Universidade Federal
do Rio de Janeiro
Escola Politécnica

INFOMOVIE:
SISTEMA DE EXTRAÇÃO DE INFORMAÇÃO COM INTERFACE PARA LINGUAGEM NATURAL

Lívia Pimentel Ribeiro

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Sérgio Palma da Justa Medeiros
Coorientadora: Ester José Casado de Lima de Campos

Rio de Janeiro
Abril de 2013

INFOMOVIE
SISTEMA DE EXTRAÇÃO DE INFORMAÇÃO COM INTERFACE PARA LINGUAGEM NATURAL

Lívia Pimentel Ribeiro

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinado por:

_____
Prof. Sergio Palma da Justa Medeiros, D. Sc.

_____
Eng. Ester José Casado de Lima de Campos, M. Sc.

_____
Prof. José Arthur da Rocha, M. Sc.

RIO DE JANEIRO, RJ – BRASIL
ABRIL de 2013

Ribeiro, Lívia Pimentel

InfoMovie: Sistema de Extração de Informação com Interface para Linguagem Natural/ Lívia Pimentel Ribeiro –Rio de Janeiro: UFRJ/ Escola Politécnica, 2013.

IX, 44p.: il.; 29,7 cm.

Orientadores: Sergio Palma da Justa Medeiros e Ester José Casado de Lima

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2013.

Referência Bibliográficas: p. 44-45

1. Extração de Informação 2. Interface para Linguagem Natural 3. Ontologia 4. Reconhecimento de Padrões I. Sergio Palma da Justa Medeiros II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Computação e Informação. III. Título.

À minha avó Sylvia Damasceno Ribeiro.

Agradecimentos

Agradeço à minha família, especialmente à minha mãe, Celeida Pimentel Ribeiro, e ao meu pai, Eden Damasceno Ribeiro, por todo o apoio e incentivo durante toda minha vida.

Aos meu amigos da UFRJ que estiveram comigo durante esses cinco anos de curso, principalmente Edimar Babilon, Luiz Alves, Mariane Martins, Thaiana Lima, Vanessa Marques e Vítor Silva.

Aos meu amigos que participaram do intercâmbio comigo, especialmente João Mello, Vinicius Gardelli, Isabela Gomide, Flávia Dias, Allyson Gomes, Izabella Simplício e Gabriela Macedo.

Ao meu orientador e professor Sergio Palma que é um exemplo de profissional para mim.

À minha coorientadora Ester Lima que foi fundamental para a realização deste projeto.

Ao professor e antigo coordenador do curso José Ferreira de Rezende por todo apoio e atenção durante a faculdade.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

InfoMovie
Sistema de Extração de Informação com Interface para Linguagem Natural

Lívia Pimentel Ribeiro

Março/2013

Orientador: Sergio Palma da Justa Medeiros
Coorientadora: Ester José Casado de Lima de Campos

Curso: Engenharia de Computação e Informação

A extração de informação relevante da internet é um grande desafio. Diversos sistemas de busca foram desenvolvidos, mas ainda é papel do usuário formular uma consulta adequada e filtrar os dados de forma a encontrar a informação que procura. Com o advento da web semântica e os avanços em inteligência artificial, cada vez mais procura-se realizar sistemas que tenham uma interface na qual o usuário possa se comunicar usando linguagem natural. Nesse projeto, os conceitos envolvendo o processo de criação de um sistema de extração de informação baseado em ontologia com interface para linguagem natural serão estudados e aplicados. O InfoMovie é um sistema que se propõe a demonstrar a extração de dados estruturados baseados na ontologia "cinema" e fornecer uma interface de linguagem natural para que o usuário possa encontrar informações relacionadas ao assunto escrevendo sua consulta na língua inglesa.

Palavras-chave: Extração de Informação, Interface para Linguagem Natural, Ontologia, Reconhecimento de Padrões

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

InfoMovie
Information Extraction System with Natural Language Interface

Lívia Pimentel Ribeiro

March/2013

Advisor: Sergio Palma da Justa Medeiros
Co-advisor: Ester José Casado de Lima de Campos

Major: Computer and Information Engineering

The extraction of relevant information on the internet is a big challenge. Many search engines were developed but it is still the user's role to formulate an appropriate query and to filter the data in order to find the information that he is searching for. With the advent of the semantic web and the progress in artificial intelligence, more and more efforts are being made to create systems with an interface that enables the user to communicate using natural language. In this project, the concepts involving the creation process of an ontology based information extraction system with natural language interface to databases are studied and applied. The InfoMovie is a system that proposes to demonstrate how to extract structured data based in the movie ontology and to provide a natural language interface in which the user is able to find information related to this subject by writing his query in English.

Keywords: Information Extraction, Natural Language Interface, Ontology, Pattern Matching

SUMÁRIO

SIGLAS

ERM - *Entity-Relationship Model*

HTML - *HiperText Markup Language*

MO – *Movie Ontology*

MTV - *Model-Template-View*

MVC – *Model-View-Controller*

NER **-** *Named Entity Recognition*

NLTK - *Natural Language ToolKit*

OBIE – *Ontology Based Information Extraction*

OWL - *Web Ontology Language*

POS *tagging* - *Part-Of-Speech tagging*

RDF **-** *Resource Description Format*

SQL - *Structured Query Language*

1 Introdução

1.1 Visão Geral

A internet funciona como um grande banco de dados, contendo informações sobre diversos assuntos e em grande quantidade. O maior desafio é extrair informações que são consideradas relevantes e fornecer essas informações de maneira clara aos usuários que as buscam.

Para isso, os sistemas de extração de informação têm o papel de buscar informações, organizá-las de forma estruturada e, assim, se tornam capazes de prover as informações obtidas. Para facilitar o acesso à informação, existem as interfaces de linguagem natural para banco de dados. Por linguagem natural entende-se qualquer linguagem desenvolvida pelo ser humano, por exemplo, a língua falada (inglês, português, francês, alemão, latim, etc.) e a linguagem dos sinais [1]. Em se tratando de interfaces para linguagem natural, normalmente refere-se a linguagens faladas. Essas interfaces permitem que o usuário digite a consulta desejada em alguma linguagem natural, sendo essa interpretada pelo sistema.

Apesar de a extração de informação ser um tema antigo, a extração de informação com base em ontologia, representação estruturada de conhecimento, vem sendo aprofundada nos últimos anos, sendo ainda um assunto recente e com grande potencial de crescimento.

1.2 Objetivo

O objetivo do projeto é demonstrar a viabilidade de se construir um sistema de *Query-Answering*, de forma simplificada, contendo todos os processos necessários para um sistema desse tipo: extração de informação baseada em ontologia, interpretação de consultas e geração de respostas. Baseado em sistemas como Evi [2] e Ask [3] que respondem a perguntas feitas por usuários, este projeto visa estudar os procedimentos de sistemas como esses e aplicar em uma proposta de sistema, o InfoMovie.

O projeto de *Query-Answering System* visa interpretar e responder corretamente às consultas dos usuários, relacionadas a cinema. Através da extração de informação, realizada a partir do banco de dados do site IMDb [4], o sistema organiza

as informações de forma estruturada de acordo com a ontologia de cinema MO (*The Movie Ontology*) [5].

O sistema apresenta uma interface para linguagem natural de acesso ao banco de dados, o que significa que o usuário pode digitar sua consulta livremente e o InfoMovie interpreta a consulta e retorna uma resposta correta para o usuário.

1.3 Organização

O texto do projeto está dividido em três partes. A primeira consiste nos conceitos estudados para a implementação do projeto. São eles: ontologia, extração de informação e algoritmos para interpretação de consultas em linguagem natural na língua inglesa.

A segunda parte trata das tecnologias utilizadas no desenvolvimento do sistema, incluindo Django e a ontologia Movie Ontology.

Na terceira parte, o projeto de desenvolvimento do sistema é mostrado, incluindo a modelagem de dados, a arquitetura, o algoritmo utilizado e os resultados.

Por fim, é feita uma conclusão e sugestão de trabalhos futuros.

2 Conceitos Estudados

Nesta seção serão apresentados os conceitos teóricos que foram estudados para implementação deste projeto de graduação.

2.1 Ontologia

A palavra Ontologia foi usada primeiramente no contexto da filosofia. Em 1613, foi criada por dois filósofos simultaneamente, Rudolf Gökel e Jacob Lohard. Na filosofia, ontologia é a ciência do que são os tipos de estrutura, processos e relações em cada área da existência real [6].

Em computação o termo ontologia se refere à especificação de uma conceituação [7], sendo conceituação uma forma abstrata de representar aquilo que encontramos na vida. Assim, a especificação de como uma conceituação deve ser elaborada é uma das definições de ontologia. A ontologia se refere aos atores, predicados e conceitos de uma linguagem usada em um determinado domínio de conhecimento [8]. Resumindo, é uma forma estruturada de representar conhecimento.

A ontologia pode ser usada para compartilhar conhecimento, de forma que o emissor de informação e o receptor consigam se comunicar sobre determinado assunto usando um padrão de linguagem [6]. Dessa forma, se torna possível responder a perguntas sobre determinada ontologia de forma consistente.

Segundo GRUBER [7], um agente que se compromete a uma ontologia não precisa responder a todas as consultas de um cliente, mas sim responder com consistência a elas. Devido a isso, o sistema que atua como agente, tem o dever de responder às perguntas corretamente, porém não retornará respostas para todas as perguntas feitas, ou seja, o foco do sistema é ter alta precisão.

Ontologias podem ser representadas através do modelo RDF (*Resource Description Format*), similar ao modelo ERM (*Entity-Relationship Model*). O modelo RDF é uma forma de representar declarações em forma de sujeito-predicado-objeto [9]. O RDF permite a representação estruturada de uma informação. Por exemplo, "Johnny Depp atuou no filme Edward Scissorhands" tem a representação: <Johnny Depp, isActorIn, Edward Scissorhands>, que corresponde ao modelo <sujeito, predicado, objeto>. Já no modelo ERM, "Johnny Depp" seria um objeto da classe

ator, “Edward Scissorhands”, um objeto da classe filme e “isActorIn”, um relacionamento entre ator e filme.

Como dito acima, “Johnny Depp” e “Edward Scissorhands” são entidades enquanto “isActorIn” representa o relacionamento. Uma entidade pode possuir diferentes *labels,* ou rótulos*,* e duas entidades podem ter o mesmo nome. Por exemplo, os atores Johnny Depp e Johnny Galecki, podem ser referidos por Johnny. Nesse caso teremos duas entidades com o mesmo nome, ou seja, uma ambiguidade. Mas também podemos encontrar apenas “Depp”, para se referir a “Johnny Depp”, sendo assim, “Depp” e “Johnny Depp” são sinônimos para a mesma entidade. A figura abaixo ilustra as ambiguidades e sinônimos que podem ser encontrados.

