**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

# DETECÇÃO DE CONTRADIÇÕES EM UM SISTEMA DE APRENDIZADO SEM FIM

**VINICIUS OLIVERIO**

**ORIENTADOR: PROF. DR. ESTEVAM RAFAEL HRUSCHKA JR.**

São Carlos - SP
Junho/2012

**UNIVERSIDADE FEDERAL DE SÃO CARLOS**

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# DETECÇÃO DE CONTRADIÇÕES EM UM SISTEMA DE APRENDIZADO SEM FIM

**VINICIUS OLIVERIO**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial.
Orientador: Dr. Estevam Rafael Hruschka Jr.

São Carlos - SP
Junho/2012

**Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar**

O48dc

Oliverio, Vinicius.
Detecção de contradições em um sistema de aprendizado sem fim / Vinicius Oliverio. -- São Carlos : UFSCar, 2012.
65 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2012.

1. Inteligência artificial. 2. Aprendizagem. 3. Ontologia. 4. Contradição. I. Título.

CDD: 006.3 (20ª)

# Universidade Federal de São Carlos

**Centro de Ciências Exatas e de Tecnologia**
**Programa de Pós-Graduação em Ciência da Computação**

# "Detecção de Contradições em um Sistema de aprendizado Sem Fim"

Vinicius Oliverio

**Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação**

**Membros da Banca:**

**Prof. Dr. Estevam Rafael Hruschka Júnior**
**(Orientador - DC/UFSCar)**

**Profa. Dra. Heloisa de Arruda Camargo**
**(DC/UFSCar)**

**Profa. Dra. Beatriz de Souza Leite Pires de Lima**
**(UFRJ)**

**São Carlos**
**Junho/2012**

# RESUMO

O NELL (Never Ending Language Learning) é um sistema que busca aprender de uma maneira contínua, extraindo informação estruturada de páginas web desestruturadas utilizando o paradigma de aprendizagem semissupervisionado como um de seus princípios básicos. O Read the Web (RTW) é o projeto no qual o sistema NELL se insere. Atualmente o NELL possui cinco módulos, todos eles trabalhando independentemente onde um desses módulos é chamado Rule Learner (RL). O RL é responsável por induzir regras de primeira ordem, as quais são utilizadas pelo sistema para identificar padrões presentes no conhecimento gerado pelos outros quatro componentes do sistema. Estas regras são induzidas e, na sequência, representadas através de uma sintaxe que tem cláusulas de Horn como base. Tais regras podem apresentar contradições, e neste contexto o presente trabalho propõe a investigação, desenvolvimento e implementação de métodos para detectar e resolver estas contradições de maneira a fazer a aprendizagem mais eficiente.

**Palavras-chave**: NELL, Read, Web, Inteligência Artificial, Aprendizagem, Semissupervisionado, Ontologia.

# ABSTRACT

NELL (Never Ending Language Learning) is a system that seeks to learn in an infinite way, extracting structured information from unstructured web pages using the semi-supervised learning paradigm as one of its basic principles. The Read the Web (RTW) project is the project where the NELL system is contained, actually it consists of 5 modules, all of them working independently where one of the modules is called Rule Learner (RL). The RL is responsible for inducing first order rules, which are used by the system to identify patterns in the knowledge generated by the other four components of the system. These rules are induced and then represented in a syntax that has Horn clauses as base. These rules can present contradictions, and in this context this paper proposes investigate, develop and implement methods to detect and solve these contradictions so that the system can learn in a more efficient way.

**Keywords**: NELL, Read, Web, Artificial Intelligence, Learning, Semi-supervised, Ontology.

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

**CD** – *Contradiction Detection*

**CMC** – *Coupled Morphological Classifier*

**CP** – *Calculo Proposicional*

**CPL** – *Coupled Pattern Learner*

**CSEAL** – *Coupled SEAL*

**FNC** – *Forma Normal Conjuntiva*

**FOIL** – *First-Order Inductive Learner*

**HTML** – *Hypertext Markup Language*

**IDE** – *Integrated Development Environment*

**KB** – *Knowledge Base*

**LPO** – *Lógica de Primeira Ordem*

**MBL** – *Meta-Bootstrap Learner*

**NELL** – *Never Ending Language Learning*

**NLP** – *Natural Language Processing*

**RL** – *Rule Learner*

**RTW** – *Read The Web*

# SUMÁRIO

# Capítulo 1

# INTRODUÇÃO

*Neste capítulo é introduzida para o leitor a ideia geral do trabalho, mostrando sua relevância e a motivação para a sua execução, assim como o objetivo que se busca ao final do trabalho. O capítulo está estruturado da seguinte maneira: Contexto, onde é discutido o sistema NELL e são apresentados alguns conceitos introdutórios. Além disso, é apresentada também a motivação para o desenvolvimento de um trabalho de investigação que aborda a detecção de contradição no contexto de um sistema de aprendizado de máquina sem fim, bem como, uma descrição mais precisa dos objetivos desta pesquisa. Para concluir este capítulo, a estrutura do trabalho é apresentada.*

## 1.1 Contexto

O termo "contradição" pode ser interpretado de diferentes formas e utilizado em diversas áreas do conhecimento com objetivos bastante distintos [Bastos Filho, 2003]. O estudo das contradições é relevante em diversos domínios tanto teóricos quanto práticos e está presente em pesquisas relacionadas à ciência da computação em geral, bem como em trabalhos na área de Inteligência Artificial (IA) [Frosini, 2009]. No contexto da lógica de primeira ordem, a detecção de contradições pode ser muito útil, por exemplo, na prova automática de teoremas e na validação dos resultados de sistemas de indução de regras.

No processo de prova automática de teoremas, a identificação de uma contradição pode ser uma das etapas principais quando se utiliza a prova por "redução ao absurdo" [Byrnes, 1999]. Na indução de regras, a identificação de contradições pode ser utilizada para identificar a abrangência (ou cobertura) de cada regra candidata, bem como, identificar exemplos negativos.

Sistemas baseados em conhecimento podem se utilizar de mecanismos de detecção de contradições para validar e revisar o conhecimento armazenado. Assim, quando a base de conhecimento é gerada automaticamente (através de algum mecanismo de aprendizado de máquina), a detecção automática de contradições pode ser vista como uma tarefa muito relevante para dar subsidios para um processo de autossupervisão[1].

O NELL (Never-Ending Language Learning) [Carlson et al., 2010] é um sistema que busca aprender como aprender de uma maneira contínua, extraindo informações estruturadas de páginas web. Com base em um conjunto inicial de categorias (ator, filme, cidade, etc.) e relações (atorAtuaEmFilme, FilmeGravadoEmCidade, etc.) denominado ontologia inicial, o sistema extrai fatos da web. Os fatos extraídos são instâncias de categorias representadas através de predicados unários, como por exemplo, ator("Harrison Ford"), cidade("São Carlos"), etc., ou instâncias de relações representadas através de predicados binários, como por exemplo, atorAtuaEmFilme("Harrison Ford", "Indiana Jones"), FilmeGravadoEmCidade("O Turista","Veneza"), etc. Além da extração destes fatos da web, o NELL também utiliza técnicas de aprendizado de máquina para identificar padrões existentes nos fatos extraídos e ampliar sua capacidade de "aprendizado" utilizando tais padrões como novo conhecimento que auxiliará o processo de aprendizado nas próximas extrações a serem efetuadas. Este é o primeiro sistema computacional construído para se analisar a viabilidade deste novo paradigma de aprendizado, o aprendizado sem fim. Resultados iniciais mostram que algumas técnicas específicas (o acoplamento de métodos de aprendizado semissupervisionado, por exemplo) podem auxiliar na construção deste tipo de sistema.

Como dito anteriormente, o NELL parte de uma ontologia inicial (fornecida como entrada para o sistema) e busca expandi-la. Neste contexto, entende-se por ontologia uma especificação explícita de uma conceitualização vinculada a uma hierarquia. Segundo Gruber [Gruber,2009], no contexto da Ciência da Computação, uma ontologia pode ser utilizada para definir um conjunto de primitivas representacionais como conceitos, atributos, relacionamentos, indivíduos e restrições, as quais modelam um domínio de conhecimento ou discurso. Em diversas aplicações ontologias têm sido utilizadas com o intuito de facilitar a comunicação, tanto entre humanos e computadores, quanto entre sistemas computacionais. Quando o conhecimento de um domínio é representado em um formalismo declarativo, o conjunto de objetos que podem ser representados é chamado universo de discurso. Este

[1] O termo autossupervisão é utilizado neste trabalho para representar a capacidade de um sistema computacional corrigir (autonomamente) erros presentes em sua base de conhecimento.

conjunto de objetos, e os relacionamentos descritíveis entre eles, são refletidos no vocabulário representacional que é o vocabulário utilizado por um sistema para representar conhecimento. Portanto, pode-se descrever a ontologia de um programa definindo-se um conjunto de termos representacionais [Gruber,1993].

Em uma ontologia, definições podem associar os nomes de entidades no universo de discurso (i.e., classes, relações, funções, ou outros objetos) a textos compreensíveis por humanos descrevendo o que os nomes significam. Além disso, podem também ser definidos axiomas formais que restringem a interpretação e o bom uso destes termos [Gruber,1993]. No NELL, este tipo de estrutura (ontologia) é utilizado e os axiomas e restrições nele implícitos são explorados para guiar o processo de aprendizado com base na ideia de aprendizado semissupervisionado.

O paradigma de aprendizado semissupervisionado [Zhu, 2009] tradicionalmente está relacionado a problemas de aprendizado onde existe um pequeno número de dados rotulados e um grande número de dados não rotulados. Assim, tal forma de aprendizado surge como uma boa opção para ser utilizado como base de um sistema de aprendizado sem fim. Um dos problemas com métodos semissupervisionado, entretanto, é que um conjunto de treinamento limitado pode ser insuficiente para guiar o processo de aprendizagem de maneira adequada, então a abordagem proposta no NELL é a utilização do acoplamento (“coupling”) de diferentes métodos e técnicas de aprendizagem para melhorar o desempenho de algoritmos de aprendizado semissupervisionado e reduzir o impacto negativo gerado pelo pequeno número de dados rotulados.

Considerando que a ideia de aprendizado sem fim é bastante nova, e o NELL foi o primeiro sistema a implementar tais princípios, há ainda muitas questões de pesquisa a serem abordadas e que carecem de atenção especial. Uma destas questões está relacionada à extração de fatos e conceitos contraditórios. Dentre as mais variadas formas de contradição, neste trabalho três formas serão abordadas e explicadas nos capítulos seguintes. Estas formas de contradição são as contradições em regras de primeira ordem, contradições por exclusividade mútua e contradições em relações funcionais (Veja definições na seção 2.5.4).

No NELL, assim como ocorre com os seres humanos, durante o processo de aprendizado a base de conhecimento vai se expandindo com o passar do tempo. Mas, nem tudo que é inicialmente aprendido é correto. Seja por falha no processo de aprendizado ou por falta de conhecimento específico, erros do aprendizado podem não ser identificados imediatamente. À medida que se adquire mais conhecimento, sobre um determinado tema, entretanto, pode ser possível identificar falhas (erros) cometidas anteriormente. Para os seres

humanos é bastante comum que, à medida que se aprende mais sobre um tema, seja possível utilizar o novo conhecimento adquirido para revisar conceitos anteriormente considerados corretos. Neste trabalho chama-se esta capacidade de autorreflexão associada à autossupervisão. Assim, detecção de contradições é um aspecto relevante na busca de melhorias na autossupervisão e autorreflexão do NELL.

Na literatura pode-se observar que o estudo de técnicas de detecção de contradições pode ser conduzido através de diferentes abordagens e as contradições são definidas de várias maneiras, dependendo do contexto em que se está trabalhando. No contexto da lógica de primeira ordem, por exemplo, duas sentenças A e B são contraditórias se não há possibilidade em que A e B sejam ambas verdadeiras, mas ambas aparecem como verdadeiras [Marneffe, 2008]. Já no contexto da lógica descritiva, Lembo [Lembo, 2010] separa as contradições em dois tipos, contradições no TBox e no ABox. O TBox contém os conceitos e relações, tais como: noz é um fruto; e o ABox contém as instâncias dos conceitos e relações, tais como, avelã é uma instância de noz, portanto as contradições no TBox são contradições apresentadas por relações erroneamente aprendidas e as contradições no ABox são contradições apresentadas por conceitos aprendidos de maneira errada. Estas duas abordagens serão exploradas com mais detalhes durante a sequência do desenvolvimento deste trabalho.

## 1.2 Motivação e Objetivos

Como visto anteriormente, a detecção de contradições é um aspecto importante na pesquisa sobre o aprendizado sem fim e pode auxiliar na superação de algumas das limitações atuais do NELL. Assim o principal objetivo deste trabalho é investigar, implementar e avaliar métodos que permitam identificar contradições na base de conhecimento auxiliando no processo de autorreflexão e autossupervisão do NELL. Para tanto, pretende-se utilizar a base de conhecimento do sistema NELL, identificar contradições existentes e implementar métodos que possam corrigi-las quando possível.

Partindo-se da base de conhecimento do NELL, têm-se então como objetivos específicos:

- Investigar e implementar métodos para identificar contradições geradas por falhas no aprendizado de instâncias de categorias;

- Investigar e implementar métodos para identificar contradições geradas pelo aprendizado incorreto de instâncias de relações;
- Investigar e implementar métodos para correção automática de contradições através da expansão da ontologia e através das eliminação de instâncias já aprendidas.

## 1.3 Organização do Trabalho

A organização da sequência deste texto está assim estruturada, no Capítulo dois, a fundamentação teórica básica é apresentada, abordando os temas Aprendizagem de Máquina, Lógica, FOIL, Read The Web.

O Capitulo três traz os trabalhos correlatos mais relevantes encontrados no campo da detecção de contradições, esses trabalhos estão divididos em três subcategorias, “Contradições” que trata de trabalhos que dizem respeito a esclarecer o que é uma contradição e como elas se dão, “Detecção de Contradições com Base em Dicionários” que são trabalhos que utilizam da ajuda de dicionários para solucionar as contradições e “Detecção de Contradições em Textos” que são trabalhos que trazem algoritmos de solução de contradições com base na informação linguística do texto.

No Capitulo quatro a proposta e a metodologia de desenvolvimento do trabalho são mostradas, também é vista a implementação onde os algoritmos implementados são explicados e demonstrados por meio de fluxogramas. Por fim as experimentações são desmonstradas.

# Capítulo 2

# FUNDAMENTAÇÃO TEÓRICA

*Neste capítulo a fundamentação teórica essencial para a compreensão do trabalho é apresentada. Neste sentido, são apresentados conceitos básicos relativos à Aprendizagem de Máquina, Lógica de primeira ordem, FOIL e NELL.*

## 2.1 Considerações Iniciais

Neste capítulo é dada a fundamentação teórica básica para a compreensão deste trabalho de mestrado. Primeiramente é abordado o tema de Aprendizagem de Máquina onde são discutidos alguns paradigmas e algoritmos, posteriormente o conceito de Lógica é introduzido, com cálculo proposicional e lógica de primeira ordem, e também é apresentado o algoritmo FOIL. Depois de abordada a fundamentação necessária para a compreensão do projeto de investigação apresentado neste texto, o projeto Read The Web é descrito e explicado em mais detalhes.

## 2.2 Aprendizagem de Máquina

Por não haver uma definição única aceita por toda a comunidade, não se pretende neste texto definir o Aprendizado de máquina de maneira rígida. Em geral, Aprendizado de Máquina pode ser visto como o processo de mudança de estrutura, programa ou dado, baseado em entradas ou em resposta a informações externas, de maneira que o desempenho futuro de um algoritmo em alguma tarefa melhore [Nilsson,1997]. Segundo [Mitchell,1997],

para se dizer que uma máquina aprende ela deve, a partir de uma experiência E, com respeito a uma classe de tarefas T e medida de performance P, melhorar sua performance na tarefa T, medida por P, com essa experiência E.

Tradicionalmente, métodos de aprendizado de máquina indutivos são classificados em 3 abordagens: aprendizado supervisionado, aprendizado não supervisionado e aprendizado semissupervisionado. A forma e a presença (ou ausência) de supervisão são aspectos primordiais para caracterizar cada uma das três diferentes abordagens. Em todos os tipos de aprendizado tem-se como base um conjunto de dados $D = \{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\}$, onde $X = \{x_1, x_2, ..., x_n\}$ é o conjunto de vetores m-dimensionais $x_i = \{x_{i,1}, x_{i,2}, ..., x_{i,m}\}$ que representam instâncias de entrada, e $Y = \{y_1, y_2, ..., y_n\}$ é o conjunto de saídas, onde cada saída (ou rótulo) yi está vinculado a um vetor xi. O processo de aprendizado normalmente é definido pela aproximação de uma função que tome xi como entrada e forneça yi como saída ($f(xi) \rightarrow yi$). Para uma dada instância i, se o rótulo yi é conhecido, então diz-se que esta é uma instância rotulada. Mas, caso o rótulo yi não seja conhecido, então diz-se que esta é uma instância não rotulada. Sendo bastante genérico, pode-se dizer que D é formado por dois subconjuntos Dr e Dn ($D = Dr \cup Dn$). O subconjunto Dr é formado por dados rotulados, já o subconjunto Dn é formado por dados não rotulados. As próximas 3 subseções dão uma visão geral sobre estas 3 diferentes abordagens.

### 2.2.1 Aprendizagem Supervisionada

Nesta abordagem, normalmente se deseja construir um classificador. Assim, o subconjunto $D_r$ é utilizado para supervisionar o processo de aprendizado. Neste sentido, numa primeira etapa o modelo de aprendizado (classificador) é treinado com base em $D_r$ e aprende a predizer o valor de y para qualquer vetor x. Assim, com base no modelo treinado, os dados presentes em $D_n$ podem ser automaticamente rotulados.

Supondo, por exemplo, que se queira aprender a classificar um carro como sendo um "carro familiar" ou "não familiar". Supondo também que se tenha um conjunto $D_r$ de dados já rotulados definindo exemplos de carros familiares e não familiares. Então, para se definir um classificador, o que se pode fazer é encontrar um conjunto de características compartilhadas pelos exemplos positivos (carros familiares), e que não sejam compartilhadas pelos exemplos negativos (carros não familiares). Assim, através do conjunto de características encontradas nos exemplos positivos e negativos é possível fazer uma predição: Dado um carro que não tenha sido visto antes (ou seja, uma instância de $D_n$) e, checando a descrição que se aprendeu,

é possível dizer se um novo automóvel que se queira classificar é um "carro familiar" ou não [Alpaydin,2010].

