UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

NAILSON BOAZ COSTA LEITE

UMA FERRAMENTA PARA A EXTRAÇÃO DE AXIOMAS DE UMA ONTOLOGIA UTILIZANDO PROGRAMAÇÃO EM LÓGICA INDUTIVA

São Luís
2013

NAILSON BOAZ COSTA LEITE

UMA FERRAMENTA PARA A EXTRAÇÃO DE AXIOMAS DE UMA ONTOLOGIA UTILIZANDO PROGRAMAÇÃO EM LÓGICA INDUTIVA

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profª Drª Rosário Girardi

São Luís

2013

Leite, Nailson Boaz Costa.

Uma ferramenta para a extração de axiomas de ontologias utilizando programação em lógica indutiva/ Nailson Boaz Costa Leite. – São Luís, 2013.

85f.

Impresso por computador (fotocópia).

Orientadora: Rosario Girardi.

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2013.

1. Aprendizagem de máquina. 2. Aprendizagem de axiomas de ontologias. 3. Programação em lógica indutiva. I. Título.

CDU 004.85

NAILSON BOAZ COSTA LEITE

UMA FERRAMENTA PARA A EXTRAÇÃO DE AXIOMAS DE ONTOLOGIAS UTILIZANDO PROGRAMAÇÃO EM LÓGICA INDUTIVA

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal do Maranhão para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em: 05 / 03 / 2013

BANCA EXAMINADORA

Prof.ª Drª Rosário Girardi (Orientadora)
Doutora em Ciência da Computação
Universidade Federal do Maranhão

Prof. Ivo José da Cunha Serra
Mestre em Engenharia de Eletricidade
Universidade Federal do Maranhão

Prof.ª Carla Gomes de Faria
Mestre em Engenharia de Eletricidade
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão

Para a minha família.

AGRADECIMENTOS

Agradeço primeiramente a minha família, principalmente aos meus pais, Nírodes e Deusanilde, que foram os meus primeiros e mais importantes educadores, por sempre zelar por minha felicidade e bem estar e por serem os meus grandes e incansáveis incentivadores em todas as minhas conquistas e superações.

Agradeço a Giulia, uma mulher espetacular, linda, inteligente e bem humorada, que preenche os meus dias com alegria e serenidade e sempre me dá forças para encontrar a paz e conforto, nos tempos em que me sinto mais vulnerável e inseguro.

Aos meus grandes amigos e companheiros do grupo GESEC, onde passei grande parte dos meus dias, em especial, Carla, Luís, Ivo e Rodrigo, que me auxiliaram com seus valiosos conselhos e ponderações.

Aos amigos que fiz no decorrer dessa dura jornada, que foi o curso de graduação. Nessa fase da minha vida fiz grandes e valorosos amigos que me apoiaram, me acompanharam e me ensinaram lições que guardarei para sempre.

Aos professores, a quem devo muito, sempre empenhados em oferecer o melhor para os seus alunos, para mim foi uma honra tê-los como tutores.

À minha orientadora Rosário Girardi, a quem tenho um enorme apreço e admiração, pela sua paciência, incentivo e orientação, que me guiaram no desenvolvimento desse trabalho.

À todos um muito obrigado, estou em eterna divida com vocês.

"Por aqui, contudo,
nós não olhamos para trás por muito tempo.
Nós seguimos em frente, abrindo novas portas e
fazendo coisas novas, porque nós somos curiosos...
e a curiosidade nos leva a novos caminhos
Siga sempre em frente."

Walt Disney

RESUMO

A Programação em Lógica Indutiva (PLI) é um campo interseção entre o Aprendizado de Máquina e a Programação em Lógica que investiga o aprendizado de hipóteses e faz uso da lógica matemática para resolver problemas, como por exemplo, induzir padrões por meio da experiência. A PLI permite a máquina aprender regras ao observar o conjunto de exemplos expressos em Lógica de Primeira Ordem. A descoberta de padrões pode servir por sua vez para a criação de regras de inferência a serem incluídas em uma base de conhecimento ou como axiomas em uma Ontologia. Uma contribuição da PLI para a aprendizagem de Ontologias poderia ser a identificação de novos axiomas a partir de suas instâncias de relacionamentos na construção de seu conjunto de treinamento.

Este trabalho vem contribuir ao Grupo de pesquisa em Engenharia de Software e Engenharia do Conhecimento (GESEC) no que tange o projeto de pesquisa Hermes: “Aprendizagem e Povoamento de Ontologias a partir de Fontes Textuais”, com a proposta de uma técnica e uma ferramenta semiautomática intitulada TAILP (Technique for Ontology Axioms extraction using ILP) para a extração de axiomas de uma ontologia povoada a partir de suas instâncias de relacionamentos não taxonômicos. Uma interface gráfica foi desenvolvida e integralizada com a ferramenta APPONTO que reúne aplicações para a construção automatizada de ontologias a partir de fontes textuais. Por fim um estudo de caso é proposto para a avaliação da técnica no domínio do direito sucessório, seus resultados são expostos e analisados.

Palavras-chave: Aprendizagem de Máquina; Aprendizagem de Axiomas de Ontologias; Programação em Lógica Indutiva.

ABSTRACT

Inductive Logic Programming (ILP) is an intersection field between Machine Learning and Logic Programming which investigates the learning of hypothesis and makes use of mathematical logic to solve problems, like for example, to induce patterns through experience. ILP allows the machine learn rules when observe the set of examples expressed in First-Order Logic. The discovery of patterns can serve in turn to the creation of inference rules to be included in a knowledge base or as axioms in an Ontology. One contribution of ILP for ontology learning could be the identification of new axioms from their instances of relationships for the construction of your training set.

This work contributes to the Research Group on Software Engineering and Knowledge Engineering (GESEC) regarding the research project Hermes: "Learning and Populating Ontologies from Textual Sources" with the proposal of a technique and a semiautomatic tool entitled TAILP (Technique for Ontology Axioms extraction using ILP) for the extraction of axioms from a populated Ontology from their instances of non-taxonomic relationships. A graphical interface was developed and integrated into the APPONTO tool that brings together applications for automated construction of ontologies from textual sources. Finally a study case is proposed to evaluate the technique in the field of inheritance law, this results are presented and analyzed. A graphical interface was developed and paid with a tool that gathers APPONTO applications for automated construction of ontologies from textual sources. Finally a case study is proposed to evaluate the technique is developed in the field of inheritance law, its results are presented and analyzed.

Keywords: Machine Learning; Learning Ontology Axioms; Inductive Logic Programming.

LISTA DE FIGURAS

Figura 1 - Camadas da Aprendizagem de Ontologias..... 24

Figura 2 - Exemplo de uma ontologia OWL. 27

Figura 3 - Headers da OWL 27

Figura 4 - Definição de taxonomia na OWL. 28

Figura 5 - Definição de relacionamento não taxonômico e sua instância..... 28

Figura 6 - Interface gráfica do Protégé, na seção de definição de instâncias de uma ontologia..... 29

Figura 7 - Programação em Lógica Indutiva, intercessão entre o Aprendizado de Máquina e a Programação em Lógica..... 30

Figura 8 - a) Exemplo de uma hipótese linear não consistente com todos os pontos. b) hipótese de polinômio de grau 2, mais consistente que a hipótese (a). c) hipótese consistente com os dados. d) hipótese de maior grau polinomial consistente com todos os pontos. 32

Figura 9 - O processo de indução de hipóteses a partir de exemplos na PLI. . 39

Figura 10- Exemplo de um grafo de refinamento, no domínio da família. 44

Figura 11 - Exemplo de arquivo de entrada para o Progol, contendo uma base de conhecimento no dominio da família. 46

Figura 12 - Exemplo de construção da arvore de refinamento no sistema Progol. 47

Figura 13 - Resultado da inferência indutiva realizada pelo Progol..... 47

Figura 14- Tarefas genéricas da técnica TAILP 49

Figura 15 – Detalhamento da tarefa “Extração de elementos da ontologia” 51

Figura 16 - Rede semântica representando as relações não taxonômicas de uma Ontologia no direito da família..... 52

Figura 17 - Geração da hipótese induzida 57

Figura 18 - Inserção de novo axioma na ontologia..... 58

Figura 19 - Seção da Aprendizagem de Ontologias da APPONTO. 61

Figura 20 - Tela inicial da ferramenta APPONTO. 61

Figura 21 - Seção de Povoamento de Ontologias da APPONTO. 63

Figura 22 - Tela da Aprendizagem da APPONTO, na tarefa da Identificação de Axiomas utilizando PLI. 64

Figura 23 - Tela inicial da TAILP, selecionando ontologia de entrada e diretório de saída..... 65

Figura 24 - Seleção de uma relação candidata a cabeça da hipótese. 65

Figura 25 - Avaliação e seleção de axiomas inferidos a serem adicionados na OWL. 67

Figura 26 - Estrutura de arquivos da APPONTO, em destaque as Classes da TAILP. 68

Figura 27 - Estrutura taxonômica da ontologia FamilyLaw. 71

Figura 28 - Classes da ontologia FamilyLaw. 71

Figura 29 - Propriedades da ontologia FamilyLaw. 72

Figura 30 - Relacionamentos não taxonômicos da ontologia FamilyLaw 72

Figura 31 - Instâncias da ontologia FamilyLaw. 73

Figura 32 - Seleção de arquivo .owl de entrada e diretório de saída para a ferramenta TAILP. 74

Figura 33 - Seleção da relação não taxonômica corpo da hipótese induzida. . 75

Figura 34 - Resultados apresentados na inferência das relações “grandfather_of” e “brother_of”. 77

Figura 35 – Axiomas induzidos visualizados pelo Protégé. 78

Figura 36 - Resultados de novo processamento de relações não taxonômicas da ontologia FamilyLaw. 78

Figura 37 - Lista de Axiomas Inferidos e utilização do plugin do Jess pelo Protégé. 79

Figura 38 - Resultados apresentados pelo SWRLJessTab na inferência dedutiva de novas instâncias de relacionamentos. 80

Figura 39 - Instâncias de relacionamentos não taxonômicos deduzidos dos axiomas induzidos pela TAILP. 80

LISTA DE TABELAS

Tabela 1 - Contabilidade das instâncias de relacionamentos não taxonômicos da ontologia FamilyLaw relevantes ao estudo de caso. 73

Tabela 2 - Resultados obitidos no estudo de caso. 81

LISTA DE SIGLAS

AM	Aprendizagem de Máquina
LPO	Lógica de Primeira Ordem
PLI	Programação em Lógica Indutiva
TAILP	Technique for Ontology Axioms extraction using ILP
SWRL	Semantic Web Rule Language
SPARQL	Protocol and RDF Query Language
OWL	Web Ontology Language
OIL	Ontology Inference Layer
DAML	DARPA Agent Markup Language
W3C	World Wide Web Consortium
RDF	Resource Description Framework
API	Application Programming Interface
XML	Extensible Markup Language
DIPPAO	Domain Independent Process for Automatic Ontology Population
DIPPAOTool	Tool for Automatic Ontology Population
JESS	Java Expert System Shell

SUMÁRIO

1 INTRODUÇÃO 17

1.1 ORGANIZAÇÃO DO TRABALHO 18

2 ONTOLOGIAS 20

2.1 ONTOLOGIAS E A WEB SEMÂNTICA 20

2.2 DEFINIÇÃO DE ONTOLOGIAS 21

2.3 APRENDIZAGEM E POVOAMENTO DE ONTOLOGIAS 23

2.4 CONSTRUÇÃO AUTOMATIZADA DE ONTOLOGIAS 24

2.5 OWL 26

2.6 PROTÉGÉ 28

3 PROGRAMAÇÃO EM LÓGICA INDUTIVA 30

3.1 APRENDIZAGEM DE MÁQUINA 30

3.1.1 Aprendizagem a partir de observações 32

3.2 PARADIGMA LÓGICO 33

3.2.1 Programação em Lógica 34

3.2.2 Clausulas de Horn 36

3.2.3 Prolog 37

3.3 INTRODUÇÃO A PLI 39

3.4 CONCEITOS BASICOS 39

3.5 MECANISMOS DE TENDÊNCIA 41

3.6 ESTRUTURAS PARA INDUÇÃO DE HIPOTESES 42

3.6.1 Utilização de modelo de regras 42

3.6.3 Resolução Inversa 44

3.7 PROGOL 45

3.7.1 Entrada do Progol 45

3.7.2 Saída do Progol 47

4 UMA TÉCNICA PARA A EXTRAÇÃO DE AXIOMAS DE UMA ONTOLOGIA UTILIZANDO a pROGRAMAÇÃO EM LÓGICA INDUTIVA 48

4.1 EXTRAÇÃO DE INSTÂNCIAS DE UMA ONTOLOGIA 50

4.2 GERAÇÃO DA HIPOTESE INDUZIDA 55

4.3 INSERÇÃO DE NOVA REGRA NA ONTOLOGIA 57

5 IMPLEMENTAÇÃO E INTEGRAÇÃO DA TAILP NA FERRAMENTA APPONTO 59

5.1 APPONTO 60

5.2 INTEGRAÇÃO DA TAILP 64

5.3 PRINCIPAIS CLASSES DA FERRAMENTA TAILP 68

6 AVALIAÇÃO 70

6.1 ONTOLOGIA FAMILYLAW 70

6.2 EXTRAINDO AXIOMAS DA ONTOLOGIA FAMILYLAW 74

6.3 RESULTADOS 81

7 CONCLUSÃO 82

REFERÊNCIAS 84

1 INTRODUÇÃO

Em meados dos anos 70, foram desenvolvidos os primeiros sistemas baseados em conhecimento, na época, chamados de sistemas de inferência orientados a padrões, que incorporavam a programação lógica em sua estrutura. A evolução desses formalismos se deu em encontro ao paradigma de orientação a objetos, em constante expansão no meio acadêmico. Com o advento da Web semântica, estes passaram a se tornar mais populares. Este cenário criou uma demanda para as Ontologias.

As Ontologias representam um elemento fundamental para a representação de conhecimento na Web semântica, estas possuem uma base formal bem estruturada facilitando a comunicação, o compartilhamento entre agentes e permitindo o reuso. Criar e povoar ontologias são tarefas árduas e caras. Classificar e estruturar um grande número de documentos manualmente é tedioso e até mesmo impraticável dependendo do tempo estipulado para sua realização, além disso, esse trabalho requer a contratação de profissionais altamente capacitados, o que resulta em um custo elevado. Uma saída para esses problemas é o desenvolvimento de mecanismos automáticos ou semiautomáticos que fornecessem suporte ao desenvolvimento de ontologias.

Os axiomas de uma ontologia são verdades em um esquema lógico, geralmente utilizados para restringir a informação contida em uma ontologia, verificar sua corretude ou deduzir novas informações. O processo de construção e manutenção de axiomas constitui-se por uma tarefa da Aprendizagem de Maquina. Automatizar esse processo é um passo importante na construção automatizada de ontologias (Freitas, 2003). A Aprendizagem de Máquina é um subcampo da Inteligência Artificial que estuda meios para simular a aprendizagem humana em aplicações computacionais. Um dos mais importantes meios de aprendizagem humana é a indução de conhecimento através a experiência.

A Programação em Lógica Indutiva (PLI) é um subcampo da Aprendizagem de Máquina que utiliza Programação em Lógica como uma representação uniforme para exemplos, conhecimentos de base e hipóteses e faz uso da Lógica Matemática e da Aprendizagem de Máquina para aprendizado indutivo

de padrões. Seu objetivo é generalizar o conhecimento para aplicar a qualquer situação nova que venha a enfrentar em seu ambiente. Uma de suas principais aplicações é a indução de hipóteses ou regras a partir de um conjunto de treinamento. A representação do conhecimento da Programação em Lógica, combinada com a habilidade de reconhecimento de padrões da Aprendizagem de Máquina, torna a PLI uma solução viável para a aprendizagem de ontologias já que elas tratam de maneira análoga o seu conhecimento. Porem a indução necessita de fatos para o seu funcionamento, isto é, a estrutura de uma ontologia não é o suficiente na para a descoberta de padrões, a indução requer fatos ou instâncias de um relacionado domínio para realizar os seus mecanismos de inferência. As instâncias de relacionamentos não taxonômicos de ontologias são ideais para a construção de uma base de conhecimento para um sistema PLI, pois além de poder apresentar uma maior diversidade de instâncias estas representam a base na construção de axiomas de ontologias.

Este trabalho apresenta uma nova abordagem para o aprendizado de axiomas de ontologias utilizando a PLI, com o desenvolvimento de uma técnica e uma ferramenta que realizam inferências indutivas a partir de suas instâncias de relacionamentos não taxonômicos.

1.1 ORGANIZAÇÃO DO TRABALHO

O **"Capitulo 2"** inicia-se com uma contextualização deste trabalho com a Web semântica, é feita uma introdução aos seus conceitos, objetivos, dificuldades e tecnologias atuais. Trata também da definição formal das Ontologias e todos os seus elementos, sua importância no contexto da Web semântica. É apresentada a definição de aprendizagem e povoamento de Ontologias e o modelo de aprendizagem em camadas. Após o levantamento de conceitos teóricos nas seções finais são destacadas as tecnologias atuais que auxiliam no desenvolvimento de Ontologias tal como uma ferramenta para a sua manipulação, utilizada no desenvolvimento desse trabalho.

Uma breve introdução de aprendizagem de máquina é apresentada no **"Capitulo 3"**, dando enfoque a aprendizagem indutiva. É dada uma explicação teórica da PLI, seus conceitos, suas estruturas de indução e uma

exemplificação com o funcionamento da ferramenta Progol, ferramenta esta que será utilizada no desenvolvimento de uma ferramenta e na sua avaliação.

O **"Capitulo 4"** propõe de uma maneira pratica e exemplificada uma técnica para a extração de axiomas de Ontologias intitulada TAILP (Technique for Ontology Axioms extraction using Inductive Logic Programming) que aplica dos conceitos da PLI juntamente com as definições formais de Ontologias para a descoberta de regras através da indução que podem vir a ser axiomas de uma ontologia.

Em seguida, o **"Capitulo 5"** apresenta uma ferramenta de software para a extração semiautomática de axiomas de Ontologias utilizando os passos da técnica TAILP e a sua integração com o ONTOLEARN, que propõe reunir todas as ferramentas desenvolvidas no laboratório de pesquisa GESEC em um processo dinâmico e incremental de aprendizagem e povoamento de Ontologias.

Por fim, **"Capitulo 6"** apresenta um estudo de caso utilizando uma ontologia no domínio do direito sucessório realizando a extração de axiomas com a ferramenta desenvolvida. A entrada para a ferramenta é a Ontologia Family Law já previamente povoada utilizando um corpus no domínio do direito sucessório. A partir de seu conjunto de instâncias e do "feedback" do usuário serão inferidas hipóteses que poderão ser ou não inseridas na Ontologia como axiomas e uma analise dos resultados obtidos é feita. E finalmente no **"Capítulo 7"** são apresentadas as conclusões e considerações finais sobre este trabalho.

2 ONTOLOGIAS

Este capítulo foca o principal objeto de estudo deste trabalho que são as Ontologias e a sua construção automatizada, dando ênfase na aprendizagem de um de seus componentes, os axiomas. Este capítulo está organizado da seguinte forma: A seção 2.1 apresenta uma contextualização das ontologias como área de pesquisa da Web semântica e descreve os conceitos e objetivos desse novo conceito de Web, assim como a importância das ontologias como meios de representação de conhecimento essenciais para o seu desenvolvimento.

A seção 2.2 apresenta uma definição formal de ontologias, aceita pelo grupo de pesquisa GESEC e utilizada no decorrer desse trabalho, descrevendo cada um de seus elementos e exemplificando-os. A seção 2.3 trata dos conceitos da aprendizagem e povoamento de ontologias. A seção 2.4, por sua vez, discute sobre a construção automatizada de ontologias, descrevendo um modelo de aprendizagem em camadas desenvolvida por Buitelaar, Cimiano e Magnini (Cimiano, Hotho, & Staab, 2004). A seção 2.5 apresenta a linguagem de representação de ontologias OWL, onde será posteriormente utilizada na construção de uma ferramenta para a extração de axiomas de uma ontologia no capítulo 5. Por fim, abordando as tecnologias existentes para criação, manipulação e instanciação de ontologias é apresentada a ferramenta Protégé, ferramenta utilizada para construção e visualização das ontologias utilizadas ao longo deste trabalho.

2.1 ONTOLOGIAS E A WEB SEMÂNTICA

Atualmente a Web 2.0 se caracteriza por um imenso conjunto de documentos em linguagens de marcação como o HTML ou XML, largamente utilizadas no desenvolvimento de sites. Essas linguagens Web descrevem o seu conteúdo e a apresentação da informação, porém, descrevem somente a sua estrutura léxica e sintática, deixando a estrutura semântica à parte dessa descrição. Somando-se à enorme quantidade de informações hoje disponíveis na Internet, recuperar alguma informação relevante tornou-se uma tarefa árdua. Devido à falta de estruturação semântica da informação disponível, a Internet é hoje, um

ambiente completamente hostil às entidades de software como os agentes inteligentes. Essas entidades poderiam ter uma larga utilização em atividades como o comércio eletrônico e sistemas de recuperação de informação.

Como resposta à demanda de aplicações capazes de processar as informações contidas na internet Tim Berner's Lee, o "inventor" da World Wide Web, em um artigo (Lee, Hendler, & Lassila, 2001) define o que vem a ser a Web semântica onde as paginas Web possuem uma estrutura semântica. O objetivo dessa definição é permitir que as máquinas façam o processamento da informação que atualmente é feito pelos seres humanos, sem a criação de sistemas inteligentes que manipulam a Web sintática atual, mas modificando a Web dando a ela um "significado".

