# Projeto

## Sistemas Baseados em Conhecimento (MAC0444)

Anderson Andrei da Silva
NUSP: 8944025

Nícolas Nogueira Lopes da Silva
NUSP: 9277541

Renato Lui Geh
NUSP: 8536030

Victor Domiciano
NUSP: 8641963

# 1 Suposições

Neste projeto tivemos de assumir certas suposições com relação ao *dataset*. Para construir o OWL, usamos o arcabouço `imdbpy`.

## 1.1 Filmes

Ao analisar os dados do dataset do IMDb, notamos que haviam vários títulos que tinham uma definição diferente de filme. Séries e miniséries de TV são consideradas pelo IMDb como filmes. Similarmente *shows* de televisão, como a apresentação dos prêmios Oscar e Emmy também entram na mesma categoria de filmes. Apesar do IMDb considerar diferentes tipos de filmes, como `tv movie` e `movie`, essa diferenciação ainda não é uma boa classificação, já que haviam exceções a regra.

Para definirmos de forma mais rigorosa o que é um filme, tivemos que supor alguns conceitos sobre filmes. Em particular, consideramos um filme um título do IMDb que segue as seguintes regras:

1. Não tem "episódios";
2. Tem um conjunto de "diretores";
3. Tem ano de lançamento;
4. Tem um conjunto de "atores";
5. É do tipo `movie`.

Com estas fortes restrições, foi possível selecionar um conjunto consistente de filmes.

Na ontologia, chamamos de `Movie` o conceito de filme. Definimos um `Movie` como uma subclasse de `Project`, conceito da ontologia `FOAF-modified` dada. Todo `Movie` possui as propriedades de dado `movieTitle` do tipo `xsd:string` e `releaseYear` do tipo `xsd:positiveInteger`. Estas propriedades representam, respectivamente, o título do filme e o ano de lançamento.

Chamaremos de `projeto` o prefixo da ontologia que criamos, e `foaf-modified` a ontologia `FOAF-modified` dada no enunciado. A relação (i.e. propriedade de objeto) `foaf-modified:maker` de um `foaf-modified:Project` foi usada para denotar todos os colaboradores (i.e. atores e diretores) de um `projeto:Movie`.

## 1.2 Atores e diretores

Para os conceitos `Actor` e `Director`, usamos o conceito `foaf-modified:Person` como superclasse para ambos. As propriedades de dado de ambas são `familyName`, `firstName` e `gender`, todas da ontologia `foaf-modified`. Estas representam, respectivamente, o sobrenome, primeiro nome e gênero de um `Person`. Para `gender`, estabelecemos que os possíveis valores poderiam ser "`male`", "`female`" ou "`nil`".

Ao criar o `.owl` a partir do dataset, fizemos as seguintes suposições sobre as pessoas:

1. O primeiro nome de uma pessoa no IMDb é o seu `firstName`;
2. O último nome de uma pessoa no IMDb é o seu `familyName`;
3. O gênero de uma pessoa foi definido a partir de consultas no `imdbpy`.

Com relação a suposição 3, o dataset do IMDb não especifica gênero de uma pessoa. No entanto, o `imdbpy` gera uma tabela em SQL com os gêneros de cada ator, tendo estes valores `'f'`, `'m'` ou um valor nulo quando não especificado. Escolhemos seguir a mesma convenção neste projeto.

A relação `foaf-modified:made` mapeia um `Person` em um `Movie`. Seu inverso é `foaf-modified:maker`. Criamos duas subrelações de `made`: `actsIn` e `directs`, mapeando, respectivamente, um ator em um filme que ele atua e um diretor em um filme que ele dirige.

Para selecionarmos a lista de pessoas iniciais (i.e. Uma Thurman, Harvey Keitel, etc.), assumimos que a pessoa de interesse é o primeiro resultado da busca de seu nome concatenado com "`(I)`". Isso garante que o homônimo escolhido é sempre o primeiro em popularidade no IMDb.

## 1.3 Relações

Resumimos as relações existentes na tabela abaixo:

| Nome | Domínio | Imagem | Superclasse |
|---|---|---|---|
| made | Person | Movie | - |
| maker | Movie | Person | - |
| actsIn | Actor | Movie | made |
| directs | Director | Movie | made |

**Tabela 1** Relações entre conceitos. As colunas indicam o nome da relação, o domínio e imagem, e a superclasse da relação. Como `made` e `maker` têm como superclasse a raíz comum de todas as relações, omitimos e no lugar indicamos com o caractere '-'.