Em outras palavras, na aprendizagem supervisionada o algoritmo aprende com a ajuda de um supervisor, que possui conhecimento sobre o problema, através de uma etapa de treinamento. Este conhecimento é normalmente representado por um conjunto de treinamento $D_r$ [Haykin,2001]. Com base neste conjunto de treinamento o algoritmo pode determinar qual a saída esperada quando novas entradas forem testadas. O conjunto $D_n$ é tradicionalmente utilizado para se testar a precisão do classificador, por este motivo, $D_n$ é comumente chamado de conjunto de testes.

Dentre os vários algoritmos de aprendizagem supervisionada podemos citar as Redes Neurais [Haykin,2001], Árvores de Decisão [Quinlan,1986], Máquinas de Vetores de Suporte [Cortes and Vapnik,1995], Naive Bayes [John and Langley,1995], entre outros. Com a finalidade de ilustrar este tipo de aprendizagem será demonstrado o funcionamento do algoritmo Naive Bayes.

#### 2.2.1.1 Naive Bayes

Também chamado de Classificador Naive Bayes, esse método é um classificador que se aplica a tarefas de aprendizagem onde o domínio de $f(x) \rightarrow v$ é finito (o problema exige um número finito de rótulos possíveis para uma dada instância de entrada $a_i$). Como já mencionado, por ser um algoritmo de aprendizado supervisionado, um conjunto de exemplos de treinamento deve ser fornecido como entrada ao algoritmo, e uma nova instância (representada por um vetor $a_i$) é apresentada. O treinamento do classificador consiste na definição da função alvo $f(x) \rightarrow v$ capaz de mapear corretamente o valor alvo, ou classificação, para uma a dada instância [Mitchell,1997]

O Naive Bayes se baseia na suposição simplificada que os atributos são condicionalmente independentes dado o valor alvo (valor de v). Em outras palavras, a suposição é que dado o valor alvo da instância *i*, a probabilidade da conjunção dos elementos de $a_i$ ($\{a_{i,1}, a_{i,2}, \ldots, a_{i,m}\}$) é dada pelo produto das probabilidades dos atributos individuais: $P(a_1, a_2 ... a_n \mid v_j) = \prod_i P(a_i, v_j)$. Portanto a abordagem utilizada pelo Naive Bayes é demonstrada na equação a seguir [Mitchell,1997].

**Classificador Naive Bayes:**

$$v_{nb} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j) \quad (2.1)$$

Onde $v_{nb}$ denota o valor alvo de saída dado pelo Naive Bayes. Resumindo, o método de aprendizagem Naive Bayes envolve um passo de aprendizagem no qual os vários termos $P(v_j)$ e $P(a_i|v_j)$ são estimados, baseados nas suas frequências no conjunto de exemplos de treinamento e no teorema de Bayes da seguinte forma:

$$P(a_i \mid v_j) = \frac{P(v_j \mid a_i)P(a_i)}{P(v_j)} \quad (2.2)$$

Para ilustrar o processo de aprendizagem temos os dados da Tabela 2.1 que serão classificados utilizando o algoritmo Naive Bayes. Este exemplo trata de um conjunto de dados onde se deve dar a probabilidade do dia ser um bom dia para se jogar tênis dada a condição climática do mesmo (adaptado de [Mitchell, 1997]).

**Tabela 2.1 - Jogar Tênis**

| Visual | Temperatura | Umidade | Vento | Jogar? |
|---|---|---|---|---|
| Ensolarado | Quente | Alta | Fraco | Sim |
| Ensolarado | Quente | Alta | Forte | Não |
| Nublado | Quente | Alta | Fraco | Sim |
| Chuvoso | Frio | Alta | Fraco | Não |

Supondo que os dados da Tabela 2.1 foram fornecidos ao Naive Bayes como conjunto de exemplos de treinamento e seja também fornecida a nova instância a ser classificada (Ensolarado,Frio,Alta,Forte). Neste exemplo, seguindo a descrição do algoritmo dada anteriormente, as instâncias são descritas por 4 atributos, ou seja, x = {$a_1$,$a_2$,$a_3$,$a_4$}, e o conjunto V de classes é composto por dois elementos V = {Sim, Não}.Para a nova instância dada o Naive Bayes vai calcular utilizando a equação (2.1) a probabilidade de Jogar = Sim e Jogar = Não para então decidir qual será o valor de saída, conforme os cálculos a seguir.

- P(Sim|Ensolarado,Frio,Alta,Forte) = P(Sim) * P(Ensolarado|Sim) * P(Frio|Sim) * P(Alta|Sim) * P(Forte|Sim) = 2/4 * 1/2 * 0/2 * 2/2 * 0/2 = 0.5 * 0.5 * 0.01 * 1 * 0.01 = 0.000025.
- P(Não|Ensolarado,Frio,Alta,Forte) = P(Não) * P(Ensolarado|Não) * P(Frio|Não) * P(Alta|Não) * P(Forte|Não) = 2/4 * 1/2 * 1/2 * 2/2 * 1/2 = 0.5 * 0.5 * 0.5 * 1 * 0.5 = 0.0625.

Tendo estas probabilidades calculadas o Naive Bayes tem como saída o mais provável que neste caso é Não. Portanto para esta nova instância a classificação realizada pelo Naive

Bayes é que a instância de entrada pertence à classe Não. Caso também se queira saber qual a probabilidade real das duas classes pode-se normalizar as probabilidades da seguinte maneira:

- P(Sim|Ensolarado,Frio,Alta,Forte) = 0.000025/(0.000025 + 0.0625) = 0.03%
- P(Não|Ensolarado,Frio,Alta,Forte) = 0.0625/(0.000025 + 0.0625) = 99.97%

O Naive Bayes nem sempre traz uma estimativa de probabilidade muito precisa de cada classe, mas tende a ter um nível de acerto muito bom quanto à ordenação das classes mais prováveis. O índice de acerto das classificações em alguns domínios se compara ao de Redes Neurais e Árvores de Decisão [Mitchell,1997]. A razão pela qual a probabilidade estimada pelo Naive Bayes não ser sempre precisa é justamente a simplificação feita por ele que desconsidera as relações entre os atributos, assumindo que eles são condicionalmente independentes dada a classe.

### 2.2.2 Aprendizagem Não Supervisionada

Como visto anteriormente, na aprendizagem supervisionada, o objetivo é induzir um mapeamento entre a entrada e a saída cujos valores são fornecidos por um supervisor. Na aprendizagem não supervisionada, não existe um supervisor e só o que se tem são os dados de entrada. O objetivo neste tipo de aprendizado é encontrar padrões nos dados de entrada que possam identificar grupos de dados homogêneos [Alpaydin,2010].

Em outras palavras, na aprendizagem não supervisionada, não existe um supervisor para guiar o algoritmo, isto significa que o conjunto de dados D é igual ao subconjunto $D_n$ e o subconjunto $D_r$ é vazio. Assim, não há exemplos rotulados para guiar o aprendizado [Haykin,2001], e por consequência, não existe uma fase de treinamento. Neste tipo de aprendizagem o que se busca é agrupar dados de acordo com a similaridade entre eles. O algoritmo indutor analisa a entrada fornecida e tenta determinar se os dados podem ser agrupados de alguma maneira, formando então grupos similares entre si.

Dentre os vários algoritmos de aprendizagem não supervisionada podem ser citados o K-Means [MacQueen,1967], Single-Link [Johnson,1967], Complete-Link [Johnson,1967], Expectation Maximization [Dempster,1977], entre outros. Com a finalidade de ilustrar este tipo de aprendizagem serão demonstrados os funcionamentos de dois algoritmos, o K-Means que foi inicialmente utilizado pelo sistema descrito no presente trabalho e o Expectation Maximization utilizando o Naive Bayes (que foi previamente explicado) como modelo

probabilístico simplificando[2] o processo de aprendizado, que na versão final do sistema proposto no presente trabalho é o algoritmo utilizado. Posteriormente serão discutidos os motivos da troca e as semelhanças e diferenças entre estes dois algoritmos.

#### 2.2.2.1 K-Means

O algoritmo K-Means é um algoritmo de agrupamento clássico primeiramente proposto por James MacQueen em 1967 [MacQueen,1967] que visa particionar *n* entradas em *k* grupos onde cada entrada pertence ao grupo com o centroide mais próximo.

Primeiramente *k* centroides são instanciados ($w_1,...,w_k$), estes centroides podem ser instanciados com valores aleatórios ou com alguma das *n* entradas ($i_1,...,i_n$). Portanto,

$$w_j = i_l,\ j \in \{1,...,k\}, l \in \{1,...,n\} \quad (2.3)$$

A fórmula 2.3 mostra uma descrição de alto nível do algoritmo K-Means. $C_j$ é o j-ésimo agrupamento cujo valor é um conjunto disjunto do conjunto de entrada. A qualidade do agrupamento é determinada pela seguinte função de erro:

$$E = \sum_{j=1}^{k} \sum_{i_l \in C_j} |i_l - w_j| \quad (2.4)$$

O tamanho apropriado de *k* é dependente do problema e do domínio, existem algoritmos especificamente criados para determinar um valor ótimo de *k* para um determinado conjunto de entradas.

O número de iterações necessárias pode variar de um número alto para somente algumas iterações dependendo do conjunto de entradas, da distribuição deste conjunto de entrada e do número de clusters.

O funcionamento do algoritmo se dá pelo seguinte pseudo-código:

1. Inicialização dos centroides
2. Enquanto não convergir:
    a. Para cada entrada:
        i. Calcula o grau de similaridade ou dissimilaridade para cada centroide.
        ii. Coloca a entrada no grupo mais similar.
    b. Recalcular os centroides de acordo com os dados no agrupamento.

2 O algoritmo EM original não se utiliza do Naive Bayes para estimar a probabilidade de uma dada instância pertencer a uma classe (p(Y|X)). Mas, para efeito de simplificação, nos exemplos mostrados nesta subseção, o cáculo de p(Y|X) será realizado assumindo-se a independência condicional dos atributos de x.

Para ilustrar o processo de aprendizagem temos os dados da Tabela 2.2 que será agrupada utilizando o algoritmo K-Means com *k* igual a 2. Este exemplo trata de um conjunto de dados onde se pode tentar agrupar os dias em que se deve jogar tênis em um grupo e os dias em que não se deve jogar tênis em outro grupo.

**Tabela 2.2 – Exemplo de Entradas (Jogar Tênis).**

| Visual | Temperatura | Umidade | Vento |
|---|---|---|---|
| Ensolarado | Quente | Alta | Fraco |
| Ensolarado | Quente | Alta | Forte |
| Nublado | Quente | Alta | Fraco |
| Chuvoso | Frio | Alta | Fraco |

Como passo inicial o algoritmo define os centroides para cada *k* aleatoriamente, portanto o conjunto de dados resultante é demonstrado na Tabela 2.3 onde os dados estão com o rótulo dado pelo algoritmo.

**Tabela 2.3 – Centroides iniciais.**

| Visual | Temperatura | Umidade | Vento | Grupo |
|---|---|---|---|---|
| Ensolarado | Quente | Alta | Fraco | G1 |
| Chuvoso | Frio | Alta | Fraco | G2 |

Os centroides são inicializados aleatoriamente, pois não se tem informações a respeito dos dados, somente quais valores eles podem assumir. Estes centroides são atualizados a cada iteração, portanto no fim o que se tem são os dados agrupados de acordo com a sua similaridade. A inicialização dos centroides interfere diretamente no resultado final, portanto, diferentes inicializações podem gerar resultados diferentes.

Com os centroides já inicializados o algoritmo calcula, neste caso utilizando a similaridade entre os valores das entradas e os centroides, ou seja, quando os valores para cada um dos itens forem iguais 1 é somado a similaridade quando não são iguais 0 é somado a similaridade, calculando a possibilidade de cada instância estar contida em um determinado grupo. Então temos:

- Primeira Linha:
    - Centroide G1 = 1 + 1 + 1 + 1 = 4
    - Centroide G2 = 0 + 0 + 1 + 1 = 2
- Segunda Linha:

- Centroide G1 = 1 + 1 + 1 + 0 = 3
- Centroide G2 = 0 + 0 + 1 + 0 = 1

- Terceira Linha:
  - Centroide G1 = 0 + 1 + 1 + 1 = 3
  - Centroide G2 = 0 + 0 + 1 + 1 = 2
- Quarta Linha:
  - Centroide G1 = 0 + 0 + 1 + 1 = 2
  - Centroide G2 = 1 + 1 + 1 + 1 = 4

O agrupamento resultante pode ser visto na Tabela 2.4, onde as entradas são inseridas nos grupos onde elas possuem mais similaridade com o cluster escolhido encontrando assim os grupos mais similares entre si.

**Tabela 2.4 – Resultado Final.**

| Visual | Temperatura | Umidade | Vento | Grupo |
|---|---|---|---|---|
| Ensolarado | Quente | Alta | Fraco | G1 |
| Ensolarado | Quente | Alta | Forte | G1 |
| Nublado | Quente | Alta | Fraco | G1 |
| Chuvoso | Frio | Alta | Fraco | G2 |

Após esta etapa os centroides são recalculados, para o exemplo aqui citado o método de recálculo dos centroides é selecionar os itens que mais aparecem no grupo, portanto os centroides não serão modificados pois para o item “Visual” no grupo 1 temos predominância de “Ensolarado”, para a “Temperatura” temos predominância de “Quente”, para “Umidade”, temos predominância de “Alta” e para vento temos predominância de “Fraco”, como no grupo 2 temos somente 1 item ele permanecerá como centroide.

Como os centroides permaneceram os mesmos, o agrupamento não será modificado, permanecendo como na tabela 2.4, portanto o algoritmo convergiu e finalizou tendo como grupos os vistos acima.

#### 2.2.2.2 Expectation Maximization

O algoritmo EM [Duda and Hart, 1973] é um algoritmo de agrupamento amplamente utilizado para o aprendizado na presença de variáveis não observadas e pode ser aplicado em vários casos onde há a necessidade de estimar algum grupo de parâmetros θ que descrevam

uma distribuição de probabilidade, dado somente a porção observada do conjunto total de dados produzidos pela distribuição [Mitchell,1997].

O EM busca pela hipótese de verossimilhança máxima h', que é a hipótese que maximiza uma função de probabilidade, buscando o h' que maximize Ln P(Y|h'), onde Y é uma variável aleatória e Ln é o logaritmo natural. Este valor esperado é tomado sobre a distribuição de probabilidade que governa Y, que é determinada pelos parâmetros desconhecidos θ. O algoritmo EM utiliza sua hipótese atual h no lugar dos atuais parâmetros θ para estimar a distribuição que governa Y pela função Q(h|h') = ln P(Y|h')|h, X. A função Q é descrita no formato Q(h|h') para indicar que é definida em parte pela suposição de que h é igual a θ [Mitchell,1997].

De uma forma geral o algoritmo EM repete dois passos principais, primeiro calcula Q(h|h') utilizado a hipótese atual h e os dados observados X para estimar a distribuição de probabilidade sobre Y e depois substitui a hipótese h pela hipótese h' que maximiza esta função Q, para então repetir o processo até convergir.

O pseudo-algoritmo do EM se da como segue:

1. Entrada: Conjunto de exemplo e amostras
2. Criação de um modelo inicial com a estimativa de θ a partir apenas dos exemplos.
3. Enquanto não convergir.
    a. Expectation – Utilizando o classificador atual θ, estimar a probabilidade de cada amostra.
    b. Maximization – Recalcular o classificador θ, utilizando os exemplos e as amostras estimadas.

Para ilustrar o processo de aprendizagem temos os dados da Tabela 2.2 que será agrupada utilizando o algoritmo EM (tendo o Naive Bayes como modelo de probabilidade simplificado). Este exemplo trata de um conjunto de dados onde se pode tentar agrupar os dias em que se deve jogar tênis em um grupo e os dias em que não se deve jogar tênis em outro grupo.

Como passo inicial o algoritmo define o grupo para cada conjunto de dados aleatoriamente, portanto o conjunto de dados resultante é demonstrado na Tabela 2.5 onde os dados estão com o rótulo dado pelo algoritmo.

**Tabela 2.5 - Dados Rotulados.**

| Visual | Temperatura | Umidade | Vento | Grupo |
|---|---|---|---|---|
| Ensolarado | Quente | Alta | Fraco | G1 |
| Ensolarado | Quente | Alta | Forte | G2 |
| Nublado | Quente | Alta | Fraco | G1 |
| Chuvoso | Frio | Alta | Fraco | G2 |

Os rótulos são dados aleatoriamente pois não importa qual é o grupo inicial já que no fim o que se quer é que os dados estejam agrupados de acordo com a sua similaridade. Diferentes inicializações de rótulos podem levar a um maior ou menor tempo para a convergência, mas é possível provar que para qualquer inicialização o EM sempre converge. Inicializações inadequadas podem, entretanto, forçar a convergência para ótimos locais.

Com os dados já rotulados o algoritmo calcula, neste caso utilizando o Naive Bayes, se as instâncias já estão no grupo correto, calculando a probabilidade de cada instância estar contida em um determinado grupo. Então temos:

- Primeira Linha:
  - P(Ensolarado,Quente,Alta,Fraco|G1) = P(Ensolarado|G1) * P(Quente|G1) * P(Alta|G1) * P(Fraco|G1) = ½ * 1 * 1 * 1 = 0.5
  - P(Ensolarado,Quente,Alta,Fraco|G2) = P(Ensolarado|G2) * P(Quente|G2) * P(Alta|G2) * P(Fraco|G2) = ½ * ½ * 1 * ½ = 0.125
- Segunda Linha:
  - P(Ensolarado,Quente,Alta,Forte|G1) = P(Ensolarado|G1) * P(Quente|G1) * P(Alta|G1) * P(Forte|G1) = ½ * 1 * 1 * 0.01 = 0.005
  - P(Ensolarado,Quente,Alta,Forte|G2) = P(Ensolarado|G2) * P(Quente|G2) * P(Alta|G2) * P(Forte|G2) = ½ * ½ * 1 * ½ = 0.125
- Terceira Linha:
  - P(Nublado,Quente,Alta,Fraco|G1) = P(Nublado|G1) * P(Quente|G1) * P(Alta|G1) * P(Fraco|G1) = ½ * 1 * 1 * 1 = 0.5
  - P(Nublado,Quente,Alta,Fraco|G2) = P(Nublado|G2) * P(Quente|G2) * P(Alta|G2) * P(Fraco|G2) = 0.01 * ½ * 1 * ½ = 0.0025
- Quarta Linha:
  - P(Chuvoso,Frio,Alta,Fraco|G1) = P(Chuvoso|G1) * P(Frio|G1) * P(Alta|G1) * P(Fraco|G1) = 0.01 * 0.01 * 1 * 1 = 0.00005

- P(Chuvoso,Frio,Alta,Fraco|G2) = P(Chuvoso|G2) * P(Frio|G2) * P(Alta|G2) * P(Fraco|G2) = ½ * ½ * 1 * ½ = 0.125

Portanto as instâncias são mantidas nos grupos iniciais, e o agrupamento resultante pode ser visto na Tabela 2.6 que por coincidência é agrupamento final do algoritmo, pois após a segunda iteração ela não se altera.