Para a criação da Web semântica são necessárias estruturas de representação de conhecimento que se ajustam ao ambiente distribuído e descentralizado que é a internet. A constante evolução de sistemas de representação de conhecimento deu origem às ontologias. Esta tecnologia adapta-se perfeitamente à necessidade de compartilhamento, reuso e manipulação do conhecimento dos sistemas inteligentes.

2.2 DEFINIÇÃO DE ONTOLOGIAS

Representar o conhecimento disponível na Web não é uma tarefa fácil para a Web Semântica, são necessários mais do que algumas linguagens de representação como XML e RDF. A carência de soluções capazes de captar e representar a semântica das páginas da Web criou uma demanda de serviços que ajusta-se à aplicação de ontologias. Esta tecnologia desponta como uma forma viável de estruturar informações esparsas disponíveis na rede.

Para analisarmos a Construção de Ontologias ou até a Construção Automatizada de Ontologias (Cimiano, Hotho, & Staab, 2004) é necessário estabelecer um conceito e também ter uma definição formal ao seu respeito. Alguns autores divergem no conceito de ontologias, portanto, iremos usar a definição formal utilizada pelo GESEC para facilitar a compreensão e para um eventual reuso deste material em pesquisas futuras.

Uma Ontologia (Faria & Girardi, Um Processo Semi-Automático para o Povoamento de Ontologias a partir de Fontes Textuais, 2010) consiste em um

conjunto de conceitos e relacionamentos taxonômicos ou não taxonômicos que representam o conhecimento de um domínio especifico, a esses conceitos estão relacionados a propriedades e instâncias que dão valores a essa propriedade. Conceitos estes são organizados hierarquicamente pelos relacionamentos taxonômicos. Esse tipo de organização e definição formal possibilita e facilita a compreensão e o processamento da representação de informação por uma máquina e o compartilhamento e o reuso entre vários agentes de software distribuídos pela internet.

Formalmente, uma ontologia pode ser definida como a tupla:

$$\mathbf{O := (C, H, I, R, P, A)}$$

Onde:

a) $\mathbf{C = C^C \cup C^I}$ é o conjunto de entidades do domínio sendo modelado. O conjunto $\mathbf{C^C}$ é formado por classes, ou seja, conceitos que representam entidades que descrevem um conjunto de objetos (por exemplo, "Mãe" $\in \mathbf{C^C}$) enquanto que o conjunto C^I é formado por instâncias, ou seja, entidades únicas no domínio (por exemplo, "Anne Smith" $\in \mathbf{C^I}$).

b) $\mathbf{H = \{tipo\_de\ (c_1,c_2)\ |\ c_1 \in C^C \wedge c_2 \in C^C\ \}}$ é o conjunto de relações taxonômicas que definem a hierarquia de classes da ontologia e são denotadas por "tipo_de(c_1,c_2)" indicando que c1 é uma subclasse de c2. Um exemplo desse relacionamento é "tipo_de(Mãe,Pessoa)".

c) $\mathbf{I = \{}\boldsymbol{é\_um}\mathbf{(c_1,c_2)\ |\ c_1 \in C^I \wedge c_2 \in C^C\ \}\ U\ \{}\boldsymbol{prop_k}\mathbf{(c_i,\ valor)\ |\ }\boldsymbol{c_i} \in \mathbf{C^I\ \}\ U\ \{}\boldsymbol{rel_k}\mathbf{(c_1,\ c_2,\ \ldots,\ c_n)\ |\ \forall i,\ }\boldsymbol{c_i} \in \mathbf{C^I\}}$ é o conjunto de relacionamentos entre os elementos da ontologia e suas instâncias, por exemplo "é_um("Anne Smith", Mãe)", "data_de_nascimento ("Anne Smith", "12/02/1980")" e "mãe_de("Anne Smith", "Clara Smith")" são relacionamentos entre classes, relacionamentos, propriedades e suas instâncias.

d) $\mathbf{R = \{rel_k(c_1,c_2,\ldots,\ c_n)\ |\ \forall i,\ c_i \in C^C\}}$ é o conjunto de relacionamentos não taxonômicos de uma ontologia. Por exemplo, "mãe_de(Mãe, Filha)".

e) **$P = \{prop_K(c_i,tipo) \mid c_i \in C^C\}$** é o conjunto de propriedades das classes de uma ontologia e seu tipo de dados básico. Por exemplo, “data_de_nascimento(Mãe, dd/mm/aaaa)”.

f) **$A: = \{condition_x \Rightarrow conclusion_y(c_1, c_2,..., c_n) \mid \forall j, c_i \in C^C\}$** é um conjunto de axiomas, regras que permitem checar a consistência da ontologia e deduzir novos conhecimentos através de algum mecanismo de inferência. O termo **$Condition_x$** é dado por: **$Condition_x = \{(cond_1, cond_2,\ldots, cond_n) \mid \forall z, cond_z \in H \cup I \cup R\}$.** Por exemplo: {comprou(Cliente, Livro1), autor(Autor, Livro1), autor(Autor, Livro2)} → "provável-comprador”(Cliente, Livro2).

2.3 APRENDIZAGEM E POVOAMENTO DE ONTOLOGIAS

O suporte automático ou semiautomático na construção de Ontologias é chamado de Aprendizagem de Ontologias (Shamsfard & Barforoush, 2003). A Aprendizagem de Ontologias esta relacionada com a extração de elementos (conceitos e relações) de dados de entrada e construir uma ontologia a partir desses elementos. Essa construção pode ser manual, porém, é uma tarefa custosa e que demanda muito tempo, sendo tediosa e passível a erros, além de cara e específica ao propósito.

Uma solução para este problema é a automatização da construção das Ontologias. Essa automação, além de eliminar os custos citados, estabelece uma melhor relação entre a Ontologia e os dados de entrada como, por exemplo, um corpus (o conceito de corpus, plural corpora, seria um conjunto de documentos textuais relacionados a um determinado assunto) utilizado na construção de Ontologias a partir de fontes textuais que é o objetivo do projeto Hermes, vigente no grupo de pesquisa GESEC. Outras abordagens para a aprendizagem de Ontologias podem ser utilizadas como um processo de construção por camadas, descrito na seção 2.4, utilizando como um exemplo, o objetivo deste trabalho que é a realização da Aprendizagem dos axiomas de uma ontologia a partir de outra previamente construída através de uma técnica incremental na descoberta de novos axiomas a partir de instâncias.

Já o processo complementar é o Povoamento de Ontologias que consiste em uma abordagem para automatizar ou semi-automatizar a instanciação de extensões de classes, seus relacionamentos não taxonômicos e suas propriedades, extensões estas que são objetos únicos na ontologia, também conhecidos como instâncias. Essas instâncias são extraídas a partir do conhecimento descoberto em diferentes fontes de dados como, por exemplo, através de documentos textuais. Formalmente, o problema do Povoamento de Ontologias é extrair um subconjunto I' do conjunto I da definição de ontologia apresentada na seção 2.2.

2.4 CONSTRUÇÃO AUTOMATIZADA DE ONTOLOGIAS

Na Construção Automatizada de Ontologias, deverão ser utilizadas várias ferramentas e algoritmos para um modelo de aprendizado definido por Buitelaar, Cimiano e Magnini (Cimiano, Hotho, & Staab, 2004). Esse modelo é formado por camadas, cada uma das camadas é analisada e processada individualmente em um conjunto de tarefas que visam à extração de determinadas informações para a construção de Ontologias (Figura 1)

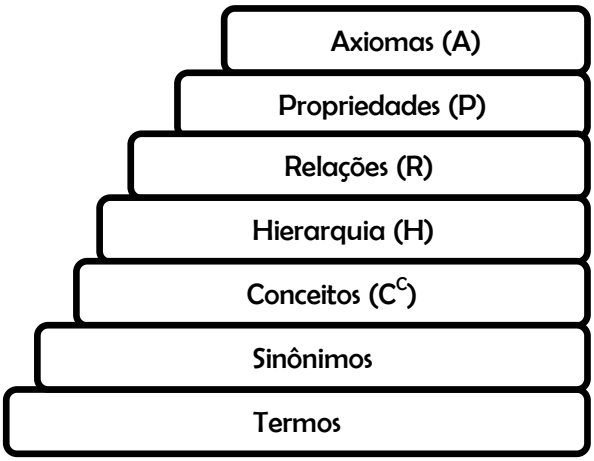

Figura 1 - Camadas da Aprendizagem de Ontologias

- Termos

A extração de termos serve de base para as camadas seguintes e é parte essencial da aprendizagem de Ontologias. Ela consiste em obter a partir de fontes textuais termos que poderão ser elementos (conceitos, relações, etc.) que constituem uma ontologia.

As técnicas utilizadas nessa etapa são inspiradas no processamento da linguagem natural e na recuperação de informação (análise estatística).

- Sinônimos

A extração de Sinônimos é importante principalmente para evitar a criação de conceitos redundantes. O WordNet (FELLBAUM, 1998) é uma ferramenta muito usada para essa finalidade na língua inglesa.

- Conceitos (C^C)

Uma definição de conceito ainda não esta bem clara e consolidada, mas segundo Buitelaar, Cimiano e Magnini, um conceito deve ter uma intensão (descrição formal do objeto), uma extensão (conjunto de instâncias), Um conjunto de realizações linguísticas (conjunto de termos em linguagem natural que o designam). Algumas abordagens como Programação em Lógica Indutiva e a Analise Formal de Conceitos realizam a extração de conceitos com base na intenção. Também existem aquelas que se focam nas realizações linguísticas, é o caso das técnicas de agrupamento.

- Hierarquia (H)

A extração de Hierarquias consiste em utilizar os conceitos adquiridos na camada anterior e o corpus para extrair as relações hierárquicas. Agrupamento conceitual (Cimiano, Hotho, & Staab, 2004) e Agrupamento Hierárquico (Maedche & Staab, 2004) são exemplos de técnicas usados para extração de hierarquias.

- Relações (R)

São as relações que não fornecem uma ideia de Hierarquia ou de instância, exemplo disso é: dono(Joaquim, Padaria).

- Propriedades (P)

Propriedades são atributos simples relacionados a conceitos e instâncias. Por exemplo, um conceito *carro*, pode ter uma propriedade *cor*, que relaciona instâncias de carro a cores (vermelho, preto, branco,...).

- Axiomas (A)

Os Axiomas são verdades em um esquema Lógico. Há dois modelos apresentados: Os Axiomas podem ser previamente definidos ou a extração de Axiomas pode ser feito como um processo de Aprendizagem. Como foi visto na construção automatizada de Ontologias os dois modelos deviam ser empregados para se obter um melhor resultado. Para a automatização completa de fato da construção de ontologias usando o modelo apresentado é necessário que cada uma dessas camadas seja extraída automaticamente de um corpus de domínio especifico utilizando técnicas em pipeline em um processamento dinâmico e interativo.

2.5 OWL

As páginas da Web Semântica são baseadas em um conhecimento estruturado em ontologias, que torna linguagens como a OWL (Web Ontology Language ou linguagem de ontologias para a Web) extremamente importantes para o seu desenvolvimento. A OWL é uma linguagem para definir e instanciar ontologias. Uma ontologia OWL pode incluir descrições de classes e suas respectivas propriedades e seus relacionamentos. Essa linguagem foi projetada para uso por aplicações que precisam processar o conteúdo da informação ao invés de apenas apresentá-la aos humanos. Ela facilita a interpretação por máquinas do conteúdo da Web em relação a XML, RDF e RDF Schema por fornecer vocabulário adicional com uma semântica formal.

A OWL é baseada nas linguagens OIL (Ontology Inference Layer, camada de inferência para ontologias) e DAML (DARPA Agent Markup Language, linguagem de anotação para agentes do Departamento de Defesa dos Estados Unidos), sendo a OWL hoje uma recomendação da W3C para a representação de ontologias para a Web semântica. Ela vem ocupando um papel importante em um número cada vez maior de aplicações e vem sendo foco de pesquisa para ferramentas, técnicas de inferência e extensões de linguagens. Nesse cenário desponta uma poderosa ferramenta para a construção e manuseio de ontologias: o ambiente Protégé (Gennari, 2003).

A Figura 2 ilustra um exemplo de uma ontologia em OWL com as definições da Classe “Person” e de suas subclasses “Man” e “Woman”.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns="http://www.owl-ontologies.com/family#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/family">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Man">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Person"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Woman">
    <rdfs:subClassOf rdf:resource="#Person"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="uncle_of">
    <rdfs:range rdf:resource="#Person"/>
    <rdfs:domain rdf:resource="#Man"/>
  </owl:ObjectProperty>
  <Man rdf:ID="Man_1">
    <uncle_of>
      <Woman rdf:ID="Woman_3"/>
    </uncle_of>
  </Man>
</rdf:RDF>
```

Figura 2 - Exemplo de uma ontologia OWL.

2.5.1 Tipos Básicos da OWL

Para definir uma ontologia em OWL, é preciso em primeiro lugar, dizer onde estão, na Web, as classes primitivas das ontologias para que se possam definir novas classes como subclasse destas. Também é necessário determinar um namespace para a nova ontologia, isto é codificado da seguinte forma num trecho da ontologia chamado de Headers, como no exemplo da Figura 3.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns="http://www.owl-ontologies.com/family#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://www.owl-ontologies.com/family">
</rdf:RDF>
```

Figura 3 - Headers da OWL

A definição de classes e subclasses é realizado pela marcação <rdfs:subclassOf/>, como a Figura 4 demonstra, que Man é subclasse de Person.

```
<owl:Class rdf:ID="Man">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Person"/>
  </rdfs:subClassOf>
</owl:Class>
```

Figura 4 - Definição de taxonomia na OWL.

A instanciação de Classes é demasiadamente simples, como é mostrado abaixo. Um fato digno de nota, é que, em lógica de descrições, a definição dos atributos não tem de estar junto com a classe.

```
<Man rdf:ID="Man_1"/>
```

Os Relacionamentos não taxonômicos são chamados de ObjectProperty e são descritos por ranges e domains, que restringem as classes que podem participar desse relacionamento. A instanciação do relacionamento é determinada como a Figura 5 demonstra.

```
<owl:ObjectProperty rdf:ID="uncle_of">
  <rdfs:range rdf:resource="#Person"/>
  <rdfs:domain rdf:resource="#Man"/>
</owl:ObjectProperty>
<Man rdf:ID="Man_1">
  <uncle_of>
    <Woman rdf:ID="Woman_3"/>
  </uncle_of>
</Man>
```

Figura 5 - Definição de relacionamento não taxonômico e sua instância.

2.6 PROTÉGÉ

As Ontologias estão se tornando cada vez mais populares em esquemas de modelagem de serviços de gestão de conhecimento e aplicações para a Web Semântica. O desenvolvimento de ferramentas para visualizar graficamente ontologias está crescendo para ajudar na sua avaliação e análise. A visualização gráfica ajuda a navegar e compreender a estrutura de uma ontologia.

O Protégé (Gennari, 2003) é uma das ferramentas mais amplamente utilizadas no desenvolvimento de ontologias, se trata de uma plataforma open source desenvolvida em Java pelo centro de pesquisa para Biomedicina de Stanford que dispõe de um conjunto ferramentas e uma API para a modelagem, criação, visualização e exportação de Ontologias em vários formatos de representação recomendados pela W3C tal como, RDF, OWL. Protégé fornece um editor intuitivo para ontologias e tem extensões para a sua visualização, gerenciamento de projetos, engenharia de software e outras tarefas de modelagem. Possui uma plataforma extensível para novos componentes com o intuito de adicionar novas funcionalidades e serviços. Existe um número crescente de plug-ins que oferecem uma variedade de características adicionais, tais como ferramentas extras de gestão de ontologia, o suporte a geração de resultados multimídia, mecanismos de pesquisa, métodos de resolução de problemas que utilizam motores de raciocínio, etc.

A sua interface gráfica (Figura 6) possui funcionalidades para adicionar classes e suas subclasses, instâncias, relacionamentos. Torna-se muito útil para instanciar classes de ontologias, assim como para definir os seus relacionamentos não taxonômicos e suas propriedades.

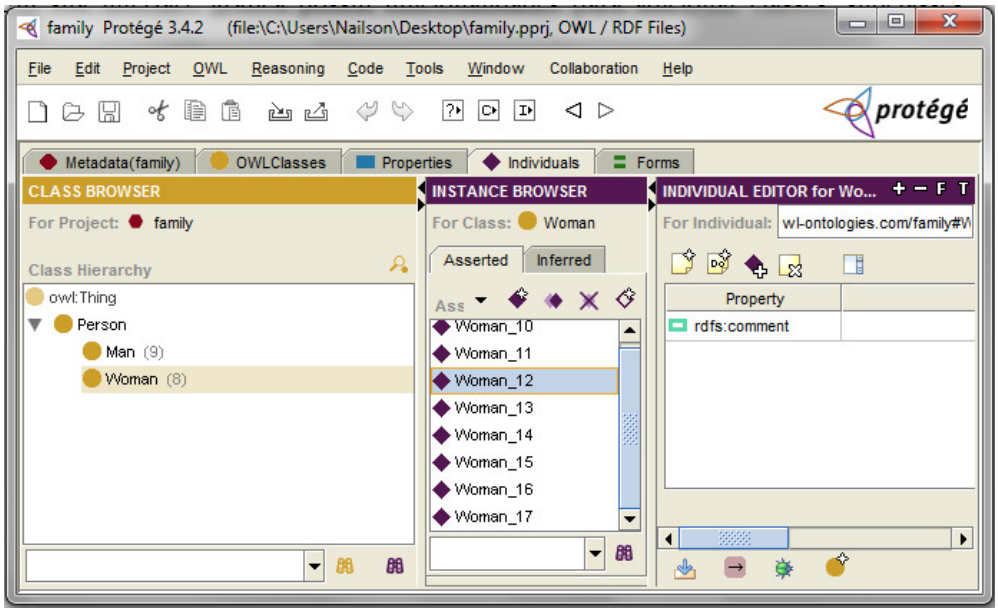

Figura 6 - Interface gráfica do Protégé, na seção de definição de instâncias de uma ontologia.

3 PROGRAMAÇÃO EM LÓGICA INDUTIVA

A Programação em Lógica Indutiva (PLI) ((Bratko, Kubat, & Michalski, 1998) (Michalski, Cambonell, & Michell, 1983) é um campo de interseção do Aprendizado de Máquina e da Programação em Lógica (Figura 7) que investiga o aprendizado de hipóteses. Uma das suas aplicações é a descoberta de padrões através da indução. A descoberta de padrões pode servir por sua vez para a criação de regras de inferência para uma base de conhecimento ou como axiomas em uma ontologia. Por exemplo, dada uma base de conhecimento o objetivo é, usando a PLI, induzir novas regras para a base de conhecimento. Esta seção apresenta os conceitos de Aprendizado de Máquina e da PLI, demostra também a ferramenta Progol que utiliza a PLI para a indução de regras.

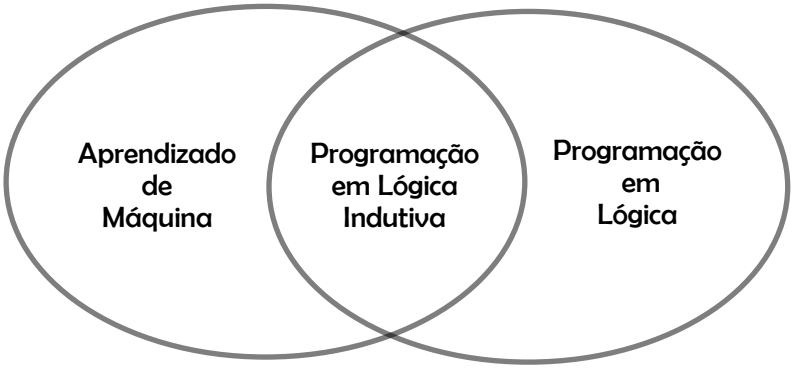

Figura 7 - Programação em Lógica Indutiva, intercessão entre o Aprendizado de Máquina e a Programação em Lógica.

3.1 APRENDIZAGEM DE MÁQUINA

As máquinas foram criadas com o objetivo de realizar tarefas de forma automática e, primeiramente, eram dotados de algoritmos estruturados e previsíveis em um paradigma que consistia somente de uma entrada e suas respectivas saídas. Porem, as percepções do ambiente e a sua interação com ele podem servir não só para gerar ações, mas também para melhorar a habilidade da máquina a reagir no futuro.

O Aprendizado de Máquina (Bishop, 2006) é uma área da Inteligência Artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o

processo do aprendizado, com algoritmos que, através da experiência permitem ao computador aperfeiçoar o seu desempenho em alguma tarefa.

O Paradigma de Aprendizado de Máquina inclui a aprendizagem Indutiva, que extrai regras e padrões de grandes conjuntos de dados a partir de observações dados, ou seja, obtêm conclusões genéricas a partir de um conjunto de exemplos (conjunto de treinamento), a aprendizagem dedutiva, que extrai regras específicas sobre um domínio a partir de um dado conhecimento de fundo[1] (background knowledge) e a aprendizagem conexionista que utiliza conceitos de redes neurais para treinamento e aprendizagem de regras. Além disso, vários Paradigmas de Aprendizagem podem ser integrados em uma estratégia múltipla (sistema de múltipla estratégia) (Lavrac & Dzeroski, 1994). Os estudos de aprendizagem de máquina costumam distinguir três casos de aprendizagem (Russel & Norving, 1995): aprendizagem supervisionada, não-supervisionada e por reforço.

