor ao nosso Programa 1, vamos mostrar, por meio de um diagrama, como o *Prolog* responde a nossa pergunta: ? - antecessor(abraham,bart), isto é, se o Abraham é um antecessor do Bart:



Para responder a nossa pergunta, o *Prolog* utiliza os fatos e regras que estão relacionados com ela: primeiro ele utiliza a regra 1 da relação antecessor, que é falsa, pois não existe o fato pai(abraham,bart). Em seguida, ele começa o processo de testar a regra 2, o fato pai(abraham,Y) é verdade para Y = homer. Agora, o *Prolog* utilizando Y = homer testa a relação antecessor(homer,bart) que, pela regra 1, é verdade pois existe o fato pai(homer,bart). E, assim, conclui que antecessor(abraham,bart) é Verdade.

4. ProbLog

Na década de 90, começaram a ser desenvolvidas diversas linguagens de programação que combinavam representações e distribuições de probabilidade com lógica (de um jeito parecido ao *Prolog*). Entre elas, podemos destacar o *SLPs*[1], *PRISM*[2]. Em 2007, foi publicado o primeiro artigo sobre a linguagem[3] junto com a sua primeira versão o *ProbLog1*. A partir deste, foram publicados diversos artigos importantes sobre tarefas de inferência, aprendizado e aplicações que podem ser feitas com essa linguagem. Muitos artigos foram aceitos em entidades de destaque como a IJCAI[4], AAAI[5] e a ECAI[6] que são as principais entidades de Inteligência Artificial da atualidade. Assim, o *ProbLog* parece ser a linguagem de programação deste tipo de maior destaque na atualidade.

Vamos definir uma linguagem lógica probabilística como uma linguagem lógica em que podemos atribuir uma medida de probabilidade aos fatos. O *ProbLog* pertence a essa categoria de linguagem de programação. Um programa desenvolvido no *ProbLog* especifica a distribuição de probabilidade de um determinado universo. Nele, é possível construir diversos tipos de modelos e algoritmos para vários tipos de tarefas de inferência que veremos adiante.

O desenvolvimento inicial foi realizado por pesquisadores nas universidades de *KU Leuven*, na Bélgica, e na Universidade do Porto, em Portugal. Atualmente, no Brasil temos pesquisas independentes nessa área sendo realizadas pelo Professor Dr. Denis Deratani Mauá docente, do Instituto de Matemática e Estatística e

[1] Stochastic Logic Programs, Muggleton 1995

[2] PRISM: A Language for Symbolic-Statistical Modeling, Sato e Kameya 2001

[3] ProbLog: A probabilistic Prolog and its application in link discovery - L. De Raedt, A. Kimmig and H. Toivonen

[4] *International Joint Conferences on Artificial Intelligence Organization*

[5] *Association for the Advancement of Artificial Intelligence*

[6] *European Conference on Artificial Intelligence*

também por membros do *Decision Making Lab* da Escola Politécnica, ambos da Universidade de São Paulo.

Neste capítulo, vamos apresentar a sintaxe, exemplos de programas que podem ser desenvolvidos, os dois principais modos existentes da linguagem, e por fim, apresentar a semântica do *ProbLog*. Todos os programas dessa seção foram desenvolvidos utilizando o compilador *online*https://dtai.cs.kuleuven.be/problog/editor.html que existe no site oficial do *ProbLog*.

4.1 A Sintaxe do ProbLog

A sintaxe de um programa do *ProbLog* consiste em duas partes: um programa lógico, que é um conjunto de regras como vimos no *Prolog* e a segunda parte é um conjunto de fatos probabilísticos.

Um fato lógico tem a mesma definição vista no *Prolog*, e é expresso como $\mathbf{h(f)}$ onde $\mathbf{h}$ é um ***átomo derivado*** (que é equivalente ao *Head* de um fato). Um exemplo seria a relação que diz que a Márcia é uma pessoa: *pessoa(marcia)*, onde temos *pessoa* como o nosso átomo derivado.

Um fato probabilístico $\mathbf{p :: f}$ é um fato $\mathbf{f}$ que acontece com a probabilidade $\mathbf{p}$, onde $\mathbf{f}$ é chamado de ***átomo probabilístico***. Na notação de probabilidade que estamos mais acostumados, isto é equivalente a: $\mathbf{P}(f = Verdade) = p$. Um exemplo simples seria o lançamento de uma moeda honesta descrito no *ProbLog* como: *0.5::saiu_coroa*. Neste caso, *saiu_coroa* é o nosso átomo probabilístico. Para representar eventos disjuntos utilizamos a notação: $\mathbf{p_1 :: h_1; ...; p_n :: h_n}$ onde $\sum_{i=1}^{n} p_i \leq 1$.

Atualmente, estamos na versão 2 do *ProbLog* e um dos seus diferenciais foi a implementação de *Probabilidades Intencionais*, isto é, expressões da forma $\mathbf{p :: f(X_1, X_2, ..., X_n) : -corpo}$ em que o **corpo** é um fato não probabilístico. A interpretação desta expressão é: Se o **corpo** for Verdade (ou o **corpo** aconteceu), o conjunto de fatos $\mathbf{X_1, X_2, ..., X_n}$ podem ocorrer com probabilidade $\mathbf{p}$.

Temos duas condições obrigatórias na sintaxe do *ProbLog*: a primeira é que o conjunto de átomos probabilísticos deve ser disjunto em relação ao conjunto de átomos derivados. E a segunda condição é que todas as variáveis que aparecem na

Head de uma Regra devem também estar definidas (positivamente) no *Corpo* de um Fato.

4.2 Primeiros algoritmos em ProbLog

Todos os exemplos desse capítulo podem ser executados e modificados pelo editor online existente do *ProbLog* https://dtai.cs.kuleuven.be/problog/editor.html. Ao final de cada exemplo, será disponibilizado um *link* que já copia o algoritmo em questão para esse editor online. Para ilustrar a sintaxe e o uso básico do *ProbLog*, vamos começar com o seguinte exemplo:

Exemplo 1: Suponha que temos duas moedas, uma honesta e uma viesada no qual a fase cara ocorre com 60% de chance. Nosso experimento consiste em jogar aleatóriamente essas moedas e observar o resultado. Assumindo que elas são independentes (o resultado de uma não influência no resultado da outra) queremos responder as seguintes perguntas:

- 1) Qual a chance de tirarmos duas caras?

```
%Fatos Probabilisticos.
0.5::cara1. %P(Moeda1=Cara) = 0.5.
0.6::cara2.

%Regras Lógicas.
%Duas Caras é Verdade se ambas as Moedas forem caras.
duasCaras :- cara1, cara2.

%Perguntamos a Probabilidade dessa Regra ser Verdadeira:
query(duasCaras).
```

Resposta: $\mathbb{P}(Moeda1 \ e \ Moeda2 = Cara) = 0.5 \cdot 0.6 = 0.3$.

- 2) Sabendo que não tiramos duas caras, quais são as probabilidades de ter saido cara em cada uma das moedas? $\mathbb{P}(Moeda1 = Cara | Ambas \ não \ são \ caras)$.

Link Questão 1
Link Questão 2

```
%Fatos Probabilisticos
0.5::cara1. 0.6::cara2.

%Regras Lógicas
duasCaras :- cara1, cara2.

%Criamos a nossa Evidencia
evidence(duasCaras,false).

query(cara1).
query(cara2).
```

Para resolver essa questão analiticamente, devemos utilizar a fórmula da Probabilidade Condicional:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

Assim,

$$\mathbb{P}(Moeda1 = Cara|Ambas\ não\ são\ caras) = \frac{\mathbb{P}(Moeda1 = Cara\ \cap\ Ambas\ não\ são\ caras)}{\mathbb{P}(Ambas\ não\ são\ caras)}$$

$$= \frac{\mathbb{P}(Moeda1 = Cara\ e\ Moeda2 = Coroa)}{\mathbb{P}(Ambas\ não\ são\ caras)} = \frac{0.2}{0.2 + 0.3 + 0.2} = \frac{0.2}{0.7} \approx 0.285714.$$

Analogamente, $\mathbb{P}(Moeda2 = Cara|Ambas\ não\ são\ caras) = \frac{0.3}{0.7} \approx 0.428571.$

Neste primeiro exemplo vimos um experimento em que os eventos são *independentes*, isto é, o resultado de uma moeda não interfere no resultado da outra. No exemplo abaixo, temos uma situação em que as variáveis aleatórias são *dependentes*.

Exemplo 2: Considere o experimento abaixo, em que jogamos uma moeda viesada ($\mathbb{P}(Cara) = 0.4$). Se obtermos *Cara*, retiramos uma bola da *Urna 1*; caso contrário, retiramos uma bola da *Urna 2*. A primeira urna tem 2 bolas brancas e 7 bolas pretas, já a segunda urna tem 5 bolas brancas e 6 bolas pretas. Vamos criar um algoritmo em *ProbLog* para responder as seguintes questões:

- 1) Qual a probabilidade de tirarmos uma bola *preta*?

- 2) Qual a probabilidade de termos tirado *cara* dado que retiramos uma bola *branca*?

Começamos modelando o nosso problema em *ProbLog*. Note que, ao contrário do Exemplo 1, agora as nossas variáveis aleatórias são dependentes, para representar essa característica utilizamos o operador **;**

```
%Fatos Probabilisticos -
%As v.a's moeda, urna 1 e 2 são dependentes
%Assim utilizamos o ; entre elas.
%Mas note que elas são independentes entre sí.

%Moeda
0.4::moeda(1,cara); 0.6::moeda(1,coroa).

%Urna 1
2/9 :: urna(1,brancas); 7/9 :: urna(1,pretas).
%Urna 2
5/11 :: urna(2,brancas); 6/11 :: urna(2,pretas).

%Regras Lógicas
%Retirar 1 bola branca
branca :- moeda(1,cara), urna(1,brancas).
branca :- moeda(1,coroa), urna(2,brancas).

%Retirar 1 bola preta
preta :- moeda(1,cara), urna(1,pretas).
preta :- moeda(1,coroa), urna(2,pretas).
```

- Resolução do item 1:

```
%Basta fazer as Queries:
query(branca).
query(preta).
```

Link Exemplo 2

Como as urnas são independentes entre sí, segue:
$\mathbb{P}(Branca) = \mathbb{P}(Urna1 = Branca) + \mathbb{P}(Urna2 = Branca) =$
$= \mathbb{P}(Cara) * \mathbb{P}(Urna1 = Branca) + \mathbb{P}(Coroa) * \mathbb{P}(Urna2 = Branca) =$
$= 0.4 \cdot \frac{2}{9} + 0.6 \cdot \frac{5}{11} = \frac{179}{495} \approx 0.361616.$
Pelo complementar: $\mathbb{P}(Preta) = 1 - \mathbb{P}(Branca) = \frac{316}{495} \approx 0.638383.$

- Resolução do item 2:

```
%Evidencia que tiramos uma bola branca:
evidence(branca).
%E fazer a query:
query(moeda(1,cara)).
```

Para resolvermos essa questão analiticamente, utilizamos o **Teorema de Bayes**: *A probabilidade de ocorrência do evento A_i, supondo a ocorrência do evento B, é dada por:*

$$\mathbb{P}(A_i|B) = \frac{\mathbb{P}(B|A_i)\mathbb{P}(A_i)}{\sum_{j=1}^{n} \mathbb{P}(B|A_j)\mathbb{P}(A_j)} \qquad \text{, Para } j = 1, 2, ..., n,$$

onde $A_1, ..., A_n$ são todos os resultados possíveis da partição do espaço amostral.

Analíticamente resolvemos a questão 2, utilizando o Teorema de Bayes acima com $A =$ Cara, $B =$ Bola Branca e $j = 2$.
$\mathbb{P}(Cara|Branca) = \frac{\mathbb{P}(Branca|Cara)\mathbb{P}(Cara)}{\mathbb{P}(Branca|Cara)\mathbb{P}(Cara) + \mathbb{P}(Branca|Coroa)\mathbb{P}(Coroa)} =$
$\frac{\frac{2}{9} \cdot 0.4}{\frac{2}{9} \cdot 0.4 + \frac{5}{11} \cdot 0.6} = \frac{44}{179} \approx 0.245810.$

4.3 Modos do ProbLog

Na versão 2 da linguagem, existem oito tipos de modos em que podemos utilizar o *Problog*:

1. Inferência (*Inference*)
2. Aprendizado (*Learning*)

3. Explicação Mais Provável (*MPE - Most Probable Explanation*)
4. Teoria da Decisão (*DTProbLog - Decision Theoretic ProbLog*)
5. Probabilidade Máxima a Posteriori (*MAP - Maximum a Posteriori Probability*)
6. Amostragem (*Sampling*)
7. Explicação (*Explain*)
8. Linguagem Natural (*Natural Language*)

Nas próximas seções vou apresentar os dois primeiros modos existentes do *ProbLog*, pois são os mais utilizados na atualidade e o modo de Inferência foi o selecionado para a construção do meu recomendador de filmes.

No anexo 1 deste trabalho apresentamos os modos de Explicação Mais Provável e Teoria da Decisão, caso interesse ao leitor.

4.3.1 Modo: Inferência

Todos os programas apresentados na seção 4.2 são exemplos do modo Inferência, que é o padrão do *ProbLog*. Basicamente, neste método a nossa entrada é um modelo composto por fatos e regras, sendo possível também adicionar evidências. Como parâmetros de saída, o programa calcula as probabilidades das *queries* que estipulamos.

Vamos modelar a seguinte situação abaixo que é um problema conhecido de redes bayesianas no *ProbLog*.

Suponha que uma casa tenha um alarme que dispare se ocorrer um assalto ou um terremoto. Os donos dessa casa estão viajando mas os seus vizinhos *Paulo* e *Maria* estão nas casas ao lado e prometeram ligar para o dono se ouvirem que o alarme disparou. A probabilidade de assalto é de 0.1, terremoto 0.2, e algum dos vizinhos ouvir o alarme caso ele dispare é de 0.7.

Queremos saber a probabilidade de algum dos vizinhos ou os dois ligarem para o dono da casa, e essa vai ser a nossa *query*

Temos o seguinte programa abaixo implementado em *ProbLog* que descreve esta situação:

```
%Fatos Probabilísticos:
0.1::assalto.
0.2::terremoto.

%Regras Lógicas:
pessoa(paulo).
pessoa(maria).

0.7::escuta_alarme(X) :- pessoa(X).
%Interpretamos como:
%A pessoa(X) escuta o alarme com probabilidade 0.7

%O alarme dispara com um assalto ou terremoto
alarme :- assalto.
alarme :- terremoto.

%A pessoa(X) liga para o dono da casa
%Se o alarme disparar e essa pessoa escutar o alarme
liga(X) :- alarme, escuta_alarme(X).


query(liga(X)).
```

A resposta do *ProbLog* é: liga(maria) = 0.196 e liga(paulo) = 0.196, assim a chance de ocorrer uma ligação é a soma dessas duas probabilidade que é igual a 0.392. Na seção 4.4 mostramos como o *ProbLog* chega nesses resultados e a sua interpretação. Vamos mostrar que a *query(liga(X))* é diferente do cálculo de $\mathbb{P}(liga = 1)$.

Note que nesse exemplo utilizamos uma notação concisa na regra *0.7::escuta_alarme(X) :- pessoa(X)*, esse é um exemplo de um *Probabilidade Intencional* onde a (pessoa(X)) é um fato lógico que caso seja verdadeiro permite que o outro fato (escuta_alarme(X)) aconteça com probabilidade 0.7.

Link do Modelo do Alarme

4.3.2 Modo: Aprendizado

Este modo é parecido com o de Inferência, no entando é possível aprender as probabilidades dos fatos a partir dos dados. Para isso, o *ProbLog2* (versão atual do sistema) utiliza o método do *Estimador de Máxima Verossimilhança*[7]. Quando queremos que uma probabilidade seja aprendida podemos utilizar dois tipos de notação:

1. $\mathbf{t}(\_) :: \mathbf{f}$ deste modo sinalizamos que fato probabilístico $\mathbf{f}$ deve ser aprendido, e o *ProbLog* inicia o algoritmo do estimador de máxima verossimilhança com um valor inicial aleatório.

2. $\mathbf{t}(\mathbf{p}) :: \mathbf{f}$ indica ao *ProbLog* que deve iniciar o algoritmo do estimador de máxima verossimilhança com o valor inicial $\mathbf{p}$.

Além do programa que já estamos acostumados a utilizar composto por fatos e regras, quando utilizamos este modo de aprendizado devemos fornecer dados ao *ProbLog* no formato de evidências (do mesmo jeito que fizemos nos exemplos da seção 4.2). Isto posto, ao invés de fazermos *queries* queremos que o *ProbLog* aprenda a probabilidade dos fatos, então quando executarmos o programa, ele vai cálcular as probabilidades que queremos aprender.

Vamos exemplificar este modo de Aprendizado com o seguindo problema: Considere três pessoas sentadas uma ao lado da outra (pessoa 1, pessoa 2, pessoa 3). Temos as seguintes regras: Regra 1 - Se uma pessoa está com sono ela vai bocejar e Regra 2 - Uma pessoa boceja se o seu vizinho bocejou. Queremos que o *ProbLog* estime a probabilidade do fato sono dado que a 1ª pessoa bocejou e a 2ª não. Abaixo segue o algoritmo dessa situação, seguido de suas evidências:

Exemplo de um Programa do Modo Aprendizado:

```
%Probabilidade que queremos estimar
t(_)::sono(X) :- pessoa(X).

%Regra 1:
boceja(X) :- sono(X).
```

[7]É um método para estimar os parâmetros de um modelo estatístico

```
%Regra 2:
boceja(X) :- vizinho(X,Y), boceja(Y).

pessoa(1).
pessoa(2).
pessoa(3).

vizinho(1,2).
vizinho(2,3).
```

E no segundo campo onde colocamos as nossas evidências temos:

```
evidence(boceja(1),true).
evidence(boceja(2),false).
```

Ao executarmos o comando *Learn* (Aprender), é retornado o valor $\frac{1}{3}$, que é a probabilidade que satisfaz este modelo. Podemos explicar este resultado como: Dentre as três pessoas, sabemos que a primeira bocejou e a segunda não, ou seja, como não passamos a evidência que a terceira pessoa bocejou e ela não foi influenciada pela sua vizinha então $\frac{1}{3}$ das pessoas estão com sono (apenas a pessoa 1).

4.4 A Semântica do ProbLog

Como vimos na seção 4.1 sobre a sintaxe do *ProbLog*, cada fato probabilístico $\mathbf{p} :: \mathbf{f}$ tem uma *decisão atômica*, isto é, podemos incluir o fato $\mathbf{f}$ com probabilidade $\mathbf{p}$ ou descartá-lo com probabilidade $(\mathbf{1} - \mathbf{p})$. Esta decisão atômica tem uma semelhança com o conceito de realização dentro da teoria da probabilidade, no entanto, no *ProbLog* a probabilidade $\mathbf{p}$ é fixa dentro de um programa. Definimos uma *Escolha Total* (ET) como uma *decisão atômica* para cada um dos fatos do nosso programa. Matematicamente uma ET é um subconjunto de todos os conjuntos possíveis de fatos do nosso programa. Portanto, se temos n átomos probabilístico vamos ter 2^n *Escolhas Totais*, onde cada uma dessas ET vai ter uma distribuição

Link do programa do modo Aprendizado

de probabilidade própria. Vamos definir a probabilidade de uma ET como o produto de todas as probabilidades das decisões atômicas que a compõem. Note que tomamos o produto pois as decisões atômicas são independentes.

Na próxima subseção, vamos exemplificar a definição de *Escolha Total* com o cálculo da probabilidade de cada uma dessas ET's para o nosso programa do alarme visto na seção 4.3.1. Finalizamos este capítulo com a formalização matemática da definição de *Escolha Total*.

4.4.1 Escolha Total do Programa do Alarme

Na seção 4.3.1, modelamos o problema do alarme. O seu código implementado em *ProbLog* está descrito abaixo:

```
pessoa(paulo).
pessoa(maria).


0.1::assalto.
0.2::terremoto.

0.7::escuta_alarme(X) :- pessoa(X).

alarme :- assalto.
alarme :- terremoto.

liga(X) :- alarme, escuta_alarme(X).
```

Lembrando dos conceitos apresentados na seção 4.1 sobre a sintaxe, temos que para este programa os *átomos probabilísticos* são *(assalto, terremoto* e *escuta_alarme)*, sendo que para este último temos duas opções: tanto o Paulo quanto a Maria podem escutar o alarme, de modo que temos 4 *átomos probabilísticos*. Já os termos *(pessoa, alarme* e *liga)* são os *átomos derivados*.

Pela definição de *Escolha Total*, temos nesse programa $2^4 = 16$ ET's que correspondem aos 4 átomos probabilísticos existentes. Se quisermos calcular a probabilidade de ocorrer esses 4 eventos temos:

$$\mathbb{P}(assalto, terremoto, escuta\_alarme(paulo), escuta\_alarme(maria)) =$$

$$\mathbb{P}(assalto) \,.\, \mathbb{P}(terremoto) \,.\, \mathbb{P}(escuta\_alarme(paulo)) \,.\, \mathbb{P}(escuta\_alarme(maria)) =$$
$$= 0.1 \cdot 0.2 \cdot 0.7 \cdot 0.7 = 0.0098$$

Agora suponha que queremos calcular a probabilidade apenas dos 3 primeiros eventos, ou seja, o caso que escuta_alarme(maria) é falso, segue: $0.1 \cdot 0.2 \cdot 0.7 \cdot (1 - 0.7) = 0.0042$.

Se quisermos fazer essa *queries* no *ProbLog* executamos o código abaixo:

```
pergunta1 :- assalto, terremoto, escuta_alarme(paulo),
                                 escuta_alarme(maria).
query(pergunta1).

pergunta2 :-  assalto, terremoto, escuta_alarme(paulo),
                              not(escuta_alarme(maria)).
query(pergunta2).
```

Na tabela abaixo temos as 16 Escolhas Totais existentes do programa 1, com as suas respectivas probabilidades:

i	Escolha Total (ET_i)	$\mathbb{P}(ET_i)$
1	{assalto, terremoto, escuta_alarme(paulo), escuta_alarme(maria)}	0.0098
2	{assalto, terremoto, escuta_alarme(paulo)}	0.0042
3	{assalto, terremoto, escuta_alarme(maria)}	0.0042
4	{assalto, terremoto}	0.0018
5	{assalto, escuta_alarme(paulo), escuta_alarme(maria)}	0.0392
6	{assalto, escuta_alarme(paulo)}	0.0168
7	{assalto, escuta_alarme(maria)}	0.0168
8	{assalto}	0.0072
9	{terremoto, escuta_alarme(paulo), escuta_alarme(maria)}	0.0882
10	{terremoto, escuta_alarme(paulo)}	0.0378
11	{terremoto, escuta_alarme(maria)}	0.0378
12	{terremoto}	0.0162
13	{escuta_alarme(paulo), escuta_alarme(maria)}	0.3528
14	{escuta_alarme(paulo)}	0.1512
15	{escuta_alarme(maria)}	0.1512
16	{}	0.0648

4.4.2 Formalização da Escolha Total

Vamos considerar os seguintes conjuntos que compõem um programa do *ProbLog*:

- Um conjunto $\boldsymbol{R}$ de regras.
- Um conjunto $\boldsymbol{F}$ de fatos onde cada um deles esta expresso com a sua respectiva probabilidade $\boldsymbol{p}$.