**Tabela 2.6 - Resultado Final.**

| Visual | Temperatura | Umidade | Vento | Grupo |
|---|---|---|---|---|
| Ensolarado | Quente | Alta | Fraco | G1 |
| Ensolarado | Quente | Alta | Forte | G2 |
| Nublado | Quente | Alta | Fraco | G1 |
| Chuvoso | Frio | Alta | Fraco | G2 |

#### 2.2.2.3 K-Means e Expectation Maximization

Durante o desenvolvimento do método de identificação e eliminação de contradições (proposto neste trabalho) o primeiro algoritmo de agrupamento utilizado foi o K-Means. Entretanto, com o intuito de se ter uma implementação que permita a execução de alguns pontos importantes a serem desenvolvidos como trabalhos futuros (veja a Capítulo 5), optou-se por trocá-lo pelo algoritmo EM, com um número de grupos variável. Além disso, essa mudança foi também motivada pelo fato de que ao serem empiricamente comparados os dois algoritmos no sistema em questão, foi possível observar um melhor comportamento do algoritmo EM para o problema aqui proposto.

A grande diferença entre o K-Means e o EM é o fato do EM ser um algoritmo probabilístico, o que permite a ele ter grupos menos rígidos, o fato de ter grupos menos rígido se deve pois o EM sendo um algoritmo probabilístico um elemento pode ter probabilidades diferentes de pertencer a grupos diferentes (como se fossem graus de pertinência), e de diferentes formatos espaciais, ao contrário do K-Means onde um elemento pertence a um grupo rígido e de formatos lineares.

O EM também permite ao usuário saber qual a probabilidade de um elemento pertencer a um determinado grupo, e embora o algoritmo coloque o elemento no grupo com maior probabilidade é possível criar uma variação do algoritmo que utilize essa informação de probabilidade para algum fim. Por exemplo, caso a probabilidade de um elemento pertencer a um grupo for muito similar à probabilidade dele não pertencer (e.g. probabilidade de pertencer 51% e probabilidade de não pertencer 49%) pode-se criar outro grupo para este

elemento. O que também é uma possibilidade ao se tratar o problema proposto no presente trabalho. Esta possibilidade de se explorar a probabilidade na definição dos grupos ainda não foi incluída nos resultados apresentados neste trabalho e deverá ser o foco de trabalhos futuros.

### 2.2.3 Aprendizagem Semissupervisionada

O paradigma de aprendizado semissupervisionado aborda o estudo de algoritmos e métodos capazes de aprender na presença de dados rotulados e não rotulados. Uma das maiores motivações para a investigação deste tipo de algoritmo é a grande quantidade de problemas reais em que há poucos dados rotulados disponíveis e uma grande quantidade de dados não rotulados. Além disso, a tarefa de rotular dados tende a ser muito custosa e demorada. Assim, o desenvolvimento de técnicas e métodos capazes de realizar tarefas de aprendizado de máquina necessitando de um pequeno conjunto de dados rotulados pode trazer grandes benefícios.

Tradicionalmente, o aprendizado semissupervisionado tem sido estudado tanto com foco no paradigma supervisionado (focado em tarefas de classificação) o quanto no paradigma não supervisionado (focado em tarefas de agrupamento). Investigações neste paradigma tem tido como alvo a análise e pesquisa de como a utilização desses dados rotulados e não rotulados pode mudar o comportamento do aprendizado, e criar algoritmos que se beneficiem dessa combinação [Zhu,2009].

Dentre as diversas linhas de pesquisa da área de aprendizagem semissupervisionada, uma traz algoritmos que combinam dados rotulados e não rotulados para autonomamente, gerar novos exemplos de treinamento para prosseguir sua aprendizagem. Isto pode ser feito utilizando o rótulo que o próprio algoritmo dá aos dados não rotulados. Assim, numa abordagem semissupervisionada, o conjunto de dados $D = \{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\}$, utilizado para o aprendizado, é composto por 3 subconjuntos, $D = D_r \cup D_n \cup D_{ar}$ (ao invés de $D = D_r \cup D_n$), onde $D_r$ contém os dados rotulados, $D_n$ contém dados não rotulados e $D_{ar}$ contém dados que originalmente eram não rotulados e foram "automaticamente rotulados" pelo algoritmo de aprendizado. O processo de aprendizado semissupervisionado é definido de maneira iterativa. Assim, na primeira iteração tem-se $D_1 = D_{r,1} \cup D_{n,1}$ e $D_{ar,1} = \varnothing$. Após o término da primeira iteração, um subconjunto $S_1$ de $D_{n,1}$ recebe rótulos (y) gerados automaticamente. Desta forma, para a segunda iteração tem-se $D_2 = D_{r,2} \cup D_{n,2} \cup D_{ar,2}$; $D_{ar,2}$

$= S_1$; $D_{r,2} = D_{r,1} \cup D_{ar,2}$. Para uma iteração *k*, tem-se: $D_k = D_{r,k} \cup D_{n,k} \cup D_{ar,k}$; $D_{ar,k} = S_{k-1}$; $D_{r,k} = D_{r,k-1} \cup D_{ar,k}$.

Um dos problemas com este tipo de aprendizado é que esse novo conjunto de treinamento $D_{ar}$ (contendo dados rotulados pelo próprio algoritmo) pode ser muito influenciado pelo viés indutivo do algoritmo de aprendizado e pela pequena amostra rotulada dada em $D_r$, e isto pode levar a erros futuros[3]. Como poucos dados inicialmente rotulados são utilizados para treinar e classificar as novas instâncias não rotuladas, o classificador pode estar equivocado em algumas instâncias, assim inserindo rótulos "errados". Como os novos rótulos serão utilizados na sequencia do processo de aprendizado, erros inseridos nos rótulos serão propagados e a qualidade do aprendizado tende a se deteriorar conforme o número de iterações aumenta.

Para exemplificar uma abordagem semissupervisionada, considere a utilização dos mesmos dados do exemplo 2.2.1.1 (Jogar tênis). Neste cenário, a abordagem de um algoritmo semissupervisionado seria utilizar os dados rotulados ($D_r$) para treinar, um classificador (como foi realizado pelo Naive Bayes), e quando uma nova instância de teste $x_i$ for classificada (gerando $y_i$), a esta instância ($x_i$,$y_i$) agora já rotulada seria adicionada ao conjunto de treinamento $D_{ar}$ e o algoritmo, na próxima iteração, realizaria novamente a etapa de treinamento, agora com este dado que o próprio algoritmo rotulou.

### 2.2.4 Aprendizagem Sem Fim

O princípio básico do aprendizado sem fim está bastante vinculado com a ideia de aprendizado semissupervisionado, onde o processo iterativo deve se repetir por um número indefinido de iterações. Existe entretanto um problema, como visto na seção anterior, em um cenário onde há poucos exemplos rotulados e um número muito grande de exemplos não rotulado, o aprendizado semissupervisionado pode ser utilizado de maneira que novos exemplos rotulados sejam automaticamente gerados e passem a ser utilizados como auxílio na supervisão do aprendizado. Neste cenário, entretanto, dependendo da qualidade dos dados inicialmente rotulados ($D_r$) e do viés indutivo do método de aprendizado, o desvio de conceito pode surgir como um grande problema. Assim, não é viável (na maioria das aplicações) a utilização de um método semissupervisionado tradicional por um número de iterações muito

[3] Erros gerados por conta de dados rotulados pelo próprio algoritmo são normalmente chamados de desvio de conceito (ou *concept drift*).

elevado. Isto faz com que ideia de aprendizado sem fim não possa ser abordada com a simples utilização de métodos tradicionais de aprendizado semissupervisionado.

Pesquisas desenvolvidas sobre o tema de aprendizado sem fim mostram [Carlson et al., 2009; Carson 2010; Duarte et al., 2011] que o acoplamento de diferentes métodos e algoritmos de aprendizado pode auxiliar na redução do problema de desvio de conceito em abordagens de aprendizado semissupervisionado. Neste sentido, o acoplamento e a utilização de diversas visões (aprendizado multivisões) são ideias chaves que viabilizaram a construção do primeiro sistema computacional de aprendizado sem fim (como pode ser visto em [Carlson et al., 2010]). Assim, com a minimização do desvio de conceito é possível haver um processo de aprendizado (semissupervisionado) que seja executado iterativamente por um tempo indeterminado, e isto caracteriza a ideia de aprendizado sem fim.

Além de poder ser executado e continuar aprendendo por tempo indeterminado, sistemas de aprendizado sem fim (SASFs) devem, em geral, seguir os seguintes princípios:

1. A arquitetura inicial do sistema deve definir um conjunto de "tipos" de conhecimento (ou funções) a serem aprendidos de acordo com a tarefa de aprendizado a ser executada. Assim, se a tarefa a ser executada é a leitura da Web (como ocorre em [Carlson et al., 2010]), alguns tipos padrão de conhecimento devem ser aqueles usados no processamento de língua natural (PLN), tais como, entidades nomeadas (por exemplo, nomes de cidades, empresas, pessoas, etc.), relações entre entidades nomeadas (por exemplo, TrabalhaPara(Larry Page, Google)), correferência (por exemplo, mesmaPessoa(Larry Page, L. Page)), etc.

2. O sistema deve ter habilidade de autorreflexão para permitir a identificação automática das próximas tarefas de aprendizado a serem executadas. Ou seja, à medida que o SASF vai aprendendo (e armazenando conhecimento), sua busca por mais conhecimento deve ser voltada para pontos específicos onde ainda não se tem conhecimento suficiente, ou ainda não se tem a precisão necessária. Para evitar que se atinja um ponto onde nenhum novo conhecimento consiga ser extraído, é necessário identificar se houver algo impedindo ou limitando o sistema de aprender. O SASF pode, por exemplo, ter dificuldade inicialmente em extrair novos conhecimentos a partir da Web por conta da baixa qualidade de seu extrator de entidades nomeadas. Um ano após estar sendo executado, pode ser que a extração de entidades nomeadas tenha atingido um bom nível de qualidade. O que pode estar, então, limitando a sua performance pode ser a baixa qualidade (ou ausência) na identificação de correferências. Assim, o sistema precisa se auto-ajustar e focar na melhoria da tarefa que o impede de atingir resultados com a qualidade esperada.

3. O sistema deve ter habilidade para formular suas próprias tarefas de aprendizado para permitir a busca de soluções para as limitações identificadas. Isto envolve ações normalmente executadas por projetistas e desenvolvedores humanos nos sistemas computacionais tradicionais. Tais ações podem ser a escolha dos atributos a serem utilizados para se representar um exemplo, a escolha de qual melhor algoritmo a ser utilizado, etc. Sem esta capacidade, torna-se difícil se conseguir que um sistema siga aprendendo continuamente por um período de tempo indefinido.

4. Um sistema de aprendizado sem fim precisa ter a capacidade de alterar sua forma de representação do conhecimento de modo a permitir a expansão e a revisão do conhecimento previamente armazenado (i.e. expansão de sua ontologia). Assim, quando uma limitação for identificada e uma nova tarefa de aprendizado for automaticamente definida, o modelo de representação do conhecimento precisa ser ajustado às mudanças ocorridas.

5. Considerando que um sistema de aprendizado sem fim deva ser autossupervisionado, e contar com supervisão humana bastante reduzida, a utilização de dados não rotulados acaba se tornando necessária na tarefa de aprendizado. Assim, mecanismos de consistência interna devem estar presentes no sistema permitindo que um novo conhecimento seja validado antes de ser inserido como verdadeiro na base de conhecimento. O SASF deve contar com pouca supervisão humana e o aprendizado semissupervsionado com base na redundância dos fatos presentes na Web deve ser a base dos métodos de aprendizado a serem utilizados (mantendo os trabalhos iniciais já realizados em [Carlson et al., 2009, 2010]).

6. A capacidade de melhorar o desempenho e sua capacidade de aprendizado é essencial para que um sistema de aprendizado sem fim possa realizar tarefas mais complexas com o passar do tempo. Os seres humanos se utilizam de conhecimentos prévios para aprender mais. Assim, por exemplo, o ser humano utiliza seus conhecimentos sobre adição para aprender a multiplicação, e utilizam ambos (adição e multiplicação) para aprender álgebra. Em geral, o aprendizado da álgebra não acontece antes do aprendizado da adição e multiplicação. O SASF apresenta uma boa oportunidade para que a propriedade de “aprender a aprender” possa ser implementada em um sistema computacional. Imagine, por exemplo, que o sistema aprendeu alguma regra a partir de padrões existentes em sua base de conhecimento (por exemplo, “Se um funcionário F trabalha em uma empresa E, então o chefe do funcionário F normalmente trabalha na empresa E”). Esta regra pode auxiliar a melhorar o aprendizado de entidades nomeadas, pois ela pode ser utilizada para extrair nomes de

funcionários de uma empresa auxiliando, assim, os outros componentes do sistema já responsáveis pela extração de entidades nomeadas.

Os seis princípios acima descritos revelam importantes desafios a serem superados para que a construção de um sistema de aprendizado sem fim possa ser alcançada com sucesso, pois não há ainda um sistema computacional capaz de tratar todas as questões vinculadas a estes princípios. Entretanto, bons resultados já foram alcançados, como pode ser visto no sistema NELL[4] [Carlson, 2010]. O NELL buscar se enquadrar nos princípios acima mencionado e sua abordagem envolve três ideias chave [Mitchell et al., 2009], a saber:

1. Macro-Leitura ao invés de Micro-Leitura. Considere a "micro-leitura" como sendo o objetivo tradicional das técnicas de PLN, ou seja, onde se busca interpretar e entender todas as informações presentes em um texto ou documento. A "macro-leitura", por outro lado, pode ser vista como uma tarefa onde se tem como entrada uma grande quantidade de textos (a Web, por exemplo) e a saída desejada é uma coleção de fatos extraídos dos textos, mas onde não é necessário que todas as sentenças presentes nos textos sejam interpretadas. Desta forma, pode-se dizer que a macro-leitura é mais simples (e mais fácil de ser implementada com sucesso) por duas razões, a saber: i) a primeira é que não há a necessidade de se extrair todo o conhecimento presente em cada frase presente na coleção de textos de entrada. ii) a segunda é que a macro-leitura pode contar com a redundância presentes nos textos, ou seja, muitos fatos importantes estão presentes (na Web, por exemplo) milhares de vezes e são expressados de várias maneiras diferentes. Assim, se em um documento o fato A não pode ser extraído pois está descrito de maneira muito complexa (linguagem muito rebuscada ou frase muito longa), em outros documentos, este fato A pode estar presente de maneira mais clara e isto facilitará sua extração.

2. Leitura guiada por uma ontologia: uma das grandes dificuldades em se extrair conhecimento de textos livres é que tais textos podem se referir a qualquer coisa. Em contraste, a proposta de macro-leitura guiada por uma ontologia conta com informações predefinidas, tais como categorias (por exemplo, pessoa, localidade, universidade, etc.) e relações (por exemplo, "estudaEm", "moraEm", etc.). Como consequência, a macro-leitura guiada por uma ontologia pode focar apenas no subconjunto de textos (ou documentos) vinculados aos temas definidos na ontologia, e meta-informações podem ser adicionadas de modo a facilitar o aprendizado (como por exemplo, meta-informações que definam que a

[4] Never-Ending Language Learner – http://rtw.ml.cmu.edu - uma visão geral do NELL é dada na seção 2.5.

categoria “pessoa” é mutuamente exclusiva com “universidade”, e que a relação “moraEm” relaciona uma “pessoa” e uma “localidade”).

3. Ganho na precisão de algoritmos de aprendizado de máquina através do aumento da complexidade da ontologia: esta ideia tem como base a utilização de métodos de aprendizado de máquina semissupervisionados capazes de extrair padrões de texto e hipertexto que darão suporte à extração de conhecimento da Web. Assim, a abordagem de aprendizado de máquina a ser seguida envolve o aprendizado de padrões textuais (por exemplo, “prefeito de X” normalmente sugere que X é uma cidade), bem como padrões de HTML, para cada predicado (categoria ou relação) da ontologia.

Em resumo, a proposta para viabilizar a construção do primeiro SASF tem como base a utilização de métodos de aprendizado máquina supervisionado, não-supervisionados e semi-supervisionados para aprender a extrair conhecimento acerca de cada predicado dado em uma ontologia inicial, utilizando tais predicados para guiar a macro-leitura a ser realizada em milhões de documentos disponíveis na Web.

## 2.3 Lógica de Primeira Ordem

Os pioneiros na criação da Lógica de Primeira Ordem (LPO) foram Boole, Frege e Peirce no século 19, mas sua linguagem se tornou mundialmente conhecida somente com o livro de Hilbert e Ackermann [Hilbert and Ackermann,1950], primeiramente publicado em 1928 mas só traduzido para o inglês em 1950 [Hodges,2001].

No inicio a lógica de primeira ordem foi utilizada para o estudo de argumentos dedutivos, mas hoje em dia ela é utilizada em várias aplicações, e.g. representação de conhecimento, como é o caso do NELL. Na lógica de primeira ordem as sentenças não são feitas para ter um significado e sim para expressar condições que podem ser satisfeitas ou não. Isto é feito em dois passos [Hodges,2001].

Primeiro a lógica de primeira ordem possui alguns símbolos chamados de constantes não lógicas, ou primitivas. Para se utilizar uma sentença φ de primeira ordem alguma coisa do mundo real (uma pessoa, um número, uma cor, etc.) está vinculada a cada uma das constantes desta sentença φ. O que os une é chamado de interpretação, estrutura ou avaliação [Hodges,2001].

Segundo, dada uma sentença de primeira ordem φ e uma interpretação I de φ, a semântica da lógica de primeira ordem determina se I faz com que φ seja verdadeira ou falsa. Caso I faça com que φ seja verdadeira, se diz que I satisfaz φ e o valor verdade da sentença φ sobre a interpretação I é verdade. Caso contrário, se I fizer com que φ não seja verdadeira (i.e. I não satisfaça φ) se diz que o valor verdade da sentença φ sobre a interpretação I é falsa [Hodges,2001].