Na **aprendizagem supervisionada** (Monard & Baranauskas, 2003), o objetivo é gerar um classificador capaz de definir corretamente a classe de novos exemplos ainda não rotulados a partir de raciocínio sobre exemplos externos ao sistema de aprendizado, ou seja, a aprendizagem de uma nova função a partir de exemplos de sua entrada e saída de forma a se aproximar o máximo possível da função perfeita. O aprendizado supervisionado é bastante utilizado na mineração de dados para o aprendizado de Arvores de Decisão, na Programação em Lógica Indutiva e, também, na realização de treinamento de Redes Neurais.

O problema da **aprendizagem não-supervisionada** consiste da aprendizagem de padrões de entrada, quando não são fornecidas os seus valores de saída específicos, portanto, existe a incerteza sobre a saída esperada, desta forma, é necessário utilizar os métodos probabilísticos para obtenção de resultados. Uma de suas principais aplicações é desvendar a organização dos padrões existentes nos dados através de clusters (agrupamentos) consistentes, como, por exemplo, em técnicas de clustering na mineração de dados e na aprendizagem de redes bayesianas.

[1] "Background knowledge"**,** também chamado de conhecimento prévio ou conhecimento de fundo, é um conjunto de fatos, ou informações que descrevem o mundo em um dado domínio.

Por fim, na **aprendizagem por reforço** o agente não recebe um objetivo, ele deve ser aprendido por reforço (com o uso de recompensas). O agente recebe uma avaliação na execução de sua ação. Essa pode ser positiva (“recompensa”) ou negativa (“punição”), o resultado é armazenado e servirá de referência para a próxima ação do agente. Uma das primeiras aplicações da aprendizagem por reforço foi na década de 50 em um jogo de damas, ela também é muito utilizada na área da robótica.

3.1.1 Aprendizagem a partir de observações

A Aprendizagem Indutiva é um campo da Aprendizagem Lógica que faz jus ao título, ao aprender com base em induções, através de exemplos. A estrutura que aprende é chamada de agente (aprendiz). O agente aprende regras ao observar o conjunto de exemplos, e seu objetivo é generalizar o conhecimento para aplicar a qualquer situação nova que venha a enfrentar em seu ambiente, ou seja (Russel & Norving, 1995):

Dada uma coleção de exemplos de *f*, retornar uma função *h* que se aproxime de *f*.

A função *h* é chamada hipótese. A Figura 8 demostra o ajuste de uma função de uma única variável, que seria a hipótese a ser induzida a alguns pontos dados, representando os exemplos. Por fim a ultima função chega a uma hipótese consistente com os pontos dados, porque concorda com todos os dados.

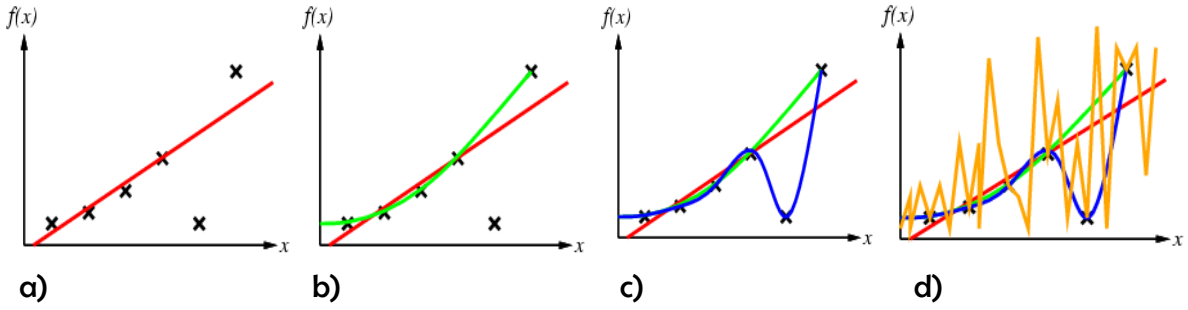

Figura 8 - a) Exemplo de uma hipótese linear não consistente com todos os pontos. b) hipótese de polinômio de grau 2, mais consistente que a hipótese (a). c) hipótese consistente com os dados. d) hipótese de maior grau polinomial consistente com todos os pontos.

A Figura 8 também ilustra o problema fundamental da indução e a primeira questão na aprendizagem indutiva que define como escolher entre as varias hipóteses disponíveis. Uma resposta é definida pela lâmina de Ockham (Russel & Norving, 1995), que consiste em escolher a hipótese mais simples consistente com os dados, pois as hipóteses mais complexas que os próprios dados deixam de extrair algum padrão deles.

No exemplo da Figura 8 podemos perceber que a ultima hipótese (*d*) atinge um grau polinomial superior ao da hipótese (*c*) que se aproxima mais dos valores esperado e ambas estão consistentes com os exemplos disponíveis. Portanto esta deverá ser a hipótese escolhida pelo agente como a melhor hipótese induzida em relação aos pontos.

Deve-se ter em mente que a possibilidade ou impossibilidade de encontrar uma hipótese simples e consistente depende fortemente do espaço de hipóteses escolhido, dizemos que um problema de aprendizagem pode ser realizável se o espaço de hipóteses contem a função verdadeira, do contrario ela é irrealizável.

3.2 PARADIGMA LÓGICO

Para a definição de Programação em Lógica Indutiva será necessário primeiramente uma revisão do paradigma lógico e da programação em lógica. Este capitulo introduz os termos e conceitos utilizados no decorrer desse trabalho e é fundamental para o entendimento das na parte teórica e pratica da PLI, discutida posteriormente na Seção 3.3.

O Paradigma Lógico é um paradigma de programação declarativo que faz uso da Lógica matemática para resolver problemas. Segundo este paradigma programas são relações matemáticas. Relações são mais genéricas do que mapeamentos, portanto programação lógica é mais alto nível que imperativa ou funcional. Os Métodos de Aprendizagem Lógicos trazem muitas vantagens na construção de sistemas inteligentes (Duarte, 2001) como a facilidade em acrescentar novos fatos a um sistema, sem mudar outros fatos e pequenos procedimentos. A Programação em Lógica tenta estender estas vantagens para todas as tarefas de programação. Na abordagem lógica qualquer

computação pode ser vista como um processo de tornarem explícitas as consequências de uma regra, ou seja, a execução de um programa implica na dedução de informações que estão implícitas nas relações contidas no programa.

3.2.1 Programação em Lógica

A Linguagem da Lógica Proposicional não é adequada ou suficiente para representar relações entre objetos, no caso, para a Programação em Lógica é utilizada a Lógica de Primeira Ordem (LPO). Por exemplo, se fôssemos usar uma Linguagem Proposicional para representar "João é pai de Maria e José é pai de João" usaríamos duas letras sentenciais diferentes para expressar ideias semelhantes, por exemplo, p para simbolizar "João é pai de Maria" e q para simbolizar "José é pai de João" e não estaríamos captando com esta representação o fato de que as duas frases falam sobre a mesma relação de parentesco entre João e Maria e entre José e João.

Outro exemplo do limite do poder de expressão da Linguagem Proposicional é sua incapacidade de representar instâncias de uma propriedade geral. Por exemplo, se quiséssemos representar em linguagem proposicional "*Qualquer objeto é igual a si mesmo*" e "*3 é igual a 3*", usaríamos letras sentenciais distintas para representar cada uma das frases, sem captar que a segunda frase é uma instância particular da primeira.

A LPO pode expressar fatos sobre todos os objetos no universo e considera o mundo com:

– Objetos (casas, cores, etc.);

– Relações (maior que, dentro, tem cor);

– Funções (pai de, melhor amigo de);

– Propriedades (grande, pequeno);

– Regras – conectivo de implicação (→) ou de equivalência (↔) que permitem expressar regras gerais.

Por exemplo, a representação em de LPO da sentença “Homens e mulheres são categorias disjuntas”. É expressa como: “homem(X)↔ ⌐ mulher(X)”.

A base da Programação em Lógica (Winston & Horn, 1981) é um modelo que contêm axiomas lógicos (proposições) que representam fatos e regras que inferem novas informações. Por exemplo: “Uma baleia é um mamífero” seria um fato ou uma informação. “Todo mamífero tem sangue quente” é uma regra que representa conhecimento. Então, por um mecanismo de inferência dedutiva podemos também inferir o fato de que “uma baleia tem sangue quente”.

Os mecanismos de inferência podem ser:

- Dedutivo: O conhecimento é inferido a partir do que já existe, mas que estava implícito, o conhecimento é sempre verdadeiro.
- Indutivo: O conhecimento obtido é intrinsecamente novo, induzido a partir de exemplos, o novo conhecimento pode ou não ser verdadeiro.
- Analógico: Dado outro problema e sua solução, relaciona os dois problemas para se chegar a uma nova solução.

Na Programação em Lógica dedutiva um programa consiste em um conjunto de fatos conhecidos e regras que formam a base para formular consultas, como por exemplo:

p: Todo homem é mortal

q: Sócrates é homem

E formular consultas que retornam valores resultantes de uma busca top-down, estes podem ser retornados verdadeiro ou falso, como por exemplo: “Sócrates é mortal?”. Nesta o programa retornaria verdadeiro, pois, dadas as premissas o programa infere através de regra de dedução chamada de silogismo que “*Sócrates é mortal*”. A consulta do tipo “*homem(X)*”, retornaria a lista de homens dispostos na base de conhecimento, que nesse caso seria “*Sócrates*”.

3.2.2 Clausulas de Horn

As Cláusulas de Horn tem um papel fundamental na Programação Lógica e para a Lógica construtiva, elas são disjunções de Cláusulas de Primeira Ordem conhecidas também como predicados.

Representam proposições na forma: se *p* então *q* na forma:

$p \rightarrow q$ ou $q \leftarrow p$

São classificadas em (Winston & Horn, 1981):

- Incondicionais: (Fatos) não contêm condições.

  Exemplo: "*3 é um número inteiro"*

  inteiro(3) ←, ou simplesmente, *inteiro(3)*

- Positivas: (Regras) possuem uma ou mais conclusões.

  Exemplo: "Todo numero natural é inteiro"

  inteiro(X) ← *natural(X)*

- Negativas: (Consultas), Não apresentam conclusões.

  Exemplo: "Será 3 um número inteiro?"

  ← *inteiro(3)*

- Definidas: (condicionais), A seguinte fórmula é um exemplo de cláusula de Horn (definida):

  $\neg p \vee \neg q \vee ... \vee \neg t \vee u$

  Usando a lógica clássica proposicional, tal fórmula pode ser reescrita ainda, de forma equivalente, no formato *cabeça* ← *corpo,* da seguinte forma:

  $u \leftarrow (p \wedge q \wedge ... \wedge t)$

  Exemplo: Para que um número seja natural, ele deve ser inteiro e positivo.

  natural(X) ← *inteiro(X)* ^ *positivo(X)*

3.2.3 Prolog

Prolog (Warren, 1983) é uma linguagem de programação em lógica, um tipo de linguagem declarativa e interpretada. Um programa em Prolog é constituído por uma base de conhecimento, que contem informações acerca de um determinado domínio. Uma base de conhecimento é constituída por um conjunto de sentenças declaradas no formato das cláusulas de Horn, estas são condicionais (regras) ou incondicionais (fatos).

A execução de um programa em Prolog é basicamente uma consulta, para tanto são feitas buscas e comparações da consulta com a base de conhecimento. Para a realização dessa busca o motor de inferência do Prolog é ativado (o motor de inferência do Prolog é um algoritmo de Backtracking[2], este segue o padrão de uma busca em profundidade de uma arvore) fazendo comparações para obtenção de um resultado para a consulta.

As clausulas em primeira ordem são descritas no modelo das cláusulas de Horn formadas por "*cabeça:-corpo"* (o '*:-*' é utilizado como operador condicional, pode ser lido como se), todas as clausulas são terminadas com o ".". Fatos são clausulas sem corpo e consultas são clausulas sem cabeça e regras são clausulas de Horn completas.

A seguir é apresentado um exemplo de uma base de conhecimento em Prolog no domínio da família:

```
% Fatos:

ser_pai(joao, pedro).

ser_pai(pedro, jose ).

% Regras, do tipo:

ser_avo(X, Y) :- ser_pai(X, Z),ser_pai(Z, Y).
```

[2] Backtracking é um algoritmo geral que utiliza a força bruta para localizar todas as possíveis soluções de um problema computacional, este faz uma busca incremental de candidatos para as soluções, e abandona cada parcial candidato, logo que determina que ele não possa ser completado com uma solução válida para o problema.

Uma consulta poderia ser:

```
?- ser_avo(joao, jose).
```

O valor retornado nessa consulta, utilizando a base de conhecimento especificada acima é:

```
yes.
```

3.3 INTRODUÇÃO A PLI

Enquanto a maioria das técnicas de Aprendizagem de Máquina faz a aquisição por meio de aplicações e métodos procedurais (métodos estatísticos, linguísticos, baseado em padrões), a PLI oferece uma forma declarativa de aquisição, facilitando a inserção de novas informações sem modificar métodos ou procedimentos. A mesma vê o programa e os dados de entrada como um comando lógico em relação ao mundo e o processo de elaborar as consequências explícitas, como um processo de inferência (Bratko, Kubat, & Michalski, 1998), utilizando uma coleção de exemplos de conhecimento e induzindo hipóteses (Figura 9).

Figura 9 - O processo de indução de hipóteses a partir de exemplos na PLI.

Em varias áreas a PLI esta sendo aplicada com sucesso e para uma grande variedade de problemas de classificação e previsão como o diagnostico de doenças em pacientes ou em plantas e na descoberta de propriedades mecânicas do aço com base nas suas características químicas.

Estes problemas podem ser resolvidos com o emprego do aprendizado Indutivo a partir de exemplos, referido como aprendizado indutivo de conceito, onde regras de classificação para um conceito específico devem ser induzidas de instâncias (ou não instâncias) daquele conceito.

3.4 CONCEITOS BASICOS

O problema em aprender em PLI pode ser descrito da seguinte forma (Duarte, 2001): Dado um conhecimento prévio **B** de um certo domínio, expresso como um conjunto de predicados em LPO no formato das clausulas de Horn invertidas; um conjunto de exemplos ou fatos positivos $\mathbf{E^+}$ (Instâncias verdadeiras da clausula a ser aprendida) e possivelmente de exemplos negativos $\mathbf{E^-}$ (exemplos falsos e restritivos da clausula a ser aprendida); um sistema que utiliza a PLI geraria um predicado lógico chamado de hipótese **H**

tal que (Muggleton & Raedt, Inductive Logic Programming: Theory and Methods., 1994):

- Todos os exemplos $\mathbf{E^+}$ podem ser derivados de **B^H**.
- Nenhum exemplo negativo $\mathbf{E^-}$ pode ser derivado logicamente de **B^H**.

Por exemplo, considere o aprendizado de um relacionamento entre pessoas de uma família, sabe-se que o seu avô é o pai de um de seus genitores, porem essa regra não esta explicita na base de conhecimento. Tendo o seguinte conhecimento prévio **B** descrito em LPO:

$$\mathbf{B} = \begin{cases} \textit{Father\_of (João, Lurdes).} \\ \textit{Mother\_of (Lurdes, Luciano).} \\ \textit{Mother\_of (Lurdes, Alice).} \end{cases}$$

Os exemplos $\mathbf{E^+}$ estabelece relacionamentos verdadeiros entre um avô e seus netos.

$$\mathbf{E^+} = \begin{cases} \textit{Grandfather\_of (João, Luciano).} \\ \textit{Grandfather\_of (João, Alice).} \end{cases}$$

Também podem ser adicionados relacionamentos que não são verdadeiros (exemplos negativos $\mathbf{E^-}$), que restringem a busca da clausula de Hipótese.

$$\mathbf{E^-} = \begin{cases} \leftarrow \textit{Grandfather\_of (Luciano, João).} \\ \leftarrow \textit{Grandfather\_of (Alice, João).} \end{cases}$$

De posse do conjunto de conhecimento prévio **B** e de fatos $\mathbf{E^+}$ e $\mathbf{E^-}$, o aprendiz deve induzir com o uso da PLI a hipótese **H**.

$$\mathbf{H} = \begin{cases} \textit{Grandfather\_of (X, Y)} \leftarrow \textit{Father\_of (X, Z), Mother\_of (Z, Y).} \end{cases}$$

A hipótese **H** satisfaz as seguintes condições (Duarte, 2001). Que são:

- ✓ **H** não é consequência de **B^E⁻**, isto é, **B^E⁻** ⊭ **H**, condição chamada Satisfação Posterior;
- ✓ Todos os exemplos **E⁺** podem ser derivados de **B^H**, isto é, **B^H** ⊨ **E⁺**, condição chamada Suficiência Posterior;

A PLI mostra-se como um eficiente método para a extração de regras, pois possibilitam meios através da indução à descoberta de padrões, as hipóteses encontradas nesse processo poderão ser novas regras acrescentadas em uma base de conhecimento, há também uma abordagem probabilística que pode ser utilizada em conjunto com a PLI para uma obtenção de uma maior confiabilidade na regra gerada.

Um método para a aquisição de novas regras para uma base de conhecimento é, com posse dos fatos contidos em uma base de conhecimento de certo domínio e expressas em LPO, o objetivo é induzir, usando a PLI, clausulas que se tornam regras para essa base de conhecimento.

3.5 MECANISMOS DE TENDÊNCIA

Qualquer coisa que influência a forma em como o aprendiz gera as inferências indutivas baseadas nas evidências é chamada de Tendência. Existem, fundamentalmente, duas formas de tendências (Duarte, 2001):

- Tendência Declarativa: define o espaço de hipótese a ser considerado pelo aprendiz, isto é, **o que procurar**;
- Tendência de Preferência: determina como procurar no espaço de hipótese, definido pela tendência declarativa, isto é, quais hipóteses levar em consideração, quais hipóteses devem ser modificadas, entre outros.

A tendência de preferência é utilizada quando existem várias hipóteses que descrevem todos os exemplos e é necessária alguma base para graduar essas hipóteses, ou seja, para decidir qual ou quais são as mais relevantes.

A tendência declarativa é dividida em dois tipos: sintática e semântica. A tendência sintática impõe restrições na forma das cláusulas permitidas na hipótese, como por exemplo, existir apenas cláusulas positivas na hipótese. A tendência semântica impõe restrições no significado, ou no comportamento das hipóteses.

Utiliza-se na maioria dos sistemas PLI, tais como, Progol (Muggleton, Inverse Entailment and Progol, 1995), FOIL (Muggleton & Raedt, Inductive Logic Programming: Theory and Methods., 1994), entre outros, os mecanismos de tendência para reduzir o espaço de busca e definir um conjunto de hipóteses/regras a serem descobertas.

3.6 ESTRUTURAS PARA INDUÇÃO DE HIPOTESES

Nesta seção são exemplificadas algumas estruturas básicas para a Inferência Indutiva de Regras, isto é, a forma como se procuram clausulas na linguagem de hipótese[3] que são consistentes com os exemplos contidos em um conjunto de treinamento. Como foi dito anteriormente, a indução é a inversão da dedução, porém na "dedução inversa" podem ser usados (assim como na dedução propriamente dita) vários tipos de estruturas para inferência de novas informações e seu uso ou não geralmente depende do formato do conhecimento prévio **B** e das evidências $\mathbf{E^+}$.

3.6.1 Utilização de modelo de regras

O objetivo de um modelo para a criação de regras é criar qualquer descrição, ou seja, um "template" para que haja uma restrição no espaço de hipóteses possíveis (Duarte, 2001), que geraria uma significativa redução de operações em algum algoritmo de busca. Por exemplo, na construção de um grafo de refinamento, vários ramos poderiam ser descartados por não se adequarem ao

[3] A linguagem de hipótese pode ser um literal qualquer contido na base de conhecimento

modelo de regra diminuindo assim o espaço de busca. Estes modelos poderiam ser dados diretamente pelo usuário ou por meio de derivações de regras aprendidas.

No sistema Progol esse mecanismo de modelo de regras é representado pela declaração de *Modes*[4]. Uma representação simples seria uma declaração do tipo:

a(X,Y):-p(X,Z),p(Z,Y).

O uso desse modelo de regra restringiria a linguagem de hipótese e reduziria o espaço de busca a clausulas com um corpo constituído por dois literais que possuem aridade dois.

3.6.2 Busca em Grafo de Refinamento

Um grafo de refinamento pode ser definido como grafo direto e acíclico, cuja raiz é o literal candidata a cabeça da clausula, os nós são cláusulas de programa e as arestas correspondem às operações básicas de refinamento: substituição de uma variável por um termo e adição de um literal ao corpo de uma cláusula.

O grafo de refinamento funciona com uma busca *top-down* que inicia com uma clausula genérica que é refinada (especializada) repetidas vezes até que a clausula não cubra mais os exemplos negativos (Lavrac & Dzeroski, 1994). Esta técnica funciona da seguinte maneira: para uma linguagem de hipóteses e um conhecimento prévio do domínio, o espaço de hipóteses de cláusulas de programa é um reticulado, isto é, um conjunto de cláusulas reduzidas, estruturado pela generalização ordenada do θ-subjugamento (Muggleton & Raedt, 1994).

A Figura 10 mostra um exemplo de arvore de refinamento gerada na indução da clausula de '*avo_de/2'*, com base em uma dada base de conhecimento.

[4] Um conjunto de declarações de mecanismos de tendência que especificam ao sistema Progol "o que procurar".

