Pós-Graduação em Ciência da Computação

Israel Felipe de Lima Araújo Silva

Chatterbot para Criação e Refinamento de Ontologias em Lógica de Descrições

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Israel Felipe de Lima Araújo Silva

Chatterbot para Criação e Refinamento de Ontologias em Lógica de Descrições

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de Concentração: inteligência computacional
Orientador: Dr. Frederico Luiz Gonçalves de Freitas
Coorientador: Dr. Ryan Ribeiro de Azevedo

Recife
2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586c Silva, Israel Felipe de Lima Araújo
Chatterbot para Criação e Refinamento de Ontologias em Lógica de Descrições/ Israel Felipe de Lima Araújo Silva – 2019.
172 f.: il., fig., tab.

Orientador: Frederico Luiz Gonçalves de Freitas.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências e apêndices.

1. Inteligência artificial. 2. Ontologias. 3. Lógica de Descrições. I. Freitas, Frederico Luiz Gonçalves de (orientador). II. Título.

006.31 CDD (23. ed.) UFPE- MEI 2019-095

Israel Felipe de Lima Araújo Silva

“Chatterbot para Criação e Refinamento de Ontologias em Lógica de Descrições”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 20/03/2019.

_____

Orientador: Prof. Dr. Frederico Luiz Gonçalves de Freitas

BANCA EXAMINADORA

_____

Profa. Dra. Bernadette Farias Lóscio
Centro de Informática / UFPE

_____

Prof. Dr. Luis Filipe Alves Pereira
Departamento de Ciência da Computação/UFRPE

_____

Prof. Dr. Ryan Ribeiro de Azevedo
Departamento de Ciência da Computação/Campus Garanhuns
(Co-Orientador)

Dedico este trabalho primeiramente à todos aqueles que desejam chegar até onde chego neste momento, mas por algum motivo não conseguiram. Um desses é meu amigo, parceiro e "quasi-irmão"Duílian, que acredito que um dia chegará muito mais longe que eu, pois capacidade possui até demais. Agora, não menos importante, à todos de minha família, aos "de sangue"e aqueles que se tornaram família além desse fato, que me deram total apoio nessa jornada.

AGRADECIMENTOS

Agradeço primeiramente à Deus, o detentor de todo o conhecimento. Conhecimento esse que tentamos redescobrir a cada dia e compilá-lo no que chamamos de Ciência.

Agradeço minha família que me apoiou sempre em cada momento difícil e riu junto nos felizes. Em especial ao meu tio Washington que sempre disse: "Você precisa fazer, pois é isso que você quer"(talvez não nessas palavras, mas foi assim que sempre entendi). Ainda agradeço muito, muito mesmo... mesmo... mesmo... a minha mãe que sempre deu tudo de si para me deixar ser quem sou da melhor maneira possível... Dona Terezinha, OBRIGADO!!! Agradeço também, a pessoa que passou bastante dificuldades com meus "abusos", Erika Sales, namorada, amiga companheira. Agradeço a ti por me aturar nos momentos estressantes, esse período sem ti, seria muito mais difícil do que podemos imaginar.

Agora, a todo vocês meu amigos... agradeço a Renan Leandro, Levy Souza, Diogo Espinhara durante esse momento acadêmico, ajuda em disciplinas e amizade fora da vida acadêmica. Carlos (*Big Chals*), Alex (Renato Russo), Jailson (*Saitama*) e Jailsim (Irmão de Saitama), Rafael, Mileide Moraes e todos vocês que (posso não ter posto seu nome aqui, mas você lembra) durante a viagem até a Universidade contribuíram significantemente através de discussões proveitosas sobre tudo (absolutamente tudo, pode crer), para meu desenvolvimento intelectual, pessoal e profissional.

Agradeço a meus colegas de trabalho que confiaram em mim muito mais do que eu mesmo confiava.

Agradeço a você Dr. Ryan Ribeiro de Azevedo, pessoa incrível, orientador e coorientador que só faltou pegar na minha mão para me ajudar a escrever, além de amigo é claro! Dr. Fred[erico] Luiz Gonçalves de Freitas, cientista sem igual (assim como Ryan sempre falou), obrigado por essa oportunidade de estar pesquisando contigo.

Por fim, agradeço ainda, a Fundação de Amparo a Ciência e Tecnologia do Estado de Pernambuco (FACEPE) pelo seu apoio financeiro à esta pesquisa, processo IBPG-1131-1.03/16. Obrigado por financiar esse garoto do interior que veio conhecer oque era energia elétrica aos 2 anos de idade, mas que com a ajuda de vocês consegui chegar até aqui.