Portanto, na LPO toda expressão é uma sentença, que representa um fato. Na LPO além de sentenças existem os termos, que representam objetos. Símbolos de constantes, variáveis e símbolos de funções são utilizados para construir termos e quantificadores, e símbolos de predicados ligados por conectivos são utilizados para construir sentenças. Uma explicação mais detalhada sobre cada elemento segue [Metakides,1996]:

- Símbolos de Constantes: *A, B, C, João, etc...*
    - Uma interpretação deve especificar qual objeto no mundo é referido por cada constante. Cada símbolo de constante nomeia exatamente um objeto, mas nem todos os objetos precisam ter um nome, e alguns podem ter vários nomes.
- Símbolos de predicados: *Irmão, Pai, etc...*
    - Uma interpretação especifica que um símbolo de predicado se refere a uma relação em particular. Por exemplo, o símbolo *Irmão* pode se referir a relação de irmandade. *Irmão* é um predicado com aridade 2 (i.e. possui dois argumentos) , e por consequência irmandade é uma relação entre pares de objetos. Em qualquer modelo, uma relação é definida por um conjunto de tuplas de objetos que a satisfazem. Uma tupla é uma coleção de objetos arranjados em uma ordem fixa, por exemplo: { (João,Pedro), (Pedro,João) }
- Símbolos de funções: *Cosseno, PaiDe, IrmãoDe, etc...*
    - Algumas relações são funcionais, isto é, um objeto é relacionado a exatamente outro objeto pela relação. Por exemplo, um ângulo tem somente um cosseno; uma pessoa tem somente uma pessoa como seu pai. Nestes casos, é mais conveniente definir um símbolo de função (e.g., *Cosseno*) que se refere à relação entre ângulos e números.
- Símbolos de conectivos: $\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$.
    - Negação: $\neg$;

- Implicação: $\Rightarrow$;
- E: $\wedge$;
- Ou: $\vee$;
- Se então: $\Leftrightarrow$.

Um termo é uma expressão lógica que se refere a um objeto. Portanto, símbolos de constantes são termos. Algumas vezes, é mais conveniente utilizar uma expressão para se referir a um objeto. Por exemplo, pode-se utilizar a expressão "A perna direita de João" ao invés de dar um nome a sua perna. Para isto existem os símbolos de funções, invés de utilizar um símbolo de constante, utiliza-se *PernaDireitaDe(João)* [Metakides,1996].

Uma sentença atômica é uma sentença formada por um símbolo de predicado seguido de uma lista de termos entre parênteses. Por exemplo, *Irmão(João,Pedro)* significa que o João é irmão do Pedro. Sentenças atômicas podem ter termos complexos como argumentos, como por exemplo, *Casados(PaiDe(João),MãeDe(Pedro))* que significa que o pai do João é casado com a Mãe do Pedro. Uma sentença atômica é verdadeira se a relação referida pelo símbolo de predicado existe entre os objetos referidos pelos argumentos [Metakides,1996].

Utilizando-se conectivos lógicos pode-se construir sentenças mais complexas, tal como em calculo proposicional. A semântica das sentenças formadas utilizando conectivos lógicos é idêntico a do calculo proposicional. Por exemplo:

- *Irmão(João,Pedro)* $\wedge$ *Irmão(Pedro,João).*
- *MaisVelho(João,30)* $\vee$ *MaisNovo(João,30).*
- $\neg$ *Irmão(João,José).*

Uma vez que exista uma lógica que permita objetos, é natural que se deseje expressar propriedades de coleções de objetos, ao invés de enumera-los. Os quantificadores permitem fazer isto. A lógica de primeira ordem contém dois quantificadores padrões, chamados universal ($\forall$) e existencial ($\exists$). Por exemplo:

- Para expressar a sentença "Todos os gatos são mamíferos" pode-se utilizar a sentença lógica "$\forall x$ *Gato(x)* $\rightarrow$ *Mamífero(x)*". $\forall$ é geralmente pronunciado "Para todo...".
- A sentença "O gato Spot possui um irmão que é um gato" pode ser expressa pela sentença lógica "$\exists x$ *Irmão(x,Spot)* $\wedge$ *Gato(x)*". $\exists$ é geralmente pronunciado "Existe...".

Para expressar sentenças mais complexas podem ser utilizados múltiplos quantificadores. O caso mais simples é quando os quantificadores são do mesmo tipo. Por exemplo, "Para todo x e todo y, se x é pai de y então y é filho de x" é expressa por "$\forall x,y$ *Pai(x,y)* $\rightarrow$ *Filho(y,x)*". $\forall x,y$ é equivalente a $\forall x\ \forall y$. Em outros casos podem haver combinações de quantificadores. Por exemplo, a sentença "Todo mundo ama alguém" pode ser expressa por "$\forall x\ \exists y$ *Ama(x,y)*".

### 2.3.1 Cláusulas de Horn

Da lógica de primeira ordem surgiu um modelo de cláusulas chamado cláusulas de Horn que vieram da necessidade de facilitar a manipulação computacional das cláusulas. Este modelo é o padrão utilizado por linguagens como o Prolog para a manipulação de sentenças lógicas, e também é o padrão utilizado pelo NELL para representar as regras de extração de conhecimentos criadas pelo componente RL, e por consequência também é o padrão utilizado pelo sistema proposto no presente trabalho.

Uma cláusula de Horn é uma fórmula na Forma Normal Conjuntiva (FNC) com apenas uma conclusão. Pode-se dizer que uma fórmula α está na FNC se α for uma conjunção $\beta_1 \wedge \beta_2 \wedge \beta_3 \wedge ... \wedge \beta_n$, em que cada $\beta_i$ $(1 \leq i \leq n)$ é uma cláusula, ou seja, é uma disjunção de literais ou um literal [Nicoletti,2009]. Então, se diz que uma fórmula está na FNC se [Nicoletti,2009]:

- Contém como conectivos apenas ∧,∨ e ¬;
- ¬ só opera sobre proposições atômicas, isto é, não tem alcance sobre ∧ e ∨;
- Não apresenta operadores de negação sucessivos como ¬¬;
- ∨ não tem alcance sobre ∧, ou seja, não existem expressões como p ∨ (q ∧ r).

O Prolog somente aceita cláusulas de Horn como forma de desenvolvimento de programas lógicos. Programa este que é um conjunto de cláusulas de Horn que realizam uma operação.

## 2.4 First-Order Inductive Learner

O *First-Order Inductive Learning* (FOIL) é um sistema para o aprendizado de definições em cláusulas de Horn de uma relação. A entrada do sistema consiste em informações sobre as relações, uma delas (a relação alvo) será definida como uma cláusula de Horn [Quinlan,1993].

O FOIL inicia com um conjunto de treinamento com exemplos positivos e negativos do que se quer encontrar, então começa criando uma regra com uma cláusula vazia no lado esquerdo e o predicado alvo do lado direito, posteriormente ele insere do lado direito literais que fazem com que a regra seja válida para um subgrupo de exemplos positivos. Caso a regra também seja válida para algum dos exemplos negativos a regra é restringida adicionando mais literais, esse processo é repetido até que a regra somente funcione para os exemplos positivos. Depois de feito isso esta cláusula é adicionada ao conjunto de cláusulas de solução e o subgrupo de regras que geraram esta cláusula é removido do conjunto de treinamento e o processo é reiniciado até que o conjunto de treinamento não possua mais exemplos positivos, então a saída é um conjunto de cláusulas Horn considerado lista de regras [Quinlan,1993].

O algoritmo FOIL é um algoritmo de aprendizagem de máquina supervisionada, pois ele necessita de um conjunto de treinamento. O FOIL tenta construir uma cobertura sucessiva das regiões dos dados que potencialmente implicam em um objetivo específico/predicado de saída. Como o FOIL funciona com lógica booleana binária (verdadeiro e falso), ele somente pode processar predicados booleanos. Isto significa que todas as variáveis devem ser representadas em um conjunto finito de predicados booleanos [Drobics,2003].

Como um exemplo de como o FOIL aprende uma relação, temos o dado por Nicoletti [Nicoletti,2011]. Considere o grafo dado na Figura 2.1, supondo que *node(A)* representa um nó no grafo e *arc(A,B)* representa um arco no grafo entre o nó A e o nó B temos como alvo a relação *path(A,B)* que o FOIL deve aprender com base no grafo da Figura 2.1 que lhe é fornecido.

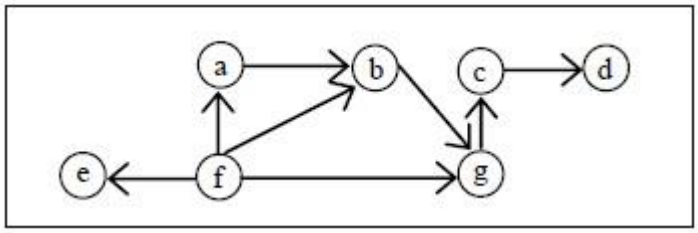


**Figura 2.1 - Um grafo definido por 7 nós e 8 arcos. Retirado de [Nicoletti,2011].**

Os dados de entrada para o FOIL mostrados na Figura 2.2 seguem a sintaxe necessária onde o cabeçalho de uma relação, que não é a relação alvo, começa com um '*' e consiste no nome da relação, tipos de argumento e chaves opcionais, seguidas por uma sequência de linhas contendo exemplos. Cada exemplo consiste de constantes separadas por vírgulas e devem aparecer em uma linha singular. Uma linha com um caractere '.' indica o fim da descrição da relação correspondente. Pode-se notar que na Figura 2.2 os exemplos negativos não foram dados.

| | | | | | |
|---|---|---|---|---|---|
| X: a,b,c,d,e,f,g. | a,g | f,c | *arc(X,X) | f,e | b |
| | b,c | f,d | a,b | f,g | c |
| path(X,X) | b,d | f,e | b,g | c,d | d |
| a,b | b,g | f,g | g,c | . | e |
| a,c | c,d | g,c | f,a | *node(X) | f |
| a,d | f,a | g,d | f,b | a | g |
| | f,b | . | | | . |

**Figura 2.2 - Entrada do FOIL para aprender a relação *path/2*. Retirado de [Nicoletti,2011].**

Considerando a entrada descrita na Figura 2.2, o FOIL induz as duas cláusulas a seguir que juntas fornecem uma descrição geral que pode ser utilizada para encontrar um caminho no grafo.

- path(A,B) :- arc(A,B).
- path(A,B) :- arc(A,C), path(C,B).

No método de identificação e eliminação de contradições proposto nesta dissertação, uma versão probabilística do FOIL foi implementada. A Seção 4.4.4 mostra mais detalhes desta versão probabilística de indução de regras de primeira ordem.

## 2.5 Read The Web

O projeto Read the Web[5] em desenvolvimento na Carnegie Mellon University desde 2008, tem como principal objetivo mostrar a viabilidade do paradigma de aprendizado sem fim. Para tanto, neste projeto foi proposto e implementado o primeiro sistema computacional

[5] http://rtw.ml.cmu.edu

de aprendizado sem fim. Tal sistema, chamado NELL (Never-Ending Language Learner), possui duas tarefas básicas a serem continuamente executadas, "extração" e "aprendizado":

Extração: extrair "conhecimento" a partir da Web visando à expansão da base de conhecimento inicial (ontologia inicial);

Aprendizado: aprender a extrair melhor e com mais precisão do que no 'dia anterior'.

O sistema atualmente consiste em cinco módulos (CPL, CSEAL, CMC, RL e MBL), que são descritos nas próximas subseções. Todos estes cinco componentes são independentes, mas interagem entre si para proporcionar uma melhoria no desempenho de cada um. A Figura 2.3 mostra o sistema como um todo.

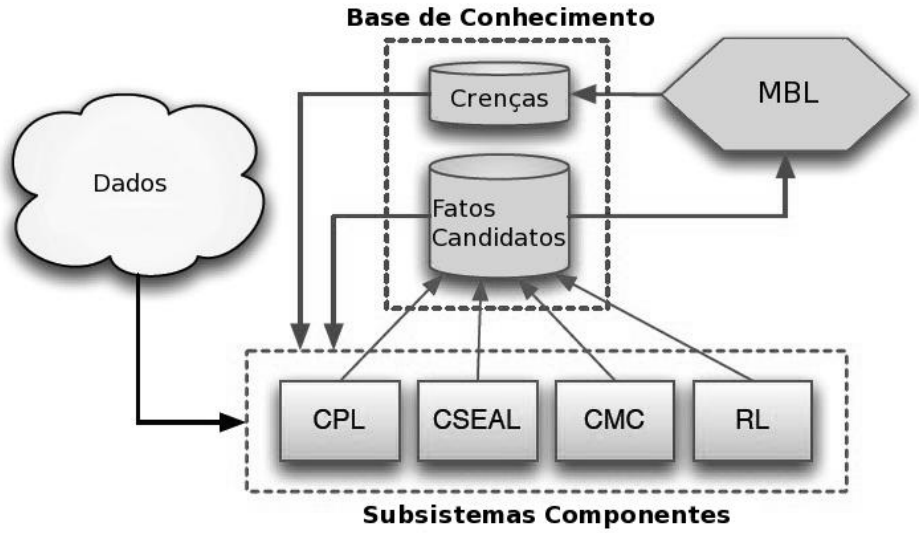


**Figura 2.3 - Projeto Read The Web. Adaptado de [Carlson et. al., 2010].**

Como pode ser visto na Figura 2.3 os componentes utilizam dados oriundos de páginas web ou de um conjunto de páginas web pré-processadas e geram fatos candidatos a crenças, o responsável por transformar estes fatos candidatos em crenças (promovê-los) é o componente MBL. A seguir é apresentada uma breve descrição de cada componente com base na abordagem dada em [Carlson et al., 2010].

### 2.5.1 CPL

O CPL (Coupled Pattern Learner) é um extrator de texto livre que aprende e utiliza padrões textuais, tais como "prefeito de X" e "X joga para Y" para extrair instâncias de categorias e relações. O CPL utiliza estatísticas de co-ocorrência entre substantivos (simples e compostos) e padrões textuais para aprender padrões de extração para cada predicado de

interesse e, então, utiliza esses padrões para encontrar instâncias adicionais de cada predicado. Na Tabela 2.9 podem ser vistos exemplos de padrões extraídos pelo CPL.

**Tabela 2.7 - Exemplo de padrões encontrados pelo CPL.**

| Predicado | Padrão |
|---|---|
| Emoção | Coração cheio de *X* |
| Bebida | Xícara de *X* |
| Time | Time *X* |
| TimeJogaNaLiga | *X* ficou em segundo no *Y* |
| AutorDoLivro | *X* escreveu *Y* |

Como pode ser visto na Tabela 2.9 para o predicado emoção o padrão encontrado pelo CPL foi “Coração cheio de *X*” onde *X* é a emoção, então quando o CPL encontrar frases como “Coração cheio de amor” ou “Coração cheio de ódio” ele classificará as palavras “amor” e “ódio” como emoções.

### 2.5.2 CSEAL

O CSEAL (Coupled SEAL) é um extrator semiestruturado que busca na web grupos de instâncias para cada categoria ou relação, e então busca listas e tabelas HTML para extrair novas instâncias do predicado correspondente. Assim como o CPL, o CSEAL utiliza relações de exclusão mútua para fornecer exemplos negativos ao processo de extração, que são utilizados para filtrar listas e tabelas muito gerais. Na Tabela 2.10 podem ser vistos exemplos de padrões extraídos pelo CSEAL.

**Tabela 2.8 - Exemplo de padrões extraídos pelo CSEAL.**

| Predicado | URL | Template de Extração |
|---|---|---|
| Campo Acadêmico | http://scholendow.ais.msu.edu/student/ScholSearch.Asp |  [X] - |
| Atleta | http://www.quotes-search.com/d%20occupation.aspx?o=+athlete | <a href=’d author.aspx?a=[X]’>- |
| Pássaro | http://www.michaelforsberg.com/stock.html | <option>[X]</option> |
| AutorDoLivro | http://lifebehindthecurve.com/ | </li> <li>[X] by [Y ] – |

### 2.5.3 CMC

O CMC (Coupled Morphological Classifier) é um conjunto de modelos de regressão logística que classificam substantivos (simples e compostos) com base nos vários recursos morfológicos (palavras, capitalizações, afixos, etc.). Os conhecimentos da base de conhecimento são utilizados como instâncias de treinamento. Assim como no CPL e no CSEAL, os relacionamentos de exclusão mútua são utilizados para identificar instâncias como exemplos negativos. O CMC examina os fatos candidatos dos outros componentes, e classifica até 30 novas instâncias por predicado, por iteração, com uma probabilidade mínima de 0.75. Essas medidas heurísticas ajudam a assegurar uma precisão alta, gerando mais evidências para fatos candidatos existentes e gerando restrições morfológicas para outros componentes. Na Tabela 2.11 podem ser observados exemplos de padrões encontrados pelo CMC.

**Tabela 2.9 - Exemplo de padrões encontrados pelo CMC.**

| Predicado | Prefixo/Sufixo |
|---|---|
| Religião | Sufixo=ismo |
| Universidade | Prefixo=Universidade |
| ArtistaMusical | Prefixo=Banda |
| ArtistaMusical | Prefixo=Grupo |

Como pode ser observado na Tabela 2.11 para o predicado religião o padrão encontrado foi o sufixo “ismo”, portanto quando o CMC encontrar palavras que terminam com “ismo”, tais como, budismo, hinduísmo, judaísmo, ele vai classificar estas palavras como sendo religiões.

### 2.5.4 RL

O RL (Rule Learner) é um algoritmo de aprendizagem relacional de primeira-ordem similar ao FOIL (descrito na Seção 2.4) que aprende cláusulas de Horn probabilísticas. Estas regras aprendidas são utilizadas para inferir novas instâncias a partir de relações que já estão na base de conhecimento [Carlson et. al. 2010]. Assim, o RL é o componente responsável por induzir regras de primeira ordem que o sistema utilizará para identificar padrões presentes na base de conhecimento. Essas regras são aprendidas e armazenadas na sintaxe Prolog. Na Tabela 2.12 podem ser observados exemplos de regras aprendidas pelo RL.

No NELL, o Rule Learner é implementado a partir de uma variação do FOIL (descrito na Seção 2.4). Uma implementação deste FOIL com base nos mesmo princípios foi realizada para os experimentos realizados neste trabalho (veja Seção 4.4.4). As entradas para o algoritmo do RL são conjuntos de exemplos positivos e negativos do consequente da regra a ser induzida. Assim, para a indução da regra R1:

R1: statelocatedincountry(x, y):- statehascapital(x, z), citylocatedincoutry(z, y).

seguindo o funcionamento do algoritmo FOIL, o RL precisa de um conjunto de exemplos positivos e um conjunto de exemplos negativos para o predicado consequente da regra statelocatedincountry(x, y). Tais conjuntos são obtidos com base nos predicados antecedentes statehascapital(x, z) e citylocatedincoutry(z, y). Ou seja, todos os pares (x,y) que satisfazem a regra R1 são incluídos no conjunto de exemplos positivos. Já os pares (x,y) que não satisfazem R1 são incluídos no conjunto de exemplos negativos.