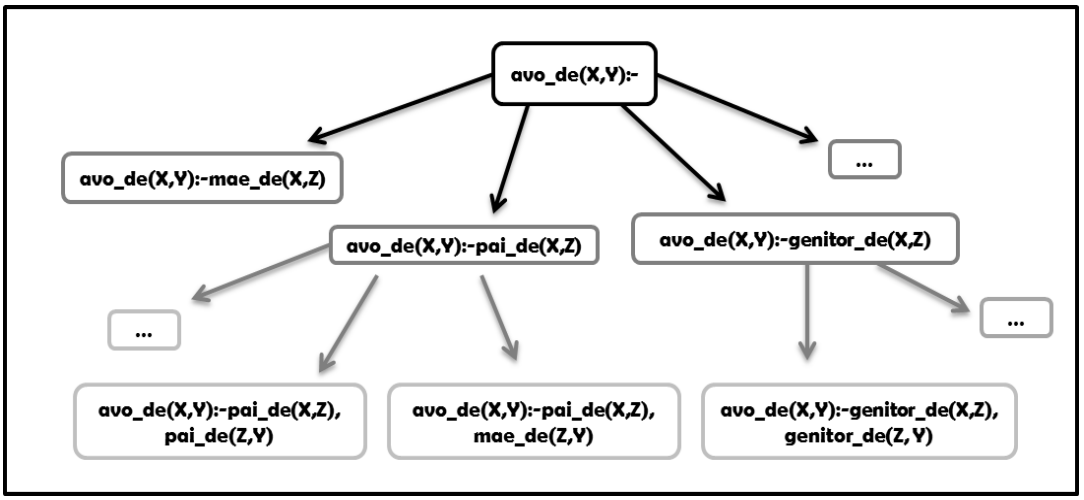

Figura 10- Exemplo de um grafo de refinamento, no domínio da família.

3.6.3 Resolução Inversa

Desde que uma regra dedutiva de resolução é completa para dedução, uma inversão da resolução deveria ser completa para indução (Lavrac & Dzeroski, 1994), ou seja, se uma regra de dedução serve para todos os casos, uma regra criada a partir da inversão desta poderia ser valida.

A resolução Inversa consiste no uso de um conjunto de operadores de implicação criados com o intuito de reverter a dedução. Entre eles estão:

- Absorção

$$\frac{q \leftarrow A \qquad p \leftarrow A, B}{q \leftarrow A \qquad p \leftarrow q, B}$$

- Identificação

$$\frac{p \leftarrow A, B \qquad p \leftarrow A, q}{q \leftarrow B}$$

- Intra-Construção

$$\frac{p \leftarrow A, B \qquad p \leftarrow A, C}{q \leftarrow B \qquad p \leftarrow A, q \qquad q \leftarrow C}$$

- Inter-Construção

$$\frac{p \leftarrow A, B \qquad q \leftarrow A, C}{p \leftarrow r, B \qquad r \leftarrow A \qquad q \leftarrow r\ , C}$$

3.7 PROGOL

O Progol é um sistema de PLI e um interpretador de Prolog que combina o modelo proposto “Inverse Entailment” (Muggleton, Inverse Entailment and Progol, 1995) com a busca da cláusula mais genérica para a mais específica através de um Grafo de Refinamento. O Progol é um tipo de sistema PLI declarativo, no qual o usuário declara explicitamente o formato da cabeça e do corpo da hipótese, estas declarações são chamadas de declarações de “*Modes”* que servem como um ‘guia’ para a construção de um grafo de refinamento.

3.7.1 Entrada do Progol

Um arquivo de entrada (Figura11) padrão para esse sistema consiste de cinco seções:

- Na seção **“Definições”** são setadas algumas condições que o Progol disponibiliza como, por exemplo, especificar ao aprendiz que apenas os exemplos positivos são levados em consideração no processo de inferência.
- A seção **“Modes”** é formada por declarações de modeh, onde são declaradas as clausulas candidatas à cabeça da hipótese e modeb, onde são declaradas as clausulas candidatas a corpo da hipótese. Todos os argumentos são “tipos” que devem ser valorados, por exemplo, João pertence a classe “Pessoa”, em LPO essa declaração poderia ser “Pessoa(João)”. Os argumentos das cláusulas podem ser do tipo +, - ou #, que indicam respectivamente, variáveis de entrada, de saída e constantes;
- Na seção **“Tipos”** são declarados os tipos dos argumentos e exemplos, como por exemplo: “*Pessoa(José)*”, ou seja existe um tipo “Pessoa” e José é um tipo de pessoa.

– Na seção **“Conhecimento prévio”** são declaradas as clausulas simples e regras que compõem o conhecimento existente.
– Na seção **“Exemplos positivos e negativos”** são declarados clausulas que condizem com a declaração de “*modeh*”.

É apresentado a seguir um exemplo de arquivo de entrada cujo objetivo é o aprendizado da regra avô_de/2 (avô de uma pessoa) (Figura 11).

```
% Aprendendo avo de genitor pai.

% Definições

:- set(posonly)?

% Declarações de Mode (H)

% Declarações da cabeça das hipoteses:

:- modeh(1,avo_de(+pessoa,+pessoa))?

% Declarações do corpo das hipoteses:

:- modeb(*,genitor_de(-pessoa,+pessoa))?
:- modeb(*,genitor_de(+pessoa,-pessoa))?

% Tipos

pessoa(joao).
pessoa(maria).
pessoa(jose).
pessoa(pedro).
pessoa(ana).
pessoa(joana).
pessoa(fred).

% Conhecimento Prévio (B)

genitor_de(X, Y) :- pai_de(X, Y).
genitor_de(X, Y) :- mae_de(X, Y).

pai_de(joao, maria).
pai_de(joao, jose).
pai_de(jose, pedro).
pai_de(pedro, joana).
mae_de(maria, ana).
mae_de(maria, fred).

% Exemplos positivos (E+)

avo_de(joao,pedro).
avo_de(jose,joana).
avo_de(joao, ana).
```

Figura 11 - Exemplo de arquivo de entrada para o Progol, contendo uma base de conhecimento no dominio da família.

3.7.2 Saída do Progol

A saída do programa (Figura 12) é gerada quando se usa o comando '*generalize(avo_de/2)?'* no exemplo '***avo.pl'*** descrito na seção anterior.

```
|- [Generalising avo_de(joao,pedro).]

avo_de(A,B) :- pai_de(A,C), pai_de(A,D), pai_de(B,E), genitor_de(A,C),
genitor_de(A,D), genitor_de(B,E), genitor_de(D,B), pai_de(D,B), mae_de(C,F),
mae_de(C,G), genitor_de(C,F), genitor_de(C,G).
[Most-specific clause reduced by 2 literals]
[Most specific clause is]

avo_de(A,B) :- pai_de(A,C), pai_de(A,D), pai_de(B,E), genitor_de(A,C),
genitor_de(A,D), genitor_de(B,E), genitor_de(D,B), pai_de(D,B), mae_de(C,F),
genitor_de(C,F).

[Learning avo_de/2 from positive examples]

[C:-0,8,7,0 avo_de(A,B).]
[C:1,8,5,0 avo_de(A,B) :- pai_de(A,C).]
[C:0,8,5,0 avo_de(A,B) :- pai_de(A,C), pai_de(A,D).]
[C:-6,4,4,0 avo_de(A,B) :- pai_de(A,C), pai_de(B,D).]
[C:0,8,5,0 avo_de(A,B) :- pai_de(A,C), genitor_de(A,C).]
[C:0,8,5,0 avo_de(A,B) :- pai_de(A,C), genitor_de(A,D).]
...
```

Figura 12 - Exemplo de construção da arvore de refinamento no sistema Progol.

A saída da uma ideia do algoritmo de busca que o Progol usa, testando, generalizando e especificando clausulas de hipóteses a partir dos exemplos positivos. No final da saída é mostrado a clausula de hipótese mais especifica e são retornados alguns detalhes da inferência realizada (Figura 12). Na saída do Progol (Figura 13) são retornados alguns detalhes da inferência indutiva e uma linha que se refere a regra induzida, nesse caso, a regra em LPO seria:

avo_de(A,B) :- pai_de(A,C), genitor_de(C,B).

```
[201 explored search nodes]
f=3,p=8,n=2,h=0
[Result of search is]

avo_de(A,B) :- pai_de(A,C), genitor_de(C,B).

[2 redundant clauses retracted]

avo_de(A,B) :- pai_de(A,C), genitor_de(C,B).

[Total number of clauses = 1]

yes
[:- generalise(avo_de/2)? - Time taken 0.04s]
```

Figura 13 - Resultado da inferência indutiva realizada pelo Progol.

4 UMA TÉCNICA PARA A EXTRAÇÃO DE AXIOMAS DE UMA ONTOLOGIA UTILIZANDO A PROGRAMAÇÃO EM LÓGICA INDUTIVA

A adição de axiomas em Ontologias é um conceito fundamental para a sua Aprendizagem, pois a representação de conhecimento proposta por elas não estaria completa sem a presença de regras em sua estrutura. O conhecimento nada mais é do que informações gerais ou regras sobre um domínio que interligam elementos de informação, com a finalidade de representar padrões. A partir desses padrões novas informações podem ser derivadas, possibilitando o reuso de um grande volume de informação. Os axiomas são definidos como verdades absolutas em um esquema lógico. Logo, devido a essa definição, estes restringem a inserção de informações contraditórias na ontologia. Além disso, a representação do conhecimento oferece uma informação semântica mais direta com o domínio modelado, aumentando a sua expressividade.

De acordo com sua definição formal, uma Ontologia, ou parte dela, poderia ser abstraída em uma base de conhecimento, os seus axiomas se tornariam regras e as suas instâncias de conceitos e instancias de relacionamentos se tornariam fatos descritos em lógica primeira ordem. Com base nessa prerrogativa podemos igualmente afirmar que o conhecimento representado por uma Ontologia poderia ser utilizada por um sistema de PLI para a representação de seu conjunto de treinamento. Isso demostra a facilidade que a PLI tem de se inserir no contexto da Aprendizagem de Ontologias. A PLI, portanto, se mostra uma alternativa bastante viável e eficaz para a manipulação do conhecimento contido em uma Ontologia, possuindo todas as ferramentas necessárias para o aprendizado e representação de seus axiomas. Este trabalho esta focado na descoberta de axiomas com base nos relacionamentos não taxonômicos e em suas instâncias.

Visando a área do aprendizado de ontologias, foi elaborada uma técnica que utiliza os conceitos da PLI e da Aprendizagem de Ontologias para a aquisição de Axiomas em Ontologias. Este capítulo descreve a TAILP (*Technique for*

Ontology Axioms extraction using Inductive Logic Programming), uma técnica semiautomática para inferência indutiva de axiomas de uma ontologia a partir de uma ontologia previamente povoada utilizando a PLI e das suas subatividades. O problema pode ser descrito como, dada uma ontologia povoada[5] de domínio estabelecido, o objetivo é induzir regras ou axiomas, com base nas instâncias de relações não taxonômicas da ontologia utilizando a PLI. Estas regras induzidas ou hipóteses, após algum critério de avaliação, tornam-se axiomas no contexto da aprendizagem de Ontologias. A técnica utiliza os conceitos da Aprendizagem Indutiva descritos no capítulo 3. Portanto para que haja aprendizado indutivo é necessária a presença de um conjunto de treinamento, que por sua vez, serão as instâncias da ontologia. Este conjunto de instâncias deve ser representativo o suficiente para que a regra seja aprendida, isto é, a regra precisa estar implícita na coleção de instâncias da ontologia para que mecanismos de indução possam, por meio de algoritmos, identificar padrões e refina-los até que estes cubram todos os exemplos disponíveis.

A Figura 14 apresenta de uma forma genérica as tarefas propostas pela técnica na extração de axiomas em uma ontologia.

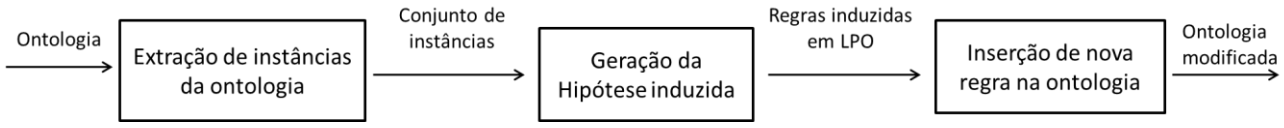

Figura 14- Tarefas genéricas da técnica TAILP

A tarefa intitulada "Extração de elementos da ontologia" recebe como entrada uma Ontologia povoada e nela é realizada a extração dos elementos da ontologia, isto é, extração do conjunto instâncias **$I' = \{rel_k(c_1, c_2, ..., c_n) \in I\}$** que participam das relações pertencentes ao conjunto **R** relevantes no processo de indução da nova regra e do conjunto das instâncias **$I'' = \{é\_um(c_1, c_2) \in I \mid c_1 \in I'\}$** necessárias para o aprendiz restringir o seu campo de busca. Em posse do conjunto de instâncias extraídas da ontologia, na tarefa "Geração

[5] Uma ontologia povoada é uma ontologia que já passou pelo processo de povoamento e, portanto, já possui instancias no seu conjunto I.

da hipótese induzida" será construída uma base de conhecimento **B = I' U I''** no padrão de alguma ferramenta PLI dada como entrada para o aprendiz. Este sistema por sua vez realiza a inferência indutiva por meio de algoritmos da PLI e retorna um conjunto de clausulas em LPO que serão as hipóteses geradas pela técnica. Por fim esse conjunto de hipóteses irá passar por uma avaliação para posteriormente vir a tornar-se um axioma da Ontologia, na tarefa final da técnica intitulada "Inserção de nova regra na ontologia".

4.1 EXTRAÇÃO DE INSTÂNCIAS DE UMA ONTOLOGIA

Uma vez que a técnica TAILP se utiliza de instâncias de relações não taxonômicas para realizar inferências é necessária a extração de determinados elementos dessa ontologia dada como entrada. Primeiramente a regra resultante do processo de indução estará em LPO no formato das cláusulas de Horn invertidas, ou seja, no formato "***a*←*b***" (se b então a). Desconhecemos qual a cabeça (conclusão) da regra a ser induzida, por isso a primeira atividade da tarefa "Extração de instâncias de uma Ontologia" é, como Figura 15 demostra, a extração de todas as relações não taxonômicas da ontologia para a posterior escolha da relação que irá vir a ser a conclusão da hipótese ou regra induzida.

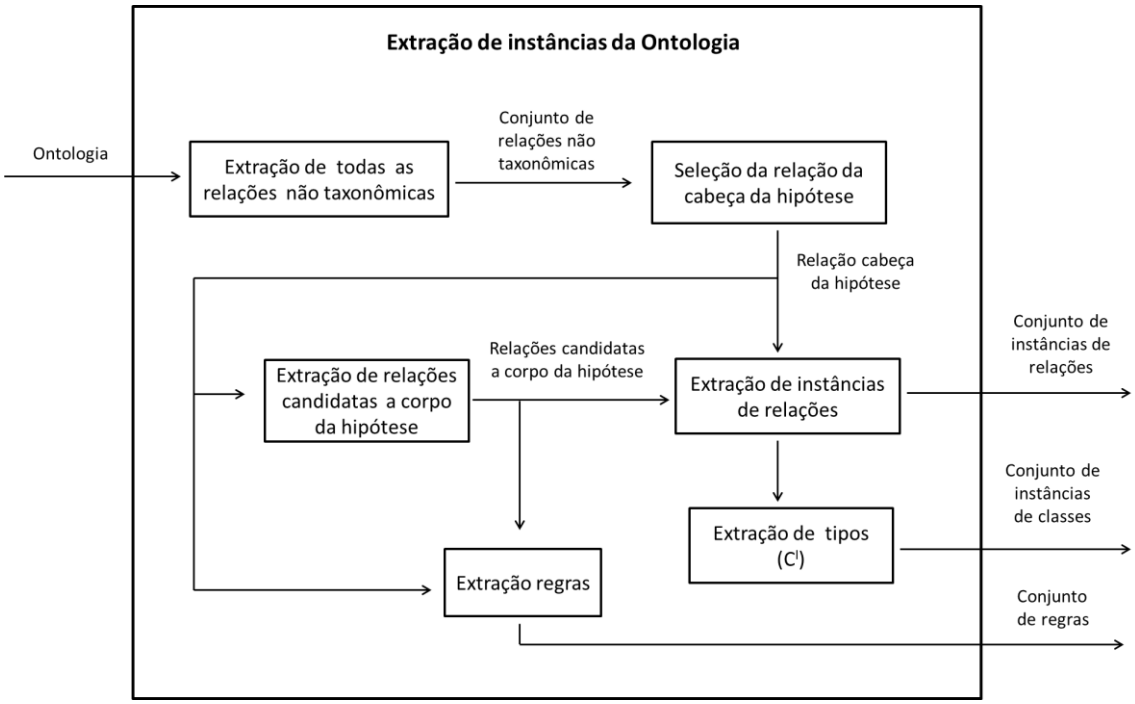

Figura 15 – Detalhamento da tarefa "Extração de elementos da ontologia"

Por se tratar de uma técnica semiautomática essa tarefa deve caber ao usuário ou especialista de domínio, porem um sistema tipo batch poderia percorrer todas as relações não taxonômicas da Ontologia e, para cada uma, aplicar os passos seguintes da técnica. Uma desvantagem para esse tipo de abordagem é a demora no tempo de execução em relação ao numero de relacionamentos contidos na Ontologia e do conjunto de instâncias que cada um possui, pois cada passo deverá ser refeito varias vezes no intuito de não inserir informações irrelevantes na base de conhecimento criada. Dependendo do tamanho dessa base de conhecimento o sistema PLI é forçado a realizar a busca da hipótese em um escopo bem maior de alternativas, realizando múltiplas vezes a mesma operação. Outra desvantagem seria a quantidade elevada de hipóteses retornadas, muitas delas não seriam relevantes ao processo de Aprendizagem da Ontologia, pois, dependendo do seu domínio especifico, nem todo relacionamento não taxonômico de uma ontologia possui uma regra lógica consistente e que dela são deduzidas novas informações.

Com posse da relação pertencente ao conjunto ***R*** que participará da regra são extraídas todas as instâncias da classe **I'** que participam dela, ou seja, o

conjunto de instâncias que participam da relação não taxonômica "$rel_x(c_i, c_j)$" escolhida e que pertencem ao conjunto **I** na atividade "Extração de instâncias

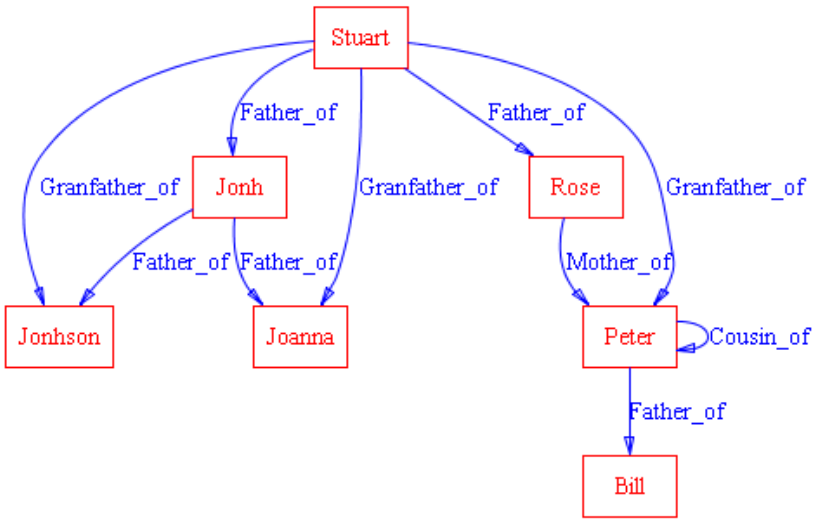

Figura 16 - Rede semântica representando as relações não taxonômicas de uma Ontologia no direito da família.

da relação". Esse conjunto de instâncias extraído se tornarão fatos na base de conhecimento e por fim será o conjunto dos exemplos positivos **E+** utilizados pelo aprendiz PLI, discutido na seção 3.3 do capítulo 3. A partir dessas informações o sistema PLI, com o uso de algoritmos e estruturas de dados, irá realizar a busca pelo conjunto hipóteses **H**. Pode-se afirmar contudo que se a relação escolhida não possuir nenhuma instância na Ontologia de origem o sistema PLI não conseguirá realizar qualquer inferência indutiva, pois, o aprendizado indutivo se baseia principalmente em seu conjunto de treinamento para realização de buscas e procedimentos.

Por exemplo, utilizando uma ontologia do domínio da família (Figura 16), o conjunto de todas as relações não taxonômicas do conjunto **R** seria:

Father_of(Man, Person);

Grandfather_of(Man, Person);

Cousin_of(Man, Person);

Sibling_of(Person, Person);

Mother_of(Woman, Person).

Se destas relações fosse escolhida a relação *Father_of /2,* as instâncias que participam dessa relação seriam em LPO:

Father_of(Stuart, John);

Father_of(Stuart, Rose);

Father_of(Peter, Bill);

Father_of(John, Johnson);

Father_of(John, Joanna).

Após essa etapa é feita a extração de relações candidatas a corpo da regra considerada como hipótese. Para a identificação de relações não taxonômicas que possam vir a participar da nova regra foi criada uma heurística: Identificando uma ou varias instâncias do relacionamento que é a cabeça da hipótese basta verificar as relações nas quais aquelas instâncias também participam. Na busca da relação candidata selecionamos um ou vários exemplos positivos aleatoriamente, nela um par de instancias de classe estão relacionadas, estas instancias podem estar associadas a outros relacionamentos, provavelmente estas são as relações não taxonômicas que estamos tentando descobrir como corpo da hipótese. Como mostrado no exemplo a seguir.

Na relação "*Grandfather_of /2"* foi identificada a seguinte instância em LPO:

"Grandfather_of(Stuart, Johnson)", cujas instâncias são {*Stuart, Johnson*};

Stuart e Johnson participam das seguintes relações:

Father_of(Stuart, John);

Father_of(Stuart, Rose);

Grandfather_of(Stuart, Joanna);

Grandfather_of(Stuart, Johnson);

Grandfather_of(Stuart, Peter);

Father_of(John, Johnson);

Cousin_of(Johnson, Peter);

Sibling_of(Johnson, Joanna).