Assumindo que todas as variáveis aleatórias do nosso programa são independentes, estamos interessados em calcular a distribuição de probabilidade para algum subconjunto F' tal que $F' \subseteq F$.

Assim $\mathbb{P}_F(F')$ representa a probabilidade de exatamente os fatos pertencentes a F' serem verdadeiros (acontecerem) dentro de um programa composto pelo conjunto total de fatos F Matematicamente:

$$\mathbb{P}_F(F') = \prod_{f_i \in F'} p_i \cdot \prod_{f_i \in F \setminus F'} (1 - p_i)$$

Vamos ilustrar como esta fórmula funciona utilizando o programa do alarme, onde queremos calcular a probabilidade da escolha total {assalto, escuta_alarme(paulo)} segue:

- $F' = \{assalto, escuta\_alarme(paulo)\}$
- $F = \{assalto, terremoto, escuta\_alarme(paulo), escuta\_alarme(maria)\}$

$$\mathbb{P}_F(F') = \underbrace{\mathbb{P}(assalto) \cdot \mathbb{P}(escuta\_alarme(paulo))}_{\mathbb{P}(\text{Eventos de } F' \text{ Acontecerem})} \cdot \underbrace{\mathbb{P}(terremoto) \cdot \mathbb{P}(escuta\_alarme(maria))}_{\mathbb{P}(\text{Eventos que Não Estão em } F' \text{ Não Acontecerem})}$$

$$\mathbb{P}_F(F') = 0.1 \cdot 0.7 \cdot (1 - 0.2) \cdot (1 - 0.7) = 0.0168.$$

Agora, vamos formalizar como o *ProbLog* calcula a probabilidade de uma *Query*.

Se fizermos a união da nossa Escolha Total F' com as regras R do nosso programa, isto é: $(F' \cup R)$ vamos obter um Programa Lógico. Adotando o conjunto $F' = \{assalto, escuta\_alarme(paulo)\}$ como a nossa Escolha Total, temos o seguinte Programa Lógico:

```
assalto.
escuta_alarme(paulo).

alarme :- assalto.
alarme :- terremoto.

liga(X) :- alarme, escuta_alarme(X).
```

Na literatura do *ProbLog* esta união é chamada de *Possible World* (referência [4]) (Mundo Possível). Neste trabalho, vou definir ω para expressar o nosso Mundo Possível. Agora já estamos aptos a definir a probabilidade de uma *query* (q) utilizando os conceitos apresentados até agora.

O nosso raciocínio é que a probabilidade de sucesso de uma *query* (q) é a soma de todas as probabilidades dos mundos possíveis (ω) em que a *query* ocorre. Em cada ω, (q) é independente e tem uma probabilidade distinta dos demais.

Equivalentemente, podemos considerar todos os nossos F' e em cada um deles observamos se existe um Mundo Possível que contém a nossa *query* (q) e, em caso positivo, somamos as suas respectivas probabilidades.

Definimos a probabilidade de sucesso de uma *query* como:

$$\mathbb{P}(q) \sum_{\substack{F' \subseteq F \\ \exists q \in (F' \cup R)}} \mathbb{P}_F(F') =$$

$$\sum_{\substack{F' \subseteq F \\ \exists q \in (F' \cup R)}} \prod_{f_i \in F'} p_i \cdot \prod_{f_i \in F \setminus F'} (1 - p_i).$$

Vamos mostrar como essa fórmula funciona fazendo o cálculo da probabilidade da *query* q = {assalto, escuta_alarme(paulo), escuta_alarme(maria)}. O conjunto F' contém os dois subconjuntos em que essa *query* é possível:

$$F' = \{assalto, escuta\_alarme(paulo), escuta\_alarme(maria)\} \cup$$

$$\{assalto, terremoto, escuta\_alarme(paulo), escuta\_alarme(maria)\}$$

A nossa *query* (q) não disse nada a respeito sobre se ocorreu ou não um terremoto, e por isso o nosso conjunto F' tem esses dois subconjuntos.

Agora vamos ao cálculo de $\mathbb{P}(q)$

$$\mathbb{P}(q) = \mathbb{P}_F(\{assalto, escuta\_alarme(paulo), escuta\_alarme(maria)\}) +$$

$$\mathbb{P}_F(\{assalto, terremoto, escuta\_alarme(paulo), escuta\_alarme(maria)\}) =$$

$$(0.1 \cdot 0.7 \cdot 0.7 \cdot (1 - 0.2)) + (0.1 \cdot 0.7 \cdot 0.7 \cdot 0.2) = 0.049.$$

A grande vantagem da semântica do *ProbLog* é que os cálculos das probabilidades das *queries* são feitos de uma maneira muito eficiente. Quando a partir de uma *query* consideramos para efeito de computação apenas os mundos possíveis em que ela pode ocorrer, não perdemos tempo considerando cenários logicamente falsos. Neste exemplo acima, calculamos apenas a probabilidade dod ω's em que a query (q) pode ocorrer(todos os outros 14 Estados Totais são automaticamente descartados).

Assim, o fato do *ProbLog* ter sido construido baseado na Programação Lógica trouxe esse beneficio na hora de responder perguntas. Dentro de um modelo não determinístico, conseguimos logicamente extrair todas as informações e variáveis que não tem significados e impactos para que o cálculo de probabilidade das nossas *queries* só consideramos os possíveis resultados com probabilidade maior que zero.

5. Construindo um Recomendador de Filmes em *ProbLog*

Toda vez que uma pessoa pede uma recomendação de algum filme, normalmente temos dois caminhos para responder essa pergunta: Se somos próximos a essa pessoa, como um amigo ou familiar, recomendamos filmes baseado nesse nosso conhecimento prévio que temos dela. E no caso em que não conhecemos essa pessoa é comum fazermos algumas dessas perguntas: Quais são os seus gêneros favoritos? Quais diretores que você mais gosta? Prefere um filme mais antigo ou mais novo? As respostas normalmente são do tipo: "Eu gosto muito de filmes de ação, de vez em quando assisto uma comédia, mas eu não suporto filmes românticos, só assisto se a minha namorada fizer muita questão", ou então, "Eu gostei desses filmes mais recentes do Tarantino[1] mas aqueles clássicos que eles dirigiu eu acho ainda mais legais!". Em ambos os casos, quando conhecemos bem ou não conhecemos a pessoa, recomendamos filmes baseado nesse conhecimento que já tínhamos ou aprendemos com a sua resposta a essas perguntas usuais. E por fim as nossas recomendações são usualmente expressas com uma expressão de incerteza, como, por exemplo; "Eu acho que você vai gostar bastante do filme (...), e talvez goste deste outro filme (...), mas com certeza você não vai gostar do filme (...)".

Queremos construir um recomendador de filmes em *ProbLog* que esteja baseado neste procedimento que já estamos acostumados a utilizar em nossas vidas. Neste capítulo, vamos criar as regras lógicas do mesmo modo que fazemos as perguntas sobre gostos cinematográficos para uma pessoa e utilizar conceitos de probabilidade para inferir quando um filme tem mais chance dessa pessoa gostar dentre os demais.

[1] Quentin Tarantino é um renomado diretor americano

Por ora, estamos interessados em desenvolver um protótipo desse recomendador para um conjunto pequeno de filmes. No próximo capítulo, vamos continuar o desenvolvimento para utilizar esse recomendador em dados reais sobre as preferências cinematográficas de uma usuária.

5.1 Luz, Câmera e Recomendação!

Vamos iniciar a nossa modelagem com a seguinte história:

"Um amigo do autor estava olhando a sua coleção de dvd's e pediu emprestado alguns dos seus filmes. Melhor do que simplesmente escolher filmes ao acaso, eu perguntei: Quais são os seus filmes favoritos? E ele respondeu os seus 6 filmes favoritos. Então, como um bom estudante de matemática que pensa muito antes de tomar qualquer decisão respondi: Ok! Espere que daqui 1 ano eu termino o meu TCC e te recomendo alguns filmes."

Esse vai ser o tema desta seção: a construção de um recomendador de filmes em *ProbLog* bem simples para um conjunto bem pequeno de filmes.

Os 6 filmes favoritos do meu amigo são:

- Filme: ET, Diretor: Steven Spielberg(SS), Gênero: Sci-Fi, Ano de Produção: 1982.
- Filme: Inteligência Artificial, Diretor: Steven Spielberg(SS), Gênero: Sci-Fi, Ano de Produção: 2001.
- Filme: Pulp Fiction, Diretor: Quentin Tarantino(QT), Gênero: Crime, Ano de Produção: 1994.
- Filme: Cães de Aluguel, Diretor: Quentin Tarantino(QT), Gênero: Ação, Ano de Produção: 1992.
- Filme: Batman Begins, Diretor: Christopher Nolan(CN), Gênero: Ação, Ano de Produção: 2005
- Filme: O Show de Truman, Diretor: Peter Weir(PW), Gênero: Comedia, Ano de Produção: 1998.

Vamos chamar esse conjunto de 6 filmes de $F6$ e supor que esses são os únicos 6 filmes que ele assistiu em sua vida.

Podemos nos perguntar agora como calcular a probabilidade deste usuário gostar de um filme do Steven Spielberg (SS) que ainda não assistiu dado esse conjunto $F6$ dos seus filmes favoritos. Definimos (*) como:

$$\mathbb{P}(SS|F6) \coloneqq \frac{\#(SS \in F6)}{\#F6} = \frac{\text{Quantidade de Filmes de SS que estão em } F6}{\text{Quantidade Total de filmes em } F6} = \frac{2}{6} = \frac{1}{3}$$

A chance dele gostar de um filme inédito do Steven Spielberg dado o conjunto dos seus filmes favoritos $F6$.

Antes de prosseguirmos com o trabalho queremos fazer um breve comentário do que está por fim nas próximas seções para amenizar eventuais choques que os estatísticos podem ter ao ver a nossa abordagem nada convencional (e um pouco informal) de probabilidade.

Queremos pedir licença aos estatísticos, pois estamos utilizando o conceito de probabilidade de um modo não muito rigoroso. Vamos utilizar o produto entre variáveis aleatórias mesmo elas não sendo independentes (a justificativa está abaixo da equação (5.4)). Já aproveitamos também para comentar que muito do que foi apresentado nos primeiros capítulos utilizou o conceito de probabilidade clássica de Laplace que, sem dúvida, já foi ultrapassado pela escola subjetivista. Como nesse trabalho queremos fazer um estudo do *ProbLog*, julgamos mais simples usar essa abordagem que a maioria das pessoas estão mais acostumadas. No entanto, daqui para frente vamos assumir uma abordagem mais próxima da escola subjetivista de probabilidade. Note que, estamos assumindo que a nossa definição (*) que diz a respeito sobre a probabilidade de uma pessoa gostar de um determinado filme com uma característica dado o conjunto dos seus filmes favoritos, como, *semelhante* a probabilidade de um filme ter uma determinada característica dado o conjunto dos seus filmes favoritos.

Caso o leitor tenha interesse em uma leitura introdutória sobre as escolas clássicas, frequentista e subjetivista de probabilidade recomendamos o capítulo 6 do livro: Decision Theory - an introduction to the mathematics of rationality do Simon French [7].

Generalizando para o caso de um diretor qualquer D que pertence ao conjunto F de filmes assistidos pelo usuário temos:

$$\boxed{\mathbb{P}(D|F) \coloneqq \frac{(\#D \in F)}{\#F} = \frac{\text{Quantidade de Filmes Dirigidos Por D em } F}{\text{Quantidade Total de filmes em } F}} \quad (5.1)$$

Vamos utilizar essa mesma definição para a inferência de Gênero de um filme, no caso do conjuntos $F6$:

$$\mathbb{P}(Crime|F6) \coloneqq \frac{\#(Crime \in F6)}{\#F6} = \frac{\text{Qnt de Filmes do Gênero Crime que estão em } F6}{\text{Quantidade Total de filmes em } F6}$$

$$\Rightarrow \mathbb{P}(Crime|F6) = \frac{1}{6}$$

Generalizando para o caso de um gênero qualquer G que pertence ao conjunto F de filmes assistidos pelo usuário temos:

$$\boxed{\mathbb{P}(G|F) \coloneqq \frac{(\#G \in F)}{\#F} = \frac{\text{Quantidade de Filmes do Gênero G em } F}{\text{Quantidade Total de filmes em } F}} \quad (5.2)$$

Já para a variável ano de produção criamos 3 intervalos possíveis:

- Antigos - filmes produzidos até 1979
- Clássicos - filmes produzidos de 1980 até 2000
- Novos - filmes produzidos a partir de 2001

Em que o cálculo da probabilidade de recomendação de um filme baseado no seu ano de produção é análogo aos do diretor e do gênero. No caso geral, para um filme da categoria *Novos*, que pertence ao conjunto F de filmes assistidos pelo usuário, temos:

$$\boxed{\mathbb{P}(Novos|F) \coloneqq \frac{(\#Novos \in F)}{\#F} = \frac{\text{Quantidade de Filmes Novos}F}{\text{Quantidade Total de filmes em } F}} \quad (5.3)$$

Agora que já definimos o cálculo das nossas 3 variáveis, Diretor, Gêneros e Ano de Produção vamos programá-las dentro do *ProbLog*:

```
%Modelando os fatos do Conjunto F6:

%Diretores:
2/6::diretor(ss); 2/6::diretor(qt);
1/6::diretor(cn); 1/6::diretor(pw).

%Gêneros:
2/6::genero(scifi); 1/6::genero(crime);
2/6::genero(acao); 1/6::genero(comedia).

%Ano:
0::ano(antigo); 4/6::ano(classico); 2/6::ano(novo).
```

Para o nosso primeiro modelo de recomendador, vamos definir que a probabilidade de um usuário gostar de um filme inédito f dado o conjunto dos seus filmes favoritos FF como:

$$\boxed{\mathbb{P}(f|FF) \coloneqq \mathbb{P}(Diretor\ de\ f|FF) \cdot \mathbb{P}(Genero\ de\ f|FF) \cdot \mathbb{P}(Ano\ de\ f|FF)} \tag{5.4}$$
$$= \frac{(\#Diretor\ de\ f \in FF)}{\#FF} \cdot \frac{(\#Genero\ de\ f \in FF)}{\#FF} \cdot \frac{(\#Ano\ de\ f \in FF)}{\#FF}$$

Isto é: A probabilidade do usuário gostar do filme f dado as suas preferências FF é o produto das probabilidades de quanto ele gosta do diretor, gênero e ano de produção de f dado FF. Escolhemos o operador produto entre essas três variáveis, mesmo elas não sendo independentes. Poderíamos ter escolhido o operador aditivo, mas pelas características de programação do *ProbLog* escolher o produto entre os termos é mais simples de programar.

Queremos enfatizar que o principal desta fórmula (5.4) não é o resultado numérico de cada filme e sim conseguirmos ordená-los de modo decrescente para criarmos um ranking de recomendação. Queremos ordenar os filmes inéditos do mais provável para o menos provável então não importa quais são exatamente os valores de cada um deles.

Agora que já definimos o critério de recomendação de um filme, falta apenas criar esta regra no *ProbLog* e, por fim, testa-lá para um conjunto de filmes, no

caso a coleção de dvd's do autor deste trabalho. Cada um desses filmes vai ser expresso como um *fato* - *filme* que é um *funcional de aridade 4* (como vimos na nomenclatura da programação lógica). Escrevendo como fatos do *ProbLog* os filmes são:

```
%Filmes da coleção de Dvd's do Autor:

%filme(título,diretor,gênero,ano).
filme(uma_guerra_muito_louca, ss, comedia, antigo).
filme(de_volta_para_o_futuro, ss, scifi, classico).

filme(kill_bill_1, qt, acao, novo).
filme(um_drink_no_inferno, qt, terror, classico).

filme(amnesia, cn, suspense, novo).
filme(o_cavaleiro_das_trevas, cn, acao, novo).

%gl = George Lucas
filme(star_wars, gl, scifi, classico).
```

Note que nesta lista de filmes surgiram 2 novos gêneros, terror e suspense. Como eles não estão no nosso conjunto $F6$, vamos atribuir probabilidade Zero para eles.

E, agora, passando a nossa regra de recomendação (5.4) de filmes para o *ProbLog*, temos:

```
%Regra
%Recomendador1:
rec(Titulo,Diretor,Genero,Ano):-  filme(Titulo,Diretor,Genero,Ano),
                                  diretor(Diretor),
                                  genero(Genero),
                                  ano(Ano).
```

Interpretamos essa regra como: Recomendamos um filme com os elementos (Título, Diretor, Gênero e Ano) se existir um filme com esses mesmos elementos. Vamos calcular a equação (5.4) para cada um deles e obter a probabilidade do indivíduo gostar dessa recomendação.

Finalmente, temos todos os elementos do nosso primeiro recomendador de filmes em *ProbLog*:

```
%Modelando os fatos do Conjunto F6:
%Diretores:
2/6::diretor(ss);2/6::diretor(qt);1/6::diretor(cn);1/6::diretor(pw).

%Gêneros:
2/6::genero(scifi); 1/6::genero(crime);
2/6::genero(acao); 1/6::genero(comedia).

%Ano:
0::ano(antigo); 4/6::ano(classico); 2/6::ano(novo).

%Filmes da coleção de Dvd's do Autor:
%filme(título,diretor,gênero,ano).
filme(uma_guerra_muito_louca, ss, comedia, antigo).
filme(de_volta_para_o_futuro, ss, scifi, classico).

filme(kill_bill_1, qt, acao, novo).
filme(um_drink_no_inferno, qt, terror, classico).

filme(amnesia, cn, suspense, novo).
filme(o_cavaleiro_das_trevas, cn, acao, novo).

filme(star_wars, gl, scifi, classico).

%Seguindo a nossa definição de recomendação devemos
%Atribuimos probabilidade 0 para os gêneros e diretor inédito:
0::terror. 0::suspense. 0::diretor(gl).

%Regra
%Recomendador1:
rec(Titulo,Diretor,Genero,Ano):-  filme(Titulo,Diretor,Genero,Ano),
                                  diretor(Diretor),
                                  genero(Genero),
                                  ano(Ano).

query(rec(Titulo,Diretor,Genero,Ano)).
```

Link do Programa

Abaixo, temos o resultado desta *query* ordenando pela probabilidade de recomendação de cada filme:

```
1) rec(de_volta_para_o_futuro,ss,scifi,classico) 0.074074074
2) rec(kill_bill_1,qt,acao,novo) 0.037037037
3) rec(o_cavaleiro_das_trevas,cn,acao,novo) 0.018518519

4) rec(uma_guerra_muito_louca,ss,comedia,antigo) 0
5) rec(um_drink_no_inferno,qt,terror,classico) 0
6) rec(amnesia,cn,suspense,novo) 0
7) rec(star_wars,gl,scifi,classico) 0
```

Observando o resultado acima, fica claro que este primeiro critério de recomendação que adotamos não é muito bom pois ele consegue ordenar apenas os filmes em que todos os fatos probabilísticos foram definidos (são maiores que zero). O filme (4) por exemplo, mesmo sendo de um dos diretores favoritos do amigo do autor e de um gênero que ele gosta, pelo simples fato de estar na categoria de filmes antigos recebeu probabilidade zero. O mesmo aconteceu com os filmes (5 e 6) que receberam probabilidade zero pois são de gêneros não avaliados (terror e suspense). E o filme (7) pelo diretor ser inédito.

Deste modo, finalizamos esta seção com a motivação de melhorar o nosso programa para que não exclua do ranking de recomendações aqueles filmes que tem alguma informação que não foi avaliada.

5.2 A Evolução do Recomendador

Nesta seção, queremos modificar o nosso programa para que ele consiga recomendar filmes nos casos em que as informações de diretor, gênero e ano de produção são inéditas. Nosso objetivo é que a situação descrita no final da seção anterior não aconteça mais. Para isso, vamos expandir a nossa fórmula de recomendação de filmes para cada uma das 3 informações vistas até agora.

5.2.1 Regra do Diretor

Na seção anterior, definimos que a probabilidade de recomendar um filme dirigido por um diretor é dada pela razão da quantidade dos filmes que assistimos e gosta-

mos desse diretor pela quantidade totais de filmes assistidos. Queremos criar agora um critério que também inclua diretores inéditos.

O nosso novo modo de calcular a probabilidade de recomendar um diretor D vai estar dividido em dois casos: Um para os diretores inéditos (aquele que não foi assistido nenhum filme) e outro para diretores conhecidos. Vamos chamar de F o conjunto de filmes já assistidos pelo usuário.

Vamos criar essas duas definições de um modo a garantir que um diretor inédito tenha sempre no máximo a metade da probabilidade de recomendação de um diretor já conhecido.

Probabilidade de Recomendação de um Diretor Inédito:

$$\mathbb{P}(D_{Inédito}|F) \coloneqq \frac{1}{\#F}$$

$$= \frac{1}{\text{Quantidade Total de Filmes Assistidos}}$$

Probabilidade de Recomendação de um Diretor D já Conhecido Como:

$$\mathbb{P}(D_{Conhecido}|F) \coloneqq \frac{(\#D \in F)}{\#F} + \mathbb{P}(D_{Inédito}|F)$$

$$= \frac{\text{Quantidade de Filmes Dirigidos Por D em } F}{\text{Quantidade Total de Filmes Assistidos}} + \mathbb{P}(D_{Inédito}|F)$$

Para passar os fatos probabilístico de diretor para o *ProbLog* do modo que esta nova definição funcione, devemos declarar os fatos **p::diretor(Nome do diretor)** com:

$$\boxed{p = \frac{(\#Diretor \in F)}{\#F + 1}} \tag{5.5}$$

para todos os diretores $\in F$.

Já para os inéditos declaramos o fato probabilístico: **p::diretor(inédito)** com:

$$\boxed{p = \frac{1}{\#F + 1}} \tag{5.6}$$

para todos os diretores $\notin F$.

Desta forma garantimos que a soma de todas as variáveis aleatórias diretor incluindo os inéditos dentro do *ProbLog* são iguais a 1. É imediato que: somando todos os caso de (5.5) teremos F elementos, e ao adicionarmos mais 1 elemento do caso (5.6) chegaremos na igualdade $\frac{\#F+1}{\#F+1} = 1$.

Abaixo, temos o código em *ProbLog* com os fatos dos diretores do conjunto $F6$ expressos dessa maneira, seguido pela nossa nova regra de diretor:

```
%Fatos Probabilísticos Seguindo a Definição Acima:
2/7::diretor(ss); 2/7::diretor(qt); 1/7::diretor(cn);
1/7::diretor(pw); 1/7::diretor(inedito).


%Caso temos um diretor Inédito,
%Apenas a segunda linha será computada
%Caso o diretor seja conhecido uma linha soma com a outra
%Do modo que definimos P(D|F)

regra_diretor(D) :- diretor(D).
regra_diretor(D) :- diretor(inedito).

%Testando essas queries temos os resultados esperados
query(regra_diretor(ss)).
query(regra_diretor(cn)).
query(regra_diretor(inedito)).
```

5.2.2 Regra dos Gêneros

Nesta seção, vamos desenvolver um modelo de recomendação para os gêneros inéditos, que ao contrário da variável diretor em que um diretor é conhecido ou não, existem diversos tipos de gêneros cinematográficos.

Por enquanto vamos considerar apenas os 6 gêneros vistos até agora: (scifi, ação, comédia, crime, terror e suspense), esses dois últimos o usuário não disse

Link para testar a regra diretor

nenhuma informação mas são os gêneros de duas possíveis recomendações de filmes. Alguns gêneros tem uma certa relação entre eles, por exemplo, scifi e filmes de suspense muita vezes estão relacionados, filmes de ação e crime também são comuns e as vezes temos até comédia com terror. Então esta vai ser a nossa motivação para construir uma regra a respeito dos gêneros: queremos conseguir recomendar filmes de gêneros ainda não avaliados e também considerar estas proximidade entre eles para deixar o nosso programa mais completo.