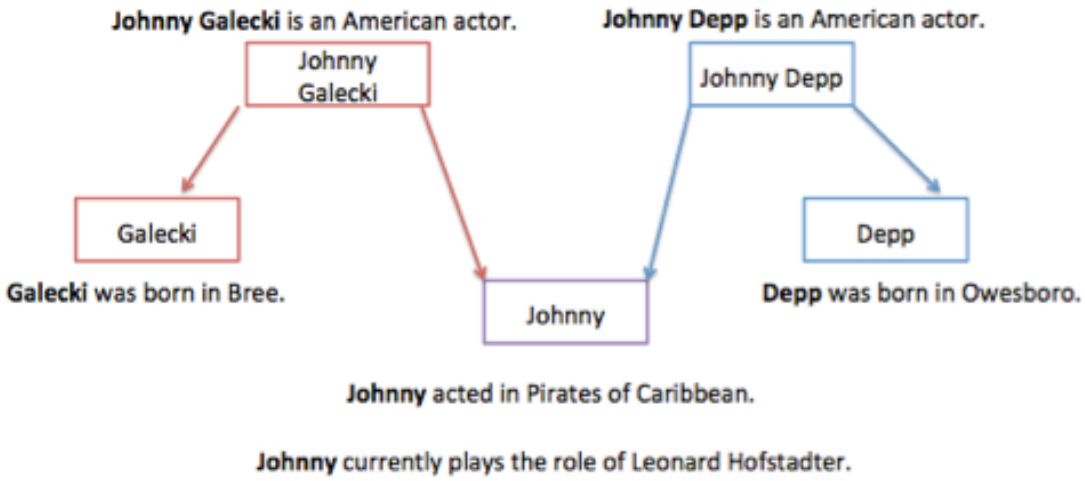

Figura 1 – Exemplo de ambiguidades e sinônimos

Na ontologia temos classes, ou conceitos, que são conjuntos de entidades similares. Temos, por exemplo, a classe ator para definir um conjunto de atores, a classe filme para definir um conjunto de filmes, a classe diretor para definir um conjunto de diretores e assim por diante. Existe também o conceito de superclasse que consiste em uma classe generalizada que contém as classes mais específicas. A classe pessoa é a superclasse de ator e diretor.

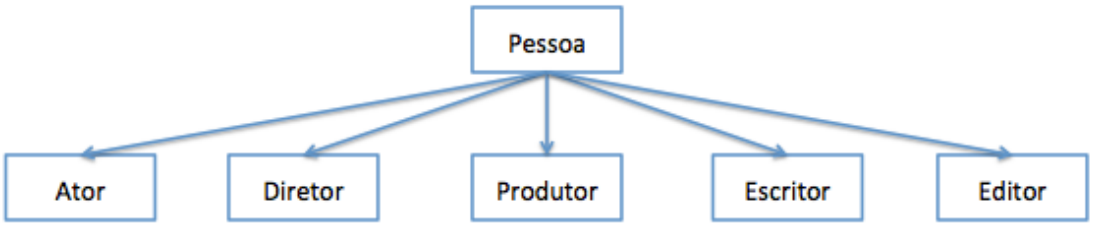

Figura 2 - Exemplo de superclasse

Uma ontologia pode ser usada para criar outra ontologia ou estender a mesma. Entenda-se por estender a obtenção de mais informações através do processo de extração de informação, que será explicado nas próximas seções. Alguns desafios desse processo são mapear as entidades extraídas com as entidades já existentes na ontologia, resolver as ambiguidades e manter a ontologia consistente, ou seja, sem informações falsas.

2.2 Extração de Informação

A extração de informação é o processo de obtenção de informação estruturada a partir de fontes que não podem ser interpretadas por máquinas, como textos [10]. Como exemplo de sistema de extração de informação, podemos ter um sistema que procure na web informações sobre uma determinada empresa, buscando em páginas de jornais, revistas e blogs. Para isso será necessário um modelo para estruturar quais informações buscar, contendo, por exemplo, empresa, número de funcionários, receita, lucro, entre outros. O sistema deve ser capaz de identificar quais informações estão dentro do campo do assunto citado e ignorar as informações irrelevantes para o estudo da empresa.

O processo de extração de informação inclui o processo de obtenção de informação, que consiste em obter os documentos que contêm as informações, e o processo de interpretar o conteúdo dos textos. Nesse último estão incluídos reconhecimento de entidades, extração de instâncias, extração de fatos e extração de informação ontológica.

2.2.1 Reconhecimento de Entidades

Do inglês, *Named Entity Recognition* (NER), o reconhecimento de entidades é o processo de extrair entidades de um texto. Por exemplo, dado o texto "Rio de Janeiro, estado localizado no sudeste do Brasil", é possível reconhecer que "Rio de Janeiro" é um estado.

Uma forma de extrair entidades é pelo processo de reconhecer padrões comuns utilizados na língua. Como no exemplo anterior: "Rio de Janeiro, estado localizado no sudeste do Brasil", reconhecemos facilmente que Rio de Janeiro é um estado. A partir do momento que se tem uma entidade reconhecida, através de regras de lógica, podemos encontrar outras entidades, como abaixo:

1. Rio de Janeiro, Brasil -> estado do Brasil
2. Rio de Janeiro e São Paulo, os maiores estados do país. -> Rio e São Paulo são estados

É possível inferir que, se Rio de Janeiro é um estado, São Paulo também é um estado, de acordo com a segunda frase. Sabendo que a classe de "Rio de Janeiro" é estado, a segunda frase nos indica que "Rio de Janeiro" e "São Paulo" são da mesma classe. Logo, São Paulo pertence à classe estado.

Da mesma forma, podemos reconhecer nomes próprios, localizações, organizações e outras entidades. Podemos encontrar também padrões que são facilmente identificados nos números, como representação de anos, sempre números de quatro dígitos, números de telefone, CEP e outros. Alguns problemas são encontrados nesse processo, como a ambiguidade. Por exemplo, tem-se "Rio de Janeiro, capital do Rio de Janeiro", nesse caso "Rio de Janeiro" representa cidade e estado, ou seja, duas entidades diferentes, escritas da mesma forma e que devem ser identificadas e classificadas corretamente.

A partir dos padrões de linguagem conhecidos, o processo para encontrá-los nos textos é feito com o uso de expressões regulares. As expressões regulares são usadas para encontrar um sequência de caracteres dentro de outra. Dado um conjunto de caracteres Σ, uma expressão regular é definida como [11]:

- Uma *string* vazia;
- Um único caractere pertencente a Σ ;
- Uma concatenação de caracteres AB, onde A e B são expressões regulares;
- A alternação de caracteres, ou seja, uma *string* na forma (A|B) onde A e B são expressões regulares;
- Uma *string* no formato (A)*, onde A é uma expressão regular.

Após a expressão definida, passamos para um compilador e em seguida fazemos a busca do padrão compilado na *string* que é passada como parâmetro. Por exemplo, se a expressão regular for (ac?|b*), as expressões “a”, “ac”, “abbb”, serão correspondentes.

Fornecendo um exemplo mais concreto, ao buscar pela data de nascimento de uma pessoa em um texto, sabemos que o padrão é “<Nome da Pessoa> *was born on* <data de nascimento>”. As datas podem seguir vários padrões, seguindo um padrão comum no inglês, “mês dia, ano”, podemos escrever a expressão regular da seguinte forma:

```
monthPattern =
"(January|February|March|April|May|June|July|August|Septe
mber|October|November|December)"

dayPattern = "([1-9]|[1|2][0-9]|30|31)"

yearPattern = "[0-9]{4}"

datePattern = "born " + monthPattern + " " + dayPattern +
", " + yearPattern
```

Ao buscar em um texto por essa expressão regular, é possível encontrar a data de nascimento da pessoa em questão. Porém, como visto acima, a data não segue um padrão universal. assim como outros padrões se diferem. Por isso, é importante normalizar o texto a fim de que os padrões sejam os mesmos ao longo dele. O processo de normalização consiste em retirar acentos, passar todos os caracteres para *lower-case* e colocar datas e outras medidas em um mesmo formato [11].

2.2.2 Extração de Instâncias

O próximo passo da extração de informação é extrair a classe de uma entidade. A obtenção de uma entidade junto com sua classe é chamada extração de instância. Para exemplificar, “Johnny Depp” pertence à classe “Ator”, “Edward Scissorhands”, à classe “Filme”, “Tim Burton”, à classe “Diretor”. A extração de instâncias muitas vezes ocorre juntamente com o reconhecimento de entidades, como no exemplo “Rio de Janeiro, estado localizado no sudeste do Brasil”. No momento em que Rio de Janeiro é reconhecido como uma entidade, sua classe é reconhecida como “Estado”.

A classe nada mais é do que um hiperônimo da entidade e a entidade um hipônimo da classe. Um hipônimo é uma palavra que é mais específica que outra palavra, no caso, a palavra menos específica é um hiperônimo. Pela visão de um falante de inglês, L0 é um hipônimo de L1 se “An L0 is a (kind of) L1” [12]. Por exemplo “**Tim Burton** is a **director**”, sendo L0, “Tim Burton” e L1, “director”.

Assim como no exemplo acima, existem outros padrões da língua inglesa que são confiáveis para a extração de instâncias. HEARST [12] criou padrões, chamados de *Hearst Patterns,* que satisfazem as seguintes condições:

1) Ocorrem com frequência em textos de diferentes gêneros;
2) Quase sempre indicam o relacionamento de interesse;
3) Podem ser reconhecidos com pequeno ou nenhum conhecimento pré-codificado.

A primeira condição garante que o padrão vai possibilitar a recuperação de diversas instâncias, independentemente do gênero do texto. A segunda condição garante que as instâncias serão recuperadas corretamente. A última condição se refere a frases nas quais não conhecemos o significado das entidades, porém podemos inferir suas classes apenas por sua estrutura. Como no exemplo apresentado em [12] “The bow lute, such as the Bambara ndang, is plucked and has an individual curved neck for each string”. Mesmo sem conhecer o vocabulário da frase, é possível ver que “Bambara ndang” é um tipo de “bow lute”, ou seja, é da classe “bow lute”. É importante perceber que a extração de informação não extrai o significado das entidades. Continuamos não entendendo o que é “Bambara ndang” nem o que é “bow

lute", mas já temos a informação de que "Bambara ndang" é um hipônimo de "bow lute".

Alguns *Hearst Patterns* que podem ser usados para reconhecimentos de hipônimos são mostrados abaixo:

1) NP_0 such as {NP_1, NP_2, NP_3, … , (and|or) } NP_n
2) Such NP as {NP ,}* {(or|and)} NP
3) NP {, NP}*{,} (or|and) other NP
4) NP {,} (including|especially) {NP ,} * {or|and} NP

É possível que esses não sejam os únicos padrões encontrados na língua inglesa. A partir das instâncias descobertas através dos padrões acima, podemos identificar outros padrões, encontrando as entidades e as classes previamente conhecidas em outras estruturas.

Analogamente, podemos encontrar padrões semelhantes para a língua portuguesa, porém não serão estudados nesse projeto, pois o sistema extrai dados apresentados em inglês, logo não será realizada uma análise de padrões na língua portuguesa.

Uma forma de facilitar a extração de instâncias consiste em realizar o POS *tagging* (*Part-Of-Speech tagging*), que significa classificar as palavras em suas classes gramaticais. Esse processo é importante, pois sabemos que uma entidade sempre será um substantivo, portanto se o texto já estiver marcado com as classes gramaticais, o reconhecimento de entidades se torna mais fácil. O POS *tagging* é muito útil para resolver ambiguidades, que é o caso de palavras escritas iguais, porém com significados diferentes. Por exemplo, "book" pode significar não só livro, mas também o verbo "to book", de reservar. Usando o POS *tagging*, fica claro que são duas palavras diferentes visto que uma é verbo e outra é substantivo.