Todas as regras são induzidas inicialmente como regras bem gerais, e progressivamente, o algoritmo vai especializando cada uma das regras de maneira que elas "possuam" um grande número de exemplos positivos e, ao mesmo tempo, um pequeno número de exemplos negativos. Após uma regra R ter sido gerada (para um dado consequente específico), os exemplos que satisfazem R são retirados da base de dados de treinamento e o algoritmo continua buscando mais regras para o consequente [Lao et al., 2011].

Para que o algoritmo possa tratar a incerteza associada ao processo de indução das regras, ao invés de se utilizar o princípio "crisp" do FOIL, o RL calcula uma probabilidade condicional P(consequente|antecedentes) usando uma priori de Dirichlet de acordo com a equação (2.5):

$$P = (N_+ + m * prior) / (N_+ + N_- + m) \quad (2.5)$$

onde $N_+$ é o número de instâncias do conjunto de treinamento que satisfazem a regra (exemplos positivos), $N_-$ é o número de instâncias do conjunto de treinamento que não satisfazem a regra (exemplos negativos), m é uma constante definida com o valor 5 (m = 5) e prior = 0.5.

**Tabela 2.10 - Exemplo de regras aprendidas pelo RL.**

| Prob. | Consequência | Antecedentes |
|---|---|---|

| | | |
|---|---|---|
| 0.95 | atletaPraticaEsporte(*X*,Basquete) | := atletaNaLiga(*X*,NBA) |
| 0.91 | timeJogaNaLiga(*X*,NHL) | := timeGanhouTroféu(*X*,Stanley Cup) |
| 0.90 | atletaNaLiga(*X*,*Y*) | := atletaJogaParaTime(*X*,*Z*), timeJogaNaLiga(*Z*,*Y*) |
| 0.88 | cidadeNoEstado(*X*,*Y*) | := cidadeCapitalDoEstado(*X*,*Y*), cidadeNoPais(*X*,USA) |

Nem todas as regras aprendidas pelo RL são relações funcionais. Entretanto, como será visto no Capítulo 4, para ser útil para um dos métodos de identificação e eliminação de contradições (proposto neste trabalho – Seção 4.3.3) as regras precisam representar relações funcionais.

***Definição 1:*** *Uma relação funcional é um predicado binário do tipo "predicado(arg1,arg2)" em que para cada valor de arg1 só pode haver um único valor de arg2.*

### 2.5.5 MBL

O MBL (Meta-Bootstrap Learner) tem a função de acoplar o aprendizado de múltiplas técnicas de extração utilizando uma restrição múltipla que requer que todas as técnicas concordem sobre a validade de uma determinada instância. Em outras palavras o MBL verifica se os componentes aprenderam o mesmo conceito para poder dizer que este conceito é valido.

Os algoritmos subordinados ao MBL são o CSEAL, o CPL, o CMC e o RL, que quando utilizados com o MBL, os algoritmos subordinados não promovem instâncias por conta própria. Ao invés, eles reportam as evidências sobre cada fato candidato a conhecimento para o MBL, e o MBL fica responsável por promover estas instâncias.

***Definição 2:*** *Promover um fato significa inseri-lo na base de conhecimento de maneira que ele seja considerado verdadeiro com um grau de certeza.*

Utilizando um método de combinação simples (descrito em [Carlson,2010] na seção 4.4 onde a implementação é descrita), o MBL promove qualquer instância que seja recomendada por ambas as técnicas desde que obedeçam as restrições de exclusão mútua e de checagem de tipo especificadas na ontologia.

## 2.6 Considerações Finais

Neste capítulo foram mostrados os conceitos fundamentais para o entendimento do trabalho passando pelos tipos de aprendizagem, onde alguns algoritmos utilizados pelo presente trabalho foram discutidos, o conceito de lógica o FOIL e por fim o projeto RTW que é o projeto no qual o sistema criado irá ser inserido.

# Capítulo 3

# TRABALHOS CORRELATOS

*Neste capítulo são abordados alguns trabalhos correlatos importantes para o entendimento do que tem sido pesquisado nesta área de detecção de contradições e importantes para o presente trabalho.*

Na área de detecção de contradições foram encontrados vários artigos sendo que são descritos aqui alguns dos que estão mais próximos do domínio desta proposta de mestrado. Pôde-se perceber que o foco da detecção de contradições atualmente é muito voltado para um tipo de sistema chamado de Questioning Answering, onde um sistema computacional responde perguntas feitas por uma pessoa e para isso é importante detectar contradições nas perguntas. Outro fato observado é que este tema é uma linha de pesquisa bastante ativa, pois muito material recente foi encontrado.

### 3.1.1 Contradições

Na área de contradições foram encontrados alguns trabalhos recentes, dentre eles os trabalhos de [Harabagiu,2006], [Sanchez-Graillet and Poesio,2007], [Voorhees,2008], [Ritter et al.,2008], [Marneffe, 2008], [Lembo, 2010]. Nesta seção são discutidos os trabalhos de [Marneffe, 2008] e [Lembo, 2010], pois são os mais recentes encontrados e traduzem bem duas visões distintas da definição de contradições.

Na literatura as contradições são definidas de várias maneiras, dependendo do contexto em que se está trabalhando. No contexto da lógica tem-se a definição dada no trabalho de [Marneffe, 2008], onde é dito que duas sentenças A e B são contraditórias se não há possibilidade em que A e B sejam simultaneamente verdadeiras, mas ambas aparecem individualmente como verdadeiras.

O seu trabalho tem como objetivo propor uma definição de contradição apropriada para tarefas de PLN (processamento de linguagem natural) e também propõe construir uma tipologia de contradições a partir de um corpus disponível. O ponto importante para o presente trabalho é a definição dos diversos tipos de contradições, onde um destes tipos são as contradições provenientes de negações, que é um conceito utilizado na detecção de contradições proposta pelo presente trabalho.

Já [Lembo, 2010] separa as contradições em dois tipos, contradições no TBox e no ABox. Na lógica descritiva uma base de conhecimento é constituída de dois componentes, chamados TBox e ABox. O TBox contém os axiomas sancionando propriedades gerais dos conceitos e relações (tais como, Cão é_um Animal), enquanto que o ABox contém axiomas afirmando propriedades de instâncias de conceitos e relações (tais como, Bob é uma instância de Cão).

Portanto as contradições no TBox são contradições apresentadas por relações e conceitos erroneamente aprendidos (tais como, Cão é_um Automóvel) e as contradições no ABox são contradições apresentadas por instâncias de conceitos e relações aprendidos de maneira errada (tais como, "Michael Jordan" é uma instância de Futebolista).

Este último trabalho foi importante na diferenciação dos tipos de contradições que seriam encontrados e no presente trabalho ambos os tipo de contradição são tratados, no caso do ABox o método é descrito e visto na seção 4.3, mas se trata da utilização de regras em clausula de Horn para a identificação destas contradições e no caso do TBox são utilizados dois métodos também vistos na seção 4.3 que consistem basicamente na verificação da exclusividade mútua e de relações funcionais.

### 3.1.2 Detecção de Contradições com Base em Dicionários

Na área detecção de contradições com base em dicionários, dentre os trabalhos mais relevantes pode-se mencionar a abordagem de [Banko et. al.,2007], [Banko and Etzioni,2008], [Ritter et al.,2008], [Lin,2009]. Nesta seção são discutidos os trabalhos de [Ritter et al., 2008] e [Lin, 2009], pois além de serem os mais atuais esses dois artigos se baseiam nos outros artigos encontrados, assim sendo, eles trazem uma ideia mais completa da aplicação deste tipo de detecção de contradições incorporando algumas ideias transmitidas pelos outros autores.

No artigo de [Ritter et al.,2008] o autor mostra um sistema de detecção de contradições baseado em relações funcionais (e.g., nascido_em(Pessoa) := Lugar) chamado

AuContraire, que automaticamente descobre frases que denotam as relações funcionais com alta precisão.

Este sistema funciona primeiramente identificando o que os autores chamam de "frases funcionais" estatisticamente, as quais estão relacionadas ao conceito de "relações funcionais". Na sequência, o sistema utiliza estas frases para automaticamente criar um grande conjunto de possíveis contradições, por fim o sistema passa por este grupo e identifica as reais contradições utilizando conhecimento sobre sinônimos, merônimos, tipos de argumentos e ambiguidade trazidos de dicionários, tais como, wordnet e gazetteers.

Ao invés de analisar as sentenças diretamente o AuContraire deixa esta tarefa para o TextRunner Open Information Extraction System [Banko et al., 2007; Banko and Etzioni, 2008] para mapear cada sentença a uma ou mais tuplas que representam entidades nas sentenças ou relações entre elas (i.e., nascido_em(Mozart,Salzburg)), o que simplifica a tarefa de detecção de contradições.

Já no artigo de [Lin,2009] é tratada uma nova abordagem baseada na proposta do AuContraire para melhorar a detecção de contradições incorporando informações adicionais do contexto que melhorem a precisão dessa detecção.

O autor diz que o desafio central para o sistema de CD (Contradiction Detection) é determinar qual relação é funcional em um texto. Como já visto uma relação R é funcional na variável x se ela mapear somente um valor y. Em textos esta é uma situação complexa devido à ambiguidade, polissemia e outros fenômenos linguísticos.

O autor demonstra o sistema desenvolvido por ele que é um sistema de CD baseado no AuContraire, sistema do artigo de [Ritter,2008], onde a principal diferença entre os dois sistemas é que, na proposta mais recente, existe um filtro para as relações que remove as relações com entidades ambíguas.

Destes artigos foi possível compreender como as relações funcionais podem gerar contradições, bem como identificar estes tipos de relações e uma maneira de resolvê-las. Este é um ponto importante pois parte da proposta do presente trabalho de mestrado é detectar estes tipos de contradições. A diferença entre o que foi feito por estes autores e o que é proposto no presente trabalho é que neste trabalho não serão utilizados dicionários para identificar as contradições.

O motivo da não utilização de dicionários é que como o RTW é um sistema que aprende infinitamente e automaticamente, irá chegar uma hora onde todo o conhecimento do dicionário já estará contido na base de conhecimento do RTW, assim sendo, em um determinado momento o dicionário se torna obsoleto para este propósito.

### 3.1.3 Detecção de Contradições em Textos

Na área de detecção de contradições em textos foram encontrados vários trabalhos, dentre eles os trabalhos de [Harabagiu,2006], [Banko et. al.,2007], [Banko and Etzioni,2008], [Ennals,2010], [Kawahara,2010]. Nesta seção são discutidos os trabalhos de [Harabagiu,2006] e [Ennals,2010] pois esses dois artigos foram considerados os mais relevantes para o presente trabalho.

No artigo de [Harabagiu,2006] é descrito um framework para reconhecimento de contradições entre múltiplas fontes de texto, se utilizando de três formas de informação linguística: (a) negação; (b) antônimos; e (c) semântica. Neste artigo a autora cria um sistema de vínculo textual que deriva informações linguísticas de pares de textos que são dados ao sistema como entrada e a partir destes textos e destas informações encontra contradições entre eles. Essas contradições são encontradas de duas maneiras:

- Primeira: Contradições são reconhecidas identificando e removendo negações de proposições e testando os vínculos textuais.
- Segunda: As contradições são reconhecidas derivando as informações linguísticas dos textos de entrada, incluindo informações que identifiquem negações e antônimos e utiliza estas informações como entrada de um classificador que dirá se estas informações caracterizam uma contradição.

Esta abordagem atinge uma precisão de 62% na identificação de contradições. Este artigo é interessante pois aborda o tema da detecção automática de negações e antônimos sem a necessidade de um dicionário como nos artigos anteriores o que se assemelha mais com a proposta do presente trabalho.

Mas a não utilização de um método semelhante pelo presente trabalho se dá pois os algoritmos do NELL extraem informações de textos aleatórios e muitas vezes não existem dois textos sobre o mesmo assunto no momento de uma extração para que estes textos possam ser comparados e assim verificar se existem contradições.

Já no trabalho de [Ennals,2010], é descrito um sistema chamado Dispute Finder, que alerta o usuário enquanto este navega na internet que a informação que ele está visualizando contradiz uma informação de uma fonte mais confiável. O Dispute Finder examina o texto na página que está sendo exibida e destaca qualquer frase que seja contraditória com alguma outra frase de uma fonte confiável. Então caso o usuário clique na frase em destaque o Dispute Finder  mostra a lista de artigos que contradizem a frase em questão.

Este sistema constrói uma base de dados de afirmações contraditórias buscando em sites da web que já mantenham uma lista de afirmações contraditórias e também permitindo ao usuário cadastrar afirmações que ele acredite que sejam contraditórias. O Dispute Finder identifica os pedaços de texto utilizando um algoritmo de vínculo textual simples que é executado dentro do navegador.

Entretanto esta abordagem não foi escolhida para ser utilizada pelo presente trabalho pois o proposto é que o NELL seja um sistema autônomo e não dependa de outras fontes de informação para ponderar sobre seu conhecimento.

## 3.2 Considerações Finais

Neste capítulo alguns trabalhos correlatos foram discutidos, onde foi possível observar o que já foi realizado na área de detecção de contradições e as tendências desta área, bem como onde os trabalhos encontrados foram úteis para o presente trabalho.

# Capítulo 4

# DETECÇÃO E ELIMINAÇÃO DE CONTRADIÇÕES NO NELL

*Neste capítulo são apresentados os métodos propostos, a implementação destes métodos e a experimentação realizada com a finalidade de validação dos mesmos.*

## 4.1 Considerações Iniciais

Neste capítulo é apresentada a proposta de trabalho de mestrado desenvolvida. Também é apresentada a metodologia onde são detalhadas as tarefas propostas e como elas foram realizadas para se alcançar os objetivos descritos na Seção 1.2. Por fim a implementação dos métodos propostos é demonstrada bem como a experimentação destes métodos.

## 4.2 Detectando e Eliminando Contradições no NELL

Como previamente descrito, a proposta deste trabalho é investigar, implementar e avaliar métodos que permitam identificar contradições na base de conhecimento do NELL, e assim, auxiliar no processo de autorreflexão e autossupervisão deste sistema de aprendizado sem fim.

Com base nas características específicas da base de conhecimento do NELL os métodos de detecção e eliminação de contradição foram definidos com base nos seguintes temas específicos de investigação:

1. Detecção de contradições geradas por falhas no aprendizado de instâncias de relações com base no conceito de exemplos negativos nas regras de primeira ordem;
2. Eliminação das contradições encontradas no passo um através da criação de novas categorias para expandir a base de conhecimento original.
3. Detecção de contradições geradas por falhas no aprendizado de instâncias de categorias com base no conceito de exclusividade mútua;
4. Eliminação das contradições encontradas no passo três através da exclusão das instâncias envolvidas na contradição;
5. Detecção de contradições geradas por falhas no aprendizado de instâncias de relações com base no conceito de relações funcionais;
6. Eliminação das contradições encontradas no passo cinco através da exclusão das instâncias envolvidas na contradição;

## 4.3 Métodos Propostos

Para viabilizar a investigação e os testes dos métodos de detecção e correção de contradições foi utilizada uma versão já implementada do NELL. Nesta versão foram incorporados novos componentes permitindo a integração dos resultados obtidos neste trabalho de pesquisa ao NELL. Assim, foi realizada a investigação, definição e implementação de algoritmos que acessam e manipulam a base de conhecimento do NELL, podendo inserir, alterar e remover instâncias de categorias e relações. Além disso, a investigação e o estudo da literatura tiveram papel importante durante todo o desenvolvimento da pesquisa.

Tendo como foco os seis temas definidos na seção anterior foram desenvolvidos três métodos de detecção de contradições e a cada um deles há uma forma de eliminação das contradições encontradas. As próximas três subseções abordam cada um dos métodos propostos.

### 4.3.1 Detecção de contradições com base no conceito de exemplos negativos nas regras de primeira ordem.

Para a realização da etapa descrita no item um (identificação de contradições geradas por falhas no aprendizado de instâncias de relações com base no conceito de exemplos negativos nas regras de primeira ordem) da proposta, foram utilizadas as regras de primeira ordem presentes na base de conhecimento do NELL (geradas pelo componente RL - veja Seção 2.5.4), onde existem os predicados que compõem cada regra. Com esses predicados é possível identificar contradições quando se induz uma determinada regra através dos casos onde os dados da base de conhecimento contradizem a regra que esta sendo induzida. Estes casos são então tratados como possíveis candidatos à contradições.

Após serem identificados os candidatos a contradições na base de conhecimento do NELL, foram explorados métodos que possibilitam a eliminação, se possível, de todas elas. Vale ressaltar que nem todas as contradições identificadas podem ser corrigidas imediatamente. Assim, para os casos onde não havia uma maneira de se corrigir uma contradição, os fatos que envolviam tal contradição foram removidos da base de conhecimento. Inicialmente a eliminação de contradições está vinculada com a criação de novas categorias na base de conhecimento original. Esta abordagem foi utilizada para eliminar as contradições detectadas através dos exemplos negativos das regras de primeira ordem induzidas pelo componente RL do NELL. Por se tratar da abordagem mais complexa, para este tipo de eliminação de contradições foi desenvolvido um método mais elaborado como se segue.

O processo inicia-se com a utilização das regras induzidas pelo RL para se extrair da base de conhecimento as instâncias consideradas falsos positivos. Para tanto, as regras são submetidas a uma “engine Prolog” que gera dois conjuntos de instâncias para cada regra. O primeiro conjunto contém as instâncias positivas e o segundo conjunto contém as instâncias que são falsas (instâncias negativas).

Com estes conjuntos já definidos as regras são analisadas uma a uma. As informações sobre as contradições são reunidas e estes dados então são submetidos ao algoritmo de agrupamento EM. Como já mencionado, em um primeiro momento o algoritmo de agrupamento utilizado foi o K-Means, mas para se obter resultados mais satisfatórios optou-se por trocá-lo pelo algoritmo EM, com um número de grupos variável.

Depois de agrupados os dados em conjuntos homogêneos (conjuntos que contenham somente casos contraditórios ou casos não contraditórios), esses grupos são analisados e os grupos homogêneos de dados contraditórios são inseridos como uma nova subcategoria atrelada a categoria original. Esta nova categoria é então inserida na base de conhecimento e o NELL é reiniciado para que possa aprender a partir da base modificada com as novas subcategorias.

Com isso o NELL pode aprender a diferença que esses dados possuem e eliminar estas contradições. Um exemplo real (extraído do NELL) pode ser ilustrado através da regra "citylocatedinstate(?x,?y) :- citycapitalofstate(?x,?y), statelocatedincountry(?y,united_states).". A interpretação da regra seria: "uma cidade X está localizada no estado Y se esta cidade X é a capital do estado Y e o estado Y está localizado nos Estados Unidos". Essa regra gerava algumas contradições para cidades que estavam fora dos Estados Unidos (i.e. "citylocatedinstate(sao_paulo, sao_paulo) :- citycapitalofstate(sao_paulo, sao_paulo), statelocatedincountry(sao_paulo,brazil)."), então o ideal é que fossem criadas, além da categoria cidade, as subcategorias "cidades dos Estados Unidos" e "cidades fora dos Estados Unidos". Desta maneira, o RL poderia automaticamente criar uma nova regra onde não houvesse contradição alguma. Isto exemplifica como a criação de novas categorias pode auxiliar na eliminação de contradições.