Começamos adaptando uma regra comum na literatura do *Prolog* que é a regra *Equals* (Iguais) para o *ProbLog*, supondo que queremos uma regra que diga que os filmes do gênero ação estão em 80% das vezes relacionados com os filmes do gênero crime; já os filmes de scifi estão relacionados com os filmes de suspense em 60% das vezes. Abaixo temos esse programa:

```
% Relação entre os filmes de (Ação e Crime) e (SciFi e Suspense)
0.8::equals(acao,crime).
0.6::equals(scifi,suspense)).

equals(X,X).
equals(X,Y) :- equals(Y,X).

query(equals(acao,crime)).
query(equals(suspense,scifi)).
```

A primeira *query* produz o resultado de 0.8, e a segunda de 0.6 como é esperado.

Para a nossa evolução do recomendador devemos integrar essa relação de *equals* no nosso programa. Deste modo essa variável aleatória gênero vai ter os 4 valores calculados do mesmo modo da seção anterior, e vamos também definir as relações de igualdade entre os gêneros. O filme "Um Drink no Inferno" que é predominantemente do gênero terror também pode ser considerado como um filme de ação. Já o filme "Amnesia" que é do gênero suspense também tem um pouco do gênero crime. essas informações foram retiradas do IMDB (*Internet Movie Database)*[2]. Assim a variável gênero do nosso novo recomendador será definida em duas partes: a primeira, para os gêneros já avaliados pelo usuário e a segunda para as relações de igualdade entre os gêneros.

[2] https://www.imdb.com/

```
% Relação entre os filmes de (Ação e Crime) e (SciFi e Suspense)
0.8::equals(acao,crime). 0.6::equals(scifi,suspense).

equals(X,X).
equals(X,Y) :- equals(Y,X).

query(equals(acao,crime)).
query(equals(suspense,scifi)).
```

Devemos modificar a equação (5.2) pois agora temos dois possíveis casos a serem considerados nesta recomendação: os gêneros conhecidos e os gêneros inéditos.

Probabilidade de Recomendação de um Gênero Inédito:

Neste nosso exemplo, queremos recomendar um filme do gênero terror pois ele tem uma similaridade com o gênero de ação.

Assim, $\mathbb{P}(G = Terror|F) = \mathbb{P}(Terror \sim Ação|F) \cdot \mathbb{P}(G = Ação|F) = 0.8 \cdot \frac{2}{6} = \frac{4}{15}$

Generalizando esta fórmula para o caso geral, de um gênero inédito $g_{inédito}$ dentro de um conjunto de filmes F temos:

$$\mathbb{P}(G_{inédito} = g_{inédito}|F) =$$

$$\sum_{i=1}^{n} \mathbb{P}(g_{inédito} \sim g^{(i)}_{equals}|F) \cdot \mathbb{P}(g^{(i)}|F)$$

Onde n é o número de gêneros próximos de $g_{inédito}$ que existem a relação *equals* e $\mathbb{P}(g^{(i)}|F)$ é calculado como (5.2), para cada $\mathbb{P}(G = g^{(i)}|F)$

Observe que esta fórmula abrange o caso em que um gênero pode ter mais de um *equals* com ele.

Probabilidade de Recomendação de um Gênero Conhecido:

No caso em que temos um gênero pertencente ao conjunto de filmes F, devemos acrescentar um pequeno fator na fórmula (5.2) para que ela também considere os *equals* desse gênero. No caso geral, temos:

$$\mathbb{P}(G_{conhecido} = g_{conhecido}|F) =$$

$$\underbrace{\mathbb{P}(G = g_{conhecido}|F)}_{\text{Mesma Definição de (5.2)}} + \underbrace{\mathbb{P}(G_{inédito} = g_{conhecido}|F)}_{\text{Definição Para } \mathbb{P}(\text{Gêneros Inéditos}) \text{ no caso } g_{conhecido}}$$

Note que o modo de calcular os fatos probabilístico para cada um dos gêneros se manteve igual ao da seção (5.2), apenas acrescentamos o zero para os gêneros não conhecidos: o que modificamos foi o modo de calcular a probabilidade de recomendação de um filme a partir de um gênero.

Implementando esta regra no *ProbLog* temos:

```
%Gêneros:
2/6::genero(scifi); 1/6::genero(crime);
2/6::genero(acao); 1/6::genero(comedia);
0::genero(terror); 0::genero(suspense).

%Relação de Igualdade entre os Gêneros
0.8::equals(acao,terror).
0.6::equals(crime,suspense).

equals(X,X).
equals(X,Y) :- equals(Y,X).

regra_genero(X,Y) :- equals(Y,X), genero(X).

query(regra_genero(X,Y)).
```

Fazendo as *queries* abaixo vamos ter os seguintes resultados:

```
%Exemplos de Queries

query(regra_genero(comedia,_)).
% 1) Resultado: regra_genero(comedia,comedia) = 1/6

query(regra_genero(acao,_)).
% 2) Resultados: regra_genero(acao,acao) = 2/6
```

```
% 3) Resultados: regra_genero(acao,terror) = 4/15


query(regra_genero(crime,_)).
% 4) Resultados: regra_genero(crime,crime) = 1/6
% 5) Resultados: regra_genero(crime,suspense) = 1/10
```

Para obtermos a probabilidade de recomendação de um filme como sugerimos acima, devemos somar o resultado das *queries* calculadas pelo *ProbLog*. Abaixo temos a análise dos resultados:

- Equivale a $\mathbb{P}(Comedia|F) = \frac{1}{6}$, pois o gênero comédia não tem nenhum *equals*, utilizamos a fórmula de Probabilidade de Recomendação de um Gênero Conhecido.
- De (2) e (3) temos que: $\mathbb{P}(Acao|F) = \frac{2}{6} + \frac{4}{15} = \frac{3}{5}$, utilizamos a fórmula de Probabilidade de Recomendação de um Gênero Conhecido.
- (3) Equivale a $\mathbb{P}(Terror|F) = \frac{4}{15} = \frac{2}{6} \cdot \underbrace{0.8}_{\text{0.8::equals(acao,terror)}}$ utilizamos a fórmula de Probabilidade de Recomendação de um Gênero Inédito.
- De (4) e (5) temos que: $\mathbb{P}(Crime|F) = \frac{1}{6} + \frac{1}{10} = \frac{4}{15}$, utilizamos a fórmula de Probabilidade de Recomendação de um Gênero Conhecido.
- (5) Equivale a $\mathbb{P}(Terror|F) = \frac{4}{15} = \frac{2}{6} \cdot \underbrace{0.8}_{\text{0.8::equals(acao,terror)}}$, utilizamos a fórmula de Probabilidade de Recomendação de um Gênero Inédito.

Depois da análise dos resultados, observamos que a nossa *regra_genero* funciona como esperado, sendo que o cálculo de probabilidade de recomendação de um gênero conhecido deve ser feito em n etapas, onde n é o número de fatos *equals* que este gênero participa. Já para os gêneros inéditos, este calculo é imediato.

5.2.3 Regra do Ano

A última regra que devemos adaptar é a sobre o ano de produção de um filme. Como vamos manter as 3 categorias possíveis sobre o ano de produção de um filme (antigo, clássico e novo), conseguimos fazer uma adaptação simples para o cálculo da probabilidade de recomendação de um filme a respeito do seu ano de produção dado um conjunto F de filmes assistidos. A nossa ideia vai ser a de somar um δ natural maior que zero em cada uma das 3 categorias de ano de produção um filme. Segue, a equação para o caso de um filme antigo (para os filmes clássicos e novos a fórmula é análoga ,bastando substituir pela categoria que deseja calcular):

$$\mathbb{P}(Ano = Antigo|F) = \frac{(\#FilmesAntigos \in F) + \delta}{\#F + (3 \cdot \delta)} =$$

$$\frac{\text{Quantidade de Filmes Antigos em } F + \delta}{\text{Quantidade Total de filmes em } F + (3 \cdot \delta)} \text{ , onde } \delta \in \mathbb{N}^*$$

Devemos demonstrar que tal definição esta sempre contida no intervalo $(0, 1)$, isto é, define uma probabilidade para qualquer valor de δ e qualquer conjunto F:

Demonstração:

Seja o conjunto F composto por A filmes antigos, C filmes clássicos e N filmes novos. Definimos $\#F$ como o total de elementos de F, ou seja, $\#F = A + C + N$. Vamos provar que:

$$\frac{A + \delta}{\#F + (3 \cdot \delta)} + \frac{C + \delta}{\#F + (3 \cdot \delta)} + \frac{N + \delta}{\#F + (3 \cdot \delta)} = 1$$

Note que a primeira parcela desta expressão é o mesmo que:

$$\frac{A + C + N + (3 \cdot \delta)}{\#F + (3 \cdot \delta)}$$

Pela definição de $\#F = A + C + N$ segue:

$$\frac{\#F + (3 \cdot \delta)}{\#F + (3 \cdot \delta)} = 1 \text{ , e concluimos a prova.}$$

Quanto menor o valor de δ, a probabilidade de uma categoria de anos vai depender mais fortemente dos filmes do conjunto F. Se quisermos que esse critério de recomendação de filmes pelo ano de produção não seja muito impactante, escolhemos um δ grande (por exemplo, 10 vezes a quantidade de filmes de F). Deste segundo modo, ambas as 3 categorias de filmes vão estar próximas de $\frac{1}{3}$ de serem recomendadas.

No nosso caso, vamos escolher o menor δ possível ($\delta = 1$) e F como sendo o nosso conjunto $F6$ segue:

- $\mathbb{P}(Antigo|F) = \dfrac{0+1}{6+(3\cdot 1)} = \dfrac{1}{9}$
- $\mathbb{P}(Classico|F) = \dfrac{4+1}{6+3} = \dfrac{5}{9}$
- $\mathbb{P}(Novo|F) = \dfrac{2+1}{6+3} = \dfrac{3}{9}$

Declarando esses fatos no *ProbLog* temos:

```
%Ano:
1/9::ano(antigo); 5/9::ano(classico); 3/9::ano(novo).
```

5.2.4 Testando esse Novo Recomendador

Agora que já definimos as novas equações de probabilidade para as categorias de diretor, gêneros e ano de produção, vamos programar um novo recomendador juntando todas elas, e testá-las para o mesmo conjunto dos 6 filmes. A regra *rec* tem o mesmo conceito apresentado em (5.4). Ela só foi modificada para funcionar com as *regra_diretor* e *regra_genero*.

```
%Diretores:
2/7::diretor(ss); 2/7::diretor(qt); 1/7::diretor(cn);
1/7::diretor(pw); 1/7::diretor(inedito).

%Gêneros:
2/6::genero(scifi); 1/6::genero(crime);
2/6::genero(acao); 1/6::genero(comedia);
0::genero(terror); 0::genero(suspense).

%Relação de Igualdade entre os Gêneros
0.8::equals(acao,terror).
0.6::equals(crime,suspense).

%Ano:
1/9::ano(antigo); 5/9::ano(classico); 3/9::ano(novo).

%Regra Diretor
regra_diretor(D) :- diretor(D).
regra_diretor(D) :- diretor(inedito).

%Regra Gêneros
equals(X,X).
equals(X,Y) :- equals(Y,X).
regra_genero(X,Y) :- equals(Y,X), genero(X).

%Filmes da coleção de Dud's do Autor:
%filme(título,diretor,gênero,ano).
filme(uma_guerra_muito_louca, ss, comedia, antigo).
filme(de_volta_para_o_futuro, ss, scifi, classico).

filme(kill_bill_1, qt, acao, novo).
filme(um_drink_no_inferno, qt, terror, classico).

filme(amnesia, cn, suspense, novo).
filme(o_cavaleiro_das_trevas, cn, acao, novo).

%gl = George Lucas
filme(star_wars, gl, scifi, classico).
```

```
%Regra
%Recomendador2:
rec(Titulo,Diretor,Genero,Gequals,Ano) :-
                                    filme(Titulo,Diretor,Gequals,Ano),
                                    regra_diretor(Diretor),
                                    regra_genero(Genero,Gequals),
                                    ano(Ano).

query(rec(_,_,_,_,_)).
```

Existem vários tipos possíveis de *queries* que podemos fazer. Se quisermos apenas que seja calculada a chance de recomendação de um filme do Quentin Tarantino, basta fazer "query(rec(_,qt,_,_,_))."; para selecionarmos apenas filmes novos,
"query(rec(_,_,_,_,novo))." e, assim, por diante.

Abaixo, temos uma tabela com os resultados deste programa de recomendação para a *query* mais genérica, onde ordenamos os filmes na formatação de um ranking:

Posição	Filme	Informações do Filme
1	De Volta Para o Futuro	Steven Spielberg, Scifi, Clássico
2	Um Drink no Inferno	Quentin Tarantino, Terror, Clássico
3	Kill Bill 1	Quentin Tarantino, Ação, Novo
4	O Cavaleiro das Trevas	Christopher Nolan, Ação, Novo
5	Star Wars	George Lucas, Scifi, Clássico
6	Amnesia	Christopher Nolan, Suspense, Novo
7	Uma Guerra Muito Louca	Steven Spielberg, Comédia, Antigo

Link deste Novo Recomendador

Análise dos Resultados:

Ao contrário do nosso primeiro programa que conseguiu recomendar apenas 3 dos 7 filmes possíveis, já vemos uma melhora nesse segundo recomendador.

A partir do conjunto $F6$ que são os 6 filmes favoritos do amigo do autor mais as nossas inferências, chegamos nos seguintes fatos probabilísticos a respeito dos diretores, gêneros e anos e produção:

```
%Diretores:
2/7::diretor(ss); 2/7::diretor(qt); 1/7::diretor(cn);
1/7::diretor(pw); 1/7::diretor(inedito).

%Gêneros:
2/6::genero(scifi); 1/6::genero(crime);
2/6::genero(acao); 1/6::genero(comedia);
0::genero(terror); 0::genero(suspense).

%Relação de Igualdade entre os Gêneros
0.8::equals(acao,terror).
0.6::equals(crime,suspense).

%Ano:
1/9::ano(antigo); 5/9::ano(classico); 3/9::ano(novo).
```

Assim faz bastante sentido que a 1º Posição do nosso ranking de recomendação seja um filme do Spielberg, seu diretor favorito, do seu gênero favorito (Scifi) e ano de produção (clássico) também favorito.

Já na 2º e 3º posições temos dois filmes do Tarantino que é também um dos diretores que ele mais gosta. O filme "Um Drink no Inferno" é de terror mas graças a nossa regra *equals* classificamos este filme com bastante proximidade do gênero ação e como é do período clássico de produção ele ficou na frente de "Kill Bill 1" que perdeu a 2º posição devido ao amigo do autor não gostar muito de filmes da categoria novos.

É interessante observar que o filme "Star Wars", 5º posição do nosso ranking, mesmo sendo de um diretor inédito ficou na frente de filmes de diretores já conhecidos, pois é do gênero (Scifi) e período (clássico) favoritos do amigo do autor.

Deste modo, ficamos felizes com o potencial que o *ProbLog* tem para a construção de um ranking de recomendação de filmes, baseado em conhecimentos prévios sobre o gosto cinematográfico de uma pessoa. Na próxima seção, vamos criar duas regras lógicas que vão deixar o nosso recomendador com mais recursos.

5.3 Regras Lógicas de Recomendação

Nas seções anteriores, trabalhamos com fatos e regras probabilísticas. O enfoque desta seção é mostrar como agregar regras lógicas ao nosso recomendador. Vamos criar uma regra que diz a respeito a faixa etária e também a duração dos filmes. Deste modo, quando um usuário for buscar um filme ele vai ter uma gama maior de alternativas para filtrar quais são as possíveis possibilidades de recomendações que ele pode receber. Se estiver com uma criança, pode querer receber apenas sugestões de filmes infantis e de menor duração, por exemplo.

Regra Faixa Etária

Queremos criar uma regra que diga a respeito da faixa etária de um filme. Vamos definir que existam três categorias de filmes:

- Infantil, sem restrição de idade
- Juvenil, acima de 13 anos (inclui exatamente 13)
- Adultos, acima de 17 anos (inclui exatamente 17)

Exemplificando, uma pessoa de 17 anos pode assistir qualquer filme. Já uma pessoa com menos de 17 só pode assistir os filmes das categorias Juvenil e Infantil, e por fim uma criança de 12 anos só pode assistir aos filmes da categoria Infantil.

Para conseguirmos testar a implementação dessa regra, nós adicionamos mais alguns filmes a lista de dvd's da seção anterior e também o campo faixa etária foi adicionado em cada um desses filmes.

Abaixo temos a implementação dessa regra, em que fornecemos a idade de um possível usuário e o programa nos retorna quais filmes este usuário pode assistir seguindo as 3 restrições de faixa etária que criamos:

```
%Listas de possíveis recomendações de Filmes

%filme(tituto, diretor, genero, ano de produção, faixa etaria).
filme(et, ss, comedia, classico, infantil).
filme(shrek, aa, animacao, classico, infantil).
filme(de_volta_para_o_futuro, ss, scifi, classico, infantil).

filme(uma_guerra_muito_louca, ss, comedia, antigo, juvenil).
filme(o_cavaleiro_das_trevas, cn, acao, novo, juvenil).
filme(star_wars, gl, scifi, classico, juvenil).

filme(kill_bill_1, qt, acao, novo, adulto).
filme(um_drink_no_inferno, qt, terror, classico, adulto).
filme(amnesia, cn, suspense, novo, adulto).


%Regra da Faixa Etária
regra_idade(I,T) :- I < 13, filme(T,_,_,_,infantil).

regra_idade(I,T) :- (I >= 13, I<17, filme(T,_,_,_,infantil));
                    (I >= 13, I<17, filme(T,_,_,_,juvenil)).


regra_idade(I,T) :- (I >= 17, filme(T,_,_,_,infantil));
                    (I >= 17, filme(T,_,_,_,juvenil));
                    (I >= 17, filme(T,_,_,_,adulto)).


query(regra_idade(12,T)).
%Resultado:
regra_idade(12,de_volta_para_o_futuro) =  1
regra_idade(12,et) = 1
regra_idade(12,shrek) = 1

query(regra_idade(18,T)).
%Resultado: São todos os filmes.
```

Link da Implementação da Regra da Faixa Etária

Regra Duração de um Filme

Agora vamos criar uma regra que diz a respeito ao tempo de duração de um filme. Assim, vamos adotar que um filme pode ser curto, médio ou longo de acordo com os intervalos abaixo:

- Curto, menos de 112 minutos de duração
- Juvenil, de 112 a 118 minutos de duração
- Adultos, acima de 118 minutos de duração

Escolhemos categorizar a duração de um filme em intervalos qualitativos pois é o modo que estamos mais acostumados no dia a dia quando buscamos por um filme. Acreditamos que é mais intuitivo buscar uma recomendação de um filme dessa maneira do que forçar um usuário a buscar por intervalos quantitativos (numéricos).

Em cada um dos nossos 9 filmes, adicionamos a sua categoria de duração, assim o *fato - filme* é da forma: *filme(tituto, diretor, genero, ano de produção, faixa etaria, duração).*

Vamos mostrar dois modos de construir essa regra: O primeiro modo é que ela selecione apenas os filmes exatamente da categoria desejada. Transcrevendo essa regra para o *ProbLog* temos:

```
%Regra1 da Duração
regra_duracao1(I,T) :- I == curto, filme(T,_,_,_,_,curto).

regra_duracao1(I,T) :- I == medio, filme(T,_,_,_,_,medio).

regra_duracao1(I,T) :- I == longo, filme(T,_,_,_,_,longo).
```

Já o segundo modo de construir essa regra é que quando uma pessoa busca por filmes de média duração ela também recebe os de curta, e quando busca por filmes de longa duração também recebe os de média.

```
%Regra2 da Duração
regra_duracao2(I,T) :- I == curto, filme(T,_,_,_,_,curto).
```

```
regra_duracao2(I,T) :- I == medio, filme(T,_,_,_,_,medio);
                                    regra_duracao2(curto,T).

regra_duracao2(I,T) :- I == longo, filme(T,_,_,_,_,longo);
                                    regra_duracao2(medio,T).
```

Nessa segunda regra da duração, não alteramos nada em relação aos filmes curtos. No entanto, para os filmes de média duração a regra consiste em duas parte que estão separadas pelo operador ; (que é um *ou* inclusivo) na primeira parte selecionamos exatamente os filmes curtos e após o operador ; fazemos uma recursão onde pedimos os filmes que respeitam a *regra_duração2* para os filmes curtos. Utilizamos o mesmo procedimento para a construção dessa regra para os filmes longos.

Agora vamos implementar essa *regra_duracao*2 no nosso programa final. E mostrar exemplos de possíveis *queries* que podemos fazer. Fizemos uma pequena alteração na regra final de recomendação para que ela inclua as categorias de faixas etárias e também de duração dos filmes.

```
%Diretores:
2/7::diretor(ss); 2/7::diretor(qt); 1/7::diretor(cn);
1/7::diretor(pw); 1/7::diretor(inedito).

%Gêneros:
2/6::genero(scifi); 1/6::genero(crime);
2/6::genero(acao); 1/6::genero(comedia);
0::genero(terror); 0::genero(suspense).

%Relação de Igualdade entre os Gêneros
0.8::equals(acao,terror).
0.6::equals(crime,suspense).

%Ano:
1/9::ano(antigo); 5/9::ano(classico); 3/9::ano(novo).

%Regra Diretor
regra_diretor(D) :- diretor(D).
regra_diretor(D) :- diretor(inedito).
```

```
%Regra Gêneros
equals(X,X).
equals(X,Y) :- equals(Y,X).
regra_genero(X,Y) :- equals(Y,X), genero(X).

%Regra2 da Duração
regra_duracao2(I,T) :-  I == curto, filme(T,_,_,_,_,curto).

regra_duracao2(I,T) :- I == medio, filme(T,_,_,_,_,medio);
                       regra_duracao2(curto,T).

regra_duracao2(I,T) :- I == longo, filme(T,_,_,_,_,longo);
                       regra_duracao2(medio,T).

%Filmes da coleção de Dvd's do Autor:
%Listas de possíveis recomendações de Filmes
%filme(tituto,diretor,genero,ano de produção,faixa etaria,duração).
filme(et, ss, comedia, classico, infantil, medio).
filme(shrek, aa, comedia, classico, infantil, curto).
filme(de_volta_para_o_futuro, ss, scifi, classico, infantil, medio).

filme(uma_guerra_muito_louca, ss, comedia, antigo, juvenil, medio).
filme(o_cavaleiro_das_trevas, cn, acao, novo, juvenil, longo).
filme(star_wars, gl, scifi, classico, juvenil, longo).

filme(kill_bill_1, qt, acao, novo, adulto, curto).
filme(um_drink_no_inferno, qt, terror, classico, adulto, curto).
filme(amnesia, cn, suspense, novo, adulto, medio).


%Regra Recomendador3:
rec(Titulo,Diretor,Genero,Gequals,Ano,Idade,Duracao) :-
                    filme(Titulo,Diretor,Gequals,Ano,Idade,Duracao),
                    regra_diretor(Diretor),
                    regra_genero(Genero,Gequals),
                    ano(Ano), regra_duracao2(Duracao,Titulo).

%Query Mais Genérica,
query(rec(_,_,_,_,_,_,_)).
```

Sugestões das possíveis *queries* que podemos fazer ao nosso recomendador são:

```
%Recomendar apenas os filmes do Steven Spielberg
%query(rec(_,ss,_,_,_,_,_)).

%Recomendar apenas filmes do gênero Ação
%os gêneros próximos de Ação no caso
%terror também aparecem por causa da regra equals
%query(rec(_,_,acao,_,_,_,_)).

%Recomendar filmes por um Periodo de Produção
%antigo, classico, novo
%query(rec(_,_,_,_,novo,_,_)).

%Recomendar filmes por uma Faixa Etária
%infantil, juvenil, adulto
%query(rec(_,_,_,_,_,adulto,_)).

%Recomendar filmes pela sua duração máxima
%curto, medio, longo
%mudar :- filme(Titulo,Diretor,Gequals,Ano,Faixa_Etaria,_)
%query(rec(_,_,_,_,_,_,medio)).

%Recomendar filmes do gênero SciFi e de
%Media ou Longa duracao
%query(rec(_,_,scifi,_,_,_,longo)).

%Recomendar filmes adultos
%do periodo clássico
%query(rec(_,_,_,_,_,adulto,_)).
```

Com elas, conseguimos selecionar quais características queremos que uma possível recomendação de um filme tenha. Note também, que ao construirmos o nosso recomendador com esta *regra 3* existem diversas permutações de buscas que podemos realizar, resultado em uma gama muito ampla de *queries* possíveis.