A linguagem de programação *Python* fornece uma biblioteca chamada NLTK (*Natural Language ToolKit*) que possui a funcionalidade de POS *tagging*. Segue o exemplo retirado de [13] de como é o resultado da identificação de classes gramaticais:

- text = nltk.word_tokenize("And now for something completely different")
- nltk.pos_tag(text)

- [('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'), ('completely', 'RB'), ('different', 'JJ')]

Sendo "CC" *coordinating conjuction*, "RB" *adverb*, "IN" *prepositon*, "NN" *noun* e "JJ" adjective. Diversas convenções podem ser utilizada para representar as classes gramaticais. A tabela abaixo mostra os códigos de uma versão simplificada que representam as classes gramaticais com seus respectivos significados e exemplos [13]:

Tabela 1 - Tabela de códigos de classes gramaticais

Tag	Significado	Exemplo
ADJ	Adjetivo	new, old, good, high, special
ADV	Advérbio	really, already, still, now
CNJ	Conjunção	and, or, but, if, although
DET	Determinante	the, a, same, most, every, no
EX	Existencial	there, there's
FW	Palavra Estrangeira	dolce, esprit, maitre
MOD	Verbo Modal	will, can, must, should, would
N	Substantivo	home, year, actor, education
NP	Substantivo Próprio	April, Washigton, Africa
NUM	Número	Twenty-four, 2012, 18:40
PRO	Pronome	I, you, she, he, her, its, my
P	Preposição	on, in, into, at, by, under
TO	A palavra "to"	To
UH	Interjeição	ah, bang, hmpf, oops
V	Verbo	is, has, get, eat, read
VD	Verbo no passado	Was, had, got, said, took
VG	Verbo no particípio presente	making, going, playing
VN	Verbo no particípio passado	given, taken, begun, sung
WH	Determinador WH	who, which, when, what, where, how

O algoritmo que realiza o POS *tagging* se baseia em um conjunto de palavras previamente classificadas, chamado conjunto de treinamento. A partir desse conjunto, quando um texto for passado, as palavras serão classificadas de acordo com a maior probabilidade calculada em cima do conjunto de treinamento. Esse algoritmo é chamado de *N-Gram tagger* [13]. Quando N=1, o algoritmo se chama *Unigram tagger* e se baseia apenas na probabilidade de a palavra aparecer isoladamente. Quando N>1, o algoritmo verifica a probabilidade de a palavra aparecer com N-1 palavras anteriores e N+1 palavras posteriores, considerando assim, o contexto no qual a palavra se encontra. O *N-gram tagger* tem o problema de não conseguir classificar uma palavra que já tenha sido vista no conjunto de treinamento, mas que não tenha sido vista em uma mesma sequência de palavras. Por isso, é indicado combinar os *taggers* a fim de encontrar um maior equilíbrio entre precisão e revocação, medidas de avaliação que serão explicadas a seguir.

Para palavras que não estão no conjunto de treinamento, o *tagger* funciona da seguinte maneira: chama a palavra de UNK e roda o algoritmo. A maior probabilidade é de que UNK seja um substantivo, mas dependendo do contexto pode ser classificado como outras classes, por exemplo, como um verbo se for precedido de "to" [13].

Para avaliar a extração de instâncias, comparamos o resultado obtido como o resultado ideal. De acordo com SENELLART [11], há duas medidas para essa comparação: precisão e revocação. Precisão é a medida de quantos acertos foram feitos em relação ao número total de resultados encontrados. Seja O, o *output* do algoritmo e G o conjunto de respostas ideais (*Gold Standard*) , a medida é dada pela fórmula:

$$precision = \frac{|O \cap G|}{O}$$

Revocação é a medida de quantos acertos foram feitos em relação ao número total de respostas corretas, dada pela fórmula:

$$recall = \frac{|O \cap G|}{G}$$

Vejamos o seguinte exemplo [6]:

$$Output = \{Einsten, Bohr, Planck, Clinton, Obama\}$$
$$Gold\ Standard = \{Einsten, Bohr, Planck, Heisenberg\}$$
$$precision\ = 3/5$$
$$recall = 3/4$$

Um algoritmo pode ser classificado como exploratório ou conservador. Quando um algoritmo é exploratório, ele procura extrair o máximo de respostas corretas, independente do número de falso positivos que isso possa gerar, ou seja, respostas retornadas que não estão no conjunto ideal. Por exemplo [11], um algoritmo que retorne o *output* abaixo tem um bom desempenho em termos de revocação, mas um desempenho ruim em termos de precisão:

$$Output = \{Einsten, Bohr, Planck, Clinton, Obama, Elvis, Heisenberg\}$$
$$precision\ = 4/7$$
$$recall = 4/4$$

Por outro lado, um algoritmo conservador terá uma boa precisão e uma baixa revocação, visto que procura acertar o máximo possível dentro do seu conjunto, independente do número de outputs corretos que não são encontrados. Abaixo, a saída de um algoritmo conservador:

$$Output = \{Einsten\}$$
$$precision\ = 1/1$$
$$recall\ = 1/4$$

Como podemos observar, é difícil obter uma boa medida de precisão e uma boa medida de revocação ao mesmo tempo. Logo, para comparar o desempenho de dois algoritmos usa-se a medida *F1-score*, que é a média harmônica entre precisão e revocação dada por:

$$F1\ score = 2\ \frac{(precision\ x\ recall)}{precision + recall}$$

Portanto, temos que no primeiro exemplo F1 = 2/3, no segundo exemplo F1 = 8/11 e no último exemplo F1 = 2/5. Notamos que o primeiro exemplo mostra um algoritmo mais equilibrado em termos de precisão e revocação.

2.2.3 Extração de Fatos

A extração de fatos é o processo de extrair fatos sobre uma entidade. Por exemplo, considere uma entidade que já foi reconhecida como pessoa e sua classe ator já foi extraída. O próximo passo será extrair fatos sobre a pessoa, sua data de nascimento, os prêmios que já ganhou, onde nasceu, em quais filmes atuou, entre outros.

Uma forma de extrair fatos é chamada de *wrapper induction*, que consiste em extrair os fatos a partir da estrutura de uma página. No IMDb, a nota de um filme aparece sempre na mesma posição, tornando possível a extração da nota apenas conhecendo a estrutura HTML (*HiperText Markup Language*) da página [11].

Outra forma de extrair fatos é pelo reconhecimento de padrões da mesma forma usada na extração de classes. Algumas informações como data e local de nascimento, pessoa com quem é casada e idade seguem o mesmo padrão de estrutura da frase. Dessa forma, podem-se usar expressões regulares para a extração de fatos, considerando que os textos sejam escritos sempre de maneira similar.

2.2.4 Extração de Informação Baseada em Ontologia

A extração de informação baseada em ontologia (OBIE – *Ontology Based Information Extraction*) é o processo de extração de informação que busca gerar uma ontologia ou estender uma já existente. Esse tipo de extração é considerado uma subárea da extração de informação, limitada a textos não-estruturados ou semiestruturados, como a Wikipedia [10].

Esse processo utiliza a extração de informação guiada por uma ontologia, visando extrair classes, instâncias e propriedades e fornecendo a saída em formato ontológico [11]. Normalmente, esses sistemas extraem informação na estrutura sujeito-predicado-objeto, como no RDF. Por exemplo, se a seguinte informação for extraída: “Brad Pitt is married to Angelina Jolie”, teremos como saída do sistema: <Brad Pitt, isMarriedTo, Angelina Jolie>. Dessa forma a informação está representada de acordo com uma estrutura ontológica.

A extração baseada em ontologia é considerada uma área de grande potencial, dado que extrai informação automaticamente de textos escritos em linguagem natural e esses textos representam 80% dos textos na web [10]. Além disso, com o advento da web semântica, se torna cada vez mais importante representar a informação de maneira estruturada semanticamente, o que corresponde ao conceito de ontologia visto anteriormente. Por fim, o processo agrega conteúdo às ontologias existentes, trazendo maior potencial para ele, visto que obtém novas informações automaticamente, se tornando um sistema cada vez mais inteligente.

Em [10], temos uma visão geral de uma possível arquitetura para sistemas de extração de informação baseados em ontologia:

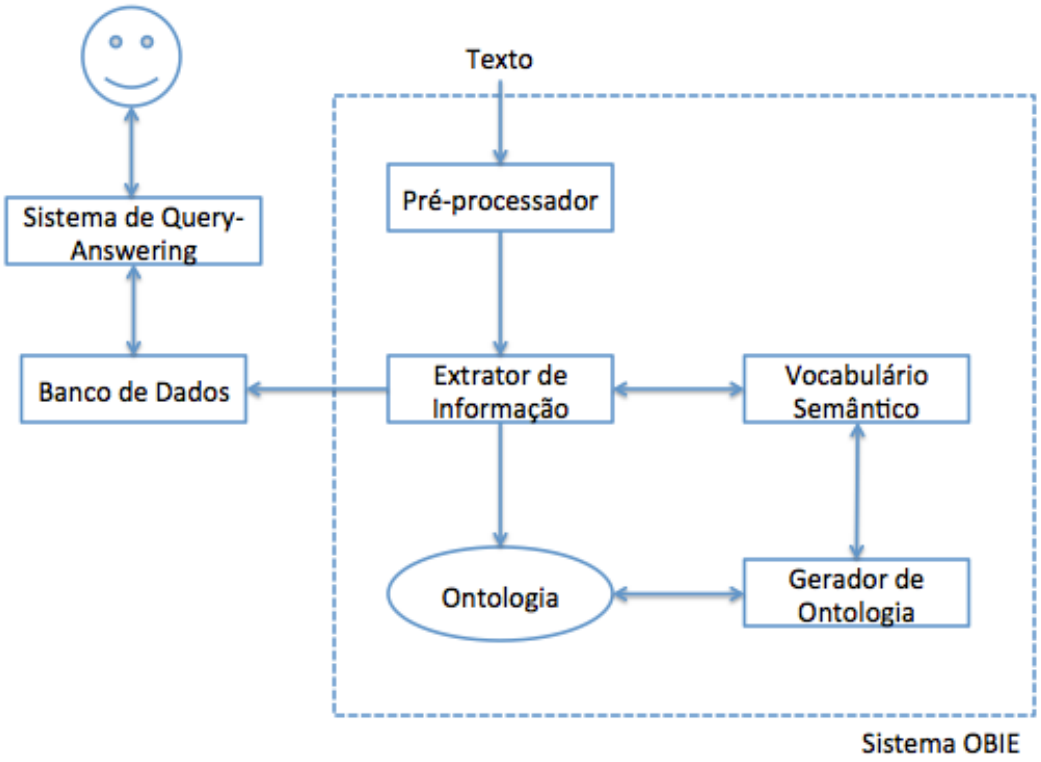

Figura 3 - Arquitetura de um sistema OBIE baseada em [10]

A arquitetura mostrada acima é um exemplo de arquitetura para um OBIE, no entanto os sistemas podem não possuir algum desses componentes, como os sistemas que não constroem uma ontologia própria. Esses sistemas não possuem o módulo "Gerador de Ontologia" e podem conter o módulo "Editor de Ontologia". Da mesma forma, o sistema pode conter alguma funcionalidade a mais e, assim, acrescentar módulos à arquitetura mostrada.

A extração de informação baseada em ontologia começa com o texto sendo retirado da web e sendo pré-processado de forma que possa passar para o módulo de extração de informação. Esse processamento pode conter as seguintes ações: passar todas as palavras para *lower-case*, padronizar números e símbolos, e, dependendo do algoritmo utilizado, aplicar o POS *tagger* no texto.

Em seguida, passamos ao módulo "Extrator de Informação" que pode usar diferentes técnicas, como as mostradas nas seções anteriores usando expressões regulares. Independentemente da técnica utilizada, o algoritmo sempre será baseado em ontologias no caso de um OBIE. Esse módulo se comunica com o editor de ontologia ou o gerador de ontologia.

O módulo "Vocabulário Semântico" é utilizado quando se tem um agrupamento de expressões ou palavras que possuem o mesmo sentido semântico. Esse módulo é opcional e o agrupamento pode ser feito por árvores semânticas. As árvores semânticas são estruturas que representam palavras similares semanticamente. A figura 4 mostra um exemplo de árvore semântica contendo termos relacionados ao cinema. Por exemplo, "movie" e "film" são termos similares, assim como podemos encontrar a pergunta "Who made the movie Argo?", usando "made" no sentido de "produced".