Portanto, as relações do conjunto {*Father_of, Grandfather_of, Cousin_of, Sibling_of*} são candidatas a corpo da regra a ser induzida. Novamente são extraídas todas as instâncias das "relações candidatas" e adicionadas ao conjunto de instâncias **I'**. Para todas as instâncias já extraídas identifica-se também a que classe esta pertence, o conjunto **I''**. Toda essa informação extraída constituirá o conjunto **B**, do conhecimento prévio. O corpo da hipótese a ser induzida esta implícita nesse conjunto, portanto sem instâncias para popular o conjunto **B** a inferência indutiva será incapaz acontecer.

Para o exemplo utilizado todas as instâncias participam da mesma classe *Person*, no tipo "*é_um(c_n, Person)*" porem, elas também pertencem as subclasses *Man* e *Woman*, para facilitar a aprendizagem serão inseridas somente as subclasses de *Person*. Portanto para este exemplo especifico a identificação de classes seriam representadas da seguinte forma em LPO:

~~*Person(Stuart);*~~

Man(Stuart);

~~*Person(Johnson)*~~

Man(Jonson);

~~*Person(Rose);*~~

Woman(Rose);

…

Esta etapa representa a "*Identificação de Tipos"* e é importante para deixar a técnica mais abrangente para outros casos de outros domínios, como por exemplo, em uma ontologia de uma livraria pode haver relações do tipo:

Buy(Client, Book);

Write(Writer, Book);

Has_interest(Client, Writer);

Nessas relações participam instâncias de classes diferentes, como por exemplo, a relação *Buy/2* estabelece uma relação entre a classe *Client* e a classe *Book*, para o aprendiz que utiliza PLI a identificação e classificação dessas instâncias em tipos serve para estreitar a busca e evitar a criação de regras que estão em desacordo com a estrutura das relações que participam dela. O conjunto de Tipos também é acrescentado no conjunto **B.**

Um passo final da subatividade é o acréscimo de regras ao conjunto **B,** essas regras são axiomas contidos na Ontologia de origem. Uma abordagem para identificar quais axiomas serão relevantes para a técnica é identificar, na cabeça e no corpo da regra, relações não taxonômicas já extraídas anteriormente, estas serão transcritas em LPO e participarão do conjunto **B**.

Terminada essa etapa podemos dizer que estamos em posse dos exemplos positivos **E+** e do conhecimento prévio **B** extraídos da Ontologia de origem indispensáveis para o processo de indução utilizando a PLI. Um elemento não menos importante para a PLI é o conjunto dos exemplos negativos **E-** porem uma questão é criada: Como identificar exemplos negativos em uma Ontologia? Quais os benefícios que o seu acréscimo acarretaria ao processo de indução de axiomas em Ontologias?

4.2 GERAÇÃO DA HIPOTESE INDUZIDA

Como a técnica TAILP faz uso de um sistema PLI para realizar inferências, as instâncias extraídas na atividade anterior serão utilizadas para a criação de um conjunto de treinamento no padrão de alguma ferramenta PLI. Essa base de

conhecimento para o sistema PLI é dividida, como o capítulo 3 descreve, dos exemplos positivos **E+** que são as instâncias de do relacionamento que será a cabeça da hipótese induzida e do conhecimento prévio **B**, que consiste das instâncias extraídas dos relações candidatas a corpo da hipótese, as instâncias de classe e também os axiomas já contidos na Ontologia que possuem no conjunto de sua regra algum relacionamento não taxonômico extraído. Após a criação desse conjunto de treinamento na atividade "Formatação de arquivo de entrada para o sistema PLI" o aprendiz utiliza técnicas em PLI, descritos na seção 3.6 do capítulo 3 para inferir e generalizar hipóteses que são consistentes com a base de dados. Utilizando o exemplo da seção passada, a cabeça da regra sendo *Grandfather_of /2* e as clausulas do corpo determinadas pela heurística descrita na atividade "Extração de elementos da ontologia", para a entrada dessa tarefa seria construída a seguinte base de conhecimento contento as instâncias das relações que participarão da criação da regra:

Grandfather_of(Stuart, Joanna);

Grandfather_of(Stuart, Johnson);

Grandfather_of(Stuart, Peter);

Father_of(Stuart, John);

Father_of(Stuart, Rose);

Father_of(John, Johnson);

Father_of(John, Joanna);

Father_of(Bill, Peter);

Cousin_of(Johnson, Peter);

Sibling_of(Johnson, Joanna);

Essa base de conhecimento[6] será a base para a criação e formatação de um arquivo de entrada para um sistema PLI na tarefa “Geração da hipótese induzida” (Figura 17) e da inserção desse arquivo como entrada para um sistema PLI. Ao processar esse arquivo de entrada, por exemplo, o sistema Progol deve retornar uma cláusula em LPO da seguinte forma:

Grandfather_of(X, Y) ← Father_of(X, Z), Father_of(Z, Y);

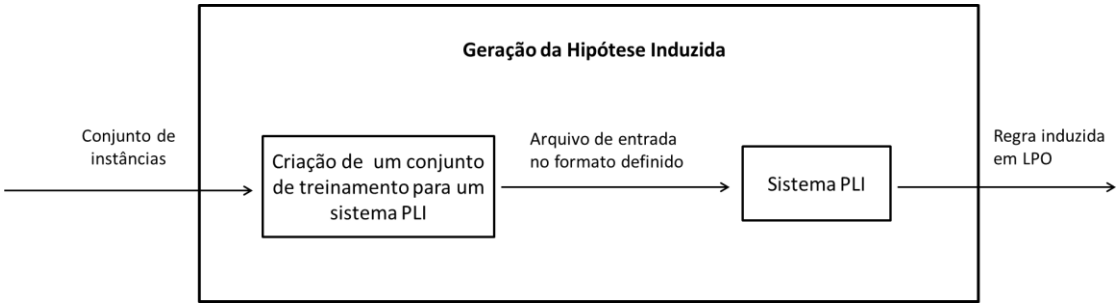

Figura 17 - Geração da hipótese induzida

4.3 INSERÇÃO DE NOVA REGRA NA ONTOLOGIA

Antes da hipótese gerada pela técnica ser considerada axioma de uma Ontologia ela necessita passar por algum tipo de avaliação, esta pode ser por meios estatísticos ou através do “feedback” do usuário (Figura 18) na atividade intitulada “Avaliação das hipóteses”. Um novo estudo sobre a validação dos axiomas poderia ser feito como, por exemplo, o uso de métodos estatísticos que utilizem um conjunto de teste e que confirmem a validade da hipótese através da analise dos elementos retornados a partir de uma inferência dedutiva dessa hipótese ou com o treinamento de uma rede neural para identificação do comportamento da Ontologia após a hipótese ser inserida como axioma. No caso da regra induzida ser verdadeira então esta será inserida no contexto da aprendizagem de Ontologias como um novo axioma da Ontologia de origem.

[6] Uma base de conhecimento é constituída de afirmações e regras que descrevem algum domínio especifico.

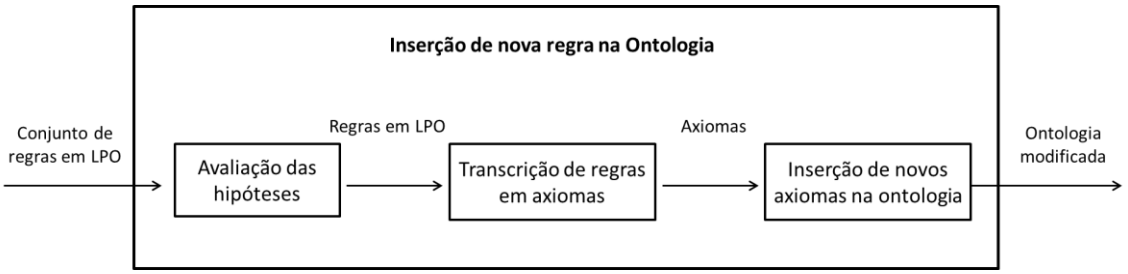

Figura 18 - Inserção de novo axioma na ontologia

A definição de Axioma de uma Ontologia definida no capítulo 2 diz que: **A = conditionx ⇒ conclusiony (c1, c2,..., cn).** Na LPO as regras são representadas no formato das clausulas de Horn invertidas: conclusão se condição, portanto uma das etapas da inserção do axioma induzido na Ontologia de destino deverá ser a inversão da regra gerada. O processo de adição de um axioma à ontologia dependerá da linguagem que a descreve, por exemplo, na ontologia OWL em algumas ferramentas como o Protégé os axiomas são descritos em SWRL[7], como segue a recomendação da W3C. Geralmente o sistema PLI utiliza a LPO para a representação do conhecimento processado e da hipótese, portanto para a inserção da hipótese gerada no arquivo que contem a ontologia em questão terá de haver primeiramente uma conversão da linguagem utilizada pelo sistema PLI para a linguagem de representação utilizada na ontologia.

[7] SWRL (sigla para semântic web rule) é uma linguagem de regra para web semântica que combina OWL e RuleML e segue o formato XML.

5 IMPLEMENTAÇÃO E INTEGRAÇÃO DA TAILP NA FERRAMENTA APPONTO

Para automatizar a execução das fases descritas no Capítulo 4, que explana como podemos fazer a extração de axiomas de uma ontologia utilizando a PLI, foi desenvolvida uma ferramenta que auxilia a aplicação da técnica TAILP.

A Ferramenta recebe como entrada uma ontologia povoada de extensão ".owl" e após algum processamento e interações com o usuário, previstas pela técnica proposta, realiza a inferência indutiva de hipóteses que serão avaliados e por fim inseridos na estrutura do arquivo OWL como axiomas em SWRL.

A TAILP foi desenvolvida na versão 7 da linguagem de programação Java, para a implementação da ferramenta foram utilizadas as seguintes APIs:

- Framework Apache Jena na versão 2.7.2, desenvolvida pela "The Apache Software Foundation". Amplamente empregada no desenvolvimento de aplicações para Web Semântica. O Jena fornece uma coleção de ferramentas e uma biblioteca Java para construção de Apps, ferramentas e servidores baseados na Web semântica.
- Biblioteca Java do Protégé, na versão 3.4.2, discutido anteriormente, na seção 2.6 do capítulo 2. Esta biblioteca faz a incorporação de várias funcionalidades para manipulação, criação e visualização de Ontologias em diversos formatos, incluindo o RDF(S), a OWL e o XML Schema.

A linguagem utilizada para definir e instanciar as ontologias na TAILP é a OWL que é a linguagem recomendada pela W3C desde fevereiro de 2004. É atualmente a tecnologia mais aceita e difundida no desenvolvimento de aplicações em Web semântica. Os Axiomas por sua vez são definidos em SWRL, linguagem de representação de regras criada como uma extensão para a instanciação de regras lógicas na OWL. A ferramenta utiliza também o sistema PLI Progol para realização de inferências Indutivas na busca de hipóteses.

Uma interface gráfica foi elaborada com o intuito de avaliar os resultados das tarefas propostas pela a técnica, assim como tornar a sua interação com o usuário de uso fácil e intuitivo. A interface foi integrada com a ferramenta

APPONTO (Monteiro, 2012), uma ferramenta em desenvolvimento pelo grupo de pesquisa GESEC que tem como objetivo final a Aprendizagem e Povoamento de Ontologias através de documentos textuais.

Este capítulo inicia-se com a breve apresentação e descrição da ferramenta APPONTO e de suas funcionalidades na seção 5.1. A seção 5.2 apresenta a integração da interface gráfica desenvolvida, a TAILP com a ferramenta APPONTO, e finalmente, um breve detalhamento da sua plataforma e um passo a passo do funcionamento da ferramenta é feita, exemplificando e detalhando todos os passos do processo de extração de axiomas de uma ontologia. Por fim a seção 5.3 é responsável pelo detalhamento das principais classes da ferramenta. Posteriormente, no capítulo 6, é apresentado um estudo de caso para a aprendizagem de axiomas em uma ontologia no domínio da família utilizando a TAILP.

5.1 APPONTO

A ferramenta APPONTO está sendo desenvolvida pelo grupo de pesquisa GESEC e visa integrar todos os resultados do projeto de pesquisa HERMES visando o seu objetivo final que é a aprendizagem e povoamento automáticos de uma Ontologia a partir de documentos textuais. Futuramente a APPONTO incluirá novas funcionalidades para a aprendizagem e povoamento incremental em que as ferramentas consigam se comunicar e trabalhar em conjunto em um processo que compreende todas as camadas da aprendizagem e povoamento de Ontologias.

A Figura 20 mostra a tela inicial da interface gráfica da APPONTO. Nela, deverá ser escolhida a tarefa na construção de Ontologias que o usuário deseja realizar, a Aprendizagem ou o Povoamento.

Figura 20 - Tela inicial da ferramenta APPONTO.

A Aprendizagem (Figura 19) compreende as técnicas destinadas a Aprendizagem de Ontologias, a entrada para a maioria das técnicas é um Corpus de um domínio especifico, por se tratarem em sua grande maioria de técnicas de Processamento de Linguagem Natural e a saída será um arquivo OWL contendo a Ontologia aprendida pelo processo selecionado.

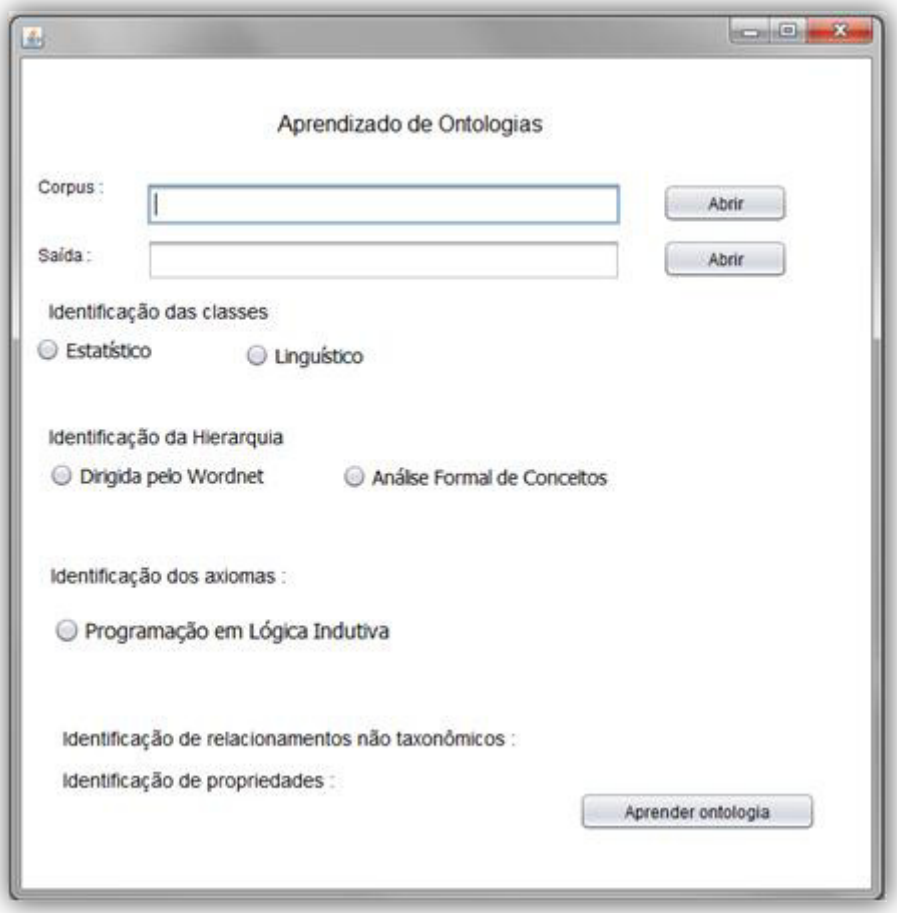

Figura 19 - Seção da Aprendizagem de Ontologias da APPONTO.

As opções para Aprendizagem disponíveis são: a identificação de Classes, a identificação de Hierarquia (relações taxonômicas) e por fim a identificação de Axiomas que utiliza a técnica TAILP desenvolvida e integrada na Ferramenta APPONTO. Outras funcionalidades serão posteriormente acrescentadas, como a identificação de relacionamentos não taxonômicos e a identificação de Propriedades. Como pode ser visto a finalidade da aprendizagem de ontologias na APPONTO é agrupar todas as camadas de aprendizagem do modelo proposto por Buitelaar, Cimiano e Magnini, descrito no capítulo 2, seção 2.3, para a construção automatizada de ontologias.

A primeira seção que podemos verificar na interface gráfica é a "Identificação das classes" que pode ser feita com o uso do método estatístico ou do método linguístico. Para o método estatístico utiliza-se a ferramenta NLPDUMPER (Macedo, 2010) que faz a extração das classes candidatas contidas em um corpus e retorna um arquivo de texto com todos os elementos encontrados.

Na "Identificação da Hierarquia" que visa fazer a extração de relações taxonômicas de uma ontologia duas abordagens estão disponíveis para utilização: a Wordnet e Análise Formal de Conceitos. Futuramente será adicionada uma ferramenta que realiza uma abordagem dirigida pela base de dados do Wordnet[8] na construção de uma OWL contendo Conceitos de Classe e relacionamentos não taxonômicos. A abordagem que utiliza a Análise Formal de Conceitos foi extraída a partir da ferramenta FCAIMPL (Galvão, 2010) que realiza a aprendizagem de Hierarquias de Conceitos e tem como resultado uma ontologia em formato OWL contendo as camadas C e H aprendidas.

A "Identificação de Axiomas" faz parte da Aprendizagem do conjunto A, dos Axiomas, de acordo com Cimiano esta seria a ultima camada a ser aprendida no processo de desenvolvimento de ontologias em camadas. Na seleção do método "Programação em Lógica Indutiva" a interface gráfica da TAILP é apresentada ao usuário, esta ferramenta é produto final do estudo da técnica que utiliza PLI para inferência de axiomas, a sua interface gráfica e a sua utilização serão discutidos nas próximas seções.

[8] O Wordnet é uma base de dados lexical organizada por significado. Ela foi desenvolvida na Universidade de Princeton por George A. Miller e categoriza conceitos e expressões em língua inglesa em quatro grupos: substantivo, verbo, adjetivo e advérbio.

Para o Povoamento de Ontologias (Figura 21) na tarefa de “Identificação de Instancias” estão disponíveis o método estatístico ou método linguístico. O método estatístico é realizado novamente pela ferramenta NLPDUMPER e preserva a funcionalidade de criar arquivos de instâncias candidatas semelhantes a sua interface original. O método Linguístico esta relacionado a ferramenta DIPPAOTool que implementa o processo DIPPAO (Faria C. G., 2013), realizando o povoamento automático de ontologias utilizando uma ontologia para a geração automática de regras para extrair instâncias a partir de textos e classifica-as como instâncias de classes da ontologia.

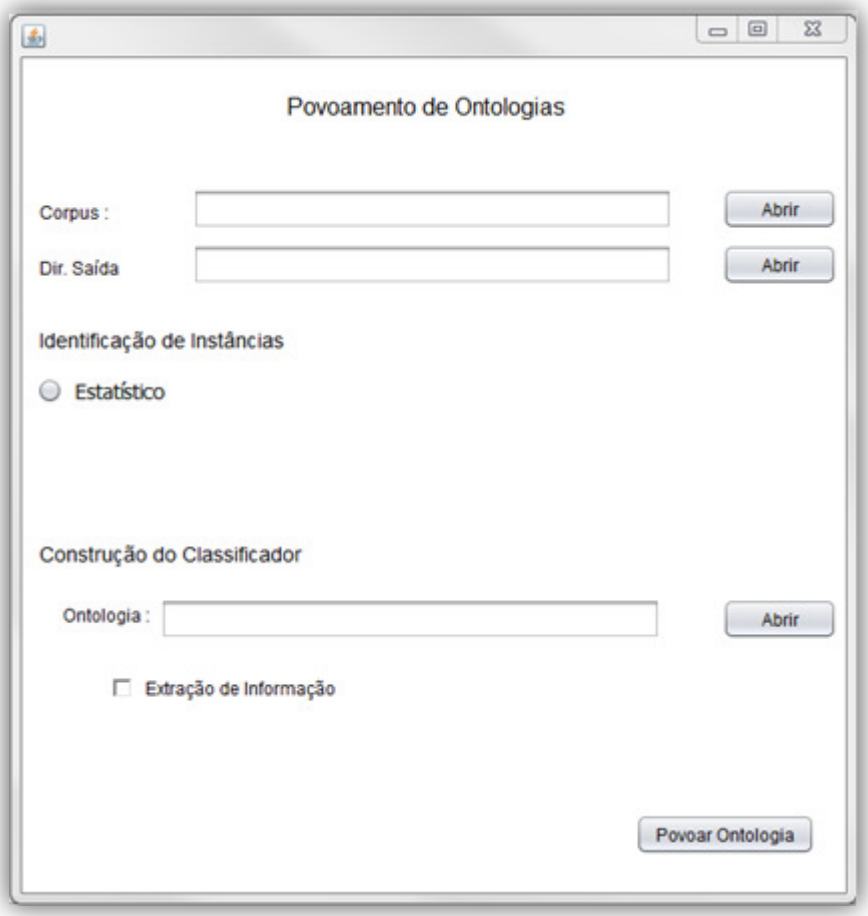

Figura 21 - Seção de Povoamento de Ontologias da APPONTO.