Na abordagem empregada, a questão da nomenclatura das subcategorias é uma questão indiferente ao sistema, essa nomenclatura poderia ser útil para o entendimento humano, mas é dispensável para o funcionamento do método proposto. Portanto, uma vez que foram criadas duas subcategorias a partir de uma categoria, elas podem significar qualquer coisa para os humanos, mas para o sistema são simplesmente dois conjuntos de elementos que tem algo em comum.

Outro ponto importante é o fato de que ao criar por exemplo, as subcategorias "cidades dos Estados Unidos" e "cidades fora dos Estados Unidos", nada impediria que o sistema em uma outra iteração resolvesse dividir a subcategoria "cidades fora dos Estados Unidos" em outras duas subcategorias, i.e. "cidades do Brasil" e "cidades fora dos Estados Unidos e Brasil". Assim sendo de uma maneira geral o algoritmo se especifica ao decorrer do processamento, e após algumas iterações pode abranger todas as possibilidades dentro de uma categoria. Portanto esse é um processo de melhoria contínua.

### 4.3.2 Detecção de contradições com base no conceito de exclusividade mútua.

Para a realização da etapa descrita no item 3 (identificação de contradições geradas por falhas no aprendizado de instâncias de categorias com base no conceito de exclusividade mútua) da proposta, foram utilizados os conceitos de "exclusividade mútua" previamente definidos na base de conhecimento do NELL. Duas categorias A e B são ditas mutuamente exclusivas se instâncias da categoria A podem ser consideradas exemplos negativos de instâncias da categoria B e vice-versa. Assim, se "cachorro" é uma instância da categoria "animal", "São Carlos" é uma instância da categoria "cidade", e ainda, "animal" e "cidade" são categorias mutuamente exclusivas, pode-se dizer que "cachorro" é um exemplo negativo de instância da categoria "cidade", assim como, "São Carlos" é um exemplo negativo de instância da categoria "animal". Com base neste conceito de exclusividade mútua, contradições podem ser encontradas na base de conhecimento do NELL.

A base de conhecimento do NELL possui as informações sobre quais as categorias são mutuamente exclusivas, então com uma varredura na base de conhecimento é possível encontrar as instâncias que estão em duas categorias que são mutuamente exclusivas, assim caracterizando uma contradição. Para a eliminação dessa contradição, todas as instâncias que representem contradições dessas categorias serão removidas da base para que o NELL possa aprender novamente esses conceitos de uma maneira correta.

### 4.3.3 Detecção de contradições com base no conceito de relações funcionais

Para a realização da etapa descrita no item 5 (identificação de contradições geradas por falhas no aprendizado de instâncias de relações com base no conceito de relações funcionais) da proposta, foram utilizados os conceitos de relações funcionais previamente definidos na base de conhecimento do NELL. Como já foi definido, em uma relação funcional "predicado(X,Y)", para cada X deve haver um único Y válido. Os casos onde, para um X específico existe mais de um valor possível para Y podem ser considerados como contradições.

A base de conhecimento do NELL possui as informações sobre quais as categorias são funcionais, então com uma varredura na base de conhecimento é possível encontrar as instâncias X que são mapeadas para mais de um Y, assim caracterizando uma contradição. Para a eliminação dessa contradição, todas as instâncias que representem contradições dessas

categorias serão removidas da base para que o NELL possa aprender novamente esses conceitos de uma maneira correta.

Para efeitos de validação dos resultados, os novos componentes de detecção e eliminação de contradições foram testados em diferentes situações, demonstradas na seção 4.4, onde os experimentos são descritos e analisados. Os resultados obtidos em todos os experimentos foram analisados comparativamente, assim sendo possível verificar se a eliminação de contradições é capaz de auxiliar no processo de aprendizado sem fim do NELL.

## 4.4 Implementação

Como pode ser visto na Seção 4.3, dos 3 métodos de detecção e eliminação de contradições propostos, o primeiro é mais elaborado e exige mais atenção na implementação. Já os dois últimos métodos propostos são mais simples e suas implementações não apresentam grandes desafios. As subseções seguintes apresentam a forma de implementação de cada um dos três métodos propostos.

### 4.4.1 Detecção de contradições com base no conceito de exemplos negativos nas regras de primeira ordem.

De uma maneira geral, o método proposto (descrito na Seção 4.3.1) para a detecção de contradições através das regras do RL foi implementado da seguinte maneira. Para cada uma das regras geradas pelo RL, são obtidos os exemplos que satisfazem a regra (chamados exemplos positivos), bem como os exemplos que não satisfazem a regra (chamados exemplos negativos).

Para a identificação dos exemplos positivos, é utilizada uma “engine Prolog” que, dada uma regra e o universo de domínio (que é a base de conhecimento do NELL, neste caso) retorna uma lista de instâncias que satisfazem a regra. Já para a identificação dos exemplos negativos, antes de se invocar a “engine PROLOG” é necessário a criação de uma nova regra R’ que seja uma versão modificada da regra original R. Neste sentido, R’ deve ser definida de tal forma que o conjunto de todas as instâncias que a satisfaçam seja exatamente igual ao conjunto de todas as instâncias que não satisfazem a regra original R. Suponha por exemplo a Regra1 dada abaixo.

Regra1: athletePlaysSport(x,Basketball) :- athletePlaysInLeague(x,NBA).

Para esta regra, a versão modificada Regra1’ seria:

Regra1’: athletePlaysSport(x,Basketball) :- athletePlaysInLeague(x,y), y/==NBA.

Ou seja, as instâncias que satisfazem Regra1 formam o conjunto de exemplos positivos desta regra, já as instâncias que satisfazem Regra1’ formam o conjunto de exemplos negativos de Regra1. O processo de criação da versão modificada da regra é facilmente automatizado através da inserção de um novo antecedente que exige que os argumentos da regra não obedeçam a regra original. De uma maneira geral, para uma dada regra R:

R: predicate1 (x,A) :- predicate2(x,B).

A versão modificada R’ sera dada por:

R’: predicate1(x,A):-predicate2(x,y), y/==B.

Para todas as regras onde há instâncias que não satisfazem a regra original (ou seja, o conjunto de exemplos negativos não é vazio) são buscadas na base todas as informações que se tem sobre estas instâncias (tanto positivas quando negativas) independente de quais sejam estas informações. Estas informações por sua vez são fornecidas como entrada do algoritmo EM (Expectation Maximization) de agrupamento. O objetivo do agrupamento é tentar identificar se existe um padrão nos dados contraditórios ou se os dados são contraditórios por causa de uma falha (ou erro) no aprendizado. Se um grupo é encontrado contendo apenas os exemplos negativos, então o sistema identifica a possibilidade de maior especificação da categoria existente na base de conhecimento. Assim, é sugerida a criação de duas novas subcategorias vinculadas à categoria original. Neste sentido, os dados contraditórios (exemplos negativos) são colocados em uma das novas subcategorias e os exemplos positivos são colocados na outra nova subcategoria. Desta forma, numa próxima iteração, quando o RL passar por esses dados novamente, ele pode aprender mais sobre essas subcategorias e com este novo aprendizado as contradições podem ser resolvidas.

A implementação deste método foi realizada utilizando a linguagem de programação Java e a interface de desenvolvimento Eclipse, e foi implementado como um módulo extra ao NELL, funcionando independentemente. A "engine PROLOG" utilizada na implementação realizada se chama tuProlog [APICe,2007] e é um projeto da Universidade de Bologna. A Figura 4.1 mostra o fluxograma que descreve o método.

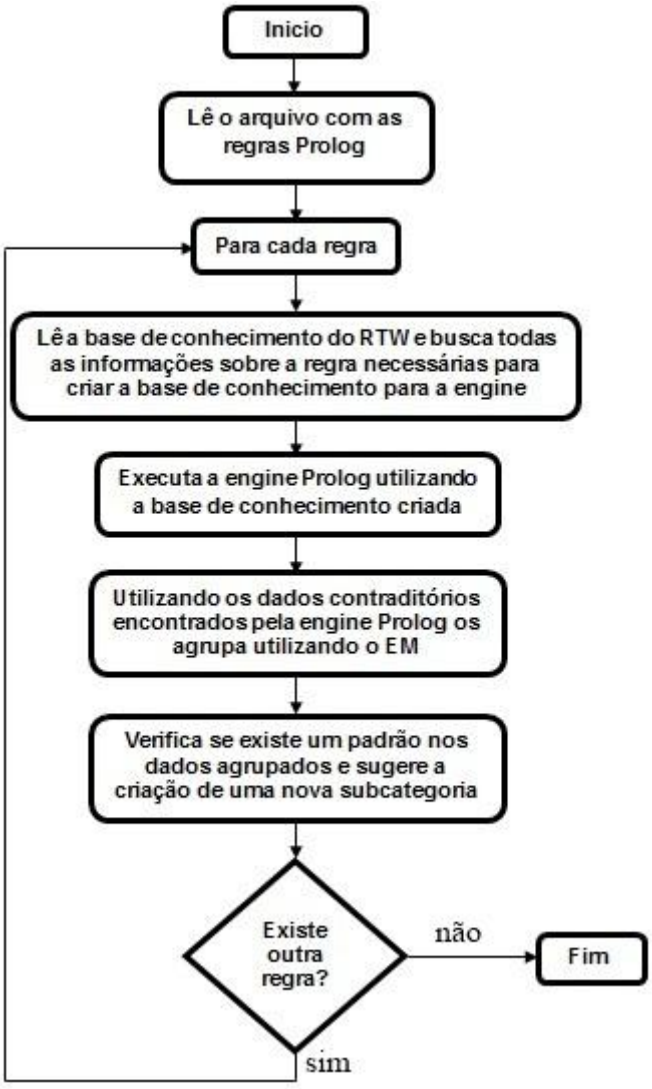


**Figura 4.1 - Fluxograma do sistema.**

Um ponto importante a ser mencionado é que o algoritmo de agrupamento (EM) implementado foi alterado para que fosse mais adequado para o problema em questão. Esta alteração envolve dois pontos principais:

1. Ao invés de se utilizar o modelo probabilístico tradicional, esta versão do EM executa um classificador Naive-Bayes para estimar P($y|x$) (assim como mostrado na seção 2.2.2.2);

2. Um novo método de identificação do número de grupos ótimo foi desenvolvido e acoplado ao EM. Assim, ao invés de se executar o EM uma única vez com um número de grupos ótimo já definido, o novo critério desenvolvido consiste na análise do resultado do agrupamento, onde enquanto o algoritmo não encontra um grupo homogêneo (um grupo com somente casos negativos ou somente com casos positivos) ou enquanto um número máximo de tentativas não for atingido o algoritmo não para a execução, sempre aumentando o número de grupos a cada iteração.

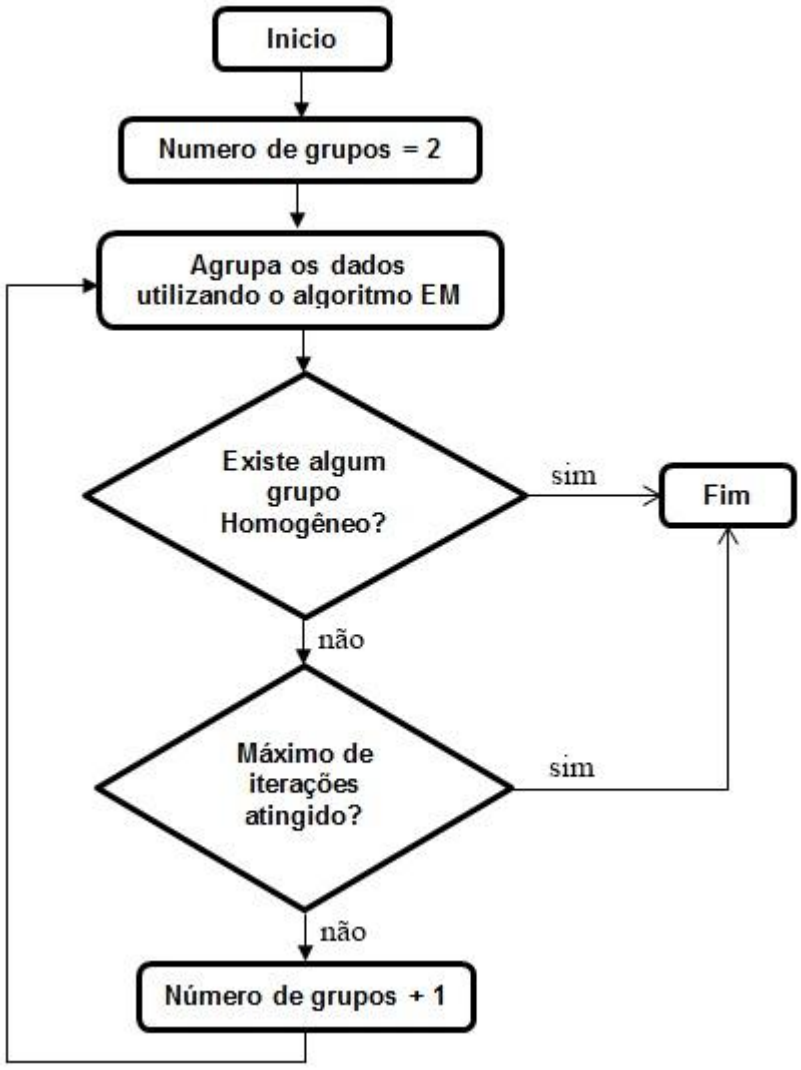


**Figura 4.2 - Fluxograma do EM.**

A motivação para a primeira modificação implementada no EM é minimizar o esforço computacional exigido pelo método. Experimentos realizados utilizando-se a implementação tradicional do EM e a versão com base no Naive-Bayes mostraram que, para o problema específico de agrupamento abordado neste trabalho, não havia diferença significativa quanto à

precisão dos resultados, por isso o modelo modificado foi definido como padrão do método proposto.

Já a segunda inovação inserida na implementação do EM foi motivada pela característica específica do problema em questão. Na busca por um grupo homogêneo de exemplos positivos (ou negativos) não é possível saber de antemão qual deve ser o número de grupos que o algoritmo de agrupamento deve buscar. Assim, haviam duas alternativas a serem seguidas: a primeira seria a utilização de um algoritmo de agrupamento que identifica automaticamente o melhor número de grupos para um determinado conjunto de dados. E a segunda alternativa seria a criação de um algoritmo capaz de identificar automaticamente o melhor número de grupos. A maior restrição a se utilizar a primeira alternativa está vinculada ao fato de que os algoritmos disponíveis na literatura (e que identificam automaticamente o melhor número de grupos) são desenvolvidos com medidas de "otimalidade de grupos" que não se enquadram com o propósito deste trabalho. Ou seja, no problema de identificação de contradições na base de conhecimento do NELL é possível saber de antemão quais dados (instâncias) devem estar agrupados em um mesmo grupo, bem como, quais devem estar em grupos diferentes. Mas, não se sabe ao certo quantos grupos realmente existem nos dados. Por este motivo, esta segunda modificação foi inserida no EM.

Devido à inserção da modificação do critério de parada do algoritmo é possível perceber que o EM em questão (desenvolvido neste trabalho) é um algoritmo semissupervisionado pois ele é parte de um algoritmo que conhece quais deveriam ser os grupos, e este algoritmo supervisor é o responsável por aumentar o número de grupos para o EM sempre que necessário. Está é uma nova abordagem do algoritmo, que foi desenvolvida por se adequar melhor ao problema em questão, onde o número de grupos não era sabido, mas a situação ideal sim, deixando assim o algoritmo mais flexível.

### 4.4.2 Detecção de contradições com base no conceito de exclusividade mútua.

Para a detecção de contradições com base no conceito de exclusividade mútua foi criado um algoritmo que verifica para todos os fatos da base de conhecimento se alguns desses fatos infringem o conceito de exclusividade mútua, i.e. caso sejam encontrados os fatos city(São_Carlos) e animal(São_Carlos) o algoritmo identifica isso como sendo uma contradição e remove este fato da base de dados. Assim, a implementação deste método é bastante simples, mas sua utilidade é bastante alta em um sistema de aprendizado sem fim.

O propósito da remoção destes fatos contraditórios da base é que o NELL possa, em uma iteração futura, aprender novamente estes conceitos e assim aprendê-los de uma maneira correta. Como com o passar do tempo o NELL agrega mais conhecimento a sua base de conhecimento, em uma iteração mais tardia ele pode aprender um fato com mais precisão que em suas iterações iniciais, pois como já foi demonstrado ele também utiliza informações de sua base para melhorar sua capacidade de aprendizado.

### 4.4.3 Detecção de contradições com base no conceito de relações funcionais

Para a detecção de contradições com base no conceito de relações funcionais foi criado um algoritmo que verifica para todos os fatos da base de conhecimento se algum desses fatos infringem esse conceito, i.e. caso sejam encontrados os fatos teamPlaysInLeague(Cardinals,NFL) e teamPlaysInLeague(Cardinals,NHL) e a relação teamPlaysInLeague seja funcional, então o algoritmo identifica isso como sendo uma contradição e remove este fato da base de dados.

Assim como ocorre na remoção das contradições por exclusividade mútua, o propósito da remoção de fatos que ferem o princípio das relações funcionais é que o NELL possa aprender novamente estes conceitos, numa iteração futura, e assim aprendê-los de uma maneira correta. Como com o passar do tempo o NELL agrega mais conhecimento a sua base de conhecimento, em uma iteração mais tardia ele pode aprender um fato com mais precisão que em suas iterações iniciais, pois como já foi demonstrado ele também utiliza informações de sua base para agregar novos conhecimentos.

### 4.4.4 FOIL Probabilístico

Para a verificação do funcionamento do algoritmo de detecção de contradições com base no conceito de exemplos negativos nas regras de primeira ordem uma versão do FOIL probabilístico foi também implementada. Esse FOIL é similar ao original [Quinlan,1993] com a diferença de ser probabilístico. Isso significa que ele não é tão rígido quando o FOIL original que corta a regra caso ela não esteja 100% correta. Já no caso do FOIL probabilístico, as regras são consideradas corretas caso tenham mais de 75% de acerto.