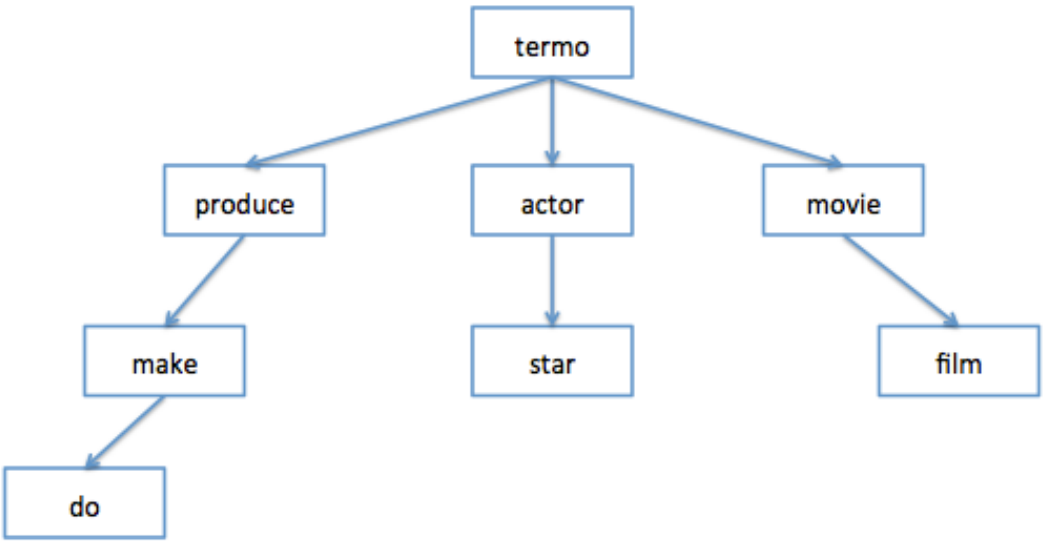

Figura 4 - Exemplo de árvore semântica

2.3 Interface de Linguagem Natural para Banco de Dados

Uma interface de linguagem natural para banco de dados consiste em um modo de interação no qual o usuário realiza as consultas usando linguagem natural e sua consulta é processada para uma linguagem de consulta a banco de dados, como o SQL (*Structured Query Language*), de forma a interagir com o banco de dados e retornar a resposta para o usuário. Existem quatro formas do processamento de linguagem natural que são mostradas nas próximas seções.

2.3.1 Sistemas de Reconhecimento de Padrões

Através de padrões linguísticos conhecidos, o sistema interpreta a consulta que o usuário deseja, de maneira semelhante à explicada na seção de reconhecimento de entidades. Por exemplo, dadas as tabelas de filme, ator e participação, podemos extrair padrões como [14]:

Padrão: ... "atores"... <filme>
Interpretação: Liste os atores do filme <filme>
SQL: SELECT actor.name FROM movie, actor, participation
WHERE movie.title = <filme>
AND participation.movie_id = movie.id
AND participation.actor_id = actor.id

Padrão: ... <ator> ... <filme>
Interpretação: O ator <ator> participa do filme <filme>?
SQL: SELECT actor.name, movie.title FROM movie, actor, participation
WHERE movie.title = <filme>
AND participation.movie_id = movie.id
AND participation.actor_id = actor.id
AND actor.name = <actor>

No primeiro caso o padrão corresponde a buscar a lista de atores do filme <filme>. O sistema identificaria perguntas como "Quais atores participam do filme True Grit?", "Gostaria de saber os atores de Titanic.", "Liste os atores do filme

Django.”. A vantagem desse sistema é justamente poder interpretar consultas escritas de diversas maneiras, dando mais liberdade ao usuário. Já a segunda consulta demanda uma resposta de “sim” ou “não”. O sistema pode responder indiretamente, simplesmente listando os atores do filme ou os filmes do ator, ou o sistema pode interpretar e responder essa consulta e retornar “sim” ou “não”.

Por outro lado, a liberdade desse sistema também o torna mais suscetível a erros, pois ele pode interpretar algo que não é um filme como sendo um filme e responder de forma errada ao usuário.

2.3.2 Sistemas Baseados em Sintaxe

O sistema baseado em sintaxe analisa os termos da frase de acordo com sua classe gramatical, conforme o algoritmo de POS *tagging* mostrado na seção 2.2.2 de extração de instâncias. Após a identificação das classes gramaticais, a frase passa pela chamada *Parse Tree*, que é uma árvore com as possíveis estruturas gramaticais das perguntas. Em [14] temos o seguinte exemplo de uma *Parse Tree*:

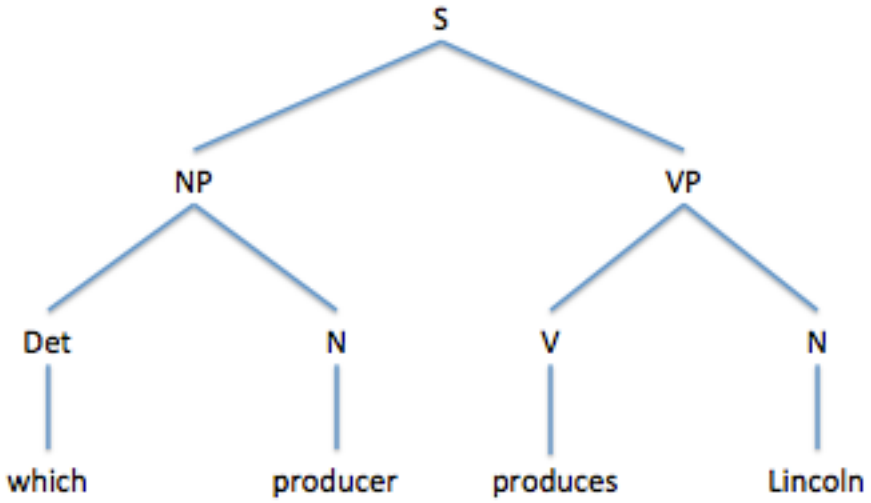

Figura 5 - *Parse Tree* Sintática baseada em [14]

De acordo com a árvore acima, uma frase (S) é formada por uma frase nominal (NP) e uma frase verbal (VP), sendo a frase nominal composta por um determinante e um substantivo e a frase verbal composta por um verbo e um

complemento. Dessa forma, para cada elemento da frase o algoritmo realiza a correspondência com a lógica que forma a consulta em SQL.

Por exemplo, na frase mostrada na figura 5, teríamos a seguinte correspondência [14]:

```
which -> for_every X
producer -> is_producer X
produces -> produces
Lincoln -> Lincoln
```

O mapeamento corresponde a linguagem intermediária:

```
for_every X      (is_producer X)
           (produces X Lincoln);
```

Tal linguagem corresponde à consulta SQL:

```
SELECT *  FROM person
WHERE person.profession == "producer"
AND person.production == "Lincoln"
```

2.3.3 Sistemas de Gramática Semântica

O sistema de gramática semântica funciona de forma similar, porém os nós internos da árvore não correspondem a classes gramaticais, mas sim a conceitos semânticos específicos da aplicação. Por esse motivo, esse tipo de análise é recomendado apenas para aplicações de um domínio específico. Em [14] um exemplo de *Parse Tree* para gramática semântica é mostrado na figura 6.

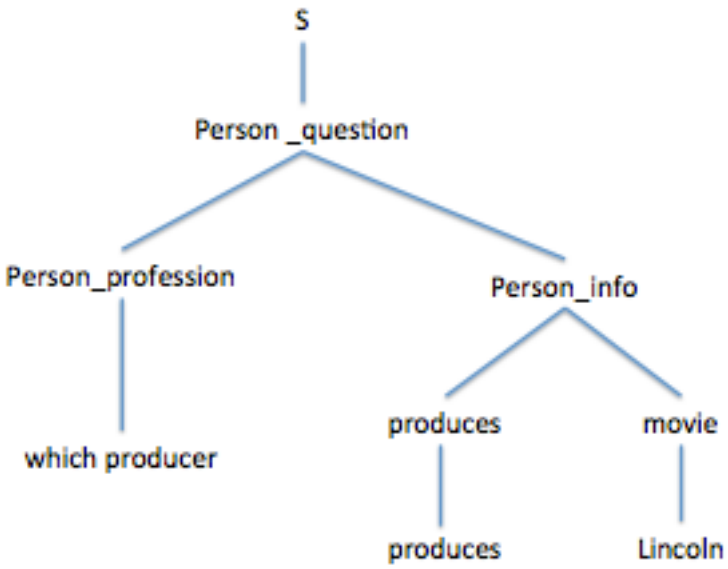

Figura 6 - *Parse Tree* Semântica baseada em [14]

Essa gramática limita as perguntas processadas pelo *parser*. Por exemplo, se o usuário coloca no lugar de "Lincoln", alguma palavra que não represente um filme, a pergunta não será processada. Ao mesmo tempo, ela facilita a transformação da consulta em uma consulta SQL ao passo que "person_profession" vai redirecionar diretamente à tabela correspondente. A dificuldade desse algoritmo é a portabilidade, visto que não pode ser reaproveitado para um sistema que envolva outro contexto. A árvore semântica deve ser refeita para cada aplicação de um contexto diferente.

2.3.4 Linguagem Intermediária de Representação

Os sistemas que utilizam linguagem intermediária de representação primeiramente transformam a consulta do usuário em uma linguagem lógica intermediária que depois é transformada em uma linguagem de comunicação com o banco de dados, como SQL [14].

Na figura 7, temos uma possível arquitetura para um sistema que utiliza linguagem intermediária de representação [14].

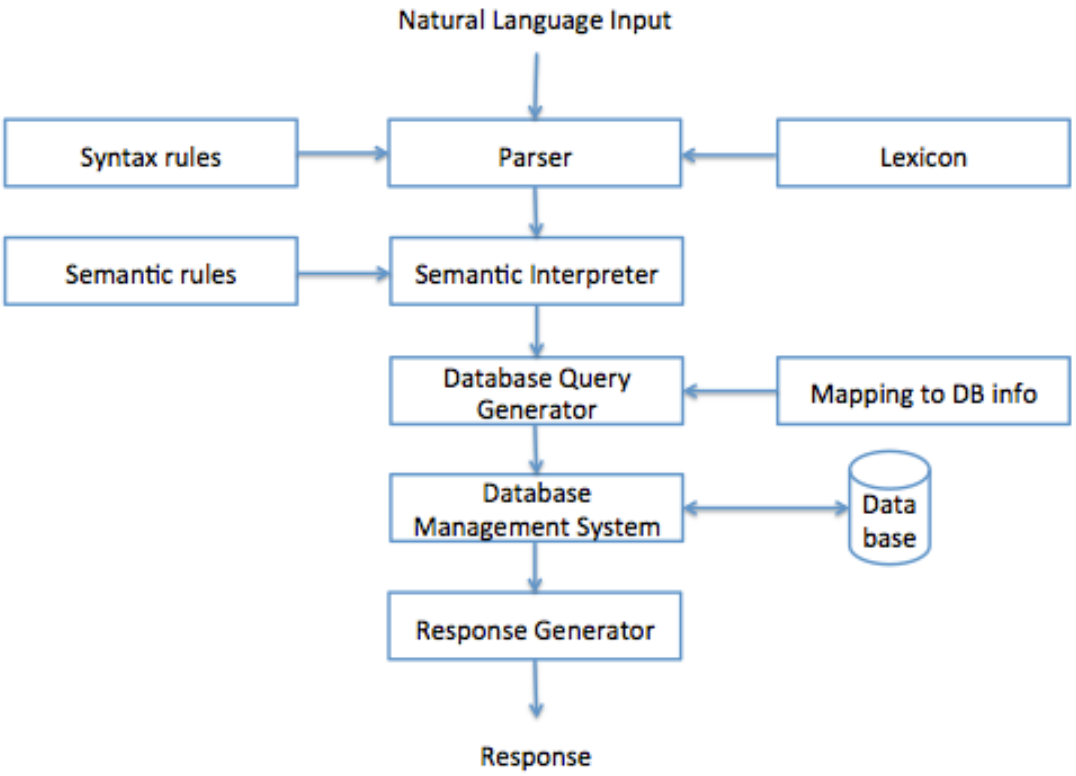

Figura 7 - Arquitetura de um sistema de linguagem intermediária de representação [11]

O texto em linguagem natural passa pelo módulo de *Parser* e depois para o interpretador, que transforma a consulta na linguagem lógica intermediária. No módulo *Database Query Generator*, a linguagem intermediária é transformada em uma consulta SQL.