Esta última versão do recomendador de filmes em *ProbLog* que contém as nossas regras e variáveis probabilísticas em conjunto com as regras lógica parece ter um

Link do Recomendador com a nova Regra Lógica

bom potencial. Seu protótipo realizado nessa escala reduzida de filmes funcionou bem, e estamos satisfeitos com os resultados.

No próximo capítulo, vamos utilizar os conceitos e estudos realizados até agora para a construção de um recomendador de filmes em *ProbLog* em escala real, isto é, para um conjunto muito maior de filmes.

6. Um Estudo de Caso

No capítulo anterior, sugerimos um possível recomendador de filmes desenvolvido utilizando o *ProbLog*. Como o nosso objetivo naquele momento era apenas o de testar possíveis regras e inferências, não nos preocupamos com uma aplicação real deste recomendador. Agora, neste capítulo, vamos fazer um estudo de caso mais abrangente sobre este tema. Nossos dados de preferência cinematográficas vão ser provenientes de um questionário de 300 filmes. Veremos que as informações sobre o gênero de um filme e a sua classificação indicativa não são tão exatas e precisas como supomos no capítulo anterior, mas utilizando algumas ideias da lógica de predicados conseguimos algumas soluções criativas para os problemas encontrados.

Uma justificativa deste estudo e desenvolvimento é que até a data de elaboração deste trabalho não existe nenhuma outra fonte ou referência sobre como usar o *ProbLog* com dados provenientes de um banco de dados e questionário reais. Um ponto importante e até então inédito é o desenvolvimento de algoritmos na linguagem de programação *R* que consigam transformar os dados do questionário realizado pela nossa usuária em *fatos* na formatação aceita pelo *ProbLog*.

O recurso que o *ProbLog* tem que está mais próximo para processamento de dados são as estruturas de listas, que são inviáveis para um banco de dados muito grande e também não é possível fazer as inferências desejadas. Deste modo, escolhemos a linguagem de programação *R* para esta tarefa pois ela é de código aberto e, nos últimos anos, tem tido muito destaque para tarefas que envolvam bancos de dados e análises estatísticas.

Resumidamente, o que queremos construir aqui é um programa na linguagem de programação *R* que receba o resultado de um questionário sobre o gosto cinematográfico de uma usuária e crie, a partir dessas informações aprendidas, um

recomendador de filmes em *ProbLog* do mesmo tipo que nos desenvolvemos no capítulo anterior. Queremos também, que todo esse processo seja o mais automatizado possível e de fácil ajuste para que ele possa ser facilmente adaptado para outros tipos de banco de dados.

6.1 Construindo um Questionário

Queremos nesse capítulo que o nosso recomendador seja desenvolvido a partir de dados reais. Assim, achamos válido construir um banco de dados proveniente de um questionário respondido por uma pessoa que o autor conhece bem. A análise final dos resultados vai ser mais fácil de interpretar visto que conhecemos a entrevistada e também teremos a avaliação e opinião desta usuária sobre o ranking de recomendação de filmes que foi gerado para ela baseado no seu questionário.

Os filmes selecionados para a realização deste questionário e posteriormente recomendação foram os 300 primeiros filmes do ranking geral do IMDB (*Internet Movie Database),*[1] que é, na atualidade o maior website de catalogação e rankings de filmes. Note que o ranking não é estático. Especialmente os filmes em posições mais baixas sofrem diversas alterações de posição ao longo do ano. A pesquisa foi realizada no mês de setembro de 2018.

A usuária tinha três opções de resposta para um filme: (1) Se gostou do filme, (0) Se não gostou do filme, e ( ) para os filmes que não viu.

A tabela abaixo mostra o resultado desta pesquisa para 4 filmes:

Posição do Ranking	**Nome**	**Classificação**	**Interpretação**
159	Deadpool	0	Não Gostou
160	Kill Bill: Vol 1	1	Gostou
161	Solaris	1	Gostou
162	Prisoners		Não Assistiu

Em resumo, o resultado foi: Apenas um único filme teve avaliação negativa, 113 filmes tiveram avaliações positivas e 186 filmes ainda não foram assistidos (futuras possíveis recomendações). No anexo B, temos os top 300 filmes do IMDB e o resultado completo desta pesquisa.

[1] https://www.imdb.com/

6.2 Seleção de Informações

Como observamos na seção 6.1, o resultado da pesquisa está disposto em uma tabela onde as três primeira colunas nos interessam. O enfoque desta seção é transformar os dados de todos os filmes que a usuária não viu (e tem idade para ver) em *fatos* na formatação aceita pelo *ProbLog*. Posteriormente, esses filmes não vistos vão ser possíveis sugestões de recomendação para a usuária.

A pesquisa realizada com os 300 filmes continha apenas o nome deles, mas agora estamos interessados em obter o máximo de informações sobre eles. Para isso utilizamos a tabela *top300.csv* que tem 14 informações para cada filme: -Título, -Nota do IMDB, -Gênero, -Classificação Indicativa, -Idioma, -País de Produção, -Duração, -Diretor, -Ator Principal, -Ator Coadjuvante, -Terceiro Ator, -Colorido ou Preto e Branco e -Palavras Chave do Filme. Para este trabalho achamos interessante e suficiente utilizar 8 destas informações para construir os fatos no *ProbLog*. Elas estão descritas na tabela abaixo para o filme "Batman: O Cavaleiro das Trevas" *The Dark Knight*:

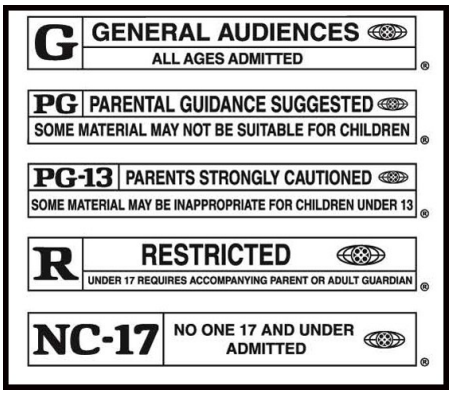
movie_title	**genres**	**content_rating**
The Dark Knight	Action Crime Drama Thriller	PG-13
language	**country**	**tittle_year**
English	USA	2008
duration	**director_name**	
152	Christopher Nolan	

Em português, essas 8 informações são: Título do Filme, Gênero, Classificação Indicativa, Idioma, País de Produção, Ano de Produção, Duração e Nome do Diretor.

Assim sendo, o enfoque desta seção é construir um algoritmo na linguagem de programação *R* que transforma esses dados da tabela de filmes em *fatos* na formatação aceita pelo *ProbLog*. Lembrando que o nosso critério de recomendação de um filme é: ser inédito e o usuário estar dentro da classificação indicativa deste filme. Nas duas próximas sub seções, veremos que são necessários alguns ajustes e modificações para os campos de classificação indicativa e gêneros.

6.2.1 Classificando as Classificações Indicativas

Vamos considerar que para recomendar um filme, ele tem que ser inédito para este usuário e também satisfazer a classificação indicativa. Para este trabalho, foi utilizada a classificação indicativa da MPAA (Motion Picture Association of America), pois a maioria dos filmes do IMDB tem essa classificação e ela é uma das mais usadas no mundo. Mas mesmo assim, ela não é quantitativamente clara como observamos na figura a seguir:



O problema desta classificação é que não está claro quando uma criança (ou adolescente) faz parte da categoria G ou PG. Para solucionar este impasse foi criada a seguinte tabela com intervalos de idades para cada uma das quatro categorias da MPAA.

MPAA	Categoria	Idades (em Anos)
G	1	0 - 7
PG	2	8 - 12
PG-13	3	13 - 16
R	4	17 - +

Na nossa pesquisa, nenhum dos filmes é NC-17. Então não consideramos esta categoria.

6.2.2 Famílias de Gêneros

Observando o campo *genres* (gêneros) do filme "Batman: O Cavaleiro das Trevas" notamos que ele é composto por 4 categorias: *Action, Crime, Drama, Thriller.*

A noção de gênero que estamos acostumados no nosso dia a dia não se aplica nesta tabela, pois um filme é composto por vários desses gêneros que estamos habituados.

Se compararmos com a continuação desta trilogia o filme "Batman: O Cavaleiro das Trevas Parte 2", o campo gênero deste segundo filme é composto por 6 categorias: *Action, Animation, Crime, Sci-Fi, Thriller* (foi tirado o gênero *Drama* e acrescentado *Animation e Sci-Fi* em relação ao primeiro filme). Se analisarmos os demais filmes que tem continuações, trilogias e sagas, esse fenômeno também acontece. Como todos os filmes tem alguns gêneros que os definem, precisamos criar um critério mais geral que englobe todos eles. Então, com essas justificativas criamos a chamada Família de Gêneros, que são 15 grandes categorias de gêneros, tentando aproximar as categorias que estamos mais acostumados de utilizar.

Nos nossos 300 filmes utilizados para este trabalho existem, 22 palavras possíveis de gêneros e se fossemos separar cada filme pelos gêneros já definidos pelo IMDB, existem 159 destas "famílias de gêneros". Por isso, foi necessária a criação desse conceito. Os dois critérios para a criação destas Famílias de Gêneros são: Sagas e continuações de filmes devem estar na mesma Família e também que seja possível buscar e recomendar filmes baseado na preferência de um usuário por cada uma dessas 15 categorias.

Na tabela abaixo, temos essas categorias e como elas foram definidas de modo que os nossos dois critérios estejam satisfeitos. Como criamos uma função implementada em R que faz essa categorização dos filmes, foram utilizamos os conectivos lógicos "e" $\wedge$, "ou" $\vee$ para isso.

Note que o "ou" no sentido lógico é diferente da conjunção semântica que estamos acostumados na nossa linguagem. Em português, "ou" é ambíguo, pois depende do contexto da sentença ele pode ser exclusivo ou inclusivo. No nosso caso, sempre interprete as regras com o "ou" ($\vee$) lógico como exclusivo e inclusivo.

Família de Gêneros	**Regra Lógica**
1	Action e (Biography ou Crime ou Drama ou History ou Thriller ou War)
2	Action e (Adventure ou Animation ou Sci-Fi ou Fantasy ou Outros)
3	(Adventure e Animation) ou Animation ou Fantasy
4	Adventure exclusivo ou com outros gêneros restantes
5	Biography e (Outros)
6	Comedy e (Outros)
7	Crime e Drama
8	Crime e Outros Distintos de 7
9	Documentary
10	Documentary mais outro gênero
11	Drama
12	Drama e (Family ou Fantasy ou Music ou Musical ou Romance)
13	Drama e (Horror ou Mistery ou Thriller ou Sci-Fi ou War)
14	Horror ou Mistery ou (Sci-Fi e Thriller)
15	Drama ou Western

Para exemplificar o conceito de família de gêneros, observamos a tabela com o nome do filme, os gêneros segundo o *IMDB* e em qual família de gêneros este filme faz parte:

Título do Filme	**Gêneros - pelo *IMDB***	**Família de Gêneros**
The Godfather	Crime - Drama	7
The Dark Knight	Action - Crime - Drama - Thriller	1
The Godfather: Part II	Crime - Drama	7
The Lord of the Rings 3	Action - Adventure - Drama	2
Fight Club	Drama	11
Forrest Gump	Comedy - Drama	6
Whiplash	Drama - Music	12
Apocalypse Now	Drama - War	13

6.3 Algoritmos em R

Nesta seção, vamos mostrar os 4 algoritmos na linguagem de programação *R* que foram desenvolvidos com os objetivos de: (1) importar os 300 filmes do IMDB, (2) importar o questionário realizado pela usuária, (3) realizar as inferências onde calculamos os três mesmos parâmetros utilizados no capítulo anterior (diretor,

gênero e ano, que são os nossos fatos probabilísticos) e, por fim, (4) gerar os *fatos* na formatação aceita pelo *ProbLog*.

Todos os algoritmos estão acompanhados de uma descrição detalhada do seu funcionamento e, em seguida, o seu código implementado no *R*, de modo que mesmo um leitor não familiarizado com esta linguagem de programação consiga entende-los.

6.3.1 Importar os Filmes e Famílias de Gêneros

Esse primeiro algoritmo é responsável por importar os 300 filmes do IMDB que estão em um arquivo chamado *top300.csv* e categorizá-los cada um desses filmes na sua família de gênero adequada. Ao final, teremos um *data frame* - *d* em que cada um dos filmes no campo *genres* vai estar com uma numeração de 1 a 15 relativa a sua família de gênero apropriada, e teremos também 4 vetores relacionados a classificação indicativa de cada filme. Abaixo, temos o código comentado para que o leitor consiga reproduzir em sua maquina e testar passo a passo se desejado.

```
#Executar o comando: getwd() para ver o diretório atual
#Mudar ele para o mesmo em que se encontra o arquivo top300.csv
#getwd() pega o diretório, setwd("/home/Alterar/Desktop/Pasta")'

setwd("/home/lucas/Desktop/R Code 2")

#Importamos o Arquivo em um data frame d = d[Linha,Coluna]"
d = read.csv("top300.csv", stringsAsFactors=FALSE)

#Tiro caracteres especiais como ç, é, ó do meu data frame
#Pois o ProbLog não suporta esses caracteres'
#Convertemos de utf-8 para ascii
iconv(d, "UTF-8", "ASCII//TRANSLIT")

#Crio um vetor com todos os Gêneros
generos = c("Action","Adventure","Animation","Biography",
"Comedy","Crime","Documentary","Drama","Family","Fantasy",
"History","Horror","Mystery","Music","Musical","News",
"Romance","Sci-Fi","Sport","Thriller","War","Western")

#Coloco os Generos de cada um dos filmes no vetor g'
```

```
g = d[,"genres"]

#Transformamos cada um dos Generos no seu Números Correspondente
# e tiramos os simbolos |
#Action = 1, Comedy = 5, etc
for (i in (1:22)){
  g=gsub(generos[i], i, g, fixed = TRUE)
}
#14al = Musical = 15 genero
g=gsub("14al",15,g)
g=gsub("[|]", " ", g)

#Função que Classifica cada filme na sua Família de Gêneros
#Apropriada conforme a definição apresentada na seção (6.2.2)
F = function(k){
  if (k[1]==1){
    if ((k[2]==4 | k[2]==6 | k[2]==8 | k[2]==11 | k[2]==20 |
           k[2]==22 | length(k)==1 )){return(1)
    } else {return(2)}
  }
  if ((k[1]==2 & k[2]==3) | k[1]==3 | k[1]==10){return(3)}
  if (k[1]==2){return(4)}
  if (k[1]==4){return(5)}
  if (k[1]==5){return(6)}
  if(k[1]==6){
    if(k[2]==8){return(7)
    } else {return(8)}
  }
  if(k[1]==7){
    if(length(k)==1){return(9)
    } else {return(10)}
  }
  if (k[1]==8){
    if (length(k)==1){return(11)}
    if (k[2]==9 | k[2]==10 | k[2]==14 | k[2]==15){return(12)
    } else {return(13)}
  }
  if ((k[1]==12 | k[1]==13 | k[1]==20 | k[1]==18)){return(14)}
  if ((k[1]==8 & k[2]==22) | (k[1]==22)){return(15)}
}
```

```
#Loop que substitui cada um dos filmes pela
#sua família de gênero apropriada
k=0
for (i in (1:300)){
  k = unlist(strsplit(g[i], split=" "))
  k = strtoi(k, base=0)
  d[i,"genres"]=F(k)
  'cat(i, F(k),"\n")'
}

$4 Vetores com os filmes ID (1:300) de cada
#filme por classificação indicativa'
g=which(d$content_rating=="G")
pg=which(d$content_rating=="PG")
pg13=which(d$content_rating=="PG-13")
r=which(d$content_rating=="R")
```

Abaixo, temos exemplos de buscas no *data frame d* com os seus respectivos resultados:

```
#Para mostrar os 5 primeiros elementos da
#primeira linha do data frame d:
d[1,1:5]

'Resultado':
movie_title           imdb_score genres content_rating language
The Shawshank Redemption 9.3        7      R            English

#Para mostrar a quantidade de filmes por cada Família de Gêneros
table(as.numeric(d[,3]))
'Resultado':
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
17 44 18 29 34 26 26  3  4  9 21  8 51  8  2

#Para mostrar todos os filmes da Família de Gênero "15"
subset.data.frame(d, genres == "15", select=1)
'Resultado':
            movie_title
8  The Good the Bad and the Ugly
```

```
30  Once Upon a Time in the West

#Para mostrar todos os filmes da classificação indicativa G = Livre
subset.data.frame(d, content_rating == "G", select=1)
'Resultado (3 primeiras linhas)':
ID       movie_title
32       Modern Times
39       The Lion King
55       WALL·E
...       ...

#Para mostrar todas as ID's dos filmes de classificação PG
pg
'Resultado':
9  14  18  22  27  31  33  44 (...)
```

6.3.2 Importando o Resultado do Questionário

Nessa segunda etapa, vamos importar o resultado do questionário realizado pela entrevistada (Veronica) e, a partir da sua classificação indicativa, vamos criar um vetor com todos os filmes que ela não viu e tem idade para ver. O *output* é o vetor *filmesnviu* com os ID's desses filmes (futuras recomendações).

```
#Importo o arquivo RankVeronica.csv no data frame user
user = read.csv("RankVeronica.csv", stringsAsFactors=FALSE)

#Tiro os caracteres especiais ç, é, ó etc problog não suporta
iconv(user, "UTF-8", "ASCII//TRANSLIT")

#Excluimos os filmes com avaliação (0) que não gostou
x = which(user[,2]==0)
user=user[-c(x),]
d=d[-c(x),]

#Salvo em avaliacoes os filmes que avaliou como (1) que gostou
#Eles ficam em uma matriz, na.omit tira os NA = filmes não viu
avaliacoes=na.omit(user)
avaliacoes=rbind(as.numeric(rownames(avaliacoes)),avaliacoes[,2])
```

```
assign("avaliacoes",avaliacoes,envir = .GlobalEnv)

#Função que recebe os filmes que viu "avaliacoes" +
#a categoria de classificação indicativa conforme a seção (6.2.1)
#Uma pessoa da categoria pg13, pode ver filmes pg13, pg e g
#Já uma pessoa da categoria pg, pode ver apenas os filmes pg e g.
#Retorna um vetor com os ID's dos filmes que não viu e pode ver

Filmesnaoviu = function(avaliacoes,Catidade){
  if(Catidade == 4){
    filmesnviu = setdiff((1:300),avaliacoes)
    return(filmesnviu)
  }
  if(Catidade == 3){
    filmesnviu = union(g,pg)
    filmesnviu = union(filmesnviu,pg13)
    filmesnviu = setdiff(filmesnviu,avaliacoes)
    return(filmesnviu)
  }
  if(Catidade == 2){
    filmesnviu= union(g,pg)
    filmesnviu = setdiff(filmesnviu,avaliacoes)
    return(filmesnviu)
  }
  if(Catidade == 1){
    filmesnviu = g
    filmesnviu = setdiff(filmesnviu,avaliacoes)
    return(filmesnviu)
  }
}

#Rodamos a função com os parametros avaliacoes
#E CatIdade = 4 (A Veronica tem mais de 18 anos)

'Veronica = CatIdade = 4'
filmesnviu=Filmesnaoviu(avaliacoes,4)

assign("filmesnviu",filmesnviu,envir = .GlobalEnv)
```

6.3.3 Inferências

Agora, estamos interessados em aprender os gostos cinematográficos da nossa entrevistada e ajustar os três parâmetros referentes aos fatos probabilísticos de gostar de cada diretor, gêneros e ano, do mesmo modo que fizemos na seção (5.2). Para isso utilizaremos os dados que estão no vetor *avaliações*. Decidimos que vamos descartar os filmes que tiveram avaliações negativas (0), pois, no caso do questionário realizado pela nossa entrevistada, apenas um único filme teve avaliação negativa, dentre 113 avaliações positivas e 186 possíveis recomendações..

Inferência Sobre Parâmetros dos Diretores Conhecidos:

Esta função é a implementação da equação (5.5). Ao executarmos a função *InfDiretores()*, criamos um vetor com o nome de todos os diretores com os seus respectivos fatos probabilísticos conforme mostrado na equação (5.5). Por exemplo: 0.04444::diretor(Steven Spielberg). O número 0.04444 foi calculado nesta função.

```
InfDiretores = function(){
  diretores = (d[(avaliacoes[1,]),"director_name"])
  #salva os nomes de todos os diretores avaliados
  #e tira os duplicados
  nomes = unique(diretores)

  qntfilmesdiretor = 0*c(1:length(nomes))
  for (i in 1:length(nomes)){
    qntfilmesdiretor[i] =  length(which(diretores==nomes[i]))
  }
  #Cria uma matriz com os nomes e as
  #quantidade de filmes de cada diretor
  qntfilmesdiretor = rbind(nomes,qntfilmesdiretor)

  #Total de filmes avaliados
  total_filmes_assistidos = length(avaliacoes[1,])

  #Calculos dos parâmetros da equação (5.5)
  for (i in 1:total_diretores){
    qntfilmesdiretor[2,i]=as.numeric((qntfilmesdiretor[2,i]))/
                                        (total_filmes_assistidos+1)
  }
```

```
  assign("qntfilmesdiretor",qntfilmesdiretor,envir = .GlobalEnv)
}
```

Ao somarmos esse vetor com a probabilidade de recomendação de um filme inédito teremos um número muito próximo (mas diferente) de 1, devido aos erros de arredondamentos cometidos pelo *R*. Quando formos imprimir esses números para o *ProbLog*, vamos arredondar todos ele em seis casas decimais e subtrair um pequeno ϵ de todos os diretores para garantir que não ultrapassemos a soma total de 1, pois, se isso acontecer, o *ProbLog* não vai funcionar.