Na parte inferior da interface contem uma funcionalidade denominada “Construção do classificador” que permite o usuário após a inserção de uma ontologia em formato OWL fazer a instanciação e gerar uma ontologia povoada utilizando o corpus selecionado.

5.2 INTEGRAÇÃO DA TAILP

Foi desenvolvida uma interface gráfica (Figura 22) utilizando plug-ins contidos na IDE do NetBeans, na versão 7.2, que sintetiza os passos da técnica TAILP para a extração de axiomas de uma ontologia e a interação com o usuário necessária. A interface está integrada na ferramenta APPONTO e está localizada na área do Aprendizado de Ontologias, na seção “Identificação de Axiomas”. A interface é exibida após a seleção do método da Programação em Lógica Indutiva na seção da Identificação dos axiomas.

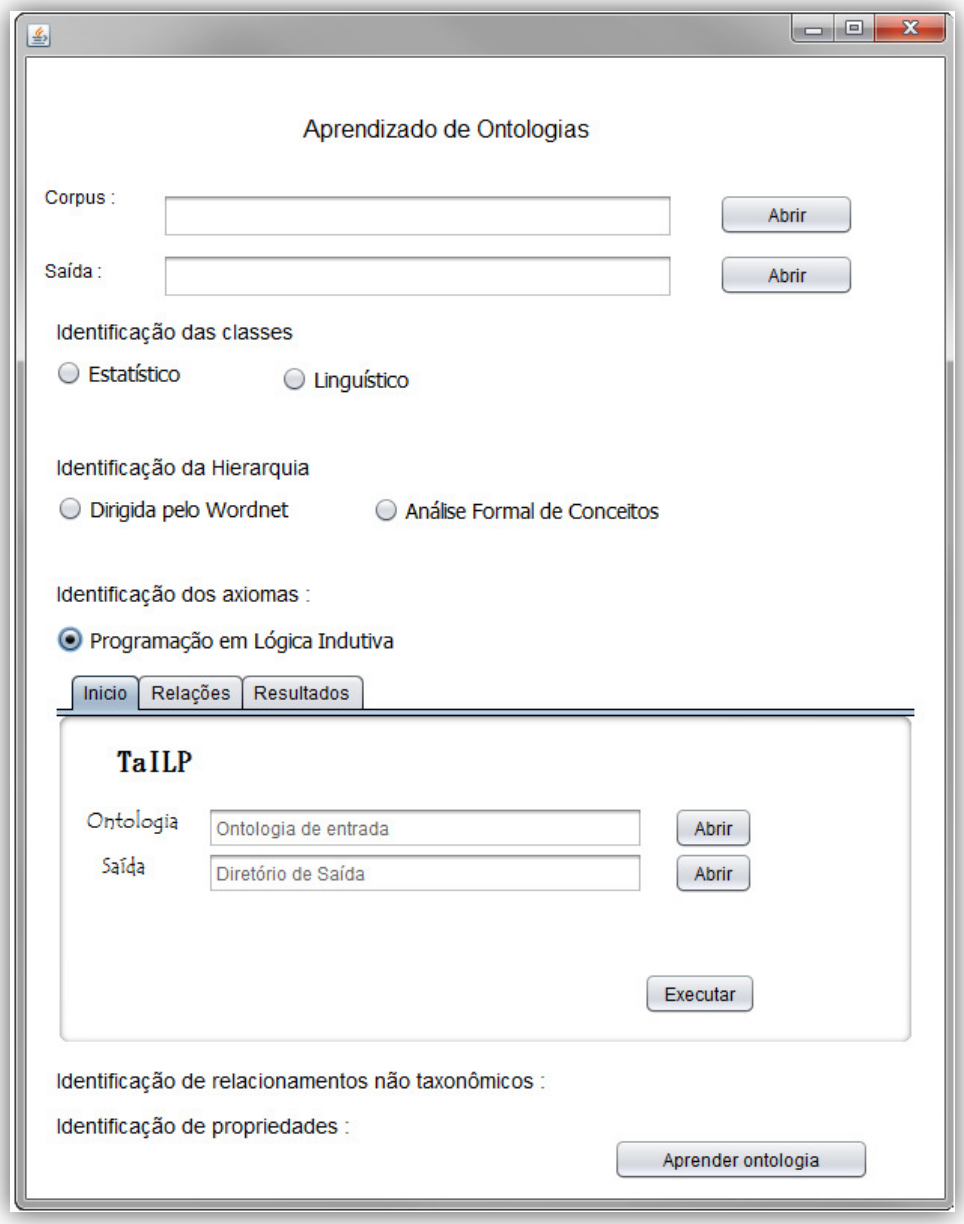

Figura 22 - Tela da Aprendizagem da APPONTO, na tarefa da Identificação de Axiomas utilizando PLI.

A ferramenta é composta por três fases representadas por guias em um painel principal na interface gráfica, cada uma demostra uma das tarefas da técnica TAILP, sendo a primeira a “Extração de elementos da Ontologia”, seguida por a “Geração da Hipótese Induzida” e por fim a “Inserção de nova regra na Ontologia”. A tela inicial da ferramenta (Figura 23) representada pela guia “*Inicio*” contem dois campos de entrada para a seleção do arquivo OWL em que se deseja aplicar a técnica e do diretório de saída para a ontologia modificada criada ao final dos passos da técnica. Acionando o botão “*Executar*” a ferramenta carrega a ontologia dada como entrada e faz a extração de todos os seus relacionamentos não taxonômicos utilizando a API do framework Jena.

Figura 23 - Tela inicial da TAILP, selecionando ontologia de entrada e diretório de saída.

Após o carregamento da ontologia e da extração de suas relações não taxonômicas uma segunda tela (Figura 24) será apresentada com uma listagem de todos os relacionamentos extraídos da OWL dada como entrada.

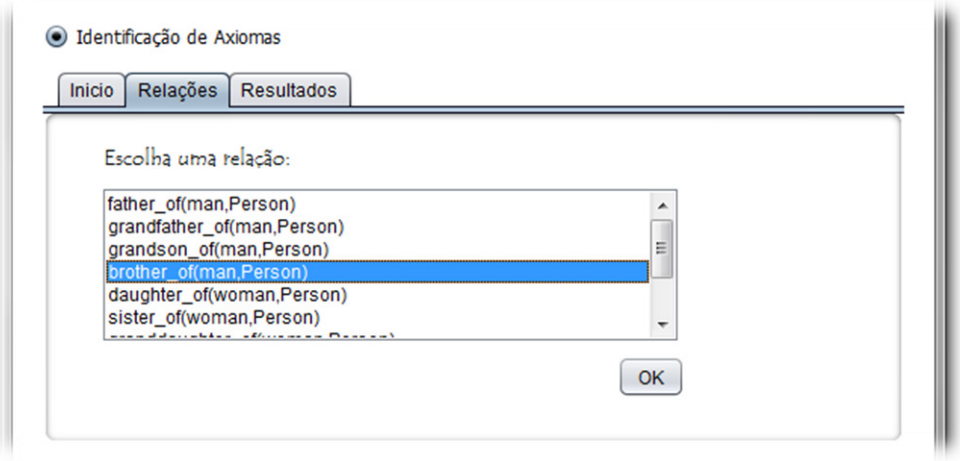

Figura 24 - Seleção de uma relação candidata a cabeça da hipótese.

O objetivo desta segunda atividade da ferramenta é a escolha da relação que virá a ser a cabeça da hipótese induzida. Após a seleção de uma das relações listadas, ao clicar no botão “OK” a ferramenta extrai todas as instâncias da relação selecionada utilizando consultas em SPARQL na Ontologia de entrada. No próximo passo a ferramenta usa a heurística proposta pela TAILP para extrair relações candidatas ao corpo de hipótese e faz a extração de todas as suas instâncias. A ferramenta ainda extrai todas as instâncias das classes que participam dos relacionamentos extraídos. Após a extração de todos os elementos requeridos é criado um arquivo de entrada para o sistema de PLI Progol, contendo a base de conhecimento com o conjunto **B** e **E+** extraídos em LPO e algumas configurações necessárias para o processo de indução. A ferramenta se conecta com o executável do sistema PLI utilizando chamadas ao sistema operacional via batch dando como argumentos o arquivo de entrada, que é a base de conhecimento construída nas etapas anteriores e o diretório de saída e o sistema PLI retorna as hipóteses induzidas pelo processo de inferência realizado.

O próximo passo da ferramenta é a avaliação da hipótese pelo usuário, para garantir que o axioma adicionado seja verdadeiro e consistente. As hipóteses inferidas serão listadas em uma nova tela de Resultados (Figura 25), caso a hipótese inferida não esteja correta o usuário poderá seleciona-la e pressionando o botão “*Remover Axioma*” a ferramenta retira essa hipótese da lista de possíveis axiomas. Após a avaliação e exclusão de hipóteses pelo usuário ao pressionar a opção “*Salvar Axioma(s)*” o arquivo “*TAILP.owl*” será criado no diretório de destino escolhido previamente contendo os axiomas inferidos incluídos em seu corpo em formato SWRL que está no padrão XML. Foi utilizada a API do Protégé para incluir os axiomas inferidos à nova ontologia OWL no padrão utilizado pelo Protégé, a linguagem SWRL.

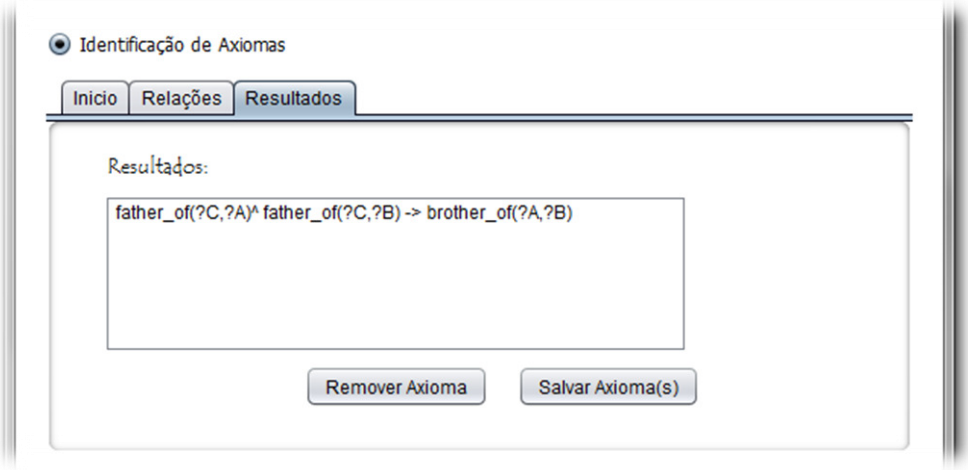

Figura 25 - Avaliação e seleção de axiomas inferidos a serem adicionados na OWL.

O usuário também tem a opção de retornar as tarefas anteriores e refaze-lo em um processo incremental quantas vezes quiser como, por exemplo, o usuário poderá voltar a função de escolha de relação da cabeça da hipótese selecionando a guia “Relações”, caso outra hipótese tiver sido inferida ela aparecerá na tela de resultados junto com os resultados da nova inferência. Após a avaliação das novas hipóteses inferidas o usuário poderá inserir toda a lista de axiomas na Ontologia pressionando o botão “Salvar Axioma(s)”. Porem a carga de ontologia só poderá ser feita uma vez, selecionada outra ontologia as outras tarefas concluídas anteriormente são descartadas.

5.3 PRINCIPAIS CLASSES DA FERRAMENTA TAILP

A Figura 26 apresenta as estruturas da ferramenta APPONTO, a seguir serão descritas as principais classes que constituem a ferramenta TAILP com o objetivo de um melhor entendimento da ferramenta desenvolvida.

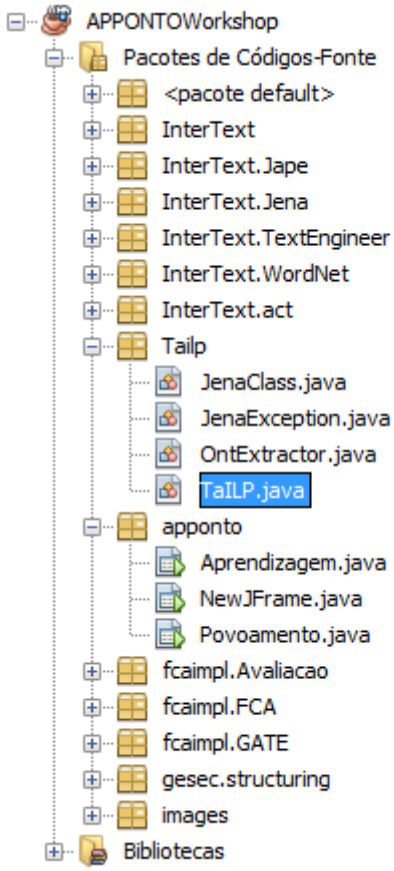

Figura 26 - Estrutura de arquivos da APPONTO, em destaque as Classes da TAILP.

• OntExtractor – A classe OntExtractor trabalha diretamente com a API do framework Jena, fazendo uso de seus métodos de extração de Conceitos e suas instâncias de Conceitos, das relações não-taxonômicas e suas instâncias. Também realiza consultas em SPARQL no documento OWL dado como entrada ao sistema. Nessa classe estão encapsuladas todas as informações necessárias para a técnica contidas no arquivo OWL de entrada. A Classe OntExtractor contem um conjunto de objetos da classe JenaClass que serve para encapsular todas as informações das Classes extraídas da Ontologia de entrada.

• JenaClass – A classe JenaClass é uma estrutura de dados que agrupa as informações de uma Classe consideradas relevantes para a técnica TAILP, como as instancias da Classe, o nome da Classe, suas propriedades que são os Object Properties e relacionamentos não taxonômicos que na OWL são referenciados como Datatype Properties.

• TaILP – É a classe principal da ferramenta TAILP. Ela é responsável por instanciar e coordenar a Classe OntExtractor chamando os seus métodos e preenchendo os seus campos com informações contidas no arquivo OWL. A classe TaILP é responsável também pela criação da base de conhecimento para o sistema PLI e usa chamadas ao sistema operacional para a sua execução e extrai os seus resultados dando como saída os axiomas inferidos. A classe TaILP trabalha principalmente com a API do Protégé para a criação e inserção de Axiomas em SWRL na ontologia OWL.

6 AVALIAÇÃO

Este capítulo tem como objetivo apresentar uma avaliação da técnica desenvolvida, a TAILP, apresentada no capítulo 4 para a extração de axiomas a partir de uma ontologia povoada, de modo a demonstrar a efetividade de suas fases e a viabilidade da utilização da PLI para a aprendizagem de axiomas de ontologias.

Um estudo de caso na área do direito de família foi conduzido, para tanto será utilizada a ferramenta desenvolvida, proposta na seção 5.2 do capítulo 5.

O estudo de caso consiste de, a partir da ontologia "*FamilyLaw.owl"*, desenvolvida pelo grupo GESEC com o uso da ferramenta Protégé e previamente povoada a partir da ferramenta DIPPAOTool (Faria, 2013), utilizar a ferramenta TAILP com o objetivo de inferir os axiomas de suas relações não taxonômicas. A seção 6.1 descreve a ontologia FamilyLaw, assim como as suas classes, subclasses, propriedades, relacionamentos e suas instâncias, visualizadas a partir da interface gráfica disponível na ferramenta Protégé. Por fim, na seção 6.2, apresenta o estudo de caso com a aplicação da ferramenta TAILP dando como entrada a ontologia FamilyLaw e são relatadas as conclusões para os resultados obtidos. Uma analise das atividades é feita com base nos resultados.

6.1 ONTOLOGIA FAMILYLAW

A ontologia adotada para os experimentos foi desenvolvida pelo grupo GESEC, com uso da ferramenta Protégé, ela descreve conhecimento acerca do ramo do Direito de Família e consiste de uma classe raiz "Family_Law", da qual derivam duas grandes classes "Family_Entity" e "Person". A primeira descreve as principais entidades familiares legalmente consideradas e a segunda discrimina os elementos pessoais constituintes de uma família. A visualização de classes da ontologia é visualisada na Figura 28 utilizando a interface gráfica do Protégé.

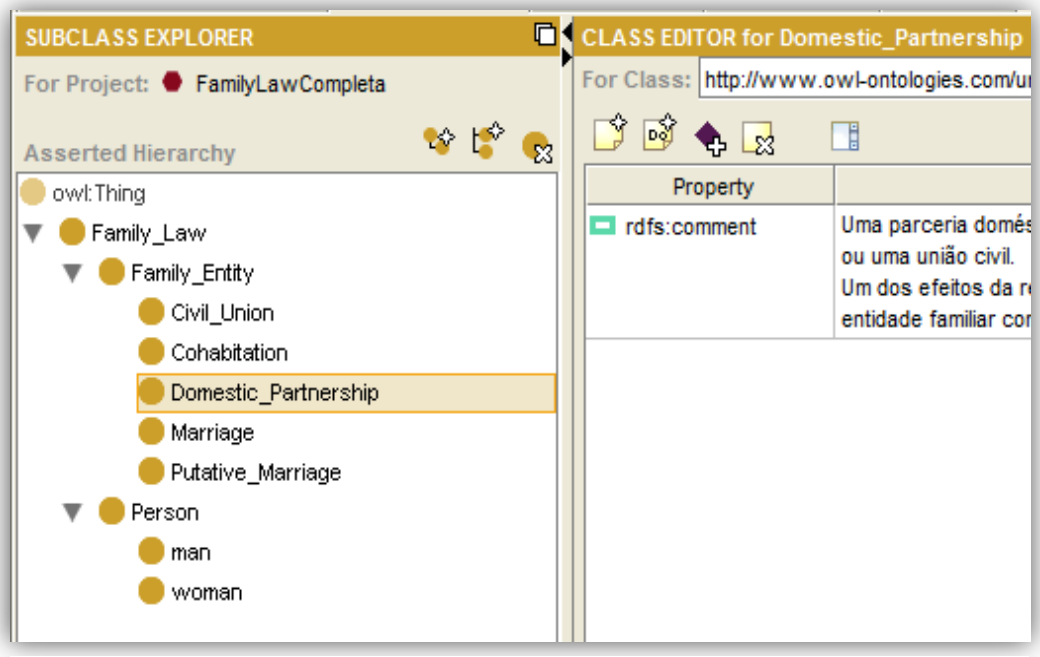

Figura 28 - Classes da ontologia FamilyLaw.

Para fins didáticos trabalharemos somente com a classe Person, suas subclasses, relações, propriedades e instâncias. A estrutura de Classes e subclasses é ilustrada pela Figura 27.

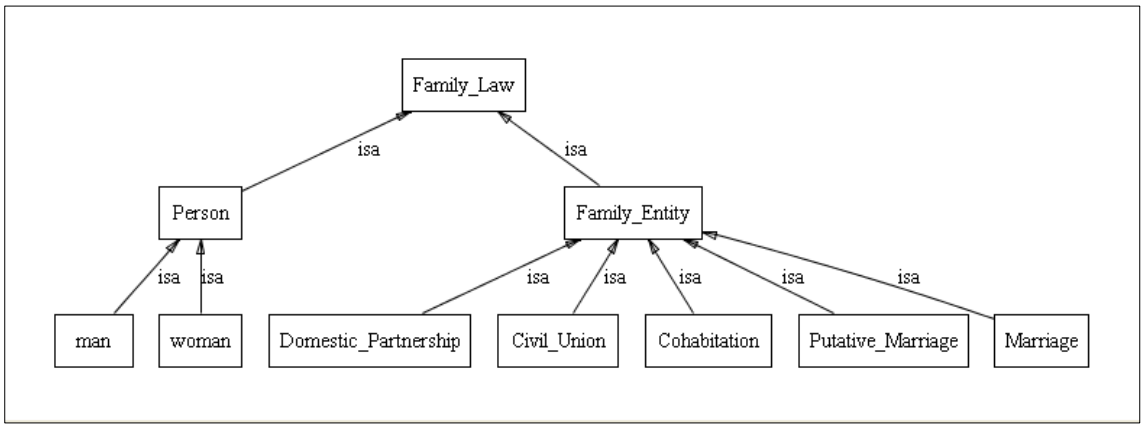

Figura 27 - Estrutura taxonômica da ontologia FamilyLaw.

Uma pessoa, descrita pela classe “Person”, é caracterizada pelas propriedades “*has_name*” e “*born_date*” (Figura 29).

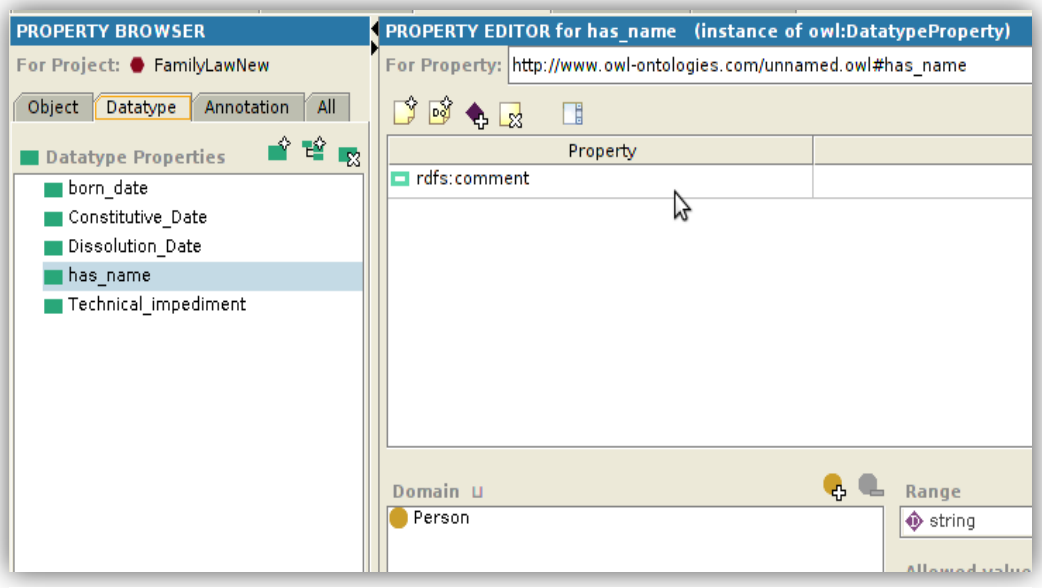

Figura 29 - Propriedades da ontologia FamilyLaw.