## 1.4 Instâncias

Foram escolhidas as instâncias de `Movie`, `Actor` e `Director` como descrito no enunciado. Depois de aplicadas as restrições descritas nas subseções 1.1 e 1.2, geramos o `.owl` com a sintaxe Manchester.

Cada instância de `Movie` é do formato:

```
1  Individual: projeto:movieTitle
2
3    Types:
4      projeto:Movie
5
6    Facts:
7      projeto:movieTitle "Movie Title",
8      foaf-modified:maker projeto:director1,
9      ⋮
10     foaf-modified:maker projeto:directorN,
11     foaf-modified:maker projeto:actor1,
12     ⋮
13     foaf-modified:maker projeto:actorM,
14     projeto:releaseYear Y
```

Onde `Y` é um inteiro positivo. A lista de atores e diretores não são ordenadas como no exemplo. Para uma instância de `Actor` ou `Director` temos o seguinte formato:

```
Individual: projeto:personName

  Types:
    projeto:Actor,
    projeto:Director

  Facts:
    projeto:actsIn projeto:movie1,
    ⋮
    projeto:actsIn projeto:movieN,
    projeto:directs projeto:movie1,
    ⋮
    projeto:directs projeto:movieM,
    foaf-modified:familyName "Name",
    foaf-modified:firstName "Person",
    foaf-modified:gender ["male" ou "female" ou "nil"]
```

Para todas as instâncias, o nome da instância é baseado no seu título ou nome. Usamos o formato conhecido como `camelCase` para formatar os nomes das instâncias. Para atores que não são diretores, omitimos a vírgula e a linha `projeto:Director` em `Types:`. Fazemos o análogo para diretores que não atuam em nenhum filme.

## 1.5 Unicidade

Quando definimos os nomes dos atores e diretores, consideramos que se alguma pessoa $p$ possui mesmo `firstName` e `familyName` que outra pessoa $q$, então $p = q$. Isso simplifica a ontologia, tratando homônimos como a mesma pessoa. Também supomos o análogo para filmes. Se um filme $m$ tem mesmo título que $n$, então $m = n$. Apesar de não ser verdade, esta suposição simplifica bastante a ontologia.

Uma consequência das suposições anteriores é que não precisamos usar a propriedade `DifferentFrom` do OWL, evitando que o arquivo cresça exponencialmente, já que toda junção de primeiro com último nome será diferente. Da mesma forma, todo título de filme será distinto, e portanto todo filme é único.

# 2 Consultas

Neste projeto tivemos que realizar consultas em SPARQL para responder as seguintes questões:

1. Quais atores participaram do filme F?
2. Quais filmes foram dirigidos pelo diretor D?
3. Em quais filmes o ator X atuou?
4. Em quais filmes o ator X atuou junto com Y?
5. Quem foram os diretores dos filmes nos quais os atores X e Y atuam juntos?
6. Qual o diretor que mais dirigiu filmes do ator X?
7. Qual o ator que mais aparece nos filmes do diretor D?
8. Entre os anos N1 e N2, quais diretores dirigiram filmes onde X e Y aparecem?
9. Entre os anos N1 e N2, quais atores atuaram juntos nos filmes onde X e Y aparecem?
10. Quais filmes do diretor do filme F possuem X ou Y como atores?

Obs.: Para estas consultas, introduzimos `FILTER`s para selecionar os resultados desejados, de forma que o filme (`Movie`) é introduzido pelo seu título `projeto:movieTitle` no filtro em questão. Para introduzir uma pessoa (`Actor` ou `Director`) utilizamos dois filtros combinados, um para seu primeiro nome `foaf-modified:firstName` e outro para seu segundo nome `foaf-modified:familyName`.