Verifique que de fato estamos muito próximos de 1

```
sum(as.numeric(qntfilmesdiretor[2,]))+1/total_filmes_assistidos
```

Inferência sobre as Famílias de Gêneros

Nesta função, implementamos a equação (5.2). Assim teremos um vetor que contém os fatos probabilísticos de cada uma das 15 famílias de gênero.

```
#O vetor ProbFamilia[1:15] contém os fatos probabilísticos
#de cada uma das 15 famílias de gênero.
#E zero para aquelas que não tem nenhum filme avaliado
InfFamiliasGeneros = function(){
  todos = as.integer(d[(avaliacoes[1,]),"genres"])
  qnttodos=length(todos)
  ProbFamilia = 0*(1:15)
  for (i in (1:15)){
    x=which(todos==i)
    ProbFamilia[i]=length(x)/qnttodos
  }
  ProbFamilia[is.na(ProbFamilia)] = 0
  #Para não correrrmos o risco de a soma passar de 1:
    for (i in (1:15)){
      if (ProbFamilia[i]>0){
        ProbFamilia[i] = ProbFamilia[i] - 0.001
      }
    }
  ProbFamilia
}
```

Rodando esta função temos:

```
InfFamiliasGeneros()
0.02172727 0.27172727 0.12400000 0.12400000 0.10127273
0.04445455 0.06718182 0.01036364 0.01036364 0.00000000
0.03309091 0.03309091 0.10127273 0.04445455 0.00000000
```

Do mesmo modo que definimos o fato probabilístico $2/6 :: genero(Acao)$, essa função efetua o cálculo deste número. No caso, a família de gêneros 1 tem 0.02172727.

Inferência sobre os Anos

A função abaixo é a implementação da equação da seção (5.2.3) para o cálculo dos fatos probabilísticos de cada uma das 3 categorias de ano. Devemos fornecer o argumento δ que é um número natural maior que zero como *input* da nossa função. O *output* é o vetor *infano* que contém as 3 probabilidade de recomendação para cada uma das categorias de ano: novo (a partir de 2001), clássico (entre 1980 e 2000) e antigo (anterior a 1980).

```
InfAnos = function(delta){
 if (delta<1){
  cat("Atenção! O Delta deve ser um natural maior que zero!")
  break
  }
 #Pega as informações dos anos a partir da avaliação do usuário
 anosuser = as.integer(d[(avaliacoes[1,]),"title_year"])
 totalanosuser = length(anosuser)
 #Equação da seção (5.2.3)
 novos=(length(which(anosuser>=2001))+delta)/(totalanosuser+3*delta)
 antigos=(length(which(anosuser<=1979))+delta)/(totalanosuser+3*delta)

 infano = c(novos,1-(novos+antigos),antigos)
 infano
}
```

Executando essa função com diferentes valores de δ, observamos a convergência esperada para $\frac{1}{3}$ a medida que aumentamos o δ (e damos menos importância para esta inferência).

```
> InfAnos(1)
[1] 0.6153846 0.2527473 0.1318681
> InfAnos(10)
[1] 0.5508475 0.2711864 0.1779661
> InfAnos(100)
[1] 0.3994845 0.3144330 0.2860825
> InfAnos(300)
[1] 0.3593117 0.3259109 0.3147773
```

Assim, encerramos esta seção onde apresentamos os três algoritmos de inferência construídos no *R* para o cálculo dos fatos probabilísticos sobre os diretores, famílias de gêneros e ano de produção de um filme.

6.3.4 Exportando os Fatos para o ProbLog

Nesta última subseção, vamos primeiro mostrar como exportar esses três fatos probabilístico calculados anteriormente. E os *equals* entre as famílias de gêneros. Finalmente na próxima função, vamos incluir todos os predicados dos filmes que podem ser recomendados (filmes inéditos e que tem idade para ver).

No nosso *data frame d*, os nomes dos diretores são expressos como o nome e o sobrenome deste diretor. O *ProbLog* não aceita átomos que sejam separados por espaços e também que eles contenham qualquer letra maiúscula ou caracteres especiais. Desta forma, substituímos todos os espaços por um "_" assim Francis Ford Coppola vira francis_ford_coppola.

Imprimindo os diretores, famílias de gêneros, anos e *equals*

```
#Função Para Imprimir o fatos dos Diretores
ProbLog_Diretor = function(){
  InfDiretores()

  qntfilmesdiretor[2,]=round(as.numeric(qntfilmesdiretor[2,]),
                                                    digits=6)

  for (i in 1:length(qntfilmesdiretor[1,])){
    cat(paste0(qntfilmesdiretor[2,i],"::diretor(",gsub("[.]", "",
              gsub(" ", "_", tolower(paste(qntfilmesdiretor[1,i],
                                           collapse=","))))),"); "))
```

```
    if (i%%3==0){cat("\n")}
  }
  P_dir_inedito = 1/(total_filmes_assistidos+1)
  P_dir_inedito=round(P_dir_inedito, digits=6) - 0.0001
  cat(paste0(P_dir_inedito,"::diretor(inedito)."))
}

#Função para imprimir as Famílias de Gêneros
ProbLog_FamiliaGeneros = function(){
  pf = InfFamilias()
  pf = round(pf, digits=6)
  for (i in 1:14){
    cat(pf[i],"::genero(fg",i,"); ",sep="")
    if(i%%3==0){cat("\n")}
  }
  i=i+1
  cat(pf[15],"::genero(fg",i,"). ",sep="")
}

#Função para Imprimir os fatos probabilisticos dos Anos
ProbLog_Ano = function(delta){
    infano = InfAnos(delta)
    cat(infano[1],"::ano(novo); ",sep="")
    cat(infano[2],"::ano(classico); ",sep="")
    cat(infano[3],"::ano(antigo). ",sep="")
}

#Adotamos os seguintes valores para a relação de igualdade
# (Equals) entre as 15 possíveis famílias de gêneros
ProbLog_Equals = function(){
  cat("0.8::equals(fg1,fg2). 0.4::equals(fg1,fg3).
    0.6::equals(fg2,fg3). 0.6::equals(f3,f4).\n")
  cat("0.6::equals(fg9,fg10). 0.4::equals(fg9,fg5).
  0.6::equals(fg10,fg5). 0.6::equals(fg7,fg8).\n")
  cat("0.4::equals(fg7,fg11). 0.6::equals(fg13,fg14).
  0.4::equals(fg7,fg15). 0.6::equals(fg7,fg8).\n")
  cat("0.6::equals(fg3,fg6). 0.4::equals(fg6,fg12).")
}

}
```

Executando as seguintes chamadas das funções, temos:

```
cat("\n % Fatos Probabilíticos para as 15 Familias de Generos: \n")
ProbLog_FamiliaGeneros()
cat("\n\n % Fatos Probabilíticos para os Equals: \n")
ProbLog_Equals()
cat("\n\n % Fatos Probabilíticos para as 3 categorias de Anos: \n")
ProbLog_Ano()
cat("\n\n % Fatos Probabilíticos para os Diretores: \n")
ProbLog_Diretor()
```

Temos os seguintes resultados:

```
% Fatos Probabilíticos para as 15 Familias de Generos:
0.034091::genero(fg1); 0.238636::genero(fg2); 0.136364::genero(fg3);
0.125::genero(fg4); 0.113636::genero(fg5); 0.045455::genero(fg6);
'Restante dos resultados foram omitidos'

% Fatos Probabilíticos para os Equals entre as Familias de Generos
0.8::equals(fg1,fg2). 0.4::equals(fg1,fg3). 0.6::equals(fg2,fg3).
0.6::equals(fg3,fg4). 0.6::equals(fg9,fg10). 0.4::equals(fg9,fg5).
'Restante dos resultados foram omitidos'

 % Fatos Probabilíticos para as 3 categorias de Anos
0.6153846::ano(novo); 0.2527473::ano(classico); 0.1318681::ano(antigo).

 % Fatos Probabilíticos para os diretores:
0.022372::diretor(francis_ford_coppola);
0.033608::diretor(christopher_nolan);
0.044844::diretor(steven_spielberg);
0.033608::diretor(quentin_tarantino);
'Restante dos resultados foram omitidos'
% último deles:
0.011136::diretor(inedito).
```

Agora, falta apenas gerar os predicados de cada um dos filmes na formatação aceita pelo *ProbLog* e incluir em cada um deles os fatos lógicos e probabilísticos que vamos utilizar no nosso recomendador.

Os elementos de um predicado *filme* são:
filme(Título, Família_de_Gênero, Diretor, ano(Novo,Clássico,Antigo), ano(Numérico), duração(Curto,Médio,Longo), duração(Numérica), Pais de produção, Idioma, Classificação Indicativa(1,2,3,4), Classificação Indicativa(G,PG,PG13,R). Por exemplo o filme Gladiador: filme(gladiator,fg1,ridley_scott,classico,2000,longo,171,usa,english,4,r).

Adicionamos os elementos Pais de produção e Idioma, pois na nossa nova versão do recomendador, vamos implementar mais essas duas variáveis lógicas. Assim a busca por um filme ideal vai ser ainda mais completa.

Imprimindo os predicados Filmes

```
ProbLog_Filmes = function(){
  #Vamos passar para o data frame d2
  #apenas as informações relevantes
  d2 = d[filmesnviu,]
  d2 = d2[c(1,3,9,7,8,6,5,4)]
  d2$epoca = "epoca"
  d2$duracao = "duracao"
  d2$classind = "classind"
  d2 = d2[c(1,2,3,9,4,10,5,6,7,11,8)]
#Adicionamos o fg antes do número da familia do genero
  d2[,2]=sub("^", "fg", d2[,2])

  for (i in 1:length(d2[,1])){
#Adicionamos os anos por categoria
    if(d2[i,5]>=2001){d2[i,4]="novo"
      } else if (d2[i,5]<=1979){d2[i,4]="antigo"
      } else {d2[i,4]="classico"}

#Adicionamos as durações por categoria
    if(d2[i,7]>=141){d2[i,6]="longo"
      } else if(d2[i,7]<114){d2[i,6]="curto"
        } else {d2[i,6]="medio"}

#Adicionamos as classificações Indicativas por categoria
    if(d2[i,11]=="R"){d2[i,10]="4"
      } else if(d2[i,11]=="PG-13"){d2[i,10]="3"
        } else if (d2[i,11]=="PG"){d2[i,10]="2"
          } else {d2[i,10]="1"}
```

```
#Tiramos e ou substituimos os caracteres especiais
 cat(paste0("\n filme(", gsub("[.]", "", gsub(":", "",gsub("'", "",
   gsub("&", "", gsub("." , "_", gsub("-", "", gsub(" ","_",
   tolower(paste(d2[i,], sep="", collapse = ","))))))))),")."))
  }
}
```

Chamando essa função e imprimindo apenas três possíveis recomendações de filmes, temos:

```
cat("\n\n % Lista filmes que são Possíveis Recomendações \n")
ProbLog_Filmes()

'Alguns Resultados:'

% Lista filmes que são Possíveis Recomendações
 filme(se7en,fg7,david_fincher,classico,1995,medio,127,
                                                usa,english,4,r).
filme(oldboy,fg13,chanwook_park,novo,2003,medio,120,
                                            south_korea,korean,4,r).
filme(snatch,fg6,guy_ritchie,classico,2000,curto,104,
                                                 uk,english,4,r).
```

6.4 Versão Final do Recomendador

Agora que já construímos os nossos algoritmos no R que conseguem gerar os fatos que vão ser utilizados no *ProbLog* a partir do questionário respondido pela nossa entrevistada, devemos fazer apenas mais algumas modificações no recomendador do capítulo anterior. Deste modo, teremos um recomendador final em que um usuário consiga buscar filmes de muitas maneiras e combinações possíveis.

Vamos acrescentar uma regra para as classificações indicativas e utilizaremos a *regra2* sobre o tempo de duração de um filme. Além destas, também será possível restringir as sugestões de recomendação para filmes produzidos em um determinado país ou então pelo idioma falado no filme. Não iremos utilizar a regra *equals* pois apenas duas categorias de famílias de gêneros não foram assistidas e também para que os resultados finais já estejam em um formato de ranking. Se o resultado não for satisfatório, iremos implementar também essa relação de igualdade entre os gêneros.

Regra Classificação Indicativa

Existem duas buscas possíveis referentes a classificação indicativa de um filme que um usuário a procura de recomendações pode estar interessado .

A primeira é uma busca exata onde apenas são sugeridos filmes da categoria da classificação indicativa procurada. Um exemplo desta busca exata seria uma mãe que for colocar um filme para o seu filho assistir provavelmente só vai querer que filmes da categoria G (livres) sejam oferecidos para esta criança. Uma outra situação seria um adulto que só quer ver filmes com emoção/cenas fortes e quer sugestões apenas da categoria R (17+).

Para o caso em que um indivíduo queira mais recomendações criamos a busca agregada, que sugere todos os filmes que este indivíduo tem idade para ver. Um adolescente de 16 anos pode ver os filmes de todas as categorias com exceção da R.

Abaixo, temos a implementação dessas duas regras. Para uma busca exata, utilizamos a regra: *class_ind_unica(X,T)* que seleciona apenas os filmes da classificação indicativa exata procurada, e temos 3 regras para as buscas agregadas para as 3 categorias de idade (não faz sentido uma busca agregada para a categoria G pois ela coincide com a busca exata).

```
%Busca Exata
class_ind_unica(X,T) :-
                      (X==1, filme(T,_,_,_,_,_,_,_,_,X,_));
                      (X==2, filme(T,_,_,_,_,_,_,_,_,X,_));
                      (X==3, filme(T,_,_,_,_,_,_,_,_,X,_));
                      (X==4, filme(T,_,_,_,_,_,_,_,_,X,_)).

%Busca Agregada
pg_mais(X,T) :- class_ind_unica(2,T); class_ind_unica(1,T).
pg13_mais(X,T) :- class_ind_unica(3,T); class_ind_unica(2,T);
                  class_ind_unica(1,T).
r_mais(X,T) :- class_ind_unica(4,T); class_ind_unica(3,T);
               class_ind_unica(2,T); class_ind_unica(1,T).
```

Note que para a busca agregada utilizamos recursões sobre a busca exata, assim conseguimos ter uma sintaxe mais reduzida.

Agora vamos para a versão final do nosso recomendador de filmes em que todos os *fatos* para o *ProbLog* são gerados de maneira automática a partir dos resultados do questionário da nossa entrevistada. Após termos executados todos os algoritmos da seção (2.3) utilizamos o código abaixo para imprimir os *fatos* sobre os diretores, famílias de gêneros, ano de produção e por fim uma lista com todos os possíveis filmes inéditos que podem ser recomendados.

```
ProbLog_PrintTotalFinal = function(){
  cat("\n % Fatos Probabilíticos para as 15 Familias de Generos: \n")
  ProbLog_FamiliaGeneros()
  cat("\n\n % Fatos Probabilíticos para as 3 categorias de Anos: \n")
  ProbLog_Ano(1)
  cat("\n\n % Fatos Probabilíticos para os diretores: \n")
  ProbLog_Diretor()
  cat("\n\n % Lista filmes que são Possíveis Recomendações \n")
  ProbLog_Filmes()
}

#Executando a Função ProbLog_PrintTotalFinal()
#Serão impressos as 3 categorias de fatos probabilísticos
#E também os predicados filmes para recomendação
```

Como estamos lidando com um volume muito grande de informações e inferências, não é possível rodar a versão final do nosso recomendador no compilador online disponível. Deste modo, instalamos a biblioteca do *ProbLog* implementada na linguagem *Python* (2 ou 3). Nos sistemas operacionais Linux, execute os seguintes comandos no terminal:

```
pip install problog
pip install problog --upgrade
```

Caso encontre algum problema siga o tutorial no site: https://problog.readthedocs.io/en/latest/install.html.

Na versão final do nosso recomendador utilizamos a: *regra_diretor(D)*, *regra_duracao2(I,T)* (pode ser trocada pela regra_duracao1 se quiser) e para a classificação indicativa utilizamos a de busca exata *class_ind_unica(X,T)* (facilmente trocada por qualquer uma das de busca agregada). Assim, temos:

```
%Regra Diretor
regra_diretor(D) :- diretor(D).
regra_diretor(D) :- diretor(inedito).

%Regra2 da Duração
regra_duracao2(I,T) :- I == curto, filme(T,_,_,_,_,curto,_,_,_,_,_).
regra_duracao2(I,T) :- I == medio, filme(T,_,_,_,_,medio,_,_,_,_,_);
                                        regra_duracao2(curto,T).

regra_duracao2(I,T) :- I == longo, filme(T,_,_,_,_,longo,_,_,_,_,_);
                                        regra_duracao2(medio,T).

%Regra Busca Exata Classificação Indicativa
class_ind_unica(X,T) :-
                      (X==1, filme(T,_,_,_,_,_,_,_,_,X,_));
                      (X==2, filme(T,_,_,_,_,_,_,_,_,X,_));
                      (X==3, filme(T,_,_,_,_,_,_,_,_,X,_));
                      (X==4, filme(T,_,_,_,_,_,_,_,_,X,_)).

%Podemos substituir por qualquer uma dessas
```

Link do Recomendador Final em uma versão reduzida - Alguns dos Fatos - com Todas as Regras

```
%pg_mais(X,T) :- class_ind_unica(2,T); class_ind_unica(1,T).
%pg13_mais(X,T) :- class_ind_unica(3,T); class_ind_unica(2,T);
                   %class_ind_unica(1,T).
%r_mais(X,T) :- class_ind_unica(4,T); class_ind_unica(3,T);
                %class_ind_unica(2,T); class_ind_unica(1,T).

%Regra do Recomendador
rec(Titulo,Fgen,Diretor,CatAno,NumAno,CatDuracao,NumDuracao,
    Pais,Lingua,NumClasInd,ClasInd) :-

    filme(Titulo,Fgen,Diretor,CatAno,NumAno,CatDuracao,NumDuracao,
                                  Pais,Lingua,NumClasInd,ClasInd),
    regra_diretor(Diretor),
    genero(Fgen),
    ano(CatAno),
    regra_duracao2(CatDuracao,Titulo),
    class_ind_unica(NumClasInd,Titulo).

%Query Mais Genérica
query(rec(_,_,_,_,_,_,_,_,_,_,_,_)).
```

No final desta página, consta um link com uma versão reduzida do recomendador, isto é, tiramos alguns dos vários fatos probabilísticos dos diretores e dos filmes para que o leitor possa ter um noção do programa em funcionamento.

Para a versão final do recomendador, vamos criar um programa no formato *.pl* que é o formato dos programas do *ProbLog*. Basta copiar o que foi impresso ao executarmos a função *ProbLog_PrintTotalFinal* em um bloco de notas (ou em qualquer editor de textos simples de sua preferência) e, embaixo desses fatos, copiar o conjunto de regras acima. Salvamos este arquivo como *recfinal.pl* e para rodarmos esse programa devemos, pelo terminal do linux, ir até o diretório desse arquivo *.pl* e executar o comando *problog recfinal.pl*. No próprio terminal vai ser exibido o resultado da nossa *query* mais genérica que são a probabilidade de recomendação de cada um dos filmes.

Nessa versão final do nosso recomendador, é possível fazer *queries* especificando 7 elementos diferentes: $query(rec(\_, FamiliadeGenero, Diretor, CategoriaAno, \_, CategoriaDuracao, \_, Pais, Lingua, ClassificacaoIndicativa, \_))$. Cada um deles pode ser ajustados individualmente ou em qualquer permutação desejada entre

eles. Deste modo, vamos ter um número absurdamente grande de diferentes tipos de *queries* que podemos fazer visando selecionar características e elementos específicos dos filmes que queremos que sejam recomendados.

6.5 Análise dos Resultados

Na parte dos anexos deste trabalho, temos tanto o questionário preenchido pela nossa entrevistada como também o ranking de recomendação em sua totalidade. Nesta seção, vamos primeiro apresentar os principais resultados do questionário e, em seguida, vamos fazer a análise teórica do nosso ranking de recomendação e, por fim, apresentamos este ranking para a nossa entrevistada que deu a sua opinião a respeito.

Análise Teórica

Abaixo temos os 3 fatos probabilísticos, família de gêneros, ano de produção e diretores, que foram calculados pelos nossos algoritmos da seção Inferências. Ordenamos os filmes de modo decrescente segundo as suas probabilidades para ficar mais claros as preferências da nossa entrevistada.

```
%Fatos Probabilíticos para as 15 Familias de Generos:
%Em ordem descrescente:
0.271727::genero(fg2); 0.124::genero(fg3); 0.124::genero(fg4);
0.101273::genero(fg5); 0.101273::genero(fg13); 0.067182::genero(fg7);
0.044455::genero(fg6); 0.044455::genero(fg14); 0.033091::genero(fg11);
0.033091::genero(fg12); 0.021727::genero(fg1); 0.010364::genero(fg8);
0.010364::genero(fg9); 0::genero(fg10); 0::genero(fg15).

% Fatos Probabilíticos para as 3 categorias de Anos:
0.5934066::ano(novo); 0.2417582::ano(classico); 0.1648352::ano(antigo).

% Fatos Probabilíticos para os Maiores Diretores:
0.05518::diretor(peter_jackson); 0.043944::diretor(steven_spielberg);
0.032708::diretor(christopher_nolan);
0.032708::diretor(quentin_tarantino);
0.032708::diretor(martin_scorsese);
0.021472::diretor(francis_ford_coppola);
0.021472::diretor(joss_whedon)
```

```
0.021472::diretor(ridley_scott)
%Inéditos:
0.010236::diretor(inedito).
```

Pelas nossas definições de probabilidade de recomendação, é esperado que os filmes das primeiras famílias de gêneros apresentadas, da categoria de filmes novos e de alguns desses diretores estejam no topo do ranking. E é exatamente isso que acontece.

Abaixo temos os top 6 filmes do nosso ranking de recomendação:

Ranking - Filme

```
1 - rec(batman_begins,fg2,christopher_nolan,novo,...)
2 - rec(serenity,fg2,joss_whedon,novo,...)
3 - rec(raiders_of_the_lost_ark,fg2,steven_spielberg,classico,...)
4 - rec(the_prestige,fg13,christopher_nolan,novo,...)
5 - rec(the_martian,fg4,ridley_scott,novo,...)
6 - rec(the_departed,fg7,martin_scorsese,novo,...)
```

Todos os filmes são de diretores conhecidos, dos gêneros favoritos 5 de 6 e são do período de ano (novo) que também é o preferencial da entrevistada.

Da sétima recomendação até a decima segunda, temos apenas diretores inéditos, mas os gêneros ainda estão entre os favoritos e o período de ano (novo) também.

```
7 - rec(batman_the_dark_knight_returns_part_2,fg2,jay_oliva,novo,...)
8 - rec(godzilla_resurgence,fg2,hideaki_anno,novo,...)
9 - rec(the_bourne_ultimatum,fg2,paul_greengrass,novo,...)
10 - rec(baahubali_the_beginning,fg2,ss_rajamouli,novo,...)
11 - rec(cinderella_man,fg5,ron_howard,novo,...)
12 - rec(incendies,fg13,denis_villeneuve,novo,...)
```

O décimo terceiro filme é de um período de ano diferente de (novo). A partir desse, a variável ano começa a variar seguindo razoavelmente bem a proporção deste fato probabilístico.

```
13 - rec(jaws,fg4,steven_spielberg,antigo,...)
```

Observando os restantes dos filmes, notamos que o fato família de gênero e a regra do diretor parecem estar funcionando bem. A proporção destas variáveis segue o que foi aprendido. No entanto, achamos que as inferências sobre o período de produção de um filme estão um pouco diferente das proporções aprendidas a partir do questionário. Para resolver isso basta escolhermos um $\delta > 1$. Abaixo temos os top 13 com um $\delta = 100$:

```
%Para um delta = 100 temos:
0.3943299::ano(novo); 0.3118557::ano(classico);
0.2938144::ano(antigo).

1 - rec(batman_begins,fg2,christopher_nolan,novo,...)
2 - rec(raiders_of_the_lost_ark,fg2,steven_spielberg,classico,...)
3 - rec(serenity,fg2,joss_whedon,novo,...)
4 - rec(jaws,fg4,steven_spielberg,antigo,...)
5 - rec(the_prestige,fg13,christopher_nolan,novo,...)
6 - rec(the_martian,fg4,ridley_scott,novo,...)
7 - rec(casino,fg5,martin_scorsese,classico,...)
8 - rec(goodfellas,fg5,martin_scorsese,classico,...)
9 - rec(raging_bull,fg5,martin_scorsese,classico,...)
10 - rec(the_departed,fg7,martin_scorsese,novo,...)
11 - rec(the_hateful_eight,fg7,quentin_tarantino,novo,...)
12 - rec(batman_the_dark_knight_returns_part_2,fg2,jay_oliva,novo,...)
13 - rec(godzilla_resurgence,fg2,hideaki_anno,novo,...)
```

Notamos que o número de filmes do período clássico aumentou de 1 ($\delta = 1$) para 4 ($\delta = 100$), o que é mais próximo do parâmetro aprendido por meio do questionário. Assim, adotamos este segundo ajuste e mostraremos o ranking geral para a avaliação final da nossa usuária.