Os relacionamentos não taxonômicos desta classe são: “father_of”, “mother_of”, “brother_of”, “sister_of”, “daugter_of”, “son_of”, “husband_of”, “wife_of”, “grandparent_of”, “grandmother_of”, “grandfather_of”, “grandson_of”, “granddaughter_of”, “lives_in_family_with_female_family_partner” e “lives_in_family_with_male_family_partner” (Figura 30).

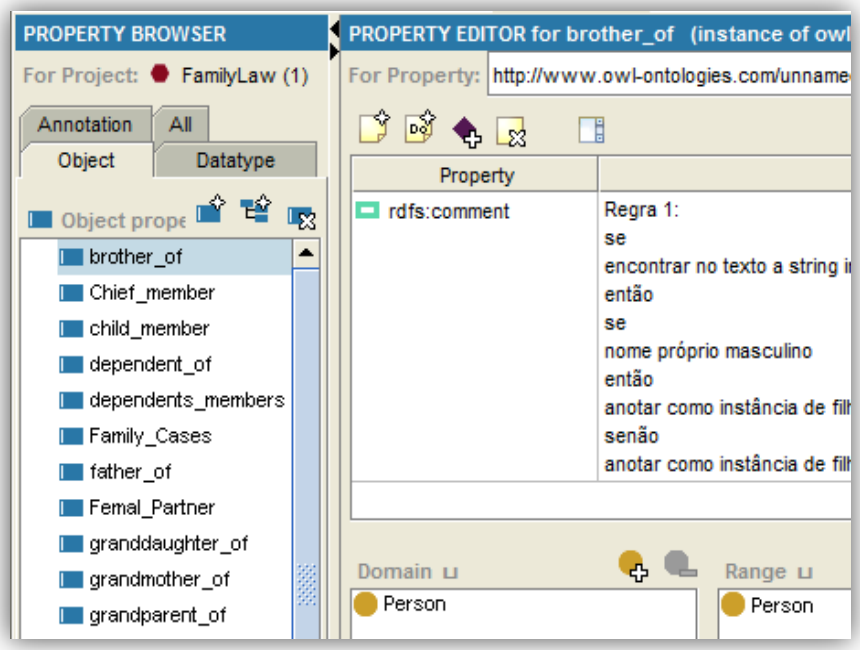

Figura 30 - Relacionamentos não taxonômicos da ontologia FamilyLaw

Utilizando a ferramenta de Povoamento de ontologias do grupo de pesquisa GESEC intitulada DIPPAOTool (Faria, 2013) que é a implementação do processo DIPPAO a ontologia é povoada automaticamente com instâncias extraídas dado um corpus do direito da família (Figura 31) e alguns de seus resultados foram modificados utilizando o Protégé.

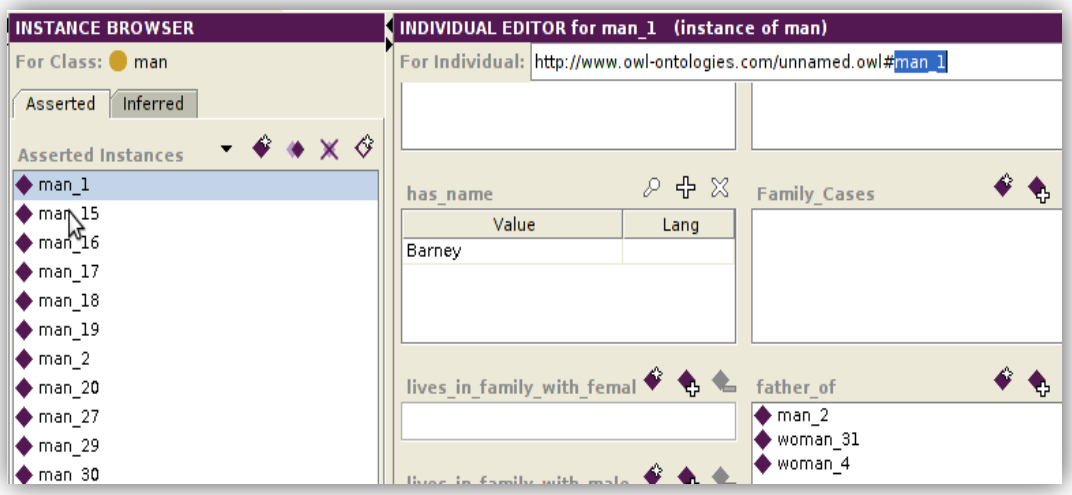

Figura 31 - Instâncias da ontologia FamilyLaw.

A ontologia *FamilyLaw* não possui inicialmente axiomas em seu arquivo OWL, a

Tabela 1 apresenta o numero de instâncias de relacionamentos taxonômicos considerados no decorrer do estudo de caso.

Relação taxonômica	Nº de instâncias
Grandfather_of	13
Brother_of	10
Married_with	5
Sister_of	8
Grandparent_of	7

Tabela 1 - Contabilidade das instâncias de relacionamentos não taxonômicos da ontologia FamilyLaw relevantes ao estudo de caso.

6.2 EXTRAINDO AXIOMAS DA ONTOLOGIA FAMILYLAW

Esta seção demostra um passo a passo da extração dos axiomas da ontologia contendo todas as interações feitas pelo usuário com a ferramenta TAILP.

A primeira interação consiste de selecionar o arquivo OWL que contem a ontologia FamilyLaw.owl (Figura 32), logo após selecionar o diretório de saída para a ontologia modificada.

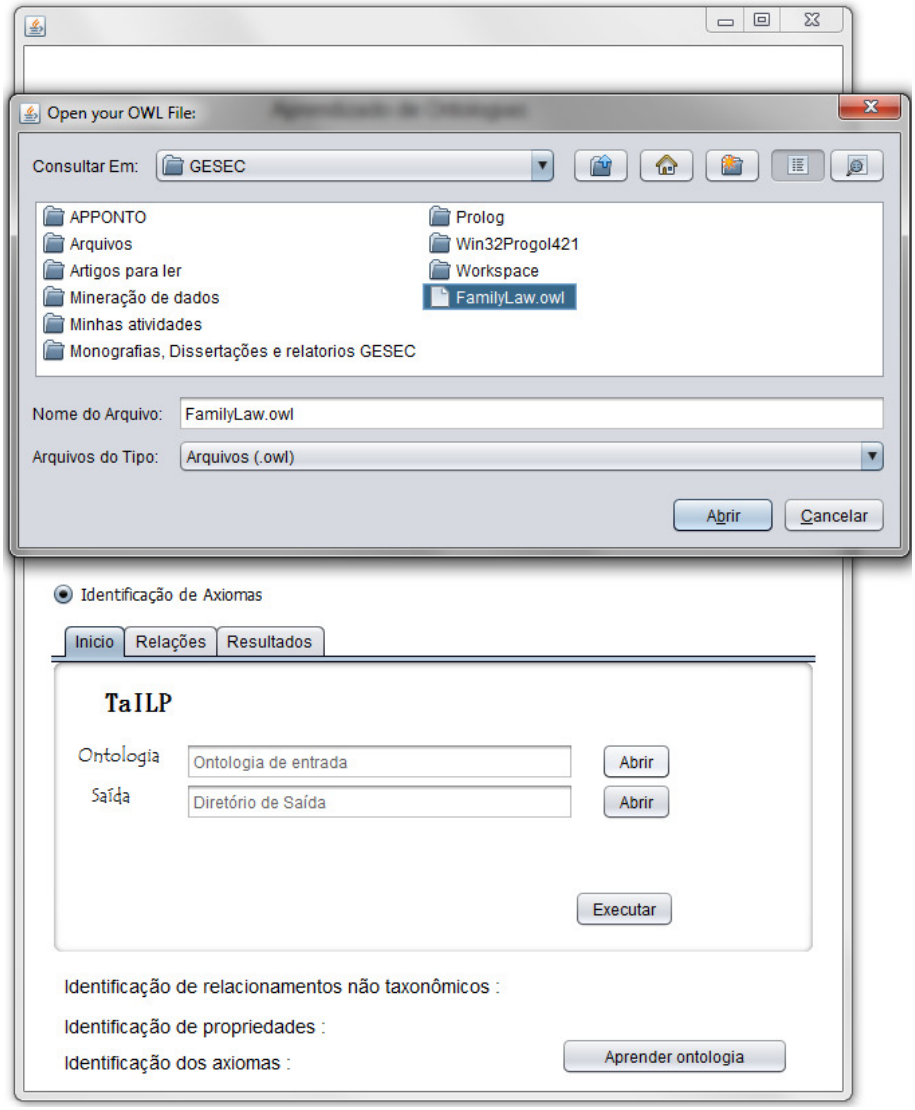

Figura 32 - Seleção de arquivo .owl de entrada e diretório de saída para a ferramenta TAILP.

Após essa etapa, são extraídos todos os relacionamentos não taxonômicos contidos no arquivo de entrada FamilyLaw.owl, em seguida são listados em um segundo painel intitulado “Relações”. O resultado para essa tarefa é

apresentado na Figura 33. Nele o usuário seleciona entre as relações apresentadas a relação que será a cabeça da regra a ser induzida, nesse caso selecionaremos o relacionamento “granfather_of(Man, Person)” e logo depois o relacionamento “brother_of(Man, Person)”.

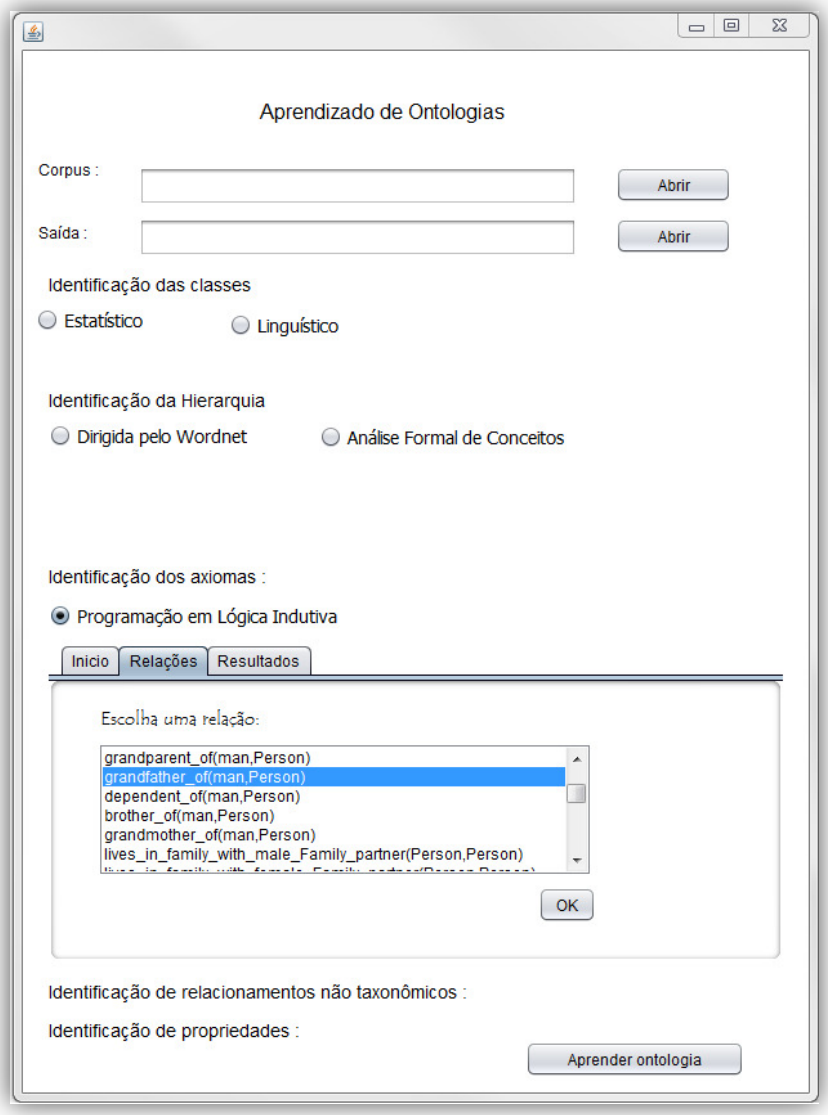

Figura 33 - Seleção da relação não taxonômica corpo da hipótese induzida.

Abaixo é descrito o processamento realizado pelo sistema PLI Progol que realiza uma busca a hipótese do relacionamento “*brother_of(Man, Person*)”. Nesse processamento podemos perceber a realização dessa busca utilizando um algoritmo de backtracking, utilizando as técnicas do Inverse Entailment e da construção de uma grafo de refinamento.

```
CProgol Version 4.2.1 for Win32

[:- modeh(100,brother_of(+man,-person))? - Time taken 0.00s]
[:- modeb(100,father_of(+man,-person))? - Time taken 0.00s]
[:- modeb(100,father_of(-man,+person))? - Time taken 0.00s]
[:- modeb(100,grandfather_of(+man,-person))? - Time taken 0.00s]
[:- modeb(100,grandfather_of(-man,+person))? - Time taken 0.00s]
[:- modeb(100,daughter_of(+woman,-person))? - Time taken 0.00s]
[:- modeb(100,daughter_of(-woman,+person))? - Time taken 0.00s]
[:- modeb(100,sister_of(+woman,-person))? - Time taken 0.00s]
[:- modeb(100,sister_of(-woman,+person))? - Time taken 0.00s]
[:- modeb(100,mother_of(+woman,-person))? - Time taken 0.00s]
[:- modeb(100,mother_of(-woman,+person))? - Time taken 0.00s]
[Testing for contradictions]
[No contradictions found]
[Generalising brother_of(man_30,man_29).]
[Most specific clause is]

brother_of(A,B) :- father_of(C,A), grandfather_of(D,A), grandfather_of(E,
       A), mother_of(F,A), father_of(C,B), father_of(C,G), father_of(D,
       C), father_of(D,H), father_of(E,F), father_of(E,I), father_of(E,
       J), grandfather_of(D,B), grandfather_of(D,G), grandfather_of(D,
       K), grandfather_of(E,B), grandfather_of(E,G), grandfather_of(E,
       K), daughter_of(F,E), daughter_of(F,L), daughter_of(J,
       E), daughter_of(H,D), sister_of(F,I), sister_of(F,J),
       sister_of(J,F), mother_of(F,B), mother_of(F,G), mother_of(F,
       K), mother_of(L,F), mother_of(M,C), father_of(N,M), grandfather_of(O,
       M), daughter_of(J,L), daughter_of(H,M), daughter_of(M,
       N), daughter_of(M,P), daughter_of(Q,K), daughter_of(R,
       K), daughter_of(S,K), sister_of(J,I), sister_of(M,T),
       sister_of(M,U), mother_of(K,Q), mother_of(K,R), mother_of(K,
       S), mother_of(L,I), mother_of(L,J), mother_of(M,H).

[C:12,14,0,2 brother_of(A,B).]
[C:12,14,0,1 brother_of(A,B) :- father_of(C,A).]
[C:10,10,0,1 brother_of(A,B) :- father_of(C,A), grandfather_of(D,A).]
[C:10,10,0,1 brother_of(A,B) :- father_of(C,A), grandfather_of(D,A).]
[C:11,14,0,1 brother_of(A,B) :- father_of(C,A), mother_of(D,A).]
[C:12,14,0,0 brother_of(A,B) :- father_of(C,A), father_of(C,B).]
[C:11,14,0,1 brother_of(A,B) :- father_of(C,A), father_of(C,D).]
[7 explored search nodes]
f=12,p=14,n=0,h=0
[Result of search is]
brother_of(A,B) :- father_of(C,A), father_of(C,B).
[Total number of clauses = 1]

[Time taken 0.002s]
```

Após a inferência de hipóteses relativa aos relacionamentos "*grandfather_of*" e "*brother_of*" os resultados obtidos são exibidos para o usuário em um novo painel, intitulado "Resultados" (Figura 34). Nela podemos perceber as hipóteses "*father_of(?A, ?C) ^ father_of(?C, ?B) -> grandfather_of(?A, ?B)"* e "*father_of(?A, ?C) ^ mother_of(?C, ?B) -> grandfather_of(?A, ?B)"* relativo a descoberta de axiomas de "*granfather_of*" e a hipótese "*father_of(?C, ?A) ^ father_of(?C, ?B) -> brother_of(?A, ?B)"* resultado da descoberta do relacionamento "*brother_of*".

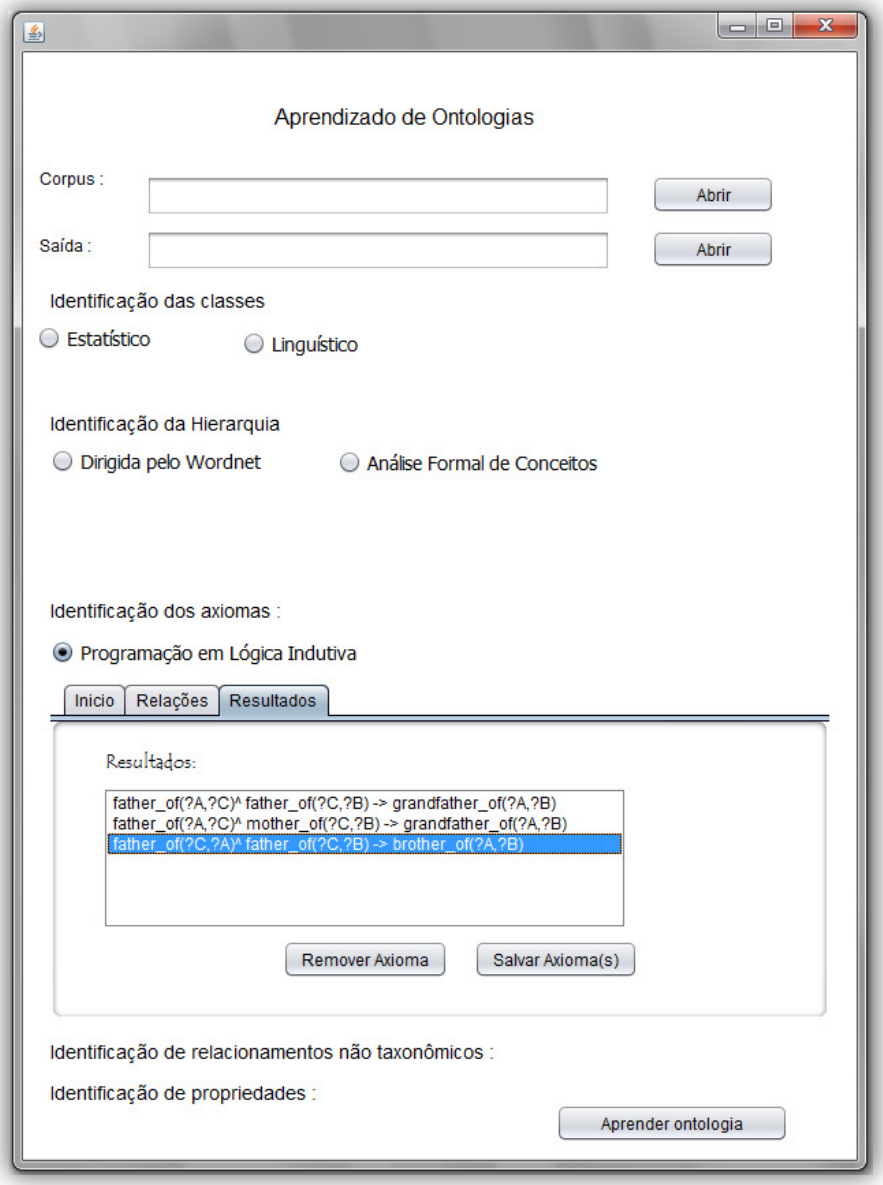

Figura 34 - Resultados apresentados na inferência das relações "grandfather_of" e "brother_of".

Analisando os resultados obtidos com essa primeira parte do estudo de caso podemos observar que as hipóteses resultantes da relação "*grandfather_of*" está correta, podendo ser inserida na ontologia como axioma, porem a hipótese retornada da relação "*brother_of*" apresenta um resultado parcial, pois o axioma correto seria: "*father_of(?C, ?A) ^ father_of(?C, ?B) ^ diferente_from(?A, ?B) → brother_of(?A, ?B)*", pois uma pessoa não pode ser considerada irmã dela mesmo. Isso demostra como o conjunto de hipóteses induzidas depende diretamente do seu conjunto de treinamento, logicamente a hipótese está correta de acordo com os exemplos positivos inseridos na sua base de conhecimento. Não há nenhuma maneira conhecida para a PLI

contornar os problemas relacionado a falta de informação suficiente para a geração de uma hipótese verdadeira senão a adição de mais informações a sua base de conhecimento, assim como é impossível para a PLI inferir hipóteses sem a presença de exemplos positivos em seu conjunto de treinamento.

Após a avaliação das hipóteses pelo usuário os axiomas considerados como verdadeiros são adicionados na ontologia de saída chamada "Tailp.owl", utilizando o Protégé para visualizar os axiomas inferidos podemos verificar a inserção dos novos axiomas através da ferramenta TAILP, utilizando o plugin SWRLtab (Figura 35).