Esse sistema tem a vantagem de ser portátil, visto que as informações específicas sobre um tema são definidas no módulo *Lexicon*, que fica externo à aplicação. De acordo com [14], é nesse módulo que é definido o significado da linguagem intermediária das palavras. Por exemplo, a palavra "capital" é traduzida pela expressão "capital_of(capital, country)". Em [14] é dado um exemplo da interpretação da pergunta "What is the capital of each country bordering Greece?" executada no sistema MASQUE/SQL [15]. Essa pergunta seria interpretada como:

```
answer([Capital, Country]):

is_country(Country), borders(Country, Greece),
capital_of(Capital, Country)
```

Dessa forma, o sistema busca os países que fazem fronteira com a Grécia e depois busca suas respectivas capitais. É importante verificar se *Country* é um país, pois o usuário pode requisitar uma resposta que o sistema não tem, e assim, o sistema verificaria de imediato que se *Country* não é um país, ele não pode responder à pergunta.

As funções da linguagem lógica não precisam se referir necessariamente a tabelas do banco nem se traduzir diretamente à linguagem SQL. É necessário passar pelo módulo *Database Query Generator,* que utiliza as informações do módulo *Mapping to DB info,* para que finalmente possam ser traduzidas para SQL.

No exemplo acima, *is_country,* corresponde à consulta SQL: "SELECT country FROM countries" [14]. Essa transformação pode gerar consultas mais complexas, envolvendo diversas tabelas do banco.

Os sistemas podem conter o módulo *Response Generator* que é responsável por transformar a resposta em uma resposta de linguagem natural. Isso é muito comum nos sistemas que mantêm um diálogo com o usuário, porém não é alvo de estudo deste projeto.

3 Tecnologias

Nesta seção será feita a apresentação das tecnologias utilizadas na implementação do InfoMovie.

3.1 Django

Django é um *framework* de desenvolvimento web de alto nível, na linguagem *Python*. A ferramenta é baseada no conceito DRY (*Don't repeat yourself*). Esse princípio é declarado em [16] como " *Every piece of knowledge must have a single, unambiguous, authoritative representation within a system*.", ou seja, cada conhecimento deve ser representado sem ambiguidade, de forma singular no sistema. O conceito foca em não repetir informação no código nem na arquitetura do sistema.

O *framework Django* adota uma arquitetura que pode ser chamada de MTV (*Model-Template-View*), que consiste em uma variação do MVC (*Model-View-Controller*). A arquitetura MVC possui o módulo *Model* que se comunica com o banco de dados, o módulo *View*, responsável pela interface com o usuário e as requisições das páginas web e por fim o módulo *Controller*, que é o intermediário entre o *View* e o *Model*. No *Django,* a arquitetura é similar, porém o conceito de *View* é descrito como os dados que serão apresentados ao usuário e não como eles serão apresentados, responsabilidade do módulo *Template*. O módulo *Controller* está por trás de tudo e pode ser considerado o framework por si só [17].

3.2 Movie Ontology

MO, *the Movie Ontology* [5], é um projeto da Universidade de Zurick que tem por objetivo a criação de uma semântica que descreve conceitos relacionados ao cinema. A ontologia foi usada como base para o entendimento da linguagem semântica usada quando se trata de cinema, para a construção do modelo relacional de dados do sistema e para a formulação do algoritmo de interpretação de consultas.

A ontologia MO representa as seguintes entidades:

- *Award*
- *Certification*
- *Consumable_Product*
- *Country*
- *Genre*
- *Movie*
- *Online_Retailer*
- *Person*
    - *Actor*
    - *Costume_Designer*
    - *Editor*
    - *Film_Director*
    - *Musical_Artist*
    - *Producer*
    - *Writer*
    - *Director*
- *Place*
- *Presentation*
- *Production_Company*
- *Territory*

E os relacionamentos diretos e inversos, quando existentes, respectivamente:

- *Movie belongsToGenre Genre – isGenreOf*
- *Territory containsCountry Country*
- *Consumable_Product enablesConsumptionOf Movie - isConsumableAs*
- *Movie hasActor Actor – isActorIn*
- *Movie hasColor ColorInfo – isColorOf*
- *Movie hasComposer Musical_Artist - isComposerIn*
- *Movie hasCostumeDesigner Costume_Designer - isCostumerDesignerIn*
- *Movie hasDirector Director – isDirectorIn*
- *Movie hasEditor Editor – isEditorIn*
- *Movie hasFilmLocation Palce isFilmLocationOf - isFilmLocationOf*
- *Movie hasProducer Producer - isProducerIn*
- *Movie hasReleasingCountry Country – isReleasingCountryOf*
- *Movie hasSoundMix Soud_Mix – isSoundMixOf*
- *Language hasTranslatedMovie Movie – isTranslatedTo*
- *Award isAwardOf Movie – isAwardedWith*
- *Certification isCertificationOf Movie – isCertifiedWith*
- *Movie isProducedBy ProductionCompany – produced*
- *Movie nominatedFor Award*
- *Country partOfTerritory Territory*

O site da MO, fornece um arquivo *movieontology.owl*, que contém a representação da ontologia no formato OWL (*Web Ontology Language*). O OWL é muito similar ao RDF, porém funciona melhor para ontologias por apresentar um vocabulário semântico.

Abaixo um trecho do arquivo *movieontology.owl* para exemplificar:

```
<!-- http://dbpedia.org/ontology/Actor -->
<owl:Class rdf:about="&ontology;Actor">
       <rdfs:subClassOf rdf:resource="&ontology;Person"/>
</owl:Class>

<!-- http://dbpedia.org/ontology/Country -->
<owl:Class rdf:about="&ontology;Country"/>

<!-- http://dbpedia.org/ontology/Language -->
<owl:Class rdf:about="&ontology;Language">
       <rdfs:subClassOf
rdf:resource="&movieontology;Presentation"/>
</owl:Class>
```

Nesse trecho encontramos as definições das classes “Actor”, “Country” e “Language”. As classes estão definidas como <owl:Class> e foram extraídas da DBpedia [18], uma enciclopédia de conhecimento que extrai dados da Wikipedia. A classe “Actor” é uma subclasse de “Person”, assim como “Language” é uma subclasse de “Presentation”, definido por “<rdfs:subClassOf>.

Vejamos agora o seguinte trecho:

```
<owl:ObjectProperty
rdf:about="&movieontology;containsCountry">
        <rdfs:range rdf:resource="&ontology;Country"/>
        <rdfs:domain rdf:resource="&movieontology;Territory"/>
</owl:ObjectProperty>
```

Nesse trecho vemos a propriedade “Country” de “Movie”, dada por “countainsCountry”. Também obtemos a informação de que “Country” esta no domínio de “Territory”.

Para a visualização da MO foi utilizado o software Protégé [19], desenvolvido em Java por uma equipe de Stanford. O Protégé é um editor de ontologia *open source,*

que suporta formatos como o RDF e OWL. O software permite visualizar, criar e editar ontologias. A figura abaixo mostra a MO vista através do Protégé.

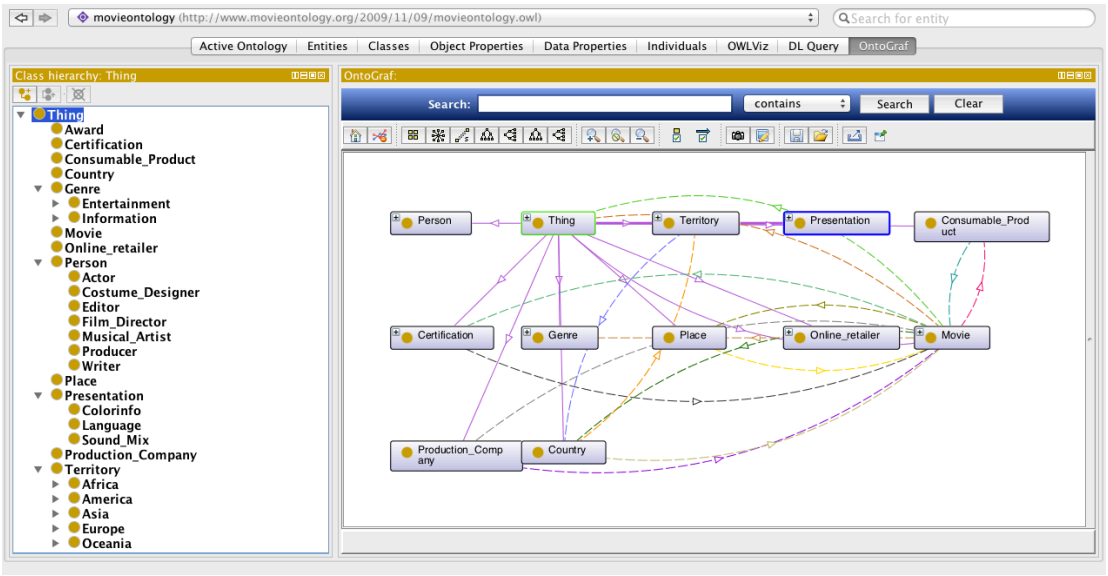

Figura 8 - *Movie Ontology* visualizada no Protégé

Podemos observar à esquerda as classes da ontologia, seguindo uma hierarquia. Todas as classes são subclasses de "Thing", a classe "Person" é um hiperônimo de "Actor", "Costume_Designer", "Editor" e assim por diante. À direita, temos a visualização das classes e seus relacionamentos.

A figura abaixo mostra a tela de visualização das propriedades dos objetos:

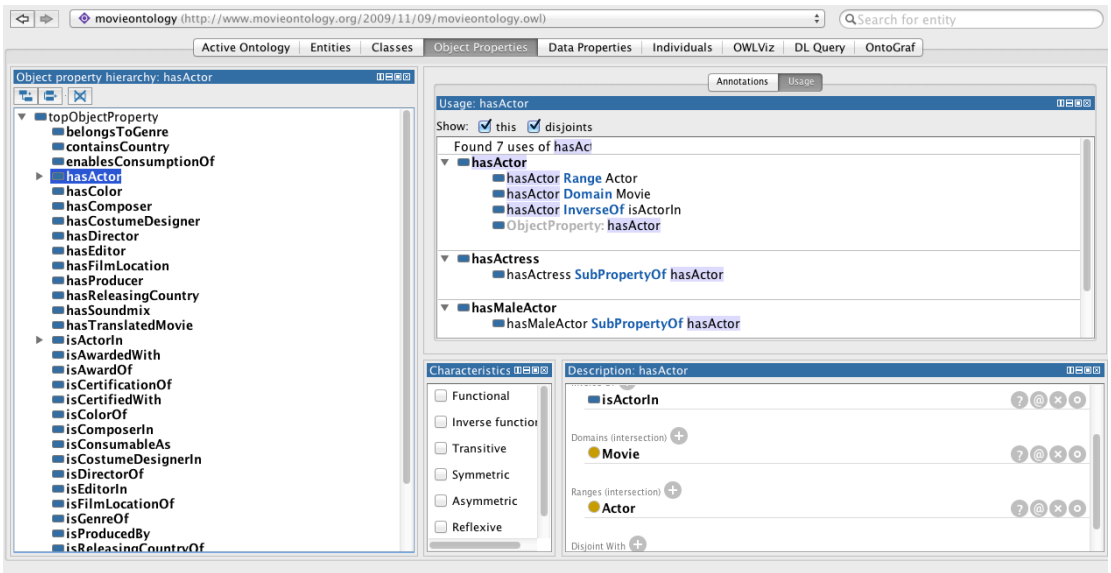

Figura 9 - Propriedades dos objetos visualizadas no Protégé

À esquerda, temos as propriedades dos objetos que são explicadas à direita. Vemos que a propriedade “hasActor” tem como domínio a classe “Movie” e como alcance a classe “Actor”, ou seja <Movie, hasActor, Actor>. Temos também que “isActorIn” é a propriedade inversa de “hasActor” e que “hasActress” e “hasMaleActor” são subpropriedades de “hasActor”.