A sua implementação é baseada na implementação do FOIL vista na seção 2.4 do presente trabalho, mas um cálculo de probabilidade é inserido ao verificar se a regra deve ser aceita ou não, conforme utilizado no NELL [Lao et al., 2011]. Uma probabilidade condicional é estimada utilizando uma priori de Dirichlet conforme pode ser observado na Seção 2.5.4. Como resultado o FOIL probabilístico aprende regras de uma maneira mais flexível que o FOIL tradicional auxiliando no tratamento da incerteza intrínseca a este problema.

## 4.5 Experimentação

Seguindo a estrutura metodológica que dividiu a abordagem de detecção e eliminação de contradições em 3 métodos distintos, os experimentos realizados também foram divididos em três etapas distintas.

Para todos os experimentos a base de conhecimento utilizada foi a mesma, a qual foi extraída após a iteração 140 do NELL. Deve-se salientar que como pode ser visto no website do NELL (http://rtw.ml.cmu.edu) esta versão da base de dados (iteração 140) recebeu uma supervisão humana superficial para corrigir algumas inconsistências e erros, portanto não era esperado um grande número de contradições nestes experimentos.

### 4.5.1 Detecção de contradições com base no conceito de exemplos negativos nas regras de primeira ordem

Para o primeiro método (Seção 4.4.1), utilizando a base de dados citada acima e as regras criadas pelo RL, foram encontradas 6 regras com contradições e um total de 11 novas subcategorias foram propostas. As 6 regras onde contradições foram encontradas são:

1. “citylocatedinstate(?x,?y) :- citycapitalofstate(?x,?y), statelocatedincountry(?y,united_states).”
2. “athleteplayssport(?x,?y) :- athleteplaysforteam(?x,?z), teamplayssport(?z,?y).”
3. “athleteplaysforteam(?x,?y) :- teammate(?x,?z), athleteplaysforteam(?z,?y).”
4. “athleteplaysinleague(?x,?y) :- athleteplaysforteam(?x,?z), teamplaysinleague(?z,?y).”
5. “teamplaysincity(?x,?y) :- teamhomestadium(?x,?z), stadiumlocatedincity(?z, ?y).”
6. “teamplaysinleague(?x,?y) :- teamplaysagainstteam(?x,?z), teamplaysinleague(?z,?y).”

Para a primeira regra houve a criação das subcategorias “Cidades nos Estados Unidos” e “Cidades fora dos Estados Unidos”, o que resolveu as contradições. Para a segunda regra tiveram que ser criadas três subcategorias, “Jogadores da MLB”, “Jogadores da NHL” e “Outros Jogadores”, isso reduziu o número de contradições em 85,5%, mas para isso o algoritmo teve de ser executado duas vezes. Para a terceira regra o sistema não conseguiu encontrar um padrão para resolver as contradições. No caso da quarta regra ela também teve a maior parte das contradições resolvidas com a criação das subcategorias “Jogadores da MBL”, “Jogadores da NHL” e “Outros Jogadores” onde o número de contradições diminuiu em 75,3%. Na penúltima regra o sistema não conseguiu identificar um padrão nas contradições, portanto não conseguiu sugerir novas subcategorias. Já na última regra a criação das subcategorias “Times da MLB”, “Times da NHL” e “Outros Times” acarretaram na eliminação de 75% das contradições em duas execuções do algoritmo.

Analisando estes resultados pôde-se observar que provavelmente os dados utilizados para aprender os jogadores e times das ligas de baseball e hóquei não estavam muito consistentes, pois foram nestes itens onde a maioria das contradições foram encontradas. Outra observação importante é que essas novas subcategorias mostram onde a base de conhecimento está deficitária, e deve ser melhorada, seja por conhecimento errado ou por falta de informações. Outro caminho a ser trilhado é a sugestão da remoção de regras que contenham muitas informações contraditórias como é o caso da terceira e da quinta regra onde o sistema não conseguiu determinar um padrão para as contradições destas regras portanto não conseguiu resolver estas contradições.

Buscando-se mais evidências empíricas da validade do método proposto, buscou-se verificar se as correções propostas pelo método resolveriam as contradições numa próxima execução do algoritmo RL na próxima iteração do NELL. Neste sentido, um ambiente simulando o ambiente real foi criado (utilizando-se a versão do FOIL probabilístico implementada (veja Seção 4.4.4) para este trabalho). Assim, a saída do método proposto foi utilizada para realimentar a base de dados e o FOIL probabilístico (simulando o RL) foi novamente executado para verificar se novas regras foram criadas ainda continham contradições. Uma das regras onde foi observada a criação de uma nova regra para corrigir as contradições foi a regra já discutida anteriormente “citylocatedinstate(?x,?y) :- citycapitalofstate(?x,?y), statelocatedincountry(?y,united_states).”. Após a execução de detecção e eliminação de contradição, foram criadas então duas novas subcategorias contendo as cidades dos estados unidos e outra subcategoria contendo as cidades fora dos estados

unidos. Depois de simular a execução do RL novamente, pôde ser observada a criação de uma nova regra e a modificação da regra já existente como segue: “citylocatedinstate(?x,?y) :- citycapitalofstate(?x,?y), statelocatedincountry(?y,?z).” onde Y pertence a subcategoria de cidades fora dos estados unidos e Z a categoria de países e “citylocatedinstate(?x, ?y) :- citycapitalofstate(?x, ?y), statelocatedincountry (?y, united_states)” onde Y pertence a subcategoria de cidades dentro dos estados unidos. Assim, pode-se notar que as novas subcategorias criadas são capazes de eliminar as contradições anteriormente existentes.

Ainda com o intuito de exploração das características do método proposto, mais um experimento foi realizado. Neste novo experimento, uma bateria de testes foi realizada para verificar até onde o algoritmo consegue encontrar as contradições. Assim, foram inseridos erros artificiais na base para configurar situações onde 20%, 30%, 40% e 50% do total de instâncias são contraditórias em uma determinada categoria da base de conhecimento. Este experimento tráz evidências empíricas sobre a eficácia do algoritmo em uma base com muitos erros.

A primeira regra escolhida para esta experimentação foi a regra “citylocatedinstate(?x,?y) :- citycapitalofstate(?x,?y), statelocatedincountry(?y,united_states).” Pois em um ambiente normal o sistema proposto conseguiu resolver todas as contradições.

Com 20% de erros na base de conhecimento, o algoritmo soluciona todos os erros para esta regra. Aumentando o número de erros para 30%, o algoritmo se comportou bem e conseguiu eliminar 82% das contradições. Para uma base com 40% de erros, o algoritmo conseguiu eliminar 67.5% das contradições e para uma base com 50% de erros, o algoritmo não conseguiu identificar um padrão nos dados contraditórios.

A segunda regra na qual o algoritmo foi executado nas mesmas situações da primeira regra foi a regra “athleteplayssport(?x,?y) :- athleteplaysforteam(?x,?z), teamplayssport(?z,?y).”. O resultado foi que para 20% de erros na base houve uma redução de 84% no número de contradições. Para 30% de erros na base, o algoritmo conseguiu solucionar 73% das contradições. Já para 40% de erros na base, o algoritmo conseguiu eliminar 61% das contradições e para 50% de erros o algoritmo não conseguiu encontrar um padrão nos dados.

A terceira regra que foi utilizada para executar o algoritmo é a regra “athleteplaysinleague(?x,?y) :- athleteplaysforteam(?x,?z), teamplaysinleague(?z,?y).”. O resultado foi que para 20% de erros na base, tivemos uma redução de 83.5% no número de contradições. Para 30% de erros na base, o algoritmo conseguiu solucionar 69% das

contradições, já para 40% de erros na base o algoritmo não conseguiu encontrar um padrão para os dados, portanto não conseguiu solucionar as contradições existentes.

A quarta regra na qual o algoritmo foi executado foi a regra “teamplaysinleague(?x,?y) :- teamplaysagainstteam(?x,?z), teamplaysinleague(?z,?y).” onde o sistema teve o pior desempenho. O resultado foi que para 20% de erros na base tivemos uma redução de 86% no número de contradições. Para 30% de erros na base o algoritmo conseguiu solucionar 71% das contradições, já para 40% de erros na base o algoritmo não conseguiu encontrar um padrão para os dados e consequentemente não conseguiu solucionar as contradições existentes.

Para as duas regras que o sistema não havia conseguido identificar um padrão nos dados contraditórios um pré-processamento manual mais complexo teve de ser realizado para ser possível a execução nesta fase da experimentação, como o número de contradições nas duas regras era muito grande estas contradições tiveram que ser identificadas e removidas a mão para ser possível chegar a um nível de 20% de contradições para a determinada regra e a partir daí crescer este número de contradições.

As duas regras nas quais o sistema não conseguiu identificar um padrão nos dados contraditórios foram “athleteplaysforteam(X,Y) :- teammate(X,Z), athleteplaysforteam(Z,Y).” e “teamplaysincity(?x,?y) :- teamhomestadium(?x,?z), stadiumlocatedincity(?z, ?y).”.

Para a primeira destas regras, com 20% de erros na base, houve uma redução de 82% no número de contradições. Para 30% de erros na base o algoritmo conseguiu solucionar 69% das contradições, já para 40% de erros na base o algoritmo não conseguiu encontrar um padrão para os dados e consequentemente não conseguiu solucionar as contradições existentes.

Para a segunda destas regras com 20% de erros na base houve uma redução de 79% no número de contradições. Para 30% de erros na base o algoritmo conseguiu solucionar 65% das contradições, já para 40% de erros na base o algoritmo não conseguiu encontrar um padrão para os dados e consequentemente não conseguiu solucionar as contradições existentes.

Além das regras que tinham contradições, acima citadas, foram também submetidas ao sistema outras 12 regras que não continham contradições para aumentar a confiabilidade dos resultados deste experimento que simula a presença de contradições através da inserção de contradições artificiais. Estas doze regras e seus respectivos resultados podem ser vistos na Tabela 4.1.

**Tabela 4.1 - Teste com regras sem contradições.**

| Regra | 20% | 30% | 40% | 50% |
|---|---|---|---|---|
| athleteplaysinleague(?x,mlb) :- athleteplaysforteam(?x,mets). | 100% | 83.5% | 68% | - |
| teamplaysinleague(?x,nfl) :- teamplayssport(?x,football). | 89% | 76% | 66% | - |
| teamplayssport(?x,basketball) :- teamplaysagainstteam(?x,raptors). | 82.5% | 71% | - | - |
| teamplaysinleague(?x,nba) :- teamwontrophy(?x, nba_championship). | 87% | 77% | 64.3% | - |
| athleteplayssport(?x,?y) :- teammate(?x,?z), athleteplayssport(?z,?y). | 84% | 69% | - | - |
| athleteplayssport(?x,baseball) :- athleteplaysforteam(?x, phillies). | 89% | 74% | 68% | - |
| teamplaysincity(?x,?y) :- stadiumhometeam(?x,?z), stadiumlocatedincity(?z,?y). | 85% | 67% | - | - |
| teamplaysinleague(?x,nfl) :- teamplaysagainstteam(?x,eagles), teamwontrophy(?x,super_bowl). | 83% | 66% | - | - |
| teamplaysinleague(?x,nfl) :- teamplayssport(?x,football), teamplaysagainstteam(?x,texans). | 84% | 65% | - | - |
| stadiumlocatedincity(?x,?y) :- stadiumhometeam(?x,?z), teamplaysincity(?z,?y). | 83% | 67% | - | - |
| teamplayssport(?x,football) :- teamplaysagainstteam(?x,packers). | 84% | 69% | - | - |
| teamplayssport(?x,basketball) :- teamwontrophy(?x,nba_championship). | 88% | 76% | 64% | - |

A Tabela 4.1 mostra que, na média o método proposto conseguiu eliminar 87% das contradições quando há 20% de dados ruidosos (contradições) na base de conhecimento. Em situações onde há 30% de contradições na base de conhecimento, o método eliminou em média 72% delas. Com o aumento da quantidade de contradições, o método encontra dificuldades em detectar e eliminar os erros da base de conhecimento. Assim, num cenário onde a base de conhecimento continha 40% de contradições, o método conseguiu detectar e eliminar apenas 28% delas em média. Já para uma situação onde há 50% de contradições o

método não foi capaz de encontrar e eliminar contradição alguma. A Figura 4.3 mostra graficamente um resumo dos resultados obtidos para a simulação de contradições nas 12 regras.

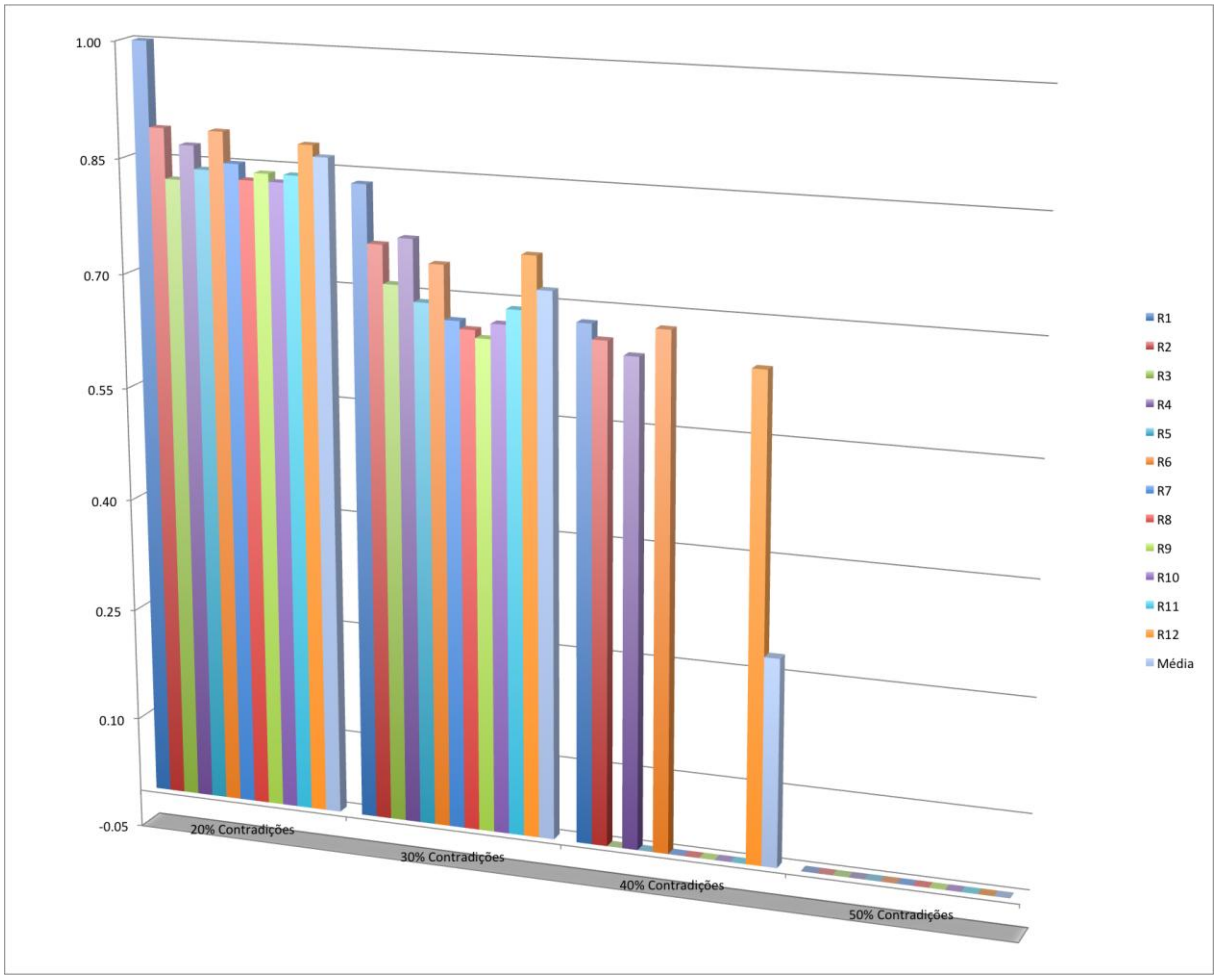


**Figura 4.3 - Gráfico da Experimentação.**

Apesar do demonstrado acima, como já dito anteriormente, o NELL possui em média 20% de erros em sua base de conhecimento, portanto o método proposto se mostra adequado na detecção e solução das contradições deste sistema de aprendizado sem fim. Outro ponto importante a ser considerado é que o método proposto tende a melhorar o funcionamento do NELL na medida que as contradições vão sendo solucionadas. Além disso, com uma taxa menor de contradições em uma determinada regra o método proposto tem uma taxa maior de acerto pois consegue determinar o padrão das contradições, e os outros métodos de detecção e eliminação de contradição aqui propostos ajudam a minimizar esse número de contradições pois removem os dados contraditórios da base de acordo com os requisitos de exclusividade mútua e relações funcionais, o que também caracterizam contradições para o primeiro método.

Uma última experimentação foi realizada para testar a eficiência deste método quando o algoritmo de agrupamento EM é trocado pelo tradicional algoritmo de agrupamento K-Means. Os algoritmos utilizados para esta experimentação são o EM (implementado com Naive Bayes) e o K-Means tradicional, e as regras utilizadas são as mesmas que geraram contradições e puderam ser resolvidas na primeira experimentação.

A primeira regra utilizada foi a regra "citylocatedinstate(?x,?y) :- citycapitalofstate(?x,?y), statelocatedincountry(?y,united_states).", para ela o número de grupos para ambos os métodos foi de 4 grupos e os tempos de execução foram de 3,4segundos para o EM e de 0,03 segundos para o K-means

A segunda regra na qual os algoritmos foram executados foi a regra "athleteplayssport(?x,?y) :- athleteplaysforteam(?x,?z), teamplayssport(?z,?y)", para ela o número de grupos para ambos os métodos foi de 4 grupos e os tempos de execução foram de 1,95 segundos para o EM e de 0,01 segundos para o K-means.

A terceira regra que foi utilizada para executar os algoritmos é a regra "athleteplaysinleague(?x,?y) :- athleteplaysforteam(?x,?z), teamplaysinleague(?z,?y).", para ela o número de grupos para o EM foi de 3 grupos e para o K-Means foi de 4 grupos e os tempos de execução foram de 1,86 segundos para o EM e de 0,02 segundos para o K-means.

A quarta regra na qual o algoritmo foi executado foi a regra "teamplaysinleague(?x,?y) :- teamplaysagainstteam(?x,?z), teamplaysinleague(?z,?y).", para ela o número de grupos para ambos os métodos foi de 3 grupos e os tempos de execução foram de 13,86 segundos para o EM e de 0,06 segundos para o K-means.