Análise da Entrevistada

Pontos Positivos:

Ao mostrarmos o ranking completo para a nossa entrevistada, de um modo geral ela ficou bastante satisfeita com as recomendações. Ela achou bem interessante que os filmes dos seus diretores favoritos estão nas primeiras posições.

Comentou também que muitos dos filmes que tem continuações ou são parte de uma saga já estavam no seu plano de assistir.

Como esse questionário foi respondido em setembro e apresentamos o ranking de recomendações no final de novembro, algumas das recomendações foram filmes que ela assistiu nesse período e nos contou que gostou destes filmes que estão bem colocados no ranking.

Algumas curiosidades foram que alguns dos diretores que matematicamente estimamos que são os seus favoritos ela não os conhecia pelos nomes mas, de fato, gostou de filmes desses diretores. As recomendações de filmes estrangeiros também foram uma surpresa e a entrevistada achou válido recomendar sucessos cinematográficos de outros países além dos EUA.

Sobre os vários tipos possíveis de *queries* que são possíveis de fazer, ela disse que a de classificação indicativa foi a que gostou mais devido a sua utilidade. Exemplificou que em momentos que vai assistir filmes com os seus sobrinhos (crianças) seria bastante útil que só fossem sugeridos filmes adequados para eles. Por fim, comentou que buscar recomendações filtrando pelo país ou ano de produção são também outros recursos interessantes.

Ponto Negativo:

Ao explicarmos a definição de família de gênero, a entrevistada entendeu a motivação de termos criado esse conceito mas não achou muito claro. Ela comentou que as recomendações por família de gêneros funcionaram, mas que não realizaria uma *query* com este elemento pois teria que ficar olhando cada uma das 15 famílias na tabela.

7. Conclusões

Neste trabalho, tivemos como principal objetivo fazer um estudo sobre a linguagem lógica probabilística *ProbLog* e o desenvolvimento de uma aplicação que foi o nosso modelo lógico probabilístico de um recomendador de filmes. Na seção subsequentes, vamos apresentar os comentários finais de tudo que foi desenvolvido, quais foram as nossas contribuições para a área e, por fim, alguns tópicos de pesquisas futuras que talvez vamos realizar, ou para motivar o leitor em possíveis ideias de trabalhos e projetos.

7.0.1 Comentários Finais

Iniciamos este trabalho com um estudo teórico da programação lógica restrita ao contexto de programas definidos positivos baseados na lógica de primeira ordem. Como exemplo de uma linguagem deste tipo estudamos alguns aspectos relevantes do *Prolog*. Acreditamos que para a melhor compreensão teórica e futuro desenvolvimentos de programas e aplicações do *ProbLog*, é essencial ter um conhecimento básico desta teoria que antecedeu a programação lógica probabilística.

No capítulo 4, apresentamos o *ProbLog*. Se não tivéssemos estudado a programação lógica e o *Prolog* anteriormente seria complicado de entender alguns aspectos da sua sintaxe e, principalmente, a motivação da criação do *ProbLog*. Mostramos questões e exemplos que foram modelados utilizando o modo de inferência do *ProbLog*. Nele é possível criar modelos de distribuições complexas que misturam probabilidade com lógica de uma maneira simples e intuitiva. Em outros paradigmas de programação existentes, muitos desses nossos modelos seriam extremamente complicados de programar, enquanto no *ProbLog* com poucos *fatos* e *regras* conseguimos o desejado.

Acreditamos que para aprofundar os nossos conhecimentos do *ProbLog* seria necessário desenvolver uma aplicação mais complexa do que os problemas apresentados no capítulo 4. Deste modo, a criação de um recomendador de filmes foi muito benéfica. O exercício de criar *fatos probabilísticos* e *regras* para a nossa aplicação foi um desafio estimulante. Propomos uma abordagem não muito convencional para as nossas inferências e ficamos muito felizes com os resultados e as inúmeras possibilidades que o *ProbLog* nos proporciona para modelar e resolver problemas. Sem esta linguagem não teríamos como construir uma distribuição lógica probabilística tão complexa e com tantos parâmetros e variáveis como a nossa.

Para a validação do nosso modelo e curiosidade se o nosso recomendador de filmes funcionaria com dados reais, desenvolvemos uma série de algoritmos que possibilitaram criar os *fatos probabilísticos* e transformar um banco de dados usual em *predicados* na formatação aceita pelo *ProbLog*. Esta tarefa, apesar de muito intuitiva, deu um trabalho considerável devido a algumas peculiaridades da linguagem *ProbLog*. Conforme apresentado na análise dos resultados, conseguimos de um modo bastante satisfatório o nosso objetivo de recomendar filmes baseado no conhecimento prévio dos gostos da nossa entrevistada.

Nossa conclusão final é que o *ProbLog* é uma linguagem muito interessante e de grande potencial. É possível programar modelos complexos de distribuições de probabilidade que contenham variáveis e regras lógicas de maneira sucinta e intuitiva.

7.0.2 Comentário Adicionais Sobre os Resultados

Gostaríamos de enfatizar dois pontos que tiveram um impacto significativo sobre a satisfação da nossa usuário sobre o resultado do recomendador de filmes:

- Conforme apresentado na seção Construindo um Questionário 6.1 dos top 300 filmes do IMDB apenas um único filme obteve avaliação negativa. Desta forma, a chance da nossa entrevistada gostar de outros filmes dessa lista é muito alta. Deste modo, acabamos recomendando filmes que muito provavelmente ela já iria gostar. No entanto, o nosso recomendador de filmes con-

seguiu criar um ranking de quais filmes recomendar mais fortemente do que outros.

- Além disso, fizemos apenas uma análise qualitativa dos resultados. Seria interessante ao trabalho se tivéssemos feito uma avaliação quantitativa do nosso recomendador de filmes como, por exemplo, utilizar a técnica de validação cruzada para a melhor avaliação dos resultados.

7.0.3 Contribuições Inéditas

- Este foi o primeiro trabalho em português sobre o *ProbLog*. Esperamos ter sido suficientemente claros para que um leitor sem conhecimento prévio da área e da linguagem consiga aprender e começar a desenvolver no *ProbLog*.
- Sugestão de um modelo de recomendador de filmes lógico probabilístico feito no *ProbLog*.
- O desenvolvimento de algoritmos na linguagem *R* para transformar um banco de dados em *fatos* aceitos pelo *ProbLog*.

7.0.4 Pesquisas Futuras

- Estudar as limitações da programação lógica de primeira ordem, e entender o que não é teoricamente possível de modelar no *ProbLog*.
- Utilizar ferramentas de *machine learning* para o ajuste de parâmetros recebidos no *ProbLog*.
- Testar o nosso recomendador com os recomendadores de filmes já existentes.

There are no two words in the English language more harmful than "good job".

– Terence Fletcher, Whiplash 2014

Bibliografia

[1] Ulf Nilsson and Jan Maluszynski. *Logic, Programming, and PROLOG*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1995.

[2] Ivan Bratko. *PROLOG Programming for Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1990.

[3] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery, 01 2007.

[4] Daan Fierens, Guy Van Den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2015.

[5] Anton Dries, Angelika Kimmig, Wannes Meert, Joris Renkens, Guy Van den Broeck, Jonas Vlasselaer, and Luc De Raedt. Problog2: Probabilistic logic programming. pages 312–315, 2015.

[6] Problog 2.1 documentation. https://problog.readthedocs.io/en/latest/index.html. Acessado ao longo de 2018.

[7] Simon French, editor. *Decision Theory: An Introduction to the Mathematics of Rationality*. Halsted Press, New York, NY, USA, 1986.

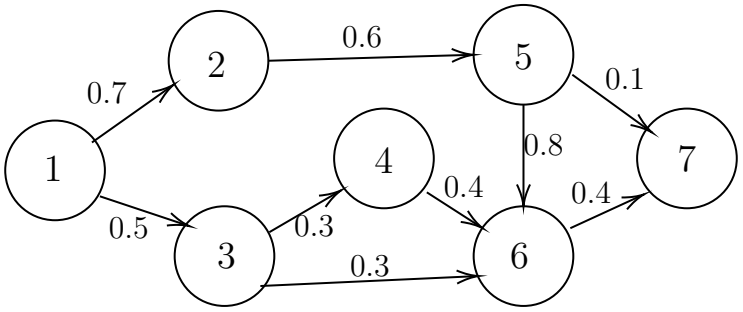
A. Anexo 1 - Dois outros Modos do ProbLog

A.1 Modo: Explicação Mais Provável

Neste modo ao contrário dos apresentados anteriormente (Inferência e Aprendizado) podem existir mais de uma maneira de satisfazer as nossas *Queries* e Evidências, assim estamos interessado em saber qual a maior probabilidade de acontecer um determinado fato. Matemáticamente:

$$EMP(e) \equiv \arg\max_{x \sim e} \mathbb{P}(x)$$

Observe o grafo probabilístico abaixo:



Descrevemos esse grafo por meio do seguinte programa:

```
%Probabilidade de cada Aresta
0.7::aresta(1,2).
0.5::aresta(2,3).
0.6::aresta(2,5).
0.3::aresta(3,4).
0.3::aresta(3,6).
```

```
0.4::aresta(4,6).
0.8::aresta(5,6).
0.1::aresta(5,7).
0.4::aresta(6,7).

%Regra Caminho
caminho(X,Y) :- aresta(X,Y).
caminho(X,Y) :- aresta(X,Z), Y \== Z, caminho(Z,Y).
```

Agora fazemos uma *query* para saber o caminho mais provável do vértice 1 ao 7 por exemplo.

```
query(caminho(1,7)).
```

Link do Modelo

Nossa resposta é: $0.13344 = 0.7 \cdot 0.6 \cdot 0.8 \cdot 0.4$. Que podemos interpretar como o caminho mais provável do vértice 1 ao 7.

A.2 Modo: Teoria da Decisão

O modo de teoria da decisão tem três características que o difere dos demais:

1. Não existem *queries* e evidências.
2. Alguns dos fatos podem ser expressos como um fato de decisão de modo que a decisão ótima deve ser tomada, isto é, a com maior valor da função de utilidade.
3. É possível também que alguns dos fatos e regras sejam expressos com a sua utilidade.

Sintaticamente expressamos um fato f do tipo (2) como:

```
?::f.
```

E para expressar alguma utilidade de uma regra ou fato:

```
utility(vencer, 10).
utility(perder, -5).
```

Vamos ilustrar o modo de Teoria da Decisão com a seguinte história narrada em primeira pessoa[1]:

"Hoje é o dia que vai entrar para a história, hoje dia 8 de abril de 2018 o meu Timão vai ser campeão. Você leitor pode não se lembrar muito bem mas no dia 31 de março perdemos o primeiro jogo de ida da final do paulistão na nossa casa lá em Itaquera, mó tristeza... Mas como diria o eterno presidente Vicente Matheus: *"Haja o que hajar, o Corinthias será campeão"*.

Confesso que estou confuso se me arrisco de ir lá no estádio do Palmeiras e ver se arranjo algum ingresso com algum cambista. Eu também posso assistir na tv mesmo mas a minha está quebrada então tem que ser na casa do meu amigo palmeirense. E a minha terceira opção é tentar ver pelo meu computador mas isso nunca dá muito certo.

Ainda bem que eu fiz um bom curso de MAE0212[2] no IME e aprendi um pouquinho de teoria da decisão para me ajudar nesses momentos díficeis da vida."

Após a leitura da nossa breve história vamos modelar o problema: Utilizando a terminologia da Teoria da Decisão, temos dois estados da natureza que são: V = vitória do Corinthians ou D = derrota. E devemos optar por uma entre três ações a_1 = assistir ao jogo no estádio, a_2 = assistir ao jogo pela tv (do amigo palmeirense), a_3 = assistir ao jogo pelo computador. Para resolver esse problema devemos atribuir valores para cada um dos seis resultados possíveis. Decidimos utilizar uma escala monetária, isto é, quanto cada ação dependendo da sua natureza "vale". Assistir a vitória do Corinthias ao vivo é sem dúvida a maior emoção, mas por outro lado vivenciar a derrota no estádio do Palmeiras é bastante amargo. Já no segundo cenário assistir na tv do amigo palmeirense tráz uma bela satisfação mas sofrer as piadas e gozações caso o Corinthians perca é chato. E por fim assistir ao jogo pelo computador ao vivo não vai ser muito legal, mas se o Corinthians

[1] Baseada em fatos reais e na imaginação do autor.
[2] Introdução à Probabilidade e à Estatística II.

perder ou tomar goleada (pouco provável) pelo menos fiquei sozinho no consolo do meu próprio lar. Segue a tabela com esses valores monetarizados:

Natureza \ Ação	a_1	a_2	a_3
V	600	300	-150
D	-300	-200	50

Vamos assumir que esses valores representam o ganho ou perda monetária em Reais (R$).

A única informação que falta é a probabilidade dos estados da natureza, vamos atribuir que a chance do Corinthians ganhar o campeonato é de 0.4.

Abaixo temos o algoritmo em *ProbLog* que modela este problema:

```
% Fatos Probabilísticos
0.4::corinthians ; 0.6::palmeiras.

% Fatos de Decisão
?::a1.
?::a2.
?::a3.

%Regras Lógicas
v_a1 :- corinthians, a1.
d_a1 :- palmeiras, a1.

v_a2 :- corinthians, a2.
d_a2 :- palmeiras, a2.

v_a3 :- corinthians, a3.
d_a3 :- palmeiras, a3.


% Utilidade de cada cenário
utility(v_a1, 600).
utility(v_a2, 300).
utility(v_a3, -150).
utility(d_a1, -300).
```

```
utility(d_a2, -200).
utility(d_a3, 50).
```

Link deste Algoritmo

Após resolver este problema no *ProbLog* nos é retornado que a melhor decisão é a: a_1 - ir assistir ao jogo no estádio.

B. Anexo 2 - Tabelas

Top 300 Filmes IMDB

movie-title	genres	title-year	director-name
The_Shawshank_Redemption	Crime—Drama	1994	Frank_Darabont
The_Godfather	Crime—Drama	1972	Francis_Ford_Coppola
The_Dark_Knight	Action—Crime—Drama—Thriller	2008	Christopher_Nolan
The_Godfather:_Part_II	Crime—Drama	1974	Francis_Ford_Coppola
The_Lord_of_the_Rings:_The_Return_of_the_King	Action—Adventure—Drama—Fantasy	2003	Peter_Jackson
Schindler's_List	Biography—Drama—History	1993	Steven_Spielberg
Pulp_Fiction	Crime—Drama	1994	Quentin_Tarantino
The_Good_the_Bad_and_the_Ugly	Western	1966	Sergio_Leone
Twelve_Angry_Men	Crime—Drama	1957	Sidney_Lumet
Inception	Action—Adventure—Sci-Fi—Thriller	2010	Christopher_Nolan
The_Lord_of_the_Rings:_The_Fellowship_of_the_Ring	Action—Adventure—Drama—Fantasy	2001	Peter_Jackson
Fight_Club	Drama	1999	David_Fincher
Forrest_Gump	Comedy—Drama	1994	Robert_Zemeckis
Star_Wars:_Episode_V_-_The_Empire_Strikes_Back	Action—Adventure—Fantasy—Sci-Fi	1980	Irvin_Kershner
The_Lord_of_the_Rings:_The_Two_Towers	Action—Adventure—Drama—Fantasy	2002	Peter_Jackson
The_Matrix	Action—Sci-Fi	1999	Lana_Wachowski
Goodfellas	Biography—Crime—Drama	1990	Martin_Scorsese
Star_Wars:_Episode_IV_-_A_New_Hope	Action—Adventure—Fantasy—Sci-Fi	1977	George_Lucas
One_Flew_Over_the_Cuckoo's_Nest	Drama	1975	Milos_Forman
City_of_God	Crime—Drama	2002	Fernando_Meirelles
Queen_of_the_Mountains	Action—Biography—Drama—History	2014	Sadyk_Sher-Niyaz
Seven_Samurai	Action—Adventure—Drama	1954	Akira_Kurosawa
Interstellar	Adventure—Drama—Sci-Fi	2014	Christopher_Nolan
Saving_Private_Ryan	Action—Drama—War	1998	Steven_Spielberg
Se7en	Crime—Drama—Mystery—Thriller	1995	David_Fincher
The_Silence_of_the_Lambs	Crime—Drama—Horror—Thriller	1991	Jonathan_Demme
Spirited_Away	Adventure—Animation—Family—Fantasy	2001	Hayao_Miyazaki
American_History_X	Crime—Drama	1998	Tony_Kaye
The_Usual_Suspects	Crime—Drama—Mystery—Thriller	1995	Bryan_Singer
Once_Upon_a_Time_in_the_West	Western	1968	Sergio_Leone
It's_a_Wonderful_Life	Drama—Family—Fantasy—Romance	1946	Frank_Capra
Modern_Times	Comedy—Drama—Family	1936	Charles_Chaplin
Casablanca	Drama—Romance—War	1942	Michael_Curtiz
The_Dark_Knight_Rises	Action—Thriller	2012	Christopher_Nolan
Gladiator	Action—Drama—Romance	2000	Ridley_Scott
Terminator_2:_Judgment_Day	Action—Sci-Fi	1991	James_Cameron
Django_Unchained	Drama—Western	2012	Quentin_Tarantino
The_Departed	Crime—Drama—Thriller	2006	Martin_Scorsese
The_Lion_King	Adventure—Animation—Drama—Family—Musical	1994	Roger_Allers
The_Green_Mile	Crime—Drama—Fantasy—Mystery	1999	Frank_Darabont
The_Prestige	Drama—Mystery—Sci-Fi—Thriller	2006	Christopher_Nolan
The_Pianist	Biography—Drama—War	2002	Roman_Polanski
Apocalypse_Now	Drama—War	1979	Francis_Ford_Coppola
Raiders_of_the_Lost_Ark	Action—Adventure	1981	Steven_Spielberg
Psycho	Horror—Mystery—Thriller	1960	Alfred_Hitchcock
Back_to_the_Future	Adventure—Comedy—Sci-Fi	1985	Robert_Zemeckis
Alien	Horror—Sci-Fi	1979	Ridley_Scott
Memento	Mystery—Thriller	2000	Christopher_Nolan
Airlift	Action—Drama—History—Thriller—War	2016	Raja_Menon
Samsara	Documentary—Music	2011	Ron_Fricke
Whiplash	Drama—Music	2014	Damien_Chazelle
The_Lives_of_Others	Drama—Thriller	2006	Florian_Henckel...
Dr_Strangelove_or...	Comedy	1964	Stanley_Kubrick
Children_of_Heaven	Drama—Family	1997	Majid_Majidi
WALL·E	Adventure—Animation—Family—Sci-Fi	2008	Andrew_Stanton
Braveheart	Biography—Drama—History—War	1995	Mel_Gibson
Amélie	Comedy—Romance	2001	Jean-Pierre_Jeunet
Baahubali:_The_Beginning	Action—Adventure—Drama—Fantasy—War	2015	S.S._Rajamouli
Lion_of_the_Desert	Biography—Drama—History—War	1980	Moustapha_Akkad
Star_Wars:_Episode_VI_-_Return_of_the_Jedi	Action—Adventure—Fantasy—Sci-Fi	1983	Richard_Marquand
Once_Upon_a_Time_in_America	Crime—Drama	1984	Sergio_Leone
Princess_Mononoke	Adventure—Animation—Fantasy	1997	Hayao_Miyazaki
The_Shining	Drama—Horror	1980	Stanley_Kubrick
Aliens	Action—Adventure—Sci-Fi	1986	James_Cameron
American_Beauty	Drama	1999	Sam_Mendes
Lawrence_of_Arabia	Adventure—Biography—Drama—History—War	1962	David_Lean
U2_3D	Documentary—Music	2007	Catherine_Owens
Das_Boot	Adventure—Drama—Thriller—War	1981	Wolfgang_Petersen
Rang_De_Basanti	Comedy—Drama—History—Romance	2006	Rakeysh_Omprakash_Mehra
Requiem_for_a_Dream	Drama	2000	Darren_Aronofsky
Batman:_The_Dark_Knight_Returns_Part_2	Action—Animation—Crime—Sci-Fi—Thriller	2013	Jay_Oliva
Oldboy	Drama—Mystery—Thriller	2003	Chan-wook_Park
Reservoir_Dogs	Crime—Drama—Thriller	1992	Quentin_Tarantino
A_Separation	Drama—Mystery	2011	Asghar_Farhadi
The_Other_Dream_Team	Documentary—Sport	2012	Marius_A._Markevicius
Toy_Story_3	Adventure—Animation—Comedy—Family—Fantasy	2010	Lee_Unkrich
Up	Adventure—Animation—Comedy—Family	2009	Pete_Docter
Inside_Out	Adventure—Animation—Comedy—Drama—Family—Fantasy	2015	Pete_Docter
Batman_Begins	Action—Adventure	2005	Christopher_Nolan