"Você pode dizer a diferença entre falar com um humano e falar com uma máquina? Ou é possível criar uma máquina capaz de conversar como um humano? Na verdade, o que é que nos torna humanos?"(WARWICK; SHAH, 2016)

RESUMO

Ontologias são especificações explícitas de conceitualizações compartilhadas, e constituem valiosos recursos que fornecem um modelo representativo de domínio de conhecimento reutilizável por diversas aplicações nas áreas de Engenharia do Conhecimento, Processamento de Linguagem Natural, Bioinformática, Sistemas de Recomendação (SR), entre outros. Deste modo, novas tecnologias, métodos e ferramentas capazes de lidar com os desafios técnicos e econômicos inerentes à construção de Ontologias e com o objetivo de minimizar o esforço manual e altamente especializado requerido se mostram necessárias. Neste trabalho o objetivo foi desenvolver um *chatterbot* inteligente para construção e refinamento de bases de conhecimento. Bases estas, modeladas como ontologias em OWL 2 DL a partir de linguagem natural. Assim construir uma ferramenta de autoria com capacidade de realizar raciocínio de inconsistência e subsunção/definição por meio de diálogos com os usuários. Os resultados alcançados demonstram que nosso *chatterbot* é capaz de (I) Modelar corretamente conhecimento em ontologias com no máximo expressividade $\mathcal{ALCOHQ}$ (*Attributive Language with Complements, Role hierarchy, Nominals and Qualified cardinality restrictions*) advindo dos diálogos com os usuários; (II) Capaz de realizar raciocínio de inconsistência e subsunção em tempo de execução. Nosso *chatterbot*, pode contribuir para facilitar o processo de desenvolvimento de ontologias expressivas a partir de Linguagem Natural. Além disso conta com o diferencial em utilizar técnicas de Aprendizado de Máquina que o torna uma ferramenta de Autoria capacitada para extrair e checar eficientemente conhecimento do texto. Ainda permite a checagem de novos fatos através da ontologia de topo SUMO, fornecendo, dessa forma, uma modelagem mais precisa e semanticamente correta.

Palavras-chave: Lógica de Descrições. Representação de Conhecimento. Raciocínio Automático. Processamento de Linguagem Natural. Aprendizado de Máquina.

ABSTRACT

Ontologies are explicit specifications of shared conceptualizations. They constitute valuable resources that provide a representative model of a reusable knowledge domain by several applications in different areas such as Knowledge Engineering, Natural Language Processing, Bioinformatics, among others. This way, new technologies, methods and tools capable of dealing with the technical and economic challenges inherent to the construction of ontologies and with the objective of minimizing the manual and highly specialized effort required are shown to be necessary. Our objective is to develop an intelligent chatterbot for building and refining knowledge bases modeled as ontologies. Knowledge bases modeled from natural language dialogs. Also provides a tool with the ability to carry out inconsistency and subsumption/definition reasoning. The results obtained demonstrate that our chatterbot is able to (I) correctly model knowledge in ontologies with max $\mathcal{ALCOHQ}$ (*Attributive Language with Complements, Role hierarchy, Nominals and Qualified cardinality restrictions*) expressivity resulting from dialogues with users; (II) Able to carry out reasoning of inconsistency and subsumption at run time. Our chatterbot can contribute to facilitate the process of development of expressive ontologies from Natural Language. It also has the advantage of using Machine Learning techniques that make it a Smart Authoring tool capable of extracting and checking text knowledge more effectively. It also allows for the checking of new facts through the top SUMO Ontology, thus providing a more accurate and semantically correct modeling.

Keywords: Description Logic. Knowledge Representation. Automatic Reasoning. Natural Language Processing. Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Ontologias são especificações explícitas de conceitualizações compartilhadas (GRUBER, 1995), e constituem valiosos recursos que fornecem um modelo representativo de domínio de conhecimento reutilizável por diversas aplicação nas áreas de Engenharia de Conhecimento, Processamento de Linguagem Natural, Bioinformática, Sistemas de Recomendação (SR), e outros. Porém a tarefa de modelar conhecimento de maneira formal em ontologias não é uma tarefa trivial (DENAUX, 2013). Geralmente a construção de ontologias requer conhecimento e esforço combinado de especialistas de domínio e engenheiros de ontologias/conhecimento, os quais na prática são escassos (VÖLKER; HITZLER; CIMIANO, 2007).