## 2.1 Consulta 1

Quais atores participaram do filme F `(?f = "Isle of Dogs")`?

```
1 SELECT ?1x ?2x
2 WHERE {
3   ?x projeto:actsIn ?m .
4   ?m projeto:movieTitle ?f .
5   ?x foaf-modified:firstName ?1x .
6   ?x foaf-modified:familyName ?2x .
7   FILTER REGEX(?f, "(^Isle of Dogs$)") .
8 }
```

## 2.2 Consulta 2

Quais filmes foram dirigidos pelo diretor D (`?1d = "Wes" ?2d = "Anderson "`)?

```
SELECT ?f
WHERE {
  ?d projeto:directs ?m .
  ?m projeto:movieTitle ?f .
  ?d foaf-modified:firstName ?1d .
  ?d foaf-modified:familyName ?2d .
  FILTER REGEX(?1d, "(^Wes?)") .
  FILTER REGEX(?2d, "(^Anderson?)") .
}
```

## 2.3 Consulta 3

Em quais filmes o ator X (`?1x = "Samuel" ?2x = "Jackson"`) atuou?

```
SELECT ?f
WHERE {
  ?x projeto:actsIn ?m .
  ?m projeto:movieTitle ?f .
  ?x foaf-modified:firstName ?1x .
  ?x foaf-modified:familyName ?2x .
  FILTER REGEX(?1x, "(^Samuel?)") .
  FILTER REGEX(?2x, "(^Jackson?)") .
}
```

## 2.4 Consulta 4

Em quais filmes o ator X (`?1x = "Uma" ?2x = "Thurman"`) atuou junto com Y (`?1y = "Samuel" ?2y = "Jackson"`)?

```
SELECT DISTINCT ?f
WHERE {
  ?x projeto:actsIn ?m .
  ?x foaf-modified:firstName ?1x .
  ?x foaf-modified:familyName ?2x .
  FILTER REGEX(?1x, "(^Uma?)") .
  FILTER REGEX(?2x, "(^Thurman?)") .
  ?y projeto:actsIn ?m .
```

```
  ?m projeto:movieTitle ?f .
  ?y foaf-modified:firstName ?1y .
  ?y foaf-modified:familyName ?2y .
  FILTER REGEX(?1y, "(^Samuel?)") .
  FILTER REGEX(?2y, "(^Jackson?)") .
}
```

### 2.5 Consulta 5

Quem foram os diretores dos filmes nos quais os atores X (`?1x = "Uma"` `?2x = "Thurman"`) e Y (`?1y = "Samuel"` `?2y = "Jackson"`) atuam juntos?

```
SELECT DISTINCT ?1d ?2d
WHERE {
  ?x projeto:actsIn ?m .
  ?x foaf-modified:firstName ?1x .
  ?x foaf-modified:familyName ?2x .
  FILTER REGEX(?1x, "(^Uma?)") .
  FILTER REGEX(?2x, "(^Thurman?)") .
  ?y projeto:actsIn ?m .
  ?y foaf-modified:firstName ?1y .
  ?y foaf-modified:familyName ?2y .
  FILTER REGEX(?1y, "(^Samuel?)") .
  FILTER REGEX(?2y, "(^Jackson?)") .
  ?d projeto:directs ?m .
  ?d foaf-modified:firstName ?1d .
  ?d foaf-modified:familyName ?2d .
}
```

### 2.6 Consulta 6

Qual o diretor que mais dirigiu filmes do ator X (`?1x = "Frances"` `?2x = "Mcdormand"`)?

```
SELECT ?1d ?2d WHERE {
  {
    SELECT ?d (COUNT(?d) AS ?count)
    WHERE {
      ?x projeto:actsIn ?m .
      ?x foaf-modified:firstName ?1x .
      ?x foaf-modified:familyName ?2x .
      FILTER REGEX(?1x, "(^Frances?)") .
```

```
      FILTER REGEX(?2x, "(^Mcdormand?)") .
      ?d projeto:directs ?m .
    } GROUP BY ?d
  }
  {
    SELECT (MAX(?count) AS ?max)
    WHERE {
      {
        SELECT ?d (COUNT(?d) AS ?count)
        WHERE {
          ?x projeto:actsIn ?m .
          ?x foaf-modified:firstName ?1x .
          ?x foaf-modified:familyName ?2x .
          FILTER REGEX(?1x, "(^Frances?)") .
          FILTER REGEX(?2x, "(^Mcdormand?)") .
          ?d projeto:directs ?m .
        } GROUP BY ?d
      }
    }
  }
  FILTER (?count = ?max)
  ?d foaf-modified:firstName ?1d .
  ?d foaf-modified:familyName ?2d .
}
```