4 O Sistema

O InfoMovie é uma proposta de sistema de *Query-Answering* que utiliza os dados extraídos do IMDb para fornecer uma interface de linguagem natural para o banco dados onde ficam armazenadas as informações baseadas na ontologia "cinema". Neste capítulo, a implementação do InfoMovie será explicada em detalhes, mostrando a relação com os conceitos estudados e tecnologias utilizadas em seu desenvolvimento.

4.1 Modelagem de Dados

A modelagem de dados foi realizada de forma a estruturar as informações extraídas do IMDb, a fim de organizá-las em um banco de dados e otimizar a busca aos dados. A modelagem foi concebida baseando-se na ontologia "cinema" extraída da *Movie Ontology*, de forma que as classes da ontologia fossem transformadas em tabelas e as propriedades em relacionamentos ou atributos.

O software utilizado para a criação do modelo entidade-relacionamento foi o Astah [20], que permite a modelagem em UML (*Unified Modeling Language*). A figura 10 mostra o modelo gerado para o InfoMovie.

No centro do modelo, encontra-se a tabela "Movie" que contém todos os atributos referentes a filme incluindo título, ano de lançamento, localização, entre outros. A tabela de filmes se relaciona com as tabelas "Place", "ColorInfo" e "Country" que representam atributos de um filme. A mesma se relaciona com as tabelas "Genre", "Production Company" e "Language" através de um relacionamento m x n. A tabela "Movie" se relaciona com a tabela "Person" através da tabela de participação "Participation". A tabela participação contém um papel, representado pela tabela "Role" que pode ser ator, diretor, produtor, escritor ou outros papéis relacionados. A participação também contém um ou mais prêmios, que podem ser apenas uma indicação ou um prêmio de fato recebido, como o Oscar.

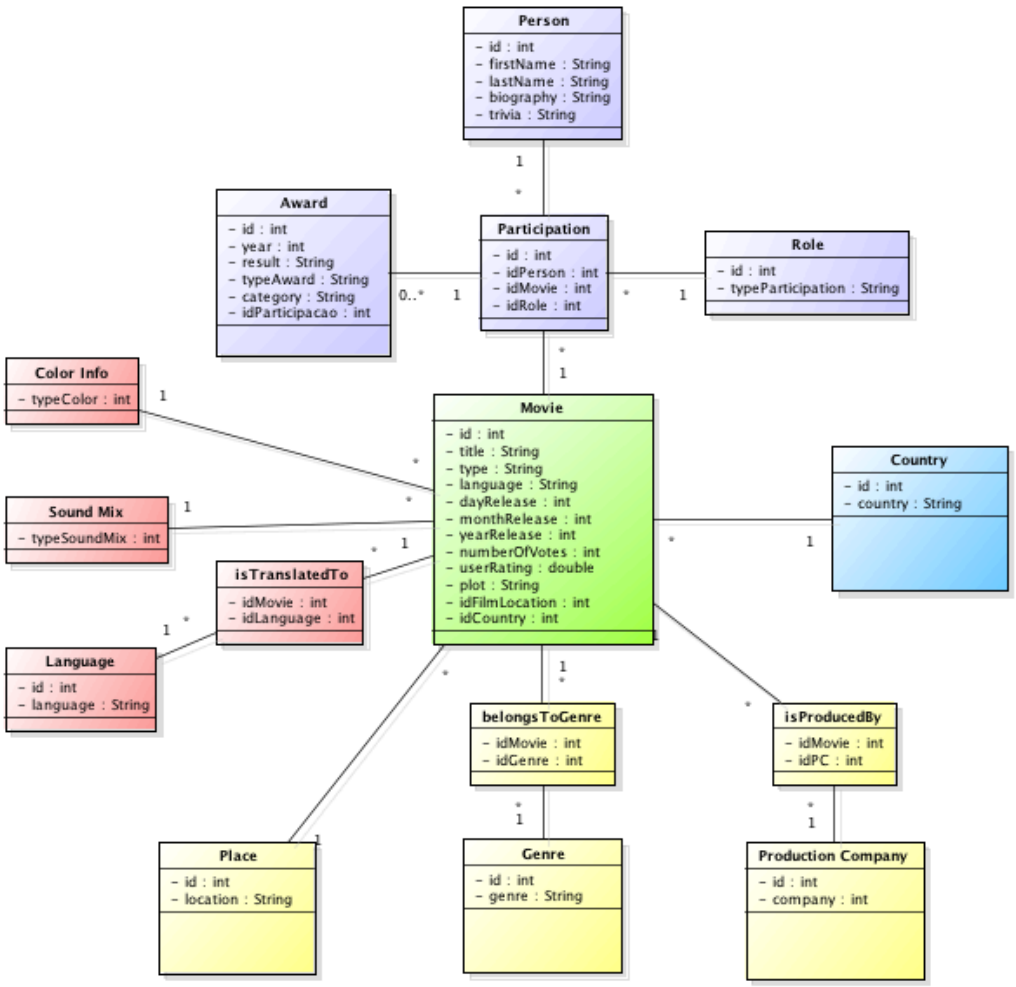

Figura 10 - Modelo de dados UML

4.2 Arquitetura

A arquitetura do sistema foi construída conforme a figura 11 e possui os seguintes componentes: interface, interpretador, gerador SQL e o banco de dados. Inicialmente, o banco de dados é preenchido através das interfaces [21] para a base de dados do IMDb, fornecidas em forma de arquivos. O usuário realiza uma consulta através da interface, essa consulta passa pelo interpretador que a transforma em uma linguagem lógica intermediária e a seguir para o gerador de SQL, que como o nome diz, gera uma consulta SQL e envia para o banco de dados. Em seguida, a resposta fornecida pelo banco de dados é enviada de volta para o usuário.

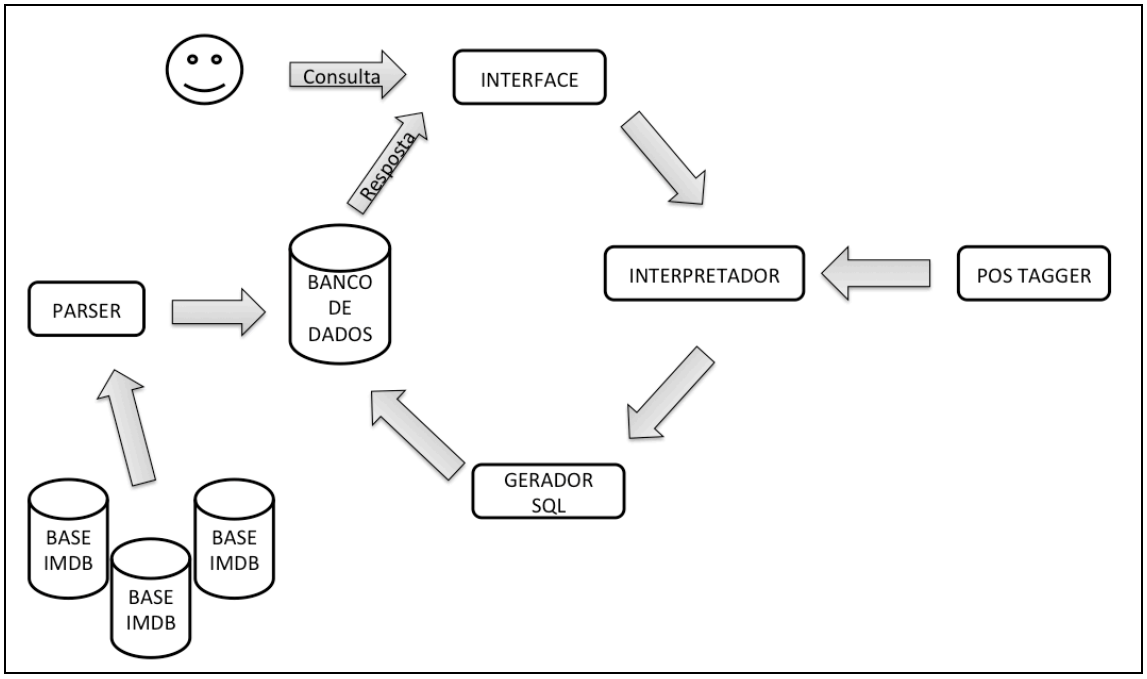

Figura 11 - Arquitetura InfoMovie

Na arquitetura mostrada acima, foram implementados o Parser, o interpretador, o gerador SQL e a interface visando atender o escopo do InfoMovie.

O Parser foi implementado para ler os arquivos no formato fornecido pelo IMDb e adicionar as informações no banco de dados modelado de acordo com a ontologia "cinema". O interpretador e o gerador SQL também foram desenvolvidos nesse projeto para interpretar perguntas referentes à ontologia de cinema. A interface foi implementada utilizando as funcionalidades fornecidas pelo Django, de criação de páginas e layout. O componente POS Tagger utiliza a função da biblioteca NLTk que realiza a classificação em classes gramaticais.

4.2.1 Parser

O IMDb fornece arquivos com listas dos dados de sua base, contendo nome dos filmes, atores, diretores, países, idiomas e outras informações que podem ser encontradas no *site*. Abaixo segue um trecho do arquivo *movies.list*:

```
Life of Pi (2012)                                          2012
Life of Pryor: The Richard Pryor Story (2006) (TV)         2006
Life of Python (1990) (TV)                                 1990
Life of Rhyme (2011) (TV)                                  2011
Life of Sam Davis: A Confederate Hero of the Sixties (1915)  1915
Life of St. Paul (1938)                                    1938
Life of St. Paul Series (1949)                             1949
Life of the Frog (1915)                                    1915
Life of the Jews of Palestine (1913)                       1913
Life of the Party (1920)                                   1920
```

O *parser* percorre o arquivo linha por linha e salva uma instância de filme, com seu título e ano, na tabela "Movie". Após ter todos os filmes armazenados no banco de dados, o *parser* percorre os outros arquivos. Para o arquivo *directors.list*, temos o trecho a seguir:

```
Lee, Ang            Brokeback Mountain (2005)
                    Chosen (2001)
                    Cleopatra (????)
                    Hulk (2003)
                    Life of Pi (2012)
                    Ride with the Devil (1999/I)
                    Se, jie (2007)
                    Sense and Sensibility (1995)
                    Taking Woodstock (2009)
                    The Ice Storm (1997)
                    Tui shou (1992)
                    Wo hu cang long (2000)
                    Xi yan (1993)
                    Yin shi nan nu (1994)

Lee, Angela (II)    From Fat to Finish Line (2013)

Lee, Angharad       ABCDad (2011)
```

Ao passar pelas linhas do arquivo *directors.list*, o *parser* salva os diretores na tabela “Person”, com seus primeiro e último nomes. A seguir, o *parser* percorre a lista de filmes correspondentes ao diretor indicado e adiciona uma participação, contendo o id do diretor, o id do filme e o id do papel, que nesse caso é diretor. Para as tabelas ator, produtor e outras tabelas relacionadas a pessoas, o *parser* funciona de maneira análoga, alterando apenas o papel que será adicionado na tabela “Participation”. Para esse projeto, apenas a lista de filmes e diretores foi adicionada no banco de dados.