movie-title	genres	title-year	director-name
Inglourious_Basterds	Adventure—Drama—War	2009	Quentin_Tarantino
Indiana_Jones_and_the_Last_Crusade	Action—Adventure—Fantasy	1989	Steven_Spielberg
LA_Confidential	Crime—Drama—Mystery—Thriller	1997	Curtis_Hanson
Toy_Story	Adventure—Animation—Comedy—Family—Fantasy	1995	John_Lasseter
Scarface	Crime—Drama	1983	Brian_De_Palma
Eternal_Sunshine_of_the_Spotless_Mind	Drama—Fantasy—Romance—Sci-Fi	2004	Michel_Gondry
Amadeus	Biography—Drama—History—Music	1984	Milos_Forman
Raging_Bull	Biography—Drama—Sport	1980	Martin_Scorsese
Metropolis	Drama—Sci-Fi	1927	Fritz_Lang
Unforgiven	Drama—Western	1992	Clint_Eastwood
Downfall	Biography—Drama—History—War	2004	Oliver_Hirschbiegel
Room	Drama	2015	Lenny_Abrahamson
Snatch	Comedy—Crime	2000	Guy_Ritchie
Two_Thousand_and_One:_A_Space_Odyssey	Adventure—Mystery—Sci-Fi	1968	Stanley_Kubrick
Good_Will_Hunting	Drama	1997	Gus_Van_Sant
Snatch	Comedy—Crime	2000	Guy_Ritchie
Lake_of_Fire	Documentary	2006	Tony_Kaye
The_Sting	Comedy—Crime—Drama	1973	George_Roy_Hill
The_Great_Escape	Adventure—Drama—History—Thriller—War	1963	John_Sturges
The_Hunt	Drama	2012	Thomas_Vinterberg
The_Apartment	Comedy—Drama—Romance	1960	Billy_Wilder
Judgment_at_Nuremberg	Drama—War	1961	Stanley_Kramer
Some_Like_It_Hot	Comedy—Music—Romance	1959	Billy_Wilder
Singing_in_the_Rain	Comedy—Musical—Romance	1952	Stanley_Donen
Inside_Job	Crime—Documentary	2010	Charles_Ferguson
No_End_in_Sight	Documentary—War	2007	Charles_Ferguson
Taxi_Driver	Crime—Drama	1976	Martin_Scorsese
Hoop_Dreams	Documentary—Drama—Sport	1994	Steve_James
Monty_Python_and_the_Holy_Grail	Adventure—Comedy—Fantasy	1975	Terry_Gilliam
The_Big_Parade	Drama—Romance—War	1925	King_Vidor
Peace_Propaganda_the_Promised_Land	Documentary	2004	Sut_Jhally
Captain_America:_Civil_War	Action—Adventure—Sci-Fi	2016	Anthony_Russo
How_to_Train_Your_Dragon	Adventure—Animation—Family—Fantasy	2010	Dean_DeBlois
Godzilla_Resurgence	Action—Adventure—Drama—Horror—Sci-Fi	2016	Hideaki_Anno
The_Wolf_of_Wall_Street	Biography—Comedy—Crime—Drama	2013	Martin_Scorsese
Finding_Nemo	Adventure—Animation—Comedy—Family	2003	Andrew_Stanton
A_Beautiful_Mind	Biography—Drama	2001	Ron_Howard
Casino	Biography—Crime—Drama	1995	Martin_Scorsese
V_for_Vendetta	Action—Drama—Thriller	2005	James_McTeigue
The_Thing	Horror—Mystery—Sci-Fi	1982	John_Carpenter
Die_Hard	Action—Thriller	1988	John_McTiernan
Blade_Runner	Sci-Fi—Thriller	1982	Ridley_Scott
Gran_Torino	Drama	2008	Clint_Eastwood
Warrior	Drama—Sport	2011	Gavin_O'Connor
Howl's_Moving_Castle	Adventure—Animation—Family—Fantasy	2004	Hayao_Miyazaki
Into_the_Wild	Adventure—Biography—Drama	2007	Sean_Penn
Pan's_Labyrinth	Drama—Fantasy—War	2006	Guillermo_del_Toro
The_Deer_Hunter	Drama—War	1978	Michael_Cimino
The_Big_Lebowski	Comedy—Crime	1998	Joel_Coen
Incendies	Drama—Mystery—War	2010	Denis_Villeneuve
The_Elephant_Man	Biography—Drama	1980	David_Lynch
Gone_with_the_Wind	Drama—History—Romance—War	1939	Victor_Fleming
The_Secret_in_Their_Eyes	Drama—Mystery—Thriller	2009	Juan_José_Campanella
Trainspotting	Drama	1996	Danny_Boyle
The_Bridge_on_the_River_Kwai	Adventure—Drama—War	1957	David_Lean
Lage_Raho_Munna_Bhai	Comedy—Drama—Romance	2006	Rajkumar_Hirani
Growing_Up_Smith	Comedy—Drama—Family	2015	Frank_Lotito
Lock_Stock_and_Two_Smoking_Barrels	Comedy—Crime	1998	Guy_Ritchie
Rebecca	Drama—Mystery—Thriller	1940	Alfred_Hitchcock
The_Act_of_Killing	Biography—Crime—Documentary—History	2012	Joshua_Oppenheimer
On_the_Waterfront	Crime—Drama—Romance	1954	Elia_Kazan
Rise_of_the_Entrepreneur...	Documentary	2014	Joe_Kenemore
Cries_Whispers	Drama	1972	Ingmar_Bergman
It_Happened_One_Night	Comedy—Romance	1934	Frank_Capra
The_Last_Waltz	Documentary—Music	1978	Martin_Scorsese
The_Avengers	Action—Adventure—Sci-Fi	2012	Joss_Whedon
Guardians_of_the_Galaxy	Action—Adventure—Sci-Fi	2014	James_Gunn
Mad_Max:_Fury_Road	Action—Adventure—Sci-Fi—Thriller	2015	George_Miller
The_Revenant	Adventure—Drama—Thriller—Western	2015	Alejandro_G._Inarritu
The_Bourne_Ultimatum	Action—Mystery—Thriller	2007	Paul_Greengrass
Pirates_of_the_Caribbean:_The_Curse_of_the_Black_Pearl	Action—Adventure—Fantasy	2003	Gore_Verbinski
Monsters_Inc	Adventure—Animation—Comedy—Family—Fantasy	2001	Pete_Docter
The_Martian	Adventure—Drama—Sci-Fi	2015	Ridley_Scott
Shutter_Island	Mystery—Thriller	2010	Martin_Scorsese
Jurassic_Park	Adventure—Sci-Fi—Thriller	1993	Steven_Spielberg
Gone_Girl	Crime—Drama—Mystery—Thriller	2014	David_Fincher
The_Truman_Show	Comedy—Drama—Sci-Fi	1998	Peter_Weir
The_Avengers	Action—Adventure—Sci-Fi	2012	Joss_Whedon
Deadpool	Action—Adventure—Comedy—Romance—Sci-Fi	2016	Tim_Miller
Kill_Bill:_Vol_1	Action	2003	Quentin_Tarantino
Solaris	Drama—Mystery—Sci-Fi	1972	Andrei_Tarkovsky
Prisoners	Crime—Drama—Mystery—Thriller	2013	Denis_Villeneuve
The_Sixth_Sense	Drama—Mystery—Thriller	1999	M._Night_Shyamalan
Sin_City	Crime—Thriller	2005	Frank_Miller
Rush	Action—Biography—Drama—Sport	2013	Ron_Howard
The_Grand_Budapest_Hotel	Adventure—Comedy—Crime—Drama	2014	Wes_Anderson
Million_Dollar_Baby	Drama—Sport	2004	Clint_Eastwood
The_Help	Drama	2011	Tate_Taylor
No_Country_for_Old_Men	Crime—Drama—Thriller	2007	Ethan_Coen
There_Will_Be_Blood	Drama	2007	Paul_Thomas_Anderson

movie-title	genres	title-year	director-name
Gandhi	Biography—Drama—History	1982	Richard_Attenborough
Twelve_Years_a_Slave	Biography—Drama—History	2013	Steve_McQueen
Spotlight	Biography—Crime—Drama—History	2015	Tom_McCarthy
Hotel_Rwanda	Drama—History—War	2004	Terry_George
Hachi_A_Dog's_Tale	Drama—Family	2009	Lasse_Hallstrom
The_Imitation_Game	Biography—Drama—Thriller—War	2014	Morten_Tyldum
The_Princess_Bride	Adventure—Family—Fantasy—Romance	1987	Rob_Reiner
Groundhog_Day	Comedy—Fantasy—Romance	1993	Harold_Ramis
The_Sea_Inside	Biography—Drama—Romance	2004	Alejandro_Amenabar
Tae_Guk_Gi_The_Brotherhood_of_War	Action—Drama—War	2004	Je-kyu_Kang
Barry_Lyndon	Adventure—Drama—History—War	1975	Stanley_Kubrick
Stand_by_Me	Adventure—Drama	1986	Rob_Reiner
Akira	Action—Animation—Sci-Fi	1988	Katsuhiro_Otomo
Elite_Squad	Action—Crime—Drama—Thriller	2007	Jose_Padilha
The_Terminator	Action—Sci-Fi	1984	James_Cameron
Platoon	Drama—War	1986	Oliver_Stone
Butch_Cassidy_and_the_Sundance_Kid	Biography—Crime—Drama—Western	1969	George_Roy_Hill
Donnie_Darko	Drama—Sci-Fi—Thriller	2001	Richard_Kelly
Annie_Hall	Comedy—Romance	1977	Woody_Allen
Network	Drama—Romance	1976	Sidney_Lumet
A_Christmas_Story	Comedy—Family	1983	Bob_Clark
The_Man_Who_Shot_Liberty_Valance	Drama—Western	1962	John_Ford
Cat_on_a_Hot_Tin_Roof	Drama—Romance	1958	Richard_Brooks
The_Wizard_of_Oz	Adventure—Family—Fantasy—Musical	1939	Victor_Fleming
Before_Sunrise	Drama—Romance	1995	Richard_Linklater
The_Best_Years_of_Our_Lives	Drama—Romance—War	1946	William_Wyler
Amores_Perros	Drama—Thriller	2000	Alejandro_G._Inarritu
Touching_the_Void	Adventure—Documentary—Drama—Sport	2003	Kevin_Macdonald
In_the_Shadow_of_the_Moon	Documentary—History	2007	David_Sington
The_Celebration	Drama	1998	Thomas_Vinterberg
Rocky	Drama—Sport	1976	John_G._Avildsen
The_Conformist	Drama	1970	Bernardo_Bertolucci
High_Noon	Thriller—Western	1952	Fred_Zinnemann
Woodstock	Documentary—History—Music	1970	Michael_Wadleigh
Nothing_But_a_Man	Drama—Romance	1964	Michael_Roemer
Ordet	Drama—Fantasy	1955	Carl_Theodor_Dreyer
X-Men_Days_of_Future_Past	Action—Adventure—Fantasy—Sci-Fi—Thriller	2014	Bryan_Singer
Ratatouille	Animation—Comedy—Family—Fantasy	2007	Brad_Bird
Star_Trek	Action—Adventure—Sci-Fi	2009	J.J._Abrams
Life_of_Pi	Adventure—Drama—Fantasy	2012	Ang_Lee
Casino_Royale	Action—Adventure—Thriller	2006	Martin_Campbell
Blood_Diamond	Adventure—Drama—Thriller	2006	Edward_Zwick
The_Incredibles	Action—Adventure—Animation—Family	2004	Brad_Bird
Cinderella_Man	Biography—Drama—Sport	2005	Ron_Howard
Big_Fish	Adventure—Drama—Fantasy	2003	Tim_Burton
The_Pursuit_of_Happyness	Biography—Drama	2006	Gabriele_Muccino
Kill_Bill_Vol_2	Action—Crime—Drama—Thriller	2004	Quentin_Tarantino
Catch_Me_If_You_Can	Biography—Crime—Drama	2002	Steven_Spielberg
The_Iron_Giant	Action—Adventure—Animation—Comedy—Drama—Family—Sci-Fi	1999	Brad_Bird
JFK	Drama—History—Thriller	1991	Oliver_Stone
Serenity	Action—Adventure—Sci-Fi—Thriller	2005	Joss_Whedon
Magnolia	Drama	1999	Paul_Thomas_Anderson
Blood_In_Blood_Out	Crime—Drama	1993	Taylor_Hackford
District_9	Action—Sci-Fi—Thriller	2009	Neill_Blomkamp
Mystic_River	Crime—Drama—Mystery—Thriller	2003	Clint_Eastwood
Aladdin	Adventure—Animation—Comedy—Family—Fantasy—Musical	1992	Ron_Clements
Winged_Migration	Documentary	2001	Jacques_Perrin
Rain_Man	Drama	1988	Barry_Levinson
Her	Drama—Romance—Sci-Fi	2013	Spike_Jonze
Dances_with_Wolves	Adventure—Drama—Western	1990	Kevin_Costner
Dead_Poets_Society	Comedy—Drama	1989	Peter_Weir
The_Artist	Comedy—Drama—Romance	2011	Michel_Hazanavicius
The_King's_Speech	Biography—Drama—History—Romance	2010	Tom_Hooper
Brazil	Drama—Sci-Fi	1985	Terry_Gilliam
In_Bruges	Comedy—Crime—Drama	2008	Martin_McDonagh
Mulholland_Drive	Drama—Mystery—Thriller	2001	David_Lynch
The_Return	Drama—Mystery—Thriller	2003	Andrey_Zvyagintsev
Slumdog_Millionaire	Drama—Romance	2008	Danny_Boyle
The_Diving_Bell_and_the_Butterfly	Biography—Drama	2007	Julian_Schnabel
Black_Swan	Drama—Thriller	2010	Darren_Aronofsky
The_Perks_of_Being_a_Wallflower	Drama—Romance	2012	Stephen_Chbosky
True_Romance	Action—Crime—Drama—Romance—Thriller	1993	Tony_Scott
Dancer_in_the_Dark	Crime—Drama—Musical	2000	Lars_von_Trier
The_Exorcist	Horror	1973	William_Friedkin
Jaws	Adventure—Drama—Thriller	1975	Steven_Spielberg
Patton	Biography—Drama—War	1970	Franklin_J._Schaffner
Casino_Royale	Action—Adventure—Thriller	2006	Martin_Campbell
Doctor_Zhivago	Drama—Romance—War	1965	David_Lean
The_Straight_Story	Biography—Drama	1999	David_Lynch

movie-title	genres	title-year	director-name
Fiddler_on_the_Roof	Drama—Family—Musical—Romance	1971	Norman_Jewison
Sicko	Documentary—Drama	2007	Michael_Moore
My_Name_Is_Khan	Adventure—Drama—Thriller	2010	Karan_Johar
The_Sound_of_Music	Biography—Drama—Family—Musical—Romance	1965	Robert_Wise
Persepolis	Animation—Biography—Drama—War	2007	Vincent_Paronnaud
The_Wild_Bunch	Action—Adventure—Western	1969	Sam_Peckinpah
Dallas_Buyers_Club	Biography—Drama	2013	Jean-Marc_Vallee
Shaun_of_the_Dead	Comedy—Horror	2004	Edgar_Wright
Sling_Blade	Drama	1996	Billy_Bob_Thornton
Night_of_the_Living_Dead	Drama—Horror—Mystery	1968	George_A._Romero
Boyhood	Drama	2014	Richard_Linklater
In_Cold_Blood	Biography—Crime—Drama—History	1967	Richard_Brooks
Rosemary's_Baby	Drama—Horror	1968	Roman_Polanski
Bowling_for_Columbine	Crime—Documentary—Drama	2002	Michael_Moore
Days_of_Heaven	Drama—Romance	1978	Terrence_Malick
Central_Station	Drama	1998	Walter_Salles
Young_Frankenstein	Comedy	1974	Mel_Brooks
The_Hustler	Drama—Sport	1961	Robert_Rossen
Before_Sunset	Drama—Romance	2004	Richard_Linklater
Waltz_with_Bashir	Animation—Biography—Documentary—Drama—History—War	2008	Ari_Folman
A_Streetcar_Named_Desire	Drama	1951	Elia_Kazan
You_Can't_Take_It_with_You	Comedy—Drama—Romance	1938	Frank_Capra
The_Lost_Weekend	Drama	1945	Billy_Wilder
Pandora's_Box	Crime—Drama—Romance	1929	Georg_Wilhelm_Pabst
Light_from_the_Darkroom	Action—Drama—Thriller	2014	Lance_McDaniel
Tupac_Resurrection	Biography—Documentary—Music	2003	Lauren_Lazin
A_Fistful_of_Dollars	Action—Drama—Western	1964	Sergio_Leone
The_Man_from_Earth	Drama—Romance—Sci-Fi	2007	Richard_Schenkman
Night_of_the_Living_Dead	Drama—Horror—Mystery	1968	George_A._Romero
Avatar	Action—Adventure—Fantasy—Sci-Fi	2009	James_Cameron
The_Hobbit_The_Desolation_of_Smaug	Adventure—Fantasy	2013	Peter_Jackson
Iron_Man	Action—Adventure—Sci-Fi	2008	Jon_Favreau
Edge_of_Tomorrow	Action—Adventure—Sci-Fi	2014	Doug_Liman
Big_Hero_6	Action—Adventure—Animation—Comedy—Drama—Family—Sci-Fi	2014	Don_Hall
The_Hobbit_An_Unexpected_Journey	Adventure—Fantasy	2012	Peter_Jackson
How_to_Train_Your_Dragon_2	Action—Adventure—Animation—Comedy—Family—Fantasy	2014	Dean_DeBlois
Toy_Story_2	Adventure—Animation—Comedy—Family—Fantasy	1999	John_Lasseter
Children_of_Men	Drama—Sci-Fi—Thriller	2006	Alfonso_Cuaron
The_Insider	Biography—Drama—Thriller	1999	Michael_Mann
The_Hateful_Eight	Crime—Drama—Mystery—Thriller—Western	2015	Quentin_Tarantino
The_Bourne_Identity	Action—Mystery—Thriller	2002	Doug_Liman
Almost_Famous	Adventure—Comedy—Drama—Music	2000	Cameron_Crowe
Captain_Phillips	Biography—Drama—Thriller	2013	Paul_Greengrass
Shrek	Adventure—Animation—Comedy—Family—Fantasy	2001	Andrew_Adamson
Hero	Action—Adventure—History	2002	Yimou_Zhang
The_Notebook	Drama—Romance	2004	Nick_Cassavetes
Glory	Drama—History—War	1989	Edward_Zwick
Walk_the_Line	Biography—Drama—Music—Romance	2005	James_Mangold
Straight_Outta_Compton	Biography—Crime—Drama—History—Music	2015	F._Gary_Gray
The_Blues_Brothers	Action—Comedy—Crime—Music	1980	John_Landis
The_Right_Stuff	Adventure—Drama—History	1983	Philip_Kaufman
Taken	Action—Thriller	2008	Pierre_Morel

Resultado Questionário

Titulo	Nota	Titulo	Nota
The_Shawshank_Redemption		The_Other_Dream_Team	
The_Godfather	1	Toy_Story_3	1
The_Dark_Knight	1	Up	1
The_Godfather_Part_II	1	Inside_Out	1
The_Lord_of_the_Rings_The_Return_of_the_King	1	Batman_Begins	
Schindler's_List	1	Inglourious_Basterds	1
Pulp_Fiction	1	Indiana_Jones_and_the_Last_Crusade	1
The_Good_the_Bad_and_the_Ugly		LA_Confidential	
Twelve_Angry_Men		Toy_Story	1
Inception	1	Scarface	
The_Lord_of_the_Rings_The_Fellowship_of_the_Ring	1	Eternal_Sunshine_of_the_Spotless_Mind	1
Fight_Club	1	Amadeus	1
Forrest_Gump	1	Raging_Bull	
Star_Wars_Episode_V_-_The_Empire_Strikes_Back	1	Metropolis	
The_Lord_of_the_Rings_The_Two_Towers	1	Unforgiven	
The_Matrix		Downfall	
Goodfellas		Room	
Star_Wars_Episode_IV_-_A_New_Hope	1	Snatch	
One_Flew_Over_the_Cuckoo's_Nest		Two_Thousand_and_One_A_Space_Odyssey	1
City_of_God		Good_Will_Hunting	
Queen_of_the_Mountains		Snatch	
Seven_Samurai		Lake_of_Fire	
Interstellar	1	The_Sting	
Saving_Private_Ryan		The_Great_Escape	
Se7en		The_Hunt	
The_Silence_of_the_Lambs		The_Apartment	
Spirited_Away		Judgment_at_Nuremberg	
American_History_X		Some_Like_It_Hot	
The_Usual_Suspects		Singin'_in_the_Rain	
Once_Upon_a_Time_in_the_West		Inside_Job	
It's_a_Wonderful_Life	1	No_End_in_Sight	
Modern_Times		Taxi_Driver	1
Casablanca		Hoop_Dreams	
The_Dark_Knight_Rises		Monty_Python_and_the_Holy_Grail	
Gladiator		The_Big_Parade	
Terminator_2_Judgment_Day		Peace_Propaganda...	
Django_Unchained	1	Captain_America_Civil_War	1
The_Departed		How_to_Train_Your_Dragon	
The_Lion_King	1	Godzilla_Resurgence	
The_Green_Mile		The_Wolf_of_Wall_Street	1
The_Prestige		Finding_Nemo	1
The_Pianist	1	A_Beautiful_Mind	1
Apocalypse_Now		Casino	
Raiders_of_the_Lost_Ark		V_for_Vendetta	
Psycho		The_Thing	
Back_to_the_Future	1	Die_Hard	
Alien	1	Blade_Runner	1
Memento		Gran_Torino	
Airlift		Warrior	
Samsara		Howl's_Moving_Castle	
Whiplash	1	Into_the_Wild	1
The_Lives_of_Others		Pan's_Labyrinth	
Dr_Strangelove_or		The_Deer_Hunter	

Titulo	Nota	Titulo	Nota
Children_of_Heaven		The_Big_Lebowski	
WALL·E	1	Incendies	
Braveheart	1	The_Elephant_Man	
Amélie	1	Gone_with_the_Wind	
Baahubali_The_Beginning		The_Secret_in_Their_Eyes	
Lion_of_the_Desert		Trainspotting	
Star_Wars_Episode_VI_-_Return_of_the_Jedi	1	The_Bridge_on_the_River_Kwai	
Once_Upon_a_Time_in_America		Lage_Raho_Munna_Bhai	
Princess_Mononoke		Growing_Up_Smith	
The_Shining		Lock_Stock_and_Two_Smoking_Barrels	
Aliens		Rebecca	
American_Beauty		The_Act_of_Killing	
Lawrence_of_Arabia		On_the_Waterfront	
U2_3D		Rise_of_the_Entrepreneur	
Das_Boot		Cries_Whispers	1
Rang_De_Basanti		It_Happened_One_Night	1
Requiem_for_a_Dream		The_Last_Waltz	
Batman_The_Dark_Knight_Returns_Part_2		The_Avengers	1
Oldboy		Guardians_of_the_Galaxy	1
Reservoir_Dogs		Mad_Max_Fury_Road	1
A_Separation		The_Revenant	
The_Bourne_Ultimatum		Aladdin	1
Pirates_of_the_Caribbean_1	1	Winged_Migration	
Monsters_Inc	1	Rain_Man	1
The_Martian		Her	1
Shutter_Island	1	Dances_with_Wolves	1
Jurassic_Park	1	Dead_Poets_Society	
Gone_Girl		The_Artist	1
The_Truman_Show		The_King's_Speech	
The_Avengers	1	Brazil	
Deadpool	0	In_Bruges	
Kill_Bill_Vol_1	1	Mulholland_Drive	
Solaris	1	The_Return	
Prisoners		Slumdog_Millionaire	
The_Sixth_Sense	1	The_Diving_Bell_and_the_Butterfly	1
Sin_City		Black_Swan	1
Rush		The_Perks_of_Being_a_Wallflower	1
The_Grand_Budapest_Hotel		True_Romance	
Million_Dollar_Baby		Dancer_in_the_Dark	1
The_Help	1	The_Exorcist	
No_Country_for_Old_Men	1	Jaws	
There_Will_Be_Blood		Patton	
Gandhi		Casino_Royale	1
Twelve_Years_a_Slave		Doctor_Zhivago	
Spotlight	1	The_Straight_Story	
Hotel_Rwanda		Fiddler_on_the_Roof	
Hachi_A_Dog's_Tale		Sicko	
The_Imitation_Game	1	My_Name_Is_Khan	
The_Princess_Bride		The_Sound_of_Music	
Groundhog_Day		Persepolis	