## 2.7 Consulta 7

Qual o ator que mais aparece nos filmes do diretor D (`?1d = "Wes" ?2d = "Anderson"`)?

```
SELECT ?1x ?2x WHERE {
  {
    SELECT ?x (COUNT(?x) AS ?count)
    WHERE {
      ?d projeto:directs ?m .
      ?d foaf-modified:firstName ?1d .
      ?d foaf-modified:familyName ?2d .
      FILTER REGEX(?1d, "(^Wes?)") .
      FILTER REGEX(?2d, "(^Anderson?)") .
      ?x projeto:actsIn ?m .
    } GROUP BY ?x
  }
  {
    SELECT (MAX(?count) AS ?max)
```

```
    WHERE {
      {
        SELECT ?x (COUNT(?x) AS ?count)
        WHERE {
          ?d projeto:directs ?m .
          ?d foaf-modified:firstName ?1d .
          ?d foaf-modified:familyName ?2d .
          FILTER REGEX(?1d, "(^Wes?)") .
          FILTER REGEX(?2d, "(^Anderson?)") .
          ?x projeto:actsIn ?m .
        } GROUP BY ?x
      }
    }
  }
  FILTER (?count = ?max) .
  ?x foaf-modified:firstName ?1x .
  ?x foaf-modified:familyName ?2x .
}
```

### 2.8 Consulta 8

Entre os anos N1 e N2, quais diretores dirigiram filmes onde X (`?1x = "Uma" ?2x = "Thurman"`) e Y (`?1y = "Samuel" ?2y = "Jackson"`) aparecem?

```
SELECT DISTINCT ?1d ?2d
WHERE {
  ?x projeto:actsIn ?m .

  ?m projeto:releaseYear ?year .
  FILTER (?year >= 1980 && ?year <= 2001) .
  ?x foaf-modified:firstName ?1x .
  ?x foaf-modified:familyName ?2x .
  FILTER REGEX(?1x, "(^Uma?)") .
  FILTER REGEX(?2x, "(^Thurman?)") .
  ?y projeto:actsIn ?m .
  ?y foaf-modified:firstName ?1y .
  ?y foaf-modified:familyName ?2y .
  FILTER REGEX(?1y, "(^Samuel?)") .
  FILTER REGEX(?2y, "(^Jackson?)") .
  ?d projeto:directs ?m .
  ?d foaf-modified:firstName ?1d .
  ?d foaf-modified:familyName ?2d .
}
```

## 2.9 Consulta 9

Entre os anos N1 e N2, quais atores atuaram juntos nos filmes onde X (`?1x = "Bill" ?2x = "Murray"`) e Y (`?1y = "Frank" ?2y = "Pellegrino"`) aparecem?

```
SELECT DISTINCT ?1a ?2a ?1b ?2b
WHERE {
  {
    SELECT ?m
    WHERE {
      ?x projeto:actsIn ?m .

      ?m projeto:releaseYear ?year .
      FILTER (?year >= 1980 && ?year <= 2004) .
      ?x foaf-modified:firstName ?1x .
      ?x foaf-modified:familyName ?2x .
      FILTER REGEX(?1x, "(^Bill?)") .
      FILTER REGEX(?2x, "(^Murray?)") .
      ?y projeto:actsIn ?m .
      ?y foaf-modified:firstName ?1y .
      ?y foaf-modified:familyName ?2y .
      FILTER REGEX(?1y, "(^Frank?)") .
      FILTER REGEX(?2y, "(^Pellegrino?)") .
    }
  }
  ?a projeto:actsIn ?m .
  ?b projeto:actsIn ?m .
  FILTER (?a != ?b) .
  ?a foaf-modified:firstName ?1a .
  ?a foaf-modified:familyName ?2a .
  ?b foaf-modified:firstName ?1b .
  ?b foaf-modified:familyName ?2b .
} ORDER BY ?1a ?2a ?1b ?2b
```