A tabela “Movie” contém 2.422.263 de registros, a tabela “Person” contém 285.293 registros e tabela participação ficou com 1.639.304 de registros. Por conter uma quantidade muito grande de registros, foi criado um índice B em cima da coluna “title” na tabela filmes. Uma árvore B organiza os dados ordenadamente de forma que seja possível realizar uma busca em O(log n). Após a criação do índice, a busca por um filme realizada pelo comando SQL “SELECT * from Movies WHERE title = ‘Edward Scissorhands’” que durava 1.59 segundos passou a durar 0.059 segundos. O mesmo foi feito para a tabela “Person”, com índice criado em cima do atributo “first_name”.

4.2.2 Interpretador

O papel do interpretador no sistema é transformar a consulta feita pelo usuário em linguagem natural em uma linguagem lógica capaz de gerar uma consulta SQL. Nesse sistema, o interpretador utiliza a técnica de reconhecimento de padrões com a análise de sintaxe usada para resolver possíveis conflitos que serão mostrados a seguir.

Para exemplificar como funciona o interpretador, seja a seguinte consulta: “who is the director of the movie Life of Pi?”, ao passar por uma lista de possíveis padrões, a pergunta será reconhecida pelo padrão “... director ... <movie>”, que corresponde ao padrão regex (*regular expression*):

1) "(director|directed|directs) ?(of the movie|of|the movie|of the film|the film)? (.*?)"

Onde, "(.*?)" será reconhecido como o filme. Notamos que essa expressão é capaz de reconhecer perguntas como “who directs the movie Life of Pi?”, “I'd like to know the name of the director of Life of Pi?” ou “Tell me who directed Life of Pi”.

A lógica intermediária que será gerada é "*isMovie (<movie>), hasDirector (<movie>)*". Primeiramente a função *isMovie* será chamada para verificar se <movie> é realmente um filme. Se sim, a função *hasDirector(<movie>)* é chamada e envia a consulta SQL correspondente.

De forma análoga, para a consulta: "which movies are directed by Ang Lee?", corresponde ao padrão "... movies ... <director>". O padrão regex para essa consulta é:

2) "(movies|films) are|were directed by (.*?)"

Há outras formas de realizar a mesma pergunta, como "which movies has Ang Lee directed?" e "what movies did Ang Lee make?". O padrão usado para perguntas nesse formato é:

3) "(movies|films) (.*?) (directed|made|direct|make)"

Notamos que no caso da pergunta "which movies are directed by Ang Lee?", ela seria reconhecida também no padrão regex 1, pois tem a palavra "directed", porém o teste "Ang Lee é um filme" falharia, então o parser continuaria procurando nos próximos padrões. A lógica por trás do padrão "... movies directed by <director>" é chamar a função *isDirector(<director>)* para verificar se <director> é mesmo um diretor e logo após a função *directorOf(<director>),* que retorna a lista de filmes nos quais o <director> tem o papel diretor na participação.

No caso de <director> é possível aplicar a função POS *tagger*, para que antes de buscar se <director> está no banco de dados como uma pessoa, verificamos se o que será passado como parâmetro é um substantivo. Fazendo isso, evitamos passar erroneamente como diretor uma palavra que não seja o nome do diretor. Por exemplo, "Tell me which movies are directed by Ang Lee, please", nesse caso, o parser 2 diria que "Ang Lee, please" é um diretor, porém utilizando o POS *tagger* evitamos esse erro, pois eliminamos a palavra "please" que não é substantivo. Outro caso seria em "which movies has Ang Lee directed?", o *parser* encontraria "has Ang Lee" como diretor. É essa função que permite a flexibilidade de escrever "has directed" ou "did direct", dado que o POS *tagger* retorna para essa frase: [('which', 'WDT'), ('movies', 'NNS'), ('has', 'VBZ'), ('Ang', 'NNP'), ('Lee', 'NNP'), ('directed', 'VBD'), ('?', '.')]. Assim, retiramos de "has Ang Lee" apenas os substantivos e passamos o nome correto para a função *isDirector(<director>).*

4.2.3 Gerador SQL

O gerador SQL é o componente responsável por interpretar a linguagem lógica intermediária, gerar a consulta SQL e retornar a resposta dada pelo SGBD. Cada função do gerador corresponde a uma ou mais consultas que serão enviadas ao bando de dados. A função *hasDirector(<movie>)* gera a seguinte consulta:

```
SELECT person.firstName, person.lastName
FROM movie, participation, director, role
WHERE movie.title = <movie>
AND movie.id = participation.movie_id
AND participation.person_id = person.id
AND role.profession = "DIRECTOR"
AND role.id = participation.role_id
```

O gerador SQL também é responsável pelas verificações *isDirector* e *isMovie*, gerando as consultas que buscam pelo diretor e pelo filme e retornam *True* ou *False* conforme as instâncias são encontradas ou não. Dessa forma, quando entram nas funções *hasDirector(<movie>)* ou *directorOf(<director>)*, já está garantido que os parâmetros foram passados corretamente. A resposta é enviada de volta à aplicação e é exibida na tela para o usuário.

4.3 Resultados

O InfoMovie se mostrou um sistema eficiente para interpretação de consultas e o envio de respostas corretas. No teste para perguntas que significam "Quem é o diretor do filme Silver Linings Playbook?" o sistema conseguiu interpretar corretamente as seguintes perguntas:

- "Who is the director of Silver Linings Playbook?"
- "What is the name of the director of Silver Linings Playbook?"
- "Tell me the director of Silver Linings Playbook"
- "Please, tell me the director of Silver Linings Playbook"
- "I'd like to know who is the director of Silver Linings Playbook?"

- “Who directs the movie Silver Linings Playbook?”
- “Who has directed Silver Linings Playbook?”
- “Who directed Silver Linings Playbook?”
- “Can you please tell me who directed Silver Linings Playbook?”
- “Do you know who directed Silver Linings Playbook?”
- “Who directs the film Silver Linings Playbook?”

O sistema falhou para a seguinte pergunta:

- “Who is the director of Silver Linings Playbook, please?”

Esse caso acontece porque o padrão "(director|directed|directs) ?(of the movie|of|the movie)? (.*?)" encontra como título do filme “Silver Linings Playbook, please” e esse título não é encontrado no banco de dados. Como os títulos de filmes não são necessariamente compostos por substantivos, não é possível aplicar o POS *tagger* como foi feito com diretor, conforme mostrado na seção 4.2.2.

Para perguntas referentes à “Liste os filmes do diretor David O. Russell”, o sistema respondeu corretamente às seguintes perguntas:

- “Which movies has David O. Russell directed?”
- “What films has David O. Russell directed?”
- “Which movies did David O. Russell direct?”
- “Which movies David O. Russell directs?”
- “Which movies David O. Russell directed?”
- “What films are directed by David O. Russell?”
- “I'd like to know what films are directed by David O. Russell”
- “Please, tell me what films are directed by David O. Russell”
- “What others movies has David O. Russell directed?”
- “Do you know any others movies that David O. Russell directed?”
- “What films were directed by David O. Russell?”

O sistema inicialmente falhou para as seguintes perguntas:

- “List the movies directed by David O. Russell”

- “Tell me the movies directed by David O. Russell”

Essas falhas aconteceram porque as perguntas não correspondem ao padrão "are|were directed by (.*?)". Esse problema pôde ser facilmente corrigido colocando “are” e “were” como opcional, da seguinte forma:

“(are|were)?directed by”

Assim, o padrão ficou mais abrangente, permitindo ao sistema reconhecer as perguntas acima e respondê-las corretamente.

Quando perguntamos o diretor de um filme, muitas vezes existem outros filmes com o mesmo título, porém de anos diferentes, por isso o sistema lista os diretores com os respectivos filmes e anos do lado, como mostram as figuras 12 e 13.

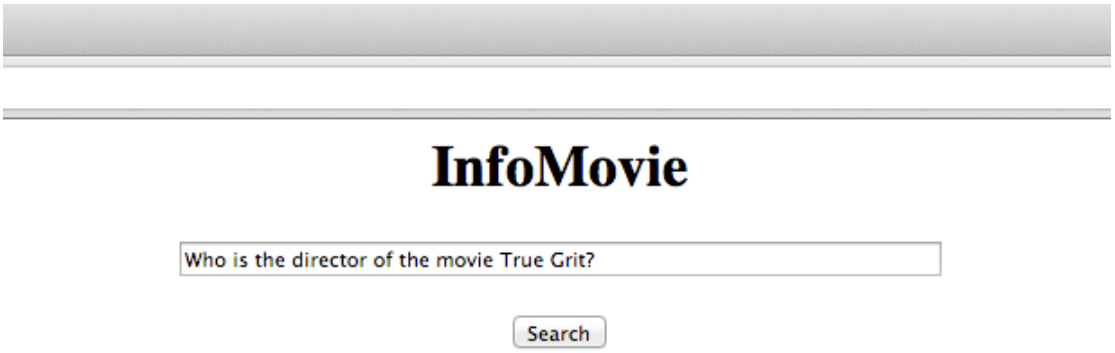

Figura 12 - Tela inicial

Figura 13 - Tela de resultado com lista de respostas

Ao perguntar a lista de filmes de um diretor, o InfoMovie mostra mais filmes do que quando procurando direto no IMDb. Isso acontece porque os filmes que ainda estão em produção já são listados e as informações desses filmes apenas são mostradas para um grupo de usuários do IMDb. No entanto, nos dados fornecidos pelo *site* já se encontram esses filmes. No arquivo "movies.list", que contém a lista de filmes, os filmes ainda não lançados aparecem no seguinte formato:

Nailed (2011)	2011
Nailed (????)	????
Nailed at the Plate (1918)	1918

Assim, sabemos que existe um filme chamado “Nailed” que ainda não tem data para ser lançado.

Vejamos o resultado da busca abaixo:

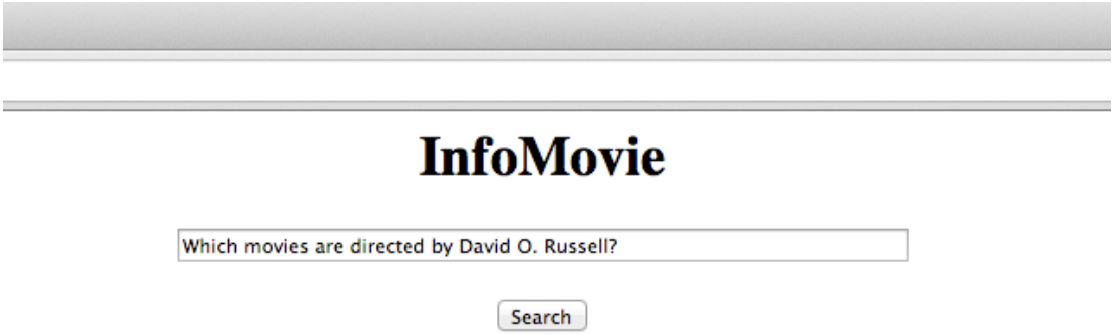

Figura 14 - Pergunta de filmes dirigidos por David O. Rusell

Figura 15 - Lista de respostas

Podemos notar que, como dito anteriormente, os filmes sem data de lançamento se encontram com a data no formato "(????)". Apesar de esses títulos não aparecerem na lista de filmes do IMDb, se buscarmos pelos títulos encontraremos a informação de que são filmes em produção. A figura abaixo mostra como o filme ainda em produção é mostrado no IMDb.