Embora seja evidente que o K-means é mais rápido que o EM, as diferenças de tempo de execução não são decisivas para o algoritmo descrito neste trabalho, pois mesmo na pior situação o EM demorou apenas 13 segundos para executar. Outro ponto que pode ser observado é que os resultados são bem similares, onde o EM superou o K-means em somente uma regra no quesito agrupamento. Mas o fator decisivo na escolha do EM como método de agrupamento foi a possibilidade de posteriormente utilizar a probabilidade calculada por ele para permitir que seja possível utilizar grupos menos rígidos quanto a homogeneidade dos mesmos. Assim se um elemento tiver em sua probabilidade apenas uma pequena diferença entre dois grupos pode ser possível permitir que este elemento entre no mais adequado, pois sabemos os rótulos dos dados previamente.

### 4.5.2 Detecção de contradições com base no conceito de exclusividade mútua.

Para este experimento, foi executado um algoritmo que "vasculha" a base de conhecimento do NELL em busca de instâncias que pertençam a mais de uma categoria e estas categorias sejam mutualmente exclusivas. Nenhuma contradição foi encontrada, mas como foi mencionado anteriormente, não foi uma grande surpresa, pois como foi dito esta base de conhecimento passou por supervisão humana.

Seguindo a ideia de inserir contradições artificiais na base de conhecimento para se obter evidências empíricas sobre a capacidade do método em detectar e corrigir contradições, foram inseridos 30 erros, tais como, sportsTeam(Cardinals) e athlete(Cardinals), male(Sam_Acho) e female(Sam_Acho), etc. O algoritmo verificou que a exclusividade mútua foi violada e encontrou todos os erros inseridos.

### 4.5.3 Detecção de contradições com base no conceito de relações funcionais

Na execução deste experimento, foram encontradas contradições em 3 relações funcionais diferentes, a primeira é a relação acquiredBy(Company,Company) que significa que uma empresa foi comprada por outra empresa, uma das instâncias contraditórias é acquiredBy(cbs, viacom) e acquiredBy(cbs, sony). Portanto, o algoritmo sugeriu a remoção destas duas instâncias. A segunda é a relação stadiumhometeam que significa que um estádio pertence a um time. Para esta relação, como exemplo de contradições foi encontrado stadiumhometeam(staples_center,la_lakers) e stadiumhometeam(staples_center,lakers), que aparentemente está correto mas para o sistema la_lakers e lakers são coisas diferentes, por isso foi considerada contradição. A última relação na qual foram encontradas contradições é ceoof que diz quem é o CEO (Chief Executive Officer) de uma empresa, um exemplo de contradição encontrada foi ceoof(charlie_ergen, echostar) e ceoof (charles_ergen, echostar).

Na última experimentação realizada para este método, foram inseridos na base 30 fatos contraditórios tais como, athleteplayssport(Sam_Acho,Baseball) e athleteplayssport(Sam_Acho, Football), teamplaysinleague(Cardinals, NFL) e teamplaysinleague(Cardinals, NBA), athleteplaysinleague(Michael_Jordan, NFL) e athleteplaysinleague(Michael_Jordan, NBA). Assim como ocorreu para o segundo método, todos estas contradições inseridas artificialmente foram encontradas e apagadas da base de conhecimento.

## 4.6 Considerações Finais

Com base nos resultados obtidos nos experimentos apresentados, pode-se observar que os métodos propostos puderam identificar e eliminar contradições presentes na base de conhecimento do NELL. Além disso, contradições inseridas artificialmente na base de conhecimento também foram identificadas e eliminadas. Mais especificamente, o método de detecção de contradições com base no conceito de exemplos negativos nas regras de primeira ordem, se mostrou bastante adequado em situações onde o volume de contradições presentes na base de conhecimento não é superior a 30%. Os métodos com base no conceito de exclusividade mútua e relações funcionais apesar de serem simples, se mostraram bastante eficazes. Além disso, os três métodos, quando utilizados em conjunto, podem gerar uma sinergia que resulte na melhora do desempenho de um modo geral.

# Capítulo 5

# CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação apresentou um estudo investigativo sobre detecção e eliminação de contradições na base de conhecimento do primeiro sistema de aprendizado de máquina sem fim já construído, o NELL. A detecção e eliminação de contradições é uma tarefa muito importante em sistemas baseados em conhecimento porque permite que a base de conhecimento possa ser corrigida e revisada automaticamente. Em sistemas de aprendizado de máquina sem fim, onde o conteúdo da base de conhecimento é automaticamente expandido e utilizado para guiar a sequencia do processo de aprendizado automático, detectar e eliminar contradições passa a ser ainda mais importante e relevante, pois tal tarefa pode evitar a propagação de erros.

Durante o trabalho de pesquisa, foram investigados vários temas os quais deram suporte para a proposta, desenvolvimento e implementação de um método de detecção com base em algoritmos de aprendizado de máquina e lógica de primeira ordem. Além deste método, foram também implementados, testados e avaliados dois mecanismos simples capazes de identificar e eliminar contradições com base simplesmente na estrutura ontológica da base de conhecimento do NELL (onde foram exploradas as relações funcionais e de exclusividade mútua).

No processo investigativo, foram estudados o aprendizado de máquina supervisionado, não supervisionado e semissupervisionado, bem como exploradas as características de métodos de aprendizado sem fim. Além disso, a lógica de primeira ordem também foi pesquisada, com foco maior em processos de indução de regras.

Dentre os algoritmos de aprendizado de máquina supervisionados, o Naive-Bayes foi implementado, e serviu de base para o método proposto. Considerando-se os algoritmos de aprendizado não supervisionado, tanto o K-Means quanto o EM (Expectation Maximization) foram investigados e implementados. A versão implementada do K-Means seguiu a definição clássica deste algoritmo, já para o algoritmo EM, duas versões foram implementadas: a primeira com base no modelo probabilístico tradicional; e a segunda na qual o algoritmo Naive-Bayes é utilizado para tornar o processo mais rápido e menos custoso (com relação ao esforço computacional exigido). Assim, o EM (que em sua versão clássica possui complexidade exponencial) com base no Naive-Bayes pode ser executado com complexidade linear (com base no número de atributos e no número de exemplos de treinamento) em cada iteração. O algoritmos EM implementado foi então utilizado na implementação do algoritmo de aprendizado semissupervisionado presente no método de identificação e eliminação de contradições proposto na Seção 4.4.1 desta dissertação.

Dentre os algoritmos de indução de regras de primeira ordem, uma versão clássica e uma versão probabilísticas do FOIL foram investigadas e implementadas. Para a implementação de ambas as versões foi utilizada uma engine Prolog chamada tuProlog. A versão probabilística do FOIL foi então utilizada na implementação da principal contribuição deste trabalho de mestrado (veja Seção 4.4.1).

Com base nas investigações de algoritmos e técnicas relacionadas à detecção de contradições, e tendo também como base as pesquisas e estudos sobre características específicas do sistema de aprendizado sem fim NELL, foram propostas 3 abordagens diferentes para tratar esta questão de detecção e eliminação de contradições na base de conhecimento do NELL. Duas destas abordagens são bastante simples e apenas replicam ideias já descritas na literatura e que não exigem grande complexidade de projeto e implementação. Estas duas abordagens simples são às baseadas em relações funcionais e em relações de exclusividade mútua (descritas nas Seções 4.4.2 e 4.4.3 respectivamente). Experimentos realizados com estas duas abordagens simples mostraram que elas são eficientes para a detecção de contradições. A metodologia utilizada para a eliminação destas contradições detectadas foi também bastante simplista, ou seja, fatos contraditórios foram excluídos da base de conhecimento. A ideia de simplesmente se excluir da base de conhecimento os fatos considerados contraditórios, mesmo sendo ingênua, torna-se mais justificável quando a base de conhecimento em questão é continuamente atualizada num processo de aprendizado sem fim. Neste contexto, como a base de conhecimento do NELL segue sendo atualizada constantemente, e considerando-se ainda que conhecimentos “errados”

(ou contraditórios) tendem a deteriorar a qualidade da tarefa de aprendizado, a eliminação de contradições através da remoção das mesmas pode ser considerada bastante eficiente neste contexto.

A terceira abordagem proposta neste trabalho, para detecção e eliminação de contradições na base de conhecimento do NELL, é também a maior contribuição deste trabalho de pesquisa. Nesta abordagem (veja Seção 4.4.1), um método de aprendizado semissupervisionado é proposto através da utilização de um algoritmo de agrupamentos (o EM) de maneira iterativa. A ideia central do método é identificar grupos homogêneos que contenham apenas exemplos positivos, ou apenas exemplos negativos para uma dada regra de primeira ordem. Assim, o método se inicia buscando gerar dois grupos distintos, caso não haja homogeneidade, é iniciada uma nova iteração com o número de grupos incrementado. O processo todo é supervisionado através da identificação prévia (e automática, através de inferências na base de conhecimento) de quais fatos são exemplos positivos e quais são exemplos negativos. Quando um grupo homogêneo é identificado, o método cria duas subcategorias na base de conhecimento do NELL e as contradições existentes tendem e ser eliminadas. Resultados obtidos nos experimentos realizados mostraram que o método proposto é eficiente quando a quantidade de contradições (presentes na base de conhecimento) é inferior a 30%.

Os resultados obtidos foram satisfatórios e os métodos propostos são promissores. Na sequência dos trabalhos vinculados a esta pesquisa os métodos propostos deverão ser incorporados ao NELL para que possam contribuir de maneira efetiva na autossupervisão do sistema e na expansão automática da ontologia.

Nos experimentos realizados neste trabalho os métodos propostos não foram efetivamente incorporados ao NELL por conta de algumas modificações no formato da base de conhecimento do sistema de aprendizado sem fim. Por este motivo, todos os experimentos foram realizado utilizando-se um ambiente especialmente implementado para simular o funcionamento do NELL e uma base de conhecimento fixa extraída após a iteração 140 do NELL. Na continuidade dos trabalhos, os métodos propostos serão adaptados ao novo formato do NELL para que a incorporação possa ocorrer.

# REFERÊNCIAS


[APICe,2007] APICe. tuProlog Guide. Disponível em: http://tuprolog.sourceforge.net/doc/2p-guide.pdf. Acesso em: 15/06/2011. Bologna, Itália, Abril 2007.

[Alpaydin,2010] Ethem Alpaydin. Introduction to Machine Learning, 2nd ed. MIT press. 2010. 579p.

[Banko et. al.,2007] Michele Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open Information extraction from the Web. In Proceedings of IJCAI. Hyderabad, India, January 2007.

[Banko and Etzioni,2008] Michele Banko and O. Etzioni. 2008. The tradeoffs between traditional and open relation extraction. In Proceedings of ACL.Columbus, USA, June 2008.

[Bastos Filho, 2003] Jenner Barretto Bastos Filho. 2003. O Que é Contradição? Algumas de suas possíveis Acepções.Caderno Brasileiro do Ensino de Física, Número 20, volume 2, p. 194-227.

[Blum and Mitchell, 1998] Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training.In: COLT. (1998)

[Byrnes, 1999] John Byrnes. Proof Search and Normal Forms in Natural Deduction. PhD thesis, Department of Philosophy, Carnegie Mellon University, May, 1999.

[Carlson et al., 2009] A. Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka Jr. and T.M. Mitchell. Coupled Semi-Supervised Learning for Information Extraction. In Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM), 2010

[Carlson, 2010] Andrew Carlson. Coupled Semi-Supervised Learning. Carnegie Mellon PhD. Thesis, Pittsburgh, USA, May 2010.

[Carlson et al., 2010] Andrew Carlson et. al. Toward An Architecture for Never-Ending Language Learning. In The Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, USA, July 2010.

[Chen,1999] Zhengxin Chen. Computational Intelligence For Decision Support. CRC Press. 1999. 389p.

[Cortes and Vapnik,1995] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. In Machine Learning Journal, volume 20, issue 3, pages 273-297. Kluwer Academic Publishers, 1995.

[Crammer and Singer,2001] Koby Crammer and Yoran Singer. Ultraconservative online algorithms for multiclass problems. In Proceedings of COLT-2001, Amsterdam, Netherlands, July 2001.

[Dempster,1977] Arthur P. Dempster, Nan M. Laird and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. In the Journal of the Royal Statistical Society, volume 39, pages 1-38. Royal Statistical Society, 1977.

[Drobics,2003] Mario Drobics, Ulrich Bodenhofer and Erich Peter Klemen. FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions. In The International Journal of Approximate Reasoning, volume 32, pages 131-152. Elsevier, 2003.

[Duarte et al., 2011] Duarte, MC and Hruschka Jr., ER and Nicoletti, M.C. Minimização do Impacto do Problema de Desvio de Conceito por Meio de Acoplamento em Ambiente de Aprendizado Sem Fim. In: Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology - STIL2011, 2011.

[Duda and Hart, 1973] Duda, R. O. and Hart, P. E. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[Ennals,2010] Rob Ennals, Beth Trushkowsky and John Mark Agosta. Highlighling Disputed Claims on the Web. In the World Wide Web Conference. Pages 341-350. Raleigh, USA, April 2010.

[Frosini, 2009] P. Frosini, Does intelligence imply contradiction? Cognitive Systems Research, Volume 10, Issue 4, December 2009, Pages 297-315.

[Galvão and Hruschka,2009] Sebastian D. C. de O. Galvão and Estevam R. Hruscka Jr. Estudo Orientado: Um Sistema de Aprendizado Sem-Fim Para Recomendação de Filmes. Relatório Técnico do Laboratório de Aprendizado de Máquina (MALL) do Departamento de Computação da UFSCar. São Carlos, Brasil, Julho 2009. 37p.

[Gruber,1993] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. In The Knowledge Acquisition Journal. volume 5, pages 199-220. Academic Press Ltd, London, UK, 1993.

[Gruber,2009] Thomas R. Gruber. Ontology. Encyclopedia of Database Systems. Pages 1963-1965. Springer, New York, USA, 2009.

[Harabagiu,2006] Sanda Harabagiu, Andrew Hickl and Finley Lacatusu. Negation, Contrast and Contradiction in Text Processing. In Proceedings of the 21st national conference on Artificial Intelligence. volume 1, pages 755-762. AAAI Press, Boston, USA, July 2006.

[Haykin,2001] Simon Haykin. Redes Neurais: Princípios e Prática. 2ª Edição. Porto Alegre: Bookman, 2001. 900p.

[Hilbert and Ackermann,1950] Hilbert, D. and Ackermann, W. 1950. Grundzuge der Theoretische Logik (1928); English translation Principles of Mathematical Logic, R. E. Luce, ed., Chelsea Publishing Company, New York.

[Hodges,2001] Wilfrid, Hodges. Classical Logic I: First-Order Logic. In Guide to Philosophical Logic, ed. Louis Goble, Blackwell, Malden Mass. 2001.

[John and Langley,1995] George H. John and Pat Langley. Estimating Continuous Distributions in Bayesian Classifiers. In Proceedings of the Eleventh Conference on Unvertainty in Atificial Intelligence. Pages 338-345. Morgan Kaufmann Publishers, San Mateo, 1995.

[Johnson,1967] Johnson S.C. Hierarchical Clustering Schemes. In the Psychometrika Journal, volume 32, pages 241-254. The Psychometric Society, 1967.

[Kawahara,2010] Daisuke Kawahara, Kentaro Inui and Sadao Kurohashi. Identifying Contradictory and Contrastive Relations between Statements to Outline Web Information on a Given Topic. In The 23rd International Conference on Computational Linguistics. Pages 534-542. Beijing, China, August 2010.

[Lao et al., 2011] N. Lao, T.M. Mitchell, W.W. Cohen. Random Walk Inference and Learning in A Large Scale Knowledge Base. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2011.

[Lembo,2010] Domenico Lembo et. al. Inconsistency-tolerant Semantics for Description Logics. In The Fourth International Conference on Web Reasoning and Rule Systems, Bressanone, Italy, September 2010.

[Lin,2009] Yuqi Lin. A New Approach to Improve CD Based On The Context. In The First International Workshop on Education Technology and Computer Science, Wuhan, China, March 2009.

[MacQueen,1967] J.B. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. In Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability. Pages 281-297. University of California Press, Berkeley, 1967.

[Marneffe,2008] Marie-Catherine de Marneffe, Anna N. Rafferty and Christopher D. Manning. Finding Contradictions in Text. In The 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Columbus, USA, June 2008.

[Metakides,1996] George Metakides and Anil Nerode. Principles of Logic and Logic Programming. volume 13. North-Holland. 1996. 329p.

[Mitchell,1997] Tom M. Mitchell. Machine Learning. McGraw-Hill Science/Engineering/Math, 1997. 421p.

[Mitchell et al., 2009] T.M. Mitchell, J.Betteridge, A. Carlson, E.R. Hruschka Jr. and R.C. Wang. Populating the Semantic Web by Macro-Reading Internet Text. Invited Paper, In Proceedings of the International Semantic Web Conference (ISWC), 2009

[Nicoletti,2009] Maria do Carmo Nicoletti. A Cartilha da Lógica, 2 ed. EdUFSCar, 2009. 233p.

[Nicoletti et al., 2011] Maria do Carmo Nicoletti, Flávia O. S. de Sá Lisboa, Estevam Rafael Hruschka Jr. Learning Temporal Interval Relations Using Inductive Logic Programming. In the First International Conference on Integrated Computing Technology, São Carlos, Brazil, June 2011.

[Nilsson,1997] Nils John Nilsson. Introduction to Machine Learning: An Early Draft of a Proposed TextBook. 1997. Stanford University, Stanford, Estados Unidos.

[Quinlan,1986] J.R. Quinlan. Induction of Decision Trees. In Machine Learning Journal, volume 1, issue 1, pages 81-106. Kluwer Academic Publishers, 1986.

[Quinlan,1993] J.R. Quinlan and R.M. Cameron-Jones. FOIL: A Midterm Report. In Proceedings of the European Conference on Machine Learning, volume 667 of Lecture Notes in Computer Science, pages 3-20. Springer-Verlag, 1993.

[Ritter et al.,2008] Alan Ritter et. al. It's a Contradiction – No, it's Not: A Case Study using Functional Relations. In Conference on Empirical Methods in Natural Language Processing, Waikiki, USA, October 2008.

[Russell and Norvig,1995] Stuart J. Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice-Hall, 1995. 946p.

[Sanchez-Graillet and Poesio,2007] Olivia Sanchez-Graillet and Massimo Poesio. Discovering contradicting protein-protein interaction in text. In BioNLP 2007: Biological, translational, and clinical language processing. Pages 195-196. Prague, Czech Republic, June 2007.

[Voorhees,2008] Ellen M. Voorhees. Contradictions and Justifications: Extensions to the Textual Entailment Task. In Proceedings of ACL-08: HLT. Pages 63-71. Columbus, USA, June 2008.

[Yarowsky, 1995] Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: ACL. (1995) 189-196

[Zhu,2009] Xiaojin Zhu and Andrew B. Goldberg. Introduction to Semi-Supervised Learning (Synthesis Lectures on Artificial Intelligence and Machine Learning). Morgan and Claypool Publishers. 2009.