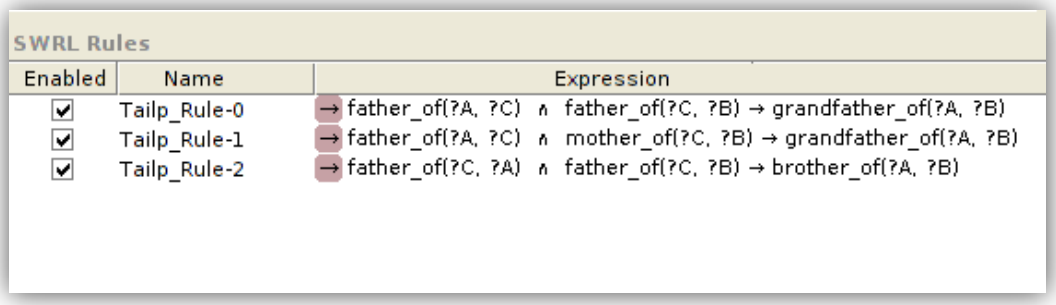

Figura 35 – Axiomas induzidos visualizados pelo Protégé.

Dando continuidade ao estudo de caso é realizado novamente a escolha de novos relacionamentos não taxonômicos "*married_with*", "*daughter_of*", "*sister_of*" e "*grandparent_of*" utilizando a mesma ontologia FamilyLaw, podemos constatar a inferência de novos axiomas que irão vir a ser inseridos na ontologia de saída (Figura 36).

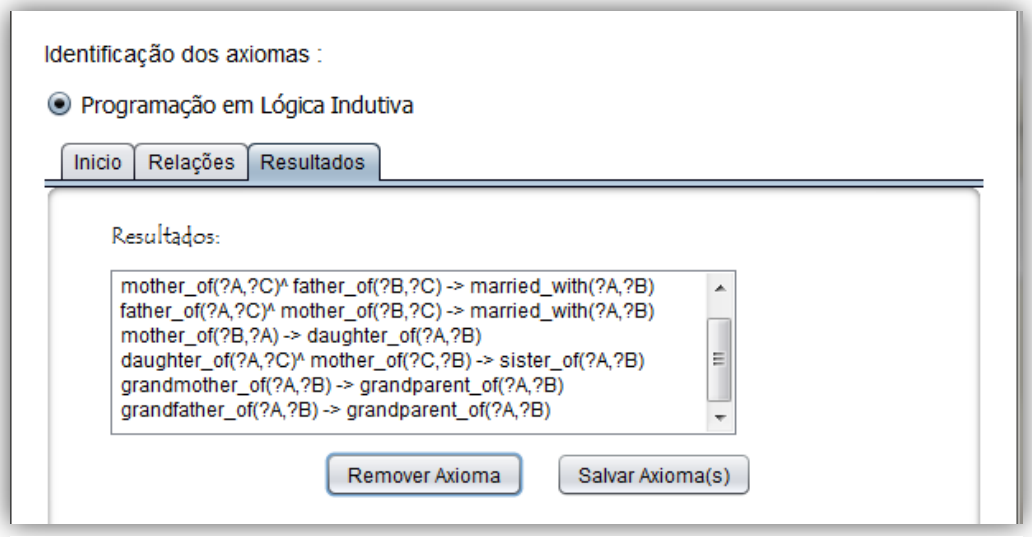

Figura 36 - Resultados de novo processamento de relações não taxonômicas da ontologia FamilyLaw.

A Figura 37 apresenta todas os axiomas inseridos na ontologia Tailp.owl. Para a validação dos axiomas inseridos e inferir novas informações, foi utilizado o plugin do Jesstab disponível na interface gráfica do Protégé. O plugin primeiramente transforma os dados da owl em conjunto com as regras em SWRL para LPO e realiza inferências deduzindo novas informações, procedimento demonstrado na Figura 38 e na Figura 39.

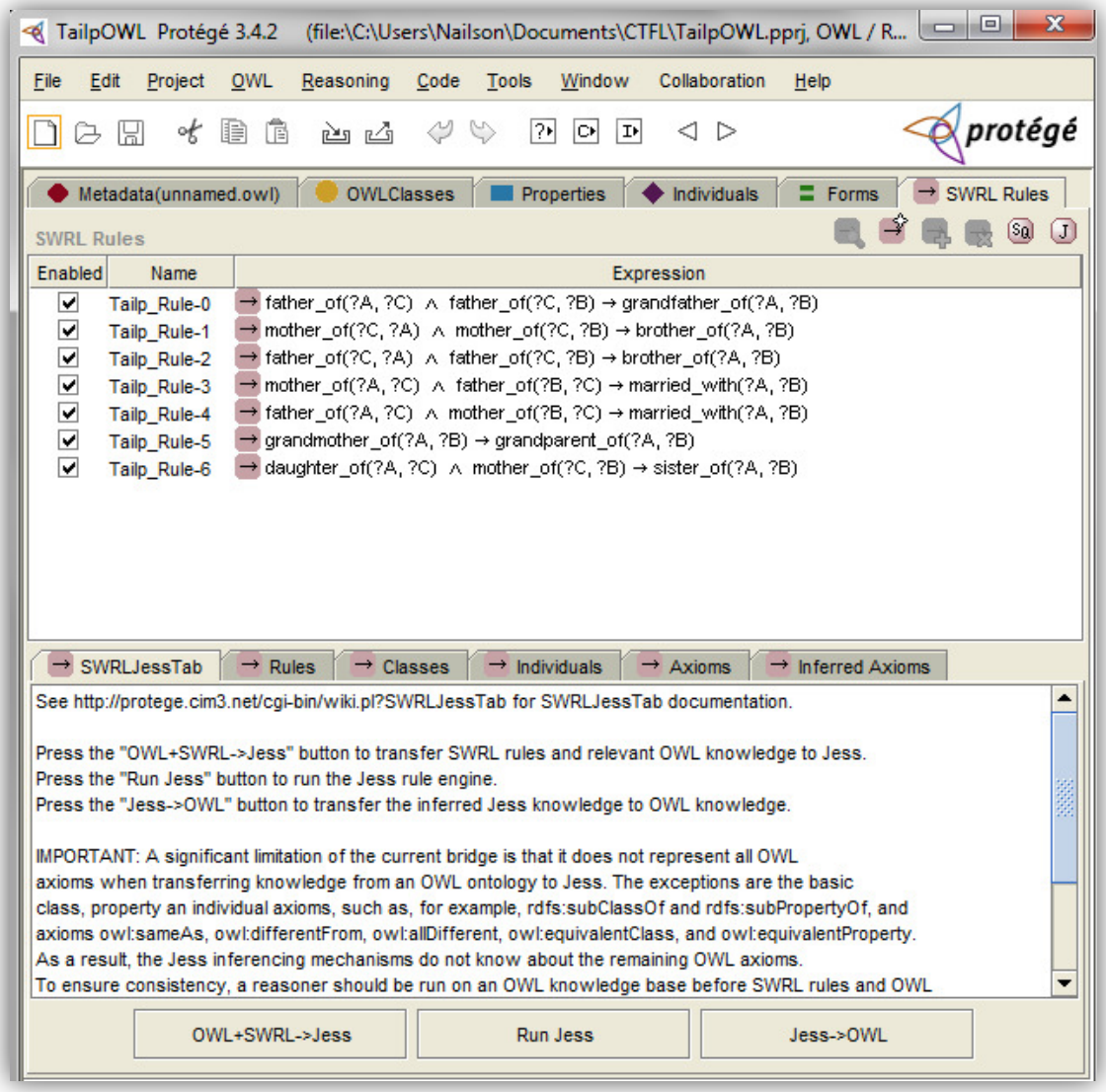

Figura 37 - Lista de Axiomas Inferidos e utilização do plugin do Jess pelo Protégé.

A Figura 38 demostra a interface do SWRLJessTab que realiza todas as fazes da geração de novas informações, ao total foram inferidas 97 novas instâncias de relacionamentos não taxonômicos .

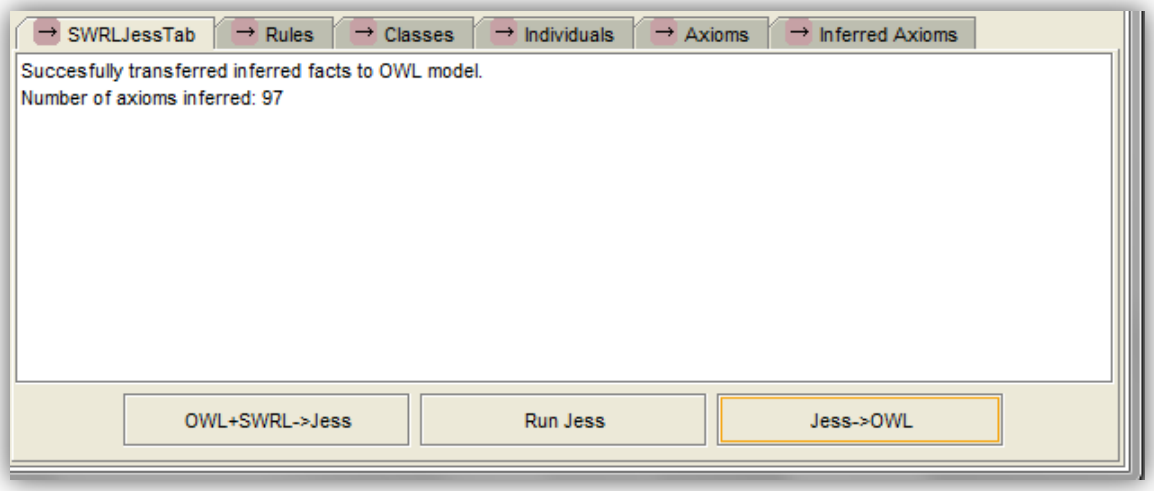

Figura 38 - Resultados apresentados pelo SWRLJessTab na inferência dedutiva de novas instâncias de relacionamentos.

A Figura 39 apresenta as 97 instâncias de relacionamentos deduzidas a partir dos axiomas da ontologia.

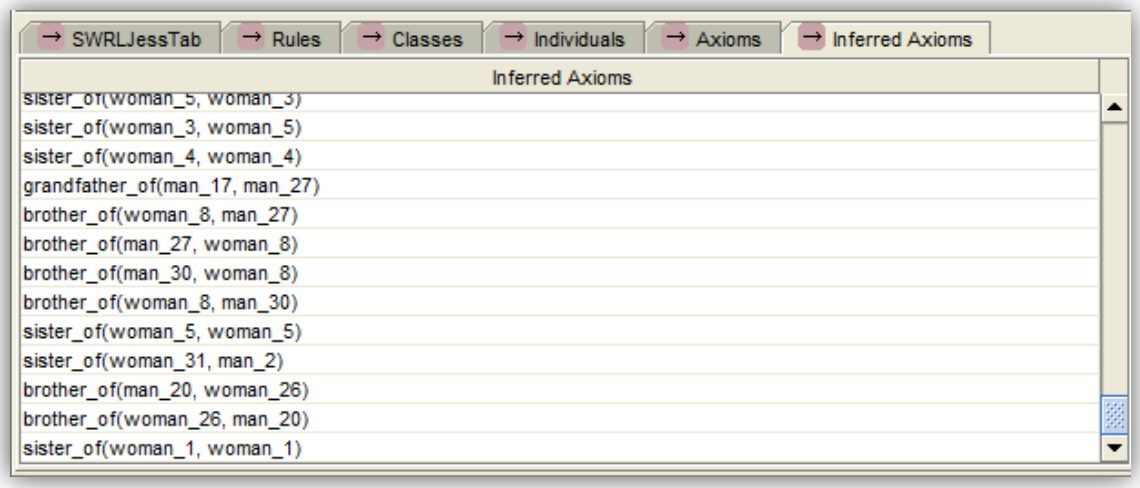

Figura 39 - Instâncias de relacionamentos não taxonômicos deduzidos dos axiomas induzidos pela TAILP.

6.3 RESULTADOS

Os resultados obtidos são apresentados na Tabela 2, que referencia a clausula escolhida e as hipóteses induzidas pela ferramenta TAILP.

Relação não taxonômica	Hipóteses Inferidas
Grandfather_of	father_of(?A,?C) ^ father_of(?C,?B) → grandfather_of(?A,?B); father_of(?A,?C) ^ mother_of(?C,?B) → grandfather_of(?A,?B).
Brother_of	father_of(?C,?A) ^ father_of(?C,?B) → brother_of(?A,?B); mother_of(?C,?A) ^ mother_of(?C,?B) → brother_of(?A,?B).
Married_with	mother_of(?A,?C) ^ father_of(?B,?C) → married_with(?A,?B); father_of(?A,?C) ^ mother_of(?B,?C) → married_with(?A,?B).
Sister_of	daughter_of(?A,?C)^mother_of(?C,?B) → sister_of(?A, ?B).
Grandparent_of	grandmother_of(?A, ?B) → grandparent_of(?A, ?B).

Tabela 2 - Resultados obitidos no estudo de caso.

Os resultados apresentados pela ferramenta podem ser considerados promissores, e conforme novas informações são adicionadas as hipóteses retornadas devem se tornar cada vez mais especializados. A solução para a extração de axiomas de ontologias apresenta-se como uma alternativa viável e genérica para qualquer ontologia que possua instâncias de relacionamentos não taxonômicos.

7 CONCLUSÃO

Este trabalho enfatizou a importância das Ontologias para a Web Semântica, descreveu formalmente os seus elementos e também os problemas enfrentados para sua criação e manutenção. Podemos destacar os elevados custos associados a obtenção do conhecimento e o tempo necessário para a criação das ontologias.

No contexto da Aprendizagem de ontologias o objetivo geral deste trabalho foi de contribuir com soluções para a aprendizagem de axiomas de uma ontologia de forma a aumentar a sua expressividade em relação ao domínio e diminuindo os custos e esforços no processo de povoamento e manutenção, reduzindo o número de instanciações de relações não taxonômicas e facilitando a coerência de suas informações.

A técnica TAILP se apresenta bem embasada nos conceitos formais das ontologias e da PLI e apresentou bons resultados na descoberta de axiomas em ontologias de domínio, no estudo de caso proposto. A ferramenta demostra muito bem os passos da técnica TAILP e a sua interação com o usuário muito bem segmentada e intuitiva, facilitando o entendimento geral do trabalho. Também se mostra útil como ferramenta de apoio, pois os axiomas inferidos são inseridos em um arquivo OWL. Para o desenvolvimento da ferramenta foram utilizadas as tecnologias mais modernas e relevantes disponíveis, no que tange a área de Ontologias e da Web semântica.

Este trabalho de pesquisa se apresentou como um grande avanço para a aprendizagem de axiomas de uma ontologia e está sendo referenciado em atividade futuras, onde a técnica desenvolvida será a incorporada a novos sistemas, sendo utilizados agentes computacionais na descoberta automática de axiomas de Ontologias. Para essa tarefa os agentes devem realizar buscas entre todos os relacionamentos não taxonômicos a fim de induzir padrões para novas hipóteses e a partir de inferências dedutivas e métodos estatísticos verificar a sua validade.

Para a obtenção de resultados mais conclusivos e satisfatórios da capacidade da PLI em aprender axiomas de ontologias deve-se utilizar uma Ontologia

melhor instanciada, isto é, com a presença de exemplos positivos conexos e confiáveis e que representem de maneira satisfatória os axiomas que se deseja induzir, pois, como bem sabemos, a construção de um conjunto de treinamento é primordial para a geração de bons resultados na Aprendizagem Indutiva.

No estudo das Ontologias este trabalho está diretamente relacionado com os conjuntos: **I** (das instâncias), o conjunto **R** (somente relações não taxonômicas) e do conjunto **A** (dos axiomas). Porém, um entendimento de toda a estrutura das Ontologias é necessário para que novas ideias venham a surgir em qualquer uma das etapas da construção automatizada de ontologias. Pensar em um passo na construção automatizada de Ontologias é abrir novos caminhos e ideias para pesquisas futuras.

Um processo contendo várias técnicas, inclusive a TAILP, faz parte de um projeto de mestrado intitulado “Processo incremental para o aprendizado e povoamento de Ontologia de aplicação”, que esta sendo desenvolvido no laboratório de pesquisa GESEC.

REFERÊNCIAS

Bishop, C. M. (2006). *Pattern recognition and machine learning.* Stringer.

Bratko, I., Kubat, M., & Michalski, R. (1998). *Machine Learning and Data Mining.* John Wiley & Sons Ltd.

Cimiano, P., Hotho, A., & Staab, S. (2004). Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Journal Of Artificial Intelligence Research*, 305-339.

Duarte, D. (2001). *Utilizando Técnicas de Programação Lógica Indutiva para Mineração de Banco de Dados Relacional.* Universidade Federal do Paraná.

Dzeroski, S. (1996). Inductive Logic Programming and Knowledge Discovery in Database. *Advences Discovery and Data Mining*, pp. 118-151.

Faria, C. G. (2013). Um processo independente de domínio para o povoamento automático de Ontologias a partir de fontes textuais. *Um processo independente de domínio para o povoamento automático de Ontologias a partir de fontes textuais*. UFMA, São Luís, Maranhão, Brasil.

Faria, C., & Girardi, R. (2010). Um Processo Semi-Automático para o Povoamento de Ontologias a partir de Fontes Textuais. *Sys - Revista Brasileira de Sistemas de Informação*, 12-12.

Faria, C., & Girardi, R. (2011). An Information Extraction Process for Semi-Automatic Ontology Population. *SOCO 2011 - Proceedings of the International Conferênce on Soft Computing Models in Industrial and Environmental Applications.* Salamanca, Spain: Springer.

FELLBAUM, C. (1998). *Wordnet: An Electronic Lexical Database.* Cambridge,: MIT Press.

Freitas, F. L. (2003). Ontologias e a Web Semântica. *Anais do XXIII Congresso da Sociedade Brasileira de Computação: SBPC*, (pp. 1-52). Campinas.

Galvão, R. (2010). *Análise do Estado da Arte da Aplicação do Processamento da Linguagem Natural na Construção Automatizada de Ontologias.* Relatório Final de Bolsista para o Comitê Local Do PIBIC., DEINF, São Luís.

Gennari, J. M. (2003). *The evolution of Protégé': an environment for.* Fonte: http://smi.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0943.html

Jr., Z. A. (29 de Janeiro de 2008). Aprendizagem de Ontologias a partir de Texto, Trabalho de Graduação. Pernambuco.

Lavrac, N., & Dzeroski, S. (1994). *Techniques and Applications.* New York: Ellis Horwood.

Lee, T. B., Hendler, J., & Lassila, O. (2001). *The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of.* Acesso em 22 de Fevereiro de 2012, disponível em scientific american: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&pageNumber=6&catID=2

Leite, N. B. (2011). *Aplicação da Programação em Lógica Indutiva na Construção Automatizada de Ontologias.* Relatório Final de Iniciação ciêntifica - PIBIC, Cnpq, UFMA, DEINF, São Luís.

Leite, N. B. (setembro de 2012). *Aquisição automática de axiomas de Ontologias utilizando a Programação em Lógica Indutiva.* Relatório Final de Iniciação ciêntifica - PIBIC, Cnpq, UFMA, DEINF, São Luís.

Macedo, M. (2010). *Processamento da linguagem natural para identificação de classes e instâncias de uma ontologia.* Monografia de Graduação em CiÊncia da Computação, UFMA, DEINF, São Luís.

Maedche, A., & Staab, S. (2004). Ontology learning. In: *Handbook on Ontologies* (pp. 173-190). Berlin: Springer.

Michalski, R. S., Cambonell, J., & Michell, T. (1983). *A Theory and Methodology of Inductive Logic Programming.* Tioga, Palo Alto, CA: TIOGA.

Michell, T. (1997). *Machine Learning.* WCB McGraw Hill.

Monard, M. C., & Baranauskas, J. A. (2003). Conceitos sobre Aprendizado de Maquina. *Sistemas Inteligentes-Fundamentos e Aplicações*, 89-114.

Monteiro, R. S. (2012). *Uma ferramenta para o povoamento e aprendizagem automáticos de ontologias.* Relatório Final de Iniciação Ciêntifica, FAPEMA, PIBIC, UFMA, DEINF, São Luís.

Muggleton, S. (1995). Inverse Entailment and Progol. *New Generation Computing*, pp. 13: 245-286.

Muggleton, S., & Raedt, L. D. (1994). Inductive Logic Programming: Theory and Methods. *Jornal of Logic Programming*, pp. pp. 19/20:629-679.

Nilsson, J. (4 de Dezembro de 1996). *Introduction to machine learning, an early draft of a proposed textbook.* Acesso em 23 de Novembro de 2011, disponível em Stanford University: http://ai.stanford.edu/~nilson/mlbook.html

Nilsson, U., & Maluszynski, J. (2001). *Logic, Programming and Prolog.* John Wiley & Sons Ltd.

Russel, S., & Norving, P. (1995). *Artificial Intelligence: A Modern Approach.* Prendice-Hall.

Shadbolt, N. R., Lee, T. B., & Hall, D. W. (2006). The Semantic Web revisited. *Intelligent Systems*, 96-101.

Shamsfard, & Barforoush, M. (2003). *The State of the Art in Ontology Learning: A Framework for Comparison. Intelligent Systems Laboratory.* Tehran, Iran.

Warren, D. (1983). The runtime environment for a prolog compiler using a copy algorithm. Suny and Stone Brook, New York, USA.

Winston, P. H., & Horn, B. K. (1981). *LISP.* Addison Wesley Publishing Company.

Zangrando, C. (1999). *Resumo do Direito do Trabalho.* Editora Destaque.