## 2.10 Consulta 10

Quais filmes do diretor do filme F (`?f = "Pulp Fiction"`) possuem X (`?1x = "Clive" ?2x = "Owen"`) ou Y (`?1y = "Jamie" ?2y = "Dunno"`) como atores?

```
SELECT DISTINCT ?f
WHERE {
  {
```

```
4       SELECT ?d
5       WHERE {
6         ?m projeto:movieTitle ?f .
7         FILTER REGEX(?f, "(^Pulp Fiction?)") .
8         ?d projeto:directs ?m .
9       }
10    }
11    ?d projeto:directs ?m .
12    ?a projeto:actsIn ?m .
13    ?a foaf-modified:firstName ?1a .
14    ?a foaf-modified:familyName ?2a .
15
16    ?m projeto:movieTitle ?f .
17    FILTER ((REGEX(?1a, "(^Clive$)") && REGEX(?2a, "(^
          Owen$)")) || (REGEX(?1a, "(^Jamie$)") && REGEX(?2
          a, "(^Dunno$)"))) .
18  }
```

# 3 Resultados

A seguir estão as tabelas com os resultados de cada consulta realizada anteriormente.

| **1x** | **2x** |
|---|---|
| Harvey | Keitel |
| Kunichi | Nomura |
| Jeff | Goldblum |
| Akira | Ito |
| Frances | Mcdormand |
| Akira | Takayama |
| Edward | Norton |
| Liev | Schreiber |
| Frank | Wood |
| F | Abraham |
| Bryan | Cranston |
| Yoko | Ono |
| Fisher | Stevens |
| Tilda | Swinton |
| Koyu | Rankin |
| Greta | Gerwig |
| Yojiro | Noda |
| Courtney | Vance |
| Mari | Natsuki |
| Kara | Hayward |
| Scarlett | Johansson |
| Bill | Murray |
| Bob | Balaban |

**Tabela 2** Resultado da Consulta 1

| f |
|---|
| Hotel Chevalier |
| Rushmore |
| Moonrise Kingdom Animated Book Short |
| The Life Aquatic with Steve Zissou |
| Bottle Rocket |
| Cousin Ben Troop Screening with Jason Schwartzman |
| Castello Cavalcanti |
| The Grand Budapest Hotel |
| The Royal Tenenbaums |
| Isle of Dogs |
| Prada Candy |
| The Darjeeling Limited |
| Come Together A Fashion Picture in Motion |
| Fantastic Mr Fox |
| Moonrise Kingdom |

**Tabela 3** Resultado da Consulta 2

| f |
|---|
| Inglourious Basterds |
| Jackie Brown |
| Django Unchained |
| Youre Still Not Fooling Anybody |
| Kill Bill Vol 2 |
| Boffo Tinseltowns Bombs and Blockbusters |
| Pulp Fiction |
| Off the Menu The Last Days of Chasens |
| The Hateful Eight |

**Tabela 4** Resultado da Consulta 3

| f |
|---|
| Youre Still Not Fooling Anybody |
| Kill Bill Vol 2 |
| Boffo Tinseltowns Bombs and Blockbusters |
| Pulp Fiction |

**Tabela 5** Resultado da Consulta 4

| 1d | 2d |
|---|---|
| Bill | Couturie |
| Mike | White |
| Quentin | Tarantino |

**Tabela 6** Resultado da Consulta 5

| 1d | 2d |
|---|---|
| Joel | Coen |
| Ethan | Coen |

**Tabela 7** Resultado da Consulta 6

| 1x | 2x |
|---|---|
| Jason | Schwartzman |
| Bill | Murray |

**Tabela 8** Resultado da Consulta 7

| 1d | 2d |
|---|---|
| Mike | White |
| Quentin | Tarantino |

**Tabela 9** Resultado da Consulta 8

| 1a | 2a | 1b | 2b |
|---|---|---|---|
| Alec | Baldwin | Bill | Murray |
| Alec | Baldwin | Edward | Saxon |
| Alec | Baldwin | Frank | Pellegrino |
| Bill | Murray | Alec | Baldwin |
| Bill | Murray | Edward | Saxon |
| Bill | Murray | Frank | Pellegrino |
| Edward | Saxon | Alec | Baldwin |
| Edward | Saxon | Bill | Murray |
| Edward | Saxon | Frank | Pellegrino |
| Frank | Pellegrino | Alec | Baldwin |
| Frank | Pellegrino | Bill | Murray |
| Frank | Pellegrino | Edward | Saxon |

**Tabela 10** Resultado da Consulta 9

| f |
|---|
| Grindhouse |
| Death Proof |
| Sin City |

**Tabela 11** Resultado da Consulta 10