Titulo	Nota	Titulo	Nota
The_Sea_Inside		The_Wild_Bunch	
Tae_Guk_Gi_The_Brotherhood_of_War		Dallas_Buyers_Club	
Barry_Lyndon		Shaun_of_the_Dead	
Stand_by_Me		Sling_Blade	
Akira		Night_of_the_Living_Dead	
Elite_Squad		Boyhood	
The_Terminator		In_Cold_Blood	
Platoon	1	Rosemary's_Baby	
Butch_Cassidy_and_the_Sundance_Kid		Bowling_for_Columbine	
Donnie_Darko		Days_of_Heaven	
Annie_Hall		Central_Station	
Network		Young_Frankenstein	
A_Christmas_Story		The_Hustler	
The_Man_Who_Shot_Liberty_Valance	1	Before_Sunset	
Cat_on_a_Hot_Tin_Roof		Waltz_with_Bashir	
The_Wizard_of_Oz		A_Streetcar_Named_Desire	
Before_Sunrise		You_Can't_Take_It_with_You	
The_Best_Years_of_Our_Lives		The_Lost_Weekend	
Amores_Perros		Pandora's_Box	
Touching_the_Void		Light_from_the_Darkroom	
In_the_Shadow_of_the_Moon		Tupac_Resurrection	
The_Celebration		A_Fistful_of_Dollars	
Rocky		The_Man_from_Earth	
The_Conformist		Night_of_the_Living_Dead	
High_Noon		Avatar	1
Woodstock		The_Hobbit_The_Desolation_of_Smaug	1
Nothing_But_a_Man		Iron_Man	1
Ordet		Edge_of_Tomorrow	
X-Men_Days_of_Future_Past	1	Big_Hero_6	1
Ratatouille	1	The_Hobbit_An_Unexpected_Journey	1
Star_Trek	1	How_to_Train_Your_Dragon_2	1
Life_of_Pi	1	Toy_Story_2	1
Casino_Royale		Children_of_Men	
Blood_Diamond	1	The_Insider	
The_Incredibles		The_Hateful_Eight	1
Cinderella_Man		The_Bourne_Identity	
Big_Fish		Almost_Famous	
The_Pursuit_of_Happyness		Captain_Phillips	
Kill_Bill_Vol_2	1	Shrek	1
Catch_Me_If_You_Can	1	Hero	
The_Iron_Giant		The_Notebook	
JFK		Glory	
Serenity		Walk_the_Line	
Magnolia		Straight_Outta_Compton	
Blood_In_Blood_Out	1	The_Blues_Brothers	
District_9		The_Right_Stuff	
Mystic_River	1	Taken	

Ranking de Recomendação final com delta igual a 100

Titulo	Ranking
rec(batman_begins.fg2.christopher_nolan.novo.2005.medio.128.usa.english.3.pg13)	1
rec(raiders_of_the_lost_ark.fg2.steven_spielberg.classico.1981.medio.115.usa.english.2.pg)	2
rec(serenity.fg2.joss_whedon.novo.2005.medio.119.usa.english.3.pg13)	3
rec(jaws.fg4.steven_spielberg.antigo.1975.medio.130.usa.english.2.pg)	4
rec(the_prestige.fg13.christopher_nolan.novo.2006.medio.130.usa.english.3.pg13)	5
rec(the_martian.fg4.ridley_scott.novo.2015.longo.151.usa.english.3.pg13)	6
rec(casino.fg5.martin_scorsese.classico.1995.longo.178.usa.english.4.r)	7
rec(goodfellas.fg5.martin_scorsese.classico.1990.longo.146.usa.english.4.r)	8
rec(raging_bull.fg5.martin_scorsese.classico.1980.medio.121.usa.english.4.r)	9
rec(the_departed.fg7.martin_scorsese.novo.2006.longo.151.usa.english.4.r)	10
rec(baahubali_the_beginning.fg2.ss_rajamouli.novo.2015.longo.159.india.telugu.3.pg13)	11
rec(batman_the_dark_knight_returns_part_2.fg2.jay_oliva.novo.2013.longo.148.usa.english.3.pg13)	12
rec(godzilla_resurgence.fg2.hideaki_anno.novo.2016.medio.120.japan.japanese.3.pg13)	13
rec(the_bourne_ultimatum.fg2.paul_greengrass.novo.2007.medio.115.usa.english.3.pg13)	14
rec(apocalypse_now.fg13.francis_ford_coppola.antigo.1979.longo.289.usa.english.4.r)	15
rec(reservoir_dogs.fg7.quentin_tarantino.classico.1992.curto.99.usa.english.4.r)	16
rec(akira.fg2.katsuhiro_otomo.classico.1988.medio.124.japan.japanese.4.r)	17
rec(aliens.fg2.james_cameron.classico.1986.longo.154.usa.english.4.r)	18
rec(terminator_2_judgment_day.fg2.james_cameron.classico.1991.longo.153.usa.english.4.r)	19
rec(the_blues_brothers.fg2.john_landis.classico.1980.longo.148.usa.english.4.r)	20
rec(the_matrix.fg2.lana_wachowski.classico.1999.medio.136.usa.english.4.r)	21
rec(the_terminator.fg2.james_cameron.classico.1984.curto.107.uk.english.4.r)	22
rec(cinderella_man.fg5.ron_howard.novo.2005.longo.144.usa.english.3.pg13)	23
rec(incendies.fg13.denis_villeneuve.novo.2010.medio.139.canada.french.4.r)	24
rec(seven_samurai.fg2.akira_kurosawa.antigo.1954.longo.202.japan.japanese.2.pg)	25
rec(the_wild_bunch.fg2.sam_peckinpah.antigo.1969.longo.144.usa.english.4.r)	26
rec(stand_by_me.fg4.rob_reiner.classico.1986.curto.89.usa.english.4.r)	27
rec(barry_lyndon.fg4.stanley_kubrick.antigo.1975.longo.184.uk.english.2.pg)	28
rec(lawrence_of_arabia.fg4.david_lean.antigo.1962.longo.227.uk.english.2.pg)	29
rec(the_bridge_on_the_river_kwai.fg4.david_lean.antigo.1957.longo.161.uk.english.2.pg)	30
rec(the_shining.fg13.stanley_kubrick.classico.1980.longo.146.usa.english.4.r)	31
rec(in_cold_blood.fg5.richard_brooks.antigo.1967.medio.134.usa.english.3.pg13)	32
rec(rosemarys_baby.fg13.roman_polanski.antigo.1968.medio.136.usa.english.4.r)	33
rec(memento.fg14.christopher_nolan.classico.2000.curto.113.usa.english.4.r)	34
rec(gone_girl.fg7.david_fincher.novo.2014.longo.149.usa.english.4.r)	35
rec(big_fish.fg4.tim_burton.novo.2003.medio.125.usa.english.3.pg13)	36
rec(howls_moving_castle.fg3.hayao_miyazaki.novo.2004.medio.119.japan.japanese.2.pg)	37
rec(my_name_is_khan.fg4.karan_johar.novo.2010.medio.128.india.hindi.3.pg13)	38
rec(persepolis.fg3.vincent_paronnaud.novo.2007.curto.89.france.french.3.pg13)	39
rec(spirited_away.fg3.hayao_miyazaki.novo.2001.medio.125.japan.japanese.2.pg)	40
rec(the_grand_budapest_hotel.fg4.wes_anderson.novo.2014.curto.99.usa.english.4.r)	41
rec(the_revenant.fg4.alejandro_g_inarritu.novo.2015.longo.156.usa.english.4.r)	42
rec(touching_the_void.fg4.kevin_macdonald.novo.2003.curto.106.uk.english.4.r)	43

Titulo	Ranking
rec(waltz_with_bashir.fg3.ari_folman.novo.2008.curto.90.israel.hebrew.4.r)	44
rec(se7en.fg7.david_fincher.classico.1995.medio.127.usa.english.4.r)	45
rec(you_cant_take_it_with_you.fg6.frank_capra.antigo.1938.medio.126.usa.english.2.pg)	46
rec(a_separation.fg13.asghar_farhadi.novo.2011.medio.123.iran.persian.3.pg13)	47
rec(before_sunset.fg13.richard_linklater.novo.2004.curto.80.usa.english.4.r)	48
rec(captain_phillips.fg5.paul_greengrass.novo.2013.medio.134.usa.english.3.pg13)	49
rec(dallas_buyers_club.fg5.jeanmarc_vallee.novo.2013.medio.117.usa.english.4.r)	50
rec(donnie_darko.fg13.richard_kelly.novo.2001.medio.133.usa.english.4.r)	51
rec(downfall.fg5.oliver_hirschbiegel.novo.2004.longo.178.germany.german.4.r)	52
rec(million_dollar_baby.fg13.clint_eastwood.novo.2004.medio.132.usa.english.3.pg13)	53
rec(mulholland_drive.fg13.david_lynch.novo.2001.longo.147.france.english.4.r)	54
rec(oldboy.fg13.chanwook_park.novo.2003.medio.120.south_korea.korean.4.r)	55
rec(slumdog_millionaire.fg13.danny_boyle.novo.2008.medio.120.uk.english.4.r)	56
rec(straight_outta_compton.fg5.f_gary_gray.novo.2015.longo.167.usa.english.4.r)	57
rec(the_act_of_killing.fg5.joshua_oppenheimer.novo.2012.curto.96.uk.indonesian.4.r)	58
rec(the_lives_of_others.fg13.florian_henckel_von.novo.2006.medio.137.germany.german.4.r)	59
rec(the_man_from_earth.fg13.richard_schenkman.novo.2007.curto.87.usa.english.2.pg)	60
rec(the_notebook.fg13.nick_cassavetes.novo.2004.medio.123.usa.english.3.pg13)	61
rec(the_pursuit_of_happyness.fg5.gabriele_muccino.novo.2006.medio.117.usa.english.3.pg13)	62
rec(the_return.fg13.andrey_zvyagintsev.novo.2003.curto.99.russia.russian.2.pg)	63
rec(the_sea_inside.fg5.alejandro_amenabar.novo.2004.medio.125.spain.spanish.3.pg13)	64
rec(the_secret_in_their_eyes.fg13.juan_jose_campanella.novo.2009.medio.129.argentina.spanish.4.r)	65
rec(tupac_resurrection.fg5.lauren_lazin.novo.2003.curto.112.usa.english.4.r)	66
rec(twelve_years_a_slave.fg5.steve_mcqueen.novo.2013.medio.134.usa.english.4.r)	67
rec(walk_the_line.fg5.james_mangold.novo.2005.longo.153.usa.english.3.pg13)	68
rec(warrior.fg13.gavin_oconnor.novo.2011.medio.140.usa.english.3.pg13)	69
rec(almost_famous.fg4.cameron_crowe.classico.2000.longo.152.usa.english.4.r)	70
rec(das_boot.fg4.wolfgang_petersen.classico.1981.longo.293.west_germany.german.4.r)	71
rec(princess_mononoke.fg3.hayao_miyazaki.classico.1997.medio.134.japan.japanese.3.pg13)	72
rec(the_right_stuff.fg4.philip_kaufman.classico.1983.longo.193.usa.english.2.pg)	73
rec(monty_python_and_the_holy_grail.fg4.terry_gilliam.antigo.1975.curto.91.uk.english.2.pg)	74
rec(the_great_escape.fg4.john_sturges.antigo.1963.longo.172.usa.english.2.pg)	75
rec(the_wizard_of_oz.fg4.victor_fleming.antigo.1939.curto.102.usa.english.1.g)	76
rec(the_dark_knight_rises.fg1.christopher_nolan.novo.2012.longo.164.usa.english.3.pg13)	77
rec(saving_private_ryan.fg1.steven_spielberg.classico.1998.longo.169.usa.english.4.r)	78
rec(amores_perros.fg13.alejandro_g_inarritu.classico.2000.medio.115.mexico.spanish.4.r)	79
rec(before_sunrise.fg13.richard_linklater.classico.1995.curto.105.usa.english.4.r)	80
rec(brazil.fg13.terry_gilliam.classico.1985.longo.142.uk.english.4.r)	81
rec(gandhi.fg5.richard_attenborough.classico.1982.longo.240.uk.english.2.pg)	82
rec(glory.fg13.edward_zwick.classico.1989.medio.122.usa.english.4.r)	83
rec(jfk.fg13.oliver_stone.classico.1991.longo.206.france.english.4.r)	84
rec(lion_of_the_desert.fg5.moustapha_akkad.classico.1980.longo.156.libya.english.2.pg)	85
rec(the_elephant_man.fg5.david_lynch.classico.1980.medio.124.usa.english.2.pg)	86
rec(the_insider.fg5.michael_mann.classico.1999.longo.157.usa.english.4.r)	87
rec(the_straight_story.fg5.david_lynch.classico.1999.curto.112.france.english.1.g)	88
rec(unforgiven.fg13.clint_eastwood.classico.1992.medio.131.usa.english.4.r)	89
rec(casablanca.fg13.michael_curtiz.antigo.1942.curto.82.usa.english.2.pg)	90

Titulo	Ranking
rec(days_of_heaven.fg13.terrence_malick.antigo.1978.curto.94.usa.english.2.pg)	91
rec(gone_with_the_wind.fg13.victor_fleming.antigo.1939.longo.226.usa.english.1.g)	92
rec(judgment_at_nuremberg.fg13.stanley_kramer.antigo.1961.longo.186.usa.english.3.pg13)	93
rec(metropolis.fg13.fritz_lang.antigo.1927.longo.145.germany.german.2.pg)	94
rec(network.fg13.sidney_lumet.antigo.1976.medio.121.usa.english.4.r)	95
rec(night_of_the_living_dead.fg13.george_a_romero.antigo.1968.curto.96.usa.english.3.pg13)	96
rec(nothing_but_a_man.fg13.michael_roemer.antigo.1964.curto.95.usa.english.2.pg)	97
rec(patton.fg5.franklin_j_schaffner.antigo.1970.longo.172.usa.english.2.pg)	98
rec(rebecca.fg13.alfred_hitchcock.antigo.1940.medio.130.usa.english.2.pg)	99
rec(rocky.fg13.john_g_avildsen.antigo.1976.longo.145.usa.english.2.pg)	100
rec(the_best_years_of_our_lives.fg13.william_wyler.antigo.1946.longo.172.usa.english.2.pg)	101
rec(the_big_parade.fg13.king_vidor.antigo.1925.longo.151.usa.english.2.pg)	102
rec(the_deer_hunter.fg13.michael_cimino.antigo.1978.longo.183.uk.english.4.r)	103
rec(the_hustler.fg13.robert_rossen.antigo.1961.medio.134.usa.english.2.pg)	104
rec(the_sound_of_music.fg5.robert_wise.antigo.1965.longo.174.usa.english.1.g)	105
rec(the_truman_show.fg6.peter_weir.classico.1998.curto.103.usa.english.2.pg)	106
rec(city_of_god.fg7.fernando_meirelles.novo.2002.medio.135.brazil.portuguese.4.r)	107
rec(dr_strangelove_....fg6.stanley_kubrick.antigo.1964.curto.95.usa.english.2.pg)	108
rec(the_sting.fg6.george_roy_hill.antigo.1973.medio.129.usa.english.2.pg)	109
rec(gladiator.fg1.ridley_scott.classico.2000.longo.171.usa.english.4.r)	110
rec(american_history_x.fg7.tony_kaye.classico.1998.curto.101.usa.english.4.r)	111
rec(la_confidential.fg7.curtis_hanson.classico.1997.medio.138.usa.english.4.r)	112
rec(once_upon_a_time_in_america.fg7.sergio_leone.classico.1984.longo.251.italy.english.4.r)	113
rec(scarface.fg7.brian_de_palma.classico.1983.longo.142.usa.english.4.r)	114
rec(the_green_mile.fg7.frank_darabont.classico.1999.longo.189.usa.english.4.r)	115
rec(the_silence_of_the_lambs.fg7.jonathan_demme.classico.1991.medio.138.usa.english.4.r)	116
rec(the_usual_suspects.fg7.bryan_singer.classico.1995.curto.106.usa.english.4.r)	117
rec(magnolia.fg11.paul_thomas_anderson.classico.1999.longo.188.usa.english.4.r)	118
rec(requiem_for_a_dream.fg11.darren_aronofsky.classico.2000.curto.102.usa.english.4.r)	119
rec(on_the_waterfront.fg7.elia_kazan.antigo.1954.curto.108.usa.english.2.pg)	120
rec(pandoras_box.fg7.georg_wilhelm_pabst.antigo.1929.curto.110.germany.german.4.r)	121
rec(twelve_angry_men.fg7.sidney_lumet.antigo.1957.curto.96.usa.english.2.pg)	122
rec(one_flew_over_the_cuckoos_nest.fg11.milos_forman.antigo.1975.medio.133.usa.english.4.r)	123
rec(growing_up_smith.fg6.frank_lotito.novo.2015.curto.102.usa.english.3.pg13)	124
rec(in_bruges.fg6.martin_mcdonagh.novo.2008.curto.107.uk.english.4.r)	125
rec(lage_raho_munna_bhai.fg6.rajkumar_hirani.novo.2006.longo.144.india.hindi.3.pg13)	126
rec(rang_de_basanti.fg6.rakeysh_omprakash_mehra.novo.2006.longo.157.india.hindi.2.pg)	127
rec(shaun_of_the_dead.fg6.edgar_wright.novo.2004.curto.99.uk.english.4.r)	128
rec(rush.fg1.ron_howard.novo.2013.medio.123.uk.english.4.r)	129
rec(groundhog_day.fg6.harold_ramis.classico.1993.curto.101.usa.english.2.pg)	130
rec(lock_stock_and_two_smoking_barrels.fg6.guy_ritchie.classico.1998.medio.120.uk.english.4.r)	131
rec(snatch.fg6.guy_ritchie.classico.2000.curto.104.uk.english.4.r)	132
rec(the_big_lebowski.fg6.joel_coen.classico.1998.medio.117.usa.english.4.r)	133
rec(the_thing.fg14.john_carpenter.classico.1982.curto.109.usa.english.4.r)	134
rec(annie_hall.fg6.woody_allen.antigo.1977.curto.93.usa.english.2.pg)	135
rec(high_noon.fg14.fred_zinnemann.antigo.1952.curto.85.usa.english.2.pg)	136
rec(modern_times.fg6.charles_chaplin.antigo.1936.curto.87.usa.english.1.g)	137
rec(psycho.fg14.alfred_hitchcock.antigo.1960.curto.108.usa.english.4.r)	138
rec(singing_in_the_rain.fg6.stanley_donen.antigo.1952.curto.103.usa.english.1.g)	139
rec(some_like_it_hot.fg6.billy_wilder.antigo.1959.medio.120.usa.english.2.pg)	140

Titulo	Ranking
rec(the_apartment.fg6.billy_wilder.antigo.1960.medio.125.usa.english.2.pg)	141
rec(young_frankenstein.fg6.mel_brooks.antigo.1974.curto.106.usa.english.2.pg)	142
rec(boyhood.fg11.richard_linklater.novo.2014.longo.165.usa.english.4.r)	143
rec(gran_torino.fg11.clint_eastwood.novo.2008.medio.116.usa.english.4.r)	144
rec(hachi_a_dogs_tale.fg12.lasse_hallstrom.novo.2009.curto.93.usa.english.1.g)	145
rec(pans_labyrinth.fg12.guillermo_del_toro.novo.2006.curto.112.spain.spanish.4.r)	146
rec(room.fg11.lenny_abrahamson.novo.2015.medio.118.ireland.english.4.r)	147
rec(the_help.fg11.tate_taylor.novo.2011.longo.146.usa.english.3.pg13)	148
rec(the_hunt.fg11.thomas_vinterberg.novo.2012.medio.115.denmark.danish.4.r)	149
rec(american_beauty.fg11.sam_mendes.classico.1999.medio.122.usa.english.4.r)	150
rec(central_station.fg11.walter_salles.classico.1998.curto.113.brazil.portuguese.4.r)	151
rec(children_of_heaven.fg12.majid_majidi.classico.1997.curto.89.iran.persian.2.pg)	152
rec(good_will_hunting.fg11.gus_van_sant.classico.1997.medio.126.usa.english.4.r)	153
rec(sling_blade.fg11.billy_bob_thornton.classico.1996.longo.148.usa.english.4.r)	154
rec(the_celebration.fg11.thomas_vinterberg.classico.1998.curto.105.denmark.danish.4.r)	155
rec(trainspotting.fg11.danny_boyle.classico.1996.curto.94.uk.english.4.r)	156
rec(a_streetcar_named_desire.fg11.elia_kazan.antigo.1951.medio.125.usa.english.2.pg)	157
rec(fiddler_on_the_roof.fg12.norman_jewison.antigo.1971.longo.181.usa.english.1.g)	158
rec(ordet.fg12.carl_theodor_dreyer.antigo.1955.medio.126.denmark.danish.2.pg)	159
rec(the_conformist.fg11.bernardo_bertolucci.antigo.1970.curto.106.italy.italian.4.r)	160
rec(the_lost_weekend.fg11.billy_wilder.antigo.1945.curto.101.usa.english.2.pg)	161
rec(airlift.fg1.raja_menon.novo.2016.medio.130.india.hindi.2.pg)	162
rec(elite_squad.fg1.jose_padilha.novo.2007.medio.115.brazil.portuguese.4.r)	163
rec(light_from_the_darkroom.fg1.lance_mcdaniel.novo.2014.curto.90.usa.english.3.pg13)	164
rec(queen_of_the_mountains.fg1.sadyk_sherniyaz.novo.2014.medio.135.kyrgyzstan.english.3.pg13)	165
rec(tae_guk_gi_the_brotherhood_of_war.fg1.jekyu_kang.novo.2004.longo.148.south_korea.korean.4.r)	166
rec(taken.fg1.pierre_morel.novo.2008.curto.93.france.english.3.pg13)	167
rec(v_for_vendetta.fg1.james_mcteigue.novo.2005.medio.132.usa.english.4.r)	168
rec(die_hard.fg1.john_mctiernan.classico.1988.medio.131.usa.english.4.r)	169
rec(a_fistful_of_dollars.fg1.sergio_leone.antigo.1964.curto.99.italy.italian.4.r)	170
rec(bowling_for_columbine.fg8.michael_moore.novo.2002.medio.120.germany.english.4.r)	171
rec(inside_job.fg8.charles_ferguson.novo.2010.curto.105.usa.english.3.pg13)	172
rec(lake_of_fire.fg9.tony_kaye.novo.2006.longo.152.usa.english.4.r)	173
rec(peace_propaganda__the_promised_land.fg9.sut_jhally.novo.2004.curto.80.usa.english.2.pg)	174
rec(rise_of_the_entrepreneur_the_search....fg9.joe_kenemore.novo.2014.curto.52.usa.english.1.g)	175
rec(hoop_dreams.fg10.steve_james.classico.1994.longo.170.usa.english.3.pg13)	176
rec(in_the_shadow_of_the_moon.fg10.david_sington.novo.2007.curto.100.uk.english.2.pg)	177
rec(no_end_in_sight.fg10.charles_ferguson.novo.2007.curto.102.usa.english.2.pg)	178
rec(once_upon_a_time_in_the_west.fg15.sergio_leone.antigo.1968.longo.145.italy.english.3.pg13)	179
rec(samsara.fg10.ron_fricke.novo.2011.curto.102.usa.none.3.pg13)	180
rec(sicko.fg10.michael_moore.novo.2007.medio.123.usa.english.3.pg13)	181
rec(the_good_the_bad_and_the_ugly.fg15.sergio_leone.antigo.1966.longo.142.italy.italian.3.pg13)	182
rec(the_last_waltz.fg10.martin_scorsese.antigo.1978.medio.117.usa.english.2.pg)	183
rec(the_other_dream_team.fg10.marius_a_markevicius.novo.2012.curto.89.usa.english.2.pg)	184
rec(u2_3d.fg10.catherine_owens.novo.2007.curto.85.usa.english.1.g)	185
rec(woodstock.fg10.michael_wadleigh.antigo.1970.longo.215.usa.english.4.r)	186