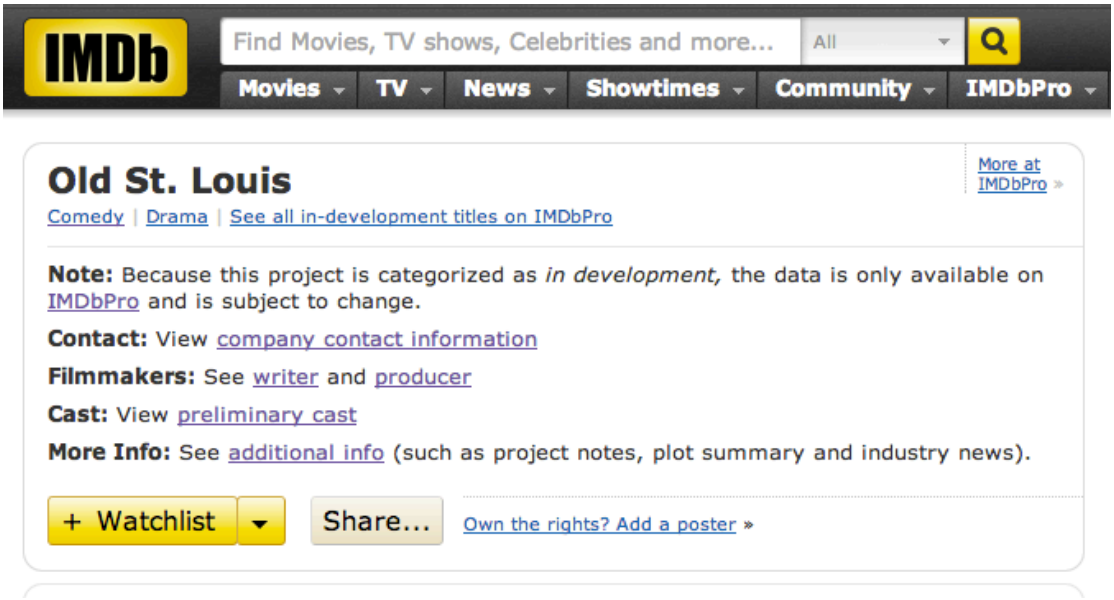

Figura 16 – Tela de informações do filme Old St. Louis no IMDb

Quando o sistema não encontra a resposta para uma pergunta, ele avisa o usuário de que a resposta não foi encontra. As telas a seguir mostram a busca pelo produtor de um filme, informação que ainda não existe no sistema:

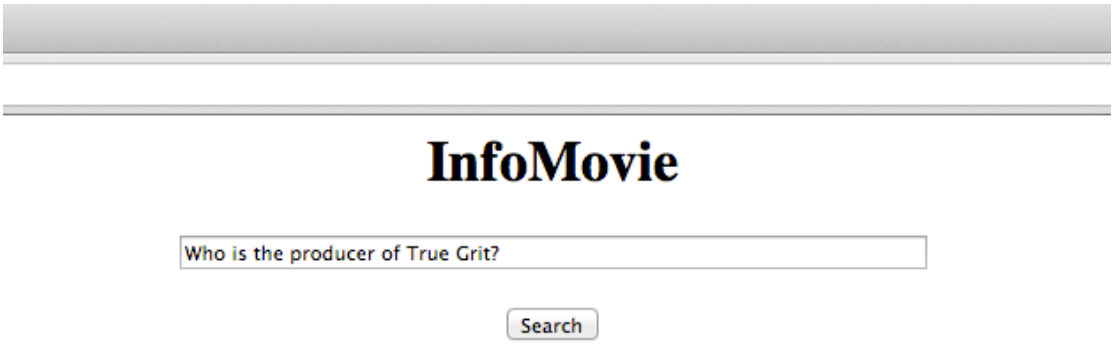

Figura 17 - Busca de informação de produtor

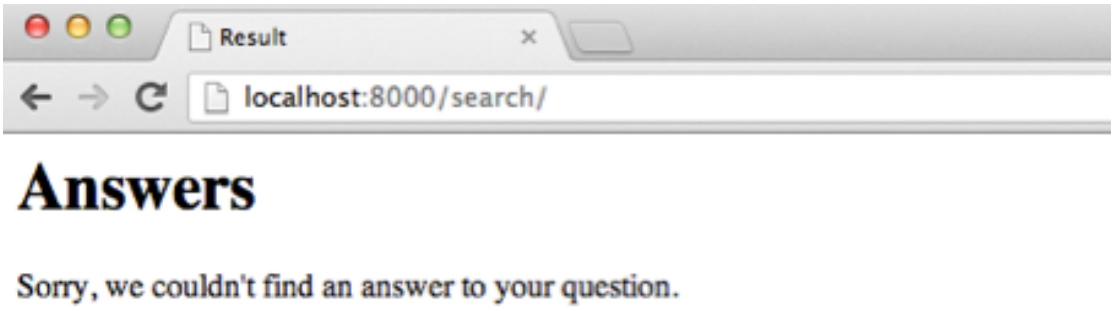

Figura 18 - Tela de resultado não encontrado

5 Conclusão

Os resultados mostraram que a maioria das perguntas que foram testadas foram respondidas corretamente. As perguntas que não possuem resposta no sistema não foram respondidas. Isso confirma que o sistema conseguiu ser 100% preciso, pois todas as perguntas que respondeu estavam corretas, porém não teve uma boa revocação, visto que não respondeu a todas as perguntas feitas.

Por ter funcionado corretamente com as perguntas relacionadas a diretores, pode-se assumir que o sistema funcionaria com outras perguntas dentro do escopo da ontologia "cinema", dado que o processo seria semelhante. Bastaria fazer o *parser* nos outros arquivos disponíveis no IMDb e adicionar funções no módulo interpretador e gerador de SQL.

As mesmas perguntas testadas na seção 4.3 foram testadas no sistema Evi [2]. O sistema não reconheceu as perguntas abaixo:

- "Which movies David O. Russell directs?"
- "Please, tell me the director of Silver Linings Playbook"
- "I'd like to know who is the director of Silver Linings Playbook"
- "Can you please tell me who directed Silver Linings Playbook?"
- "Do you know who directed Silver Linings Playbook?"
- "Who is the director of Silver Linings Playbook, please"
- "Which movies David O. Russell directs?"
- "Which movies David O. Russell directed?"
- "I'd like to know what films are directed by David O. Russell"
- "What others movies has David O. Russell directed?"
- "Do you know any others movies that David O. Russell directed?"

Notamos que o InfoMovie obteve melhor medida de revocação em suas respostas, ou seja, conseguiu responder a um maior número de perguntas dentre as perguntas testadas. O fato de ser baseado na ontologia "cinema" possibilitou uma maior atenção às possíveis formas de realizar uma mesma pergunta, criando assim, padrões mais abrangentes.

Podemos observar que perguntas contendo inglês informal, como "Which movies David O. Russell directed?" no sistema Evi não são respondidas. No

InfoMovie, devido à flexibilidade que o POS *tagger* proporcionou, esse tipo de pergunta, muito usada no inglês falado, pode ser respondida. Da mesma forma com expressões como "I'd like to know", "Can you tell me" e "please" podem ser adicionadas sem prejudicar o reconhecimento da pergunta. No entanto, como mostrado na seção 4.3, para as perguntas do tipo "Who is the director of Silver Linings Playbook, please", o InfoMovie também falha. Isso ocorre pois não é possível aplicar o POS *tagger*, visto que não é trivial reconhecer o nome de um filme, assim como o nome de um diretor que é sempre composto de substantivos.

Finalmente, o estudo realizado nesse trabalho mostrou que é possível construir um *Query-Answering System* baseado em ontologia que é capaz de encontrar respostas para perguntas relacionadas a cinema escritas em linguagem natural.

6 Trabalhos Futuros

Para o futuro, o primeiro passo seria estender o módulo de *parser*, realizando a extração de informação nos outros arquivos fornecidos e estender o módulo de interpretação de consultas e gerador SQL para responder consultas relacionadas às outras entidades.

Para melhorar a interface com o usuário, seria interessante criar um robô que visite diretamente as páginas do IMDb para obter as URLs do material recuperado e apresentá-las como links ao mostrar cada resultado. Dessa forma, o usuário poderia verificar a fonte de onde foram inferidas as respostas e se de fato estão corretas.

Ainda na interface, pode-se aplicar um algoritmo corretor, para que no caso de o usuário escrever uma palavra erroneamente o sistema seja capaz de corrigir automaticamente e, assim, não deixar de responder uma pergunta por um erro de digitação. Também seria interessante adicionar a função de *auto-complete*. Conforme o usuário escreve, o sistema prediz a pergunta que o usuário está digitando, baseando-se nas perguntas armazenadas que já foram feitas.

Para o algoritmo de interpretação de perguntas, seria interessante fornecer uma maneira de o usuário classificar se a resposta foi correta ou errada, assim os cálculos de precisão e revocação poderiam ser feitos e o algoritmo seria monitorado e melhorado conforme as avaliações.

Continuando com o algoritmo, caso uma pergunta não seja respondida, o sistema poderia permitir que o usuário escrevesse a resposta, caso ele soubesse, e salvar a pergunta de forma que essas perguntas fossem armazenadas e uma melhoria automática no algoritmo fosse feita.

Referências Bibliográficas

[1] "Língua Natural", http://pt.wikipedia.org/wiki/L%C3%ADngua_natural

[2] "Evi", www.evi.com

[3] "Ask", www.ask.com

[4] "IMDb", http://www.imdb.com/

[5] "MO – the Movie Ontology", http://www.movieontology.org/

[6] SMITH, B., WELTY, C., "Ontology: Towards a New Synthesis", *Formal Ontology in Information Systems*, ACM Press, 2001.

[7] GRUBER, T. R. "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition* 5, pp. 199–220, 1993.

[8] SOWA, J., "Ontology". Disponível em: <http://www.jfsowa.com/ontology/>.

[9] MANOLA, F., MILLER, E. "RDF Primer". Disponível em: <http://www.w3.org/TR/rdf-primer/>

[10] WIMALASURIYA, D., DEJING, D., "Ontology-Based Information Extraction: An Introduction and Survey of Current Approaches", *Journal of Information Science,* pp. 306-323, June, 2010.

[11] SUCHANEK, F., "Information Extraction", Disponível em: < http://pierre.senellart.com/enseignement/2011-2012/inf396/3-ie/slides3.pdf>.

[12] HEARST, M., "Automatic Acquisition of Hyponyms from Large Text Corpora". In*: Proceedings of the 14th International Conference on Computational Linguistics*, pp. 539-545, Stroudsburg, PA, USA, 1992.

[13] BIRD, S., KLEIN, E., LOPER, E., *Natural Language Processing with Python,* O'Relly, 2009.

[14] ANDROUTSOPOULOS, I., RITCHIE, G. D., THANISCH, P., "Natural Language Interface to Databases - An Introduction", *Journal of Natural Language Engineering*, Cambridge University Press, 1995.

[15] ANDROUTSOPOULOS, I., RITCHIE, G. D., THANISCH, P., "MASQUE/SQL – An Efficient and Portable Natural Language Query Interface for Relational Databases". In: Proceedings of the 6th international conference on Industrial and engineering applications of artificial intelligence and expert systems, pp. 327-330, 1993.

[16] HUNT, A., THOMAS, D., *The Pragmatic Programmer: From Journeyman to Master*, 1999.

[17] "Django", https://www.djangoproject.com/

[18] "DBPedia", http://dbpedia.org/

[19] "protégé", http://protege.stanford.edu/

[20] "astah", http://astah.net/

[21] "Alternative Interfaces", http://www.imdb.com/interfaces