A conversão de linguagem natural para uma linguagem de representação de conhecimento (LRC)(HORROCKS; PATEL-SCHNEIDER, 2011; BAADER et al., 2003) é um trabalho custoso em termos financeiros e de tempo. Para representar o conhecimento, que antes poderia ter sido expresso por linguagem natural, em LRC é essencial a figura do engenheiro de conhecimento. Engenheiro de conhecimento é o profissional que incorpora conhecimento em sistemas computacionais com a finalidade de solucionar problemas que, na maioria das vezes, exigem grande volume de conhecimento humano.

Além do engenheiro, o especialista de domínio tem papel fundamental nessa tarefa. O especialista de domínio é o profissional que possui conhecimento especializado sobre uma determinada área. Nesse contexto existem três componentes essenciais: Engenheiro de Conhecimento, Especialista de Domínio e a Linguagem de Representação de Conhecimento. Os dois primeiros são escassos, e por isso com alto custo para o trabalho de modelagem de conhecimento. O último, a linguagem, apresenta uma baixa curva de aprendizado(DENAUX, 2013) tomando uma porção significativa do tempo de modelagem, elevando os custos na representação de conhecimento.

As pesquisas em ontologias têm concentrado esforços na tentativa de encontrar formas para ajudar as pessoas a colaborar com os engenheiros do conhecimento na construção de Ontologias. Deste modo, novas tecnologias, métodos e ferramentas capazes de lidar com os desafios técnicos e econômicos inerentes à construção de ontologias e com o objetivo de minimizar o esforço manual e altamente especializado requerido se mostram necessárias (ZHOU; QI; SUNTISRIVARAPORN, 2013; BUITELAAR; BUITELAAR; CIMIANO, 2008; CIMIANO, 2006).

A W3C[1], órgão que regulamenta a padronização da Web atribuiu a *Ontology Web Language* (OWL, Linguagem Ontológica para Web)[2] como padrão para representação de conhecimento em ontologias.

[1] https://www.w3.org/
[2] https://www.w3.org/OWL/

O processo de "aprendizado"que consiste em modelar conhecimento em ontologias é conhecido como Aprendizado de Ontologias (AO)(LEHMANN; VOELKER, 2014). O processo de aprendizado pode ser classificado em 3 (três) tipos(El Idrissi Esserhrouchni; FRIKH; OUHBI, 2014):

- Automáticos: Um sistema computacional realiza toda a tarefa de construção, sem intervenção humana no processo.
- Semi-Automáticos: A intervenção humana ocorre durante o processo de construção das ontologias, onde axiomas são avaliados pelo usuário através de *softwares* desenvolvidos para esse propósito.
- Manuais: Ontologias são construídas manualmente utilizando de ambientes de desenvolvimento integrado (IDE), como o Protégé (NOY; FERGERSON; MUSEN, 2007) ou OntoEdit (STUDER et al., 2007).

Em relação ao aprendizado de forma manual, existe um grande esforço do engenheiro de conhecimento e especialista de domínio em termos de tempo e financeiro que provocam aumento nos custos de produção de ontologias.

Trabalhos como o LExO(VÖLKER; HITZLER; CIMIANO, 2007), OntoRich(BARBUR; BLAGA; GROZA, 2011) e o TextOntoEx(DAHAB; HASSAN; RAFEA, 2008), Chen, Dosyn e Lytvyn (2016) apostaram no processo de produção automática de ontologias, afim de facilitar o processo e minimizar seus custos para representação de conhecimento. Porém ma abordagem automática pode não proporcionar a construção de ontologias semanticamente corretas, uma vez que não há intervenção humana para interagir no processo de construção. Assim axiomas incorretos ou com baixa semântica podem ser inseridos na ontologia resultante.

A abordagem semi-automática, é uma das que se mostram em equilíbrio entre facilidade de modelagem de conhecimento e intervenção humana. Aproveitando-se dessa abordagem, várias ferramentas tem sido propostas para construção de ontologias a partir de fontes textuais (De Azevedo et al., 2014; SERRA; GIRARDI; NOVAIS, 2013; DRYMONAS; ZERVANOU; PETRAKIS, 2010).

De Azevedo et al. (2014), Ali et al. (2016), Bradesko et al. (2016) se destacam por explorar um recurso proposto por Alan Turing (1912-1954), matemático, lógico, criptoanalista e precursor da Inteligência Artificial (IA), em 1950. A utilização de um sistema computacional inteligente que dialoga com humanos afim de aprender. Nesse contexto, De Azevedo et al. (2014) e Bradesko et al. (2016) utilizam *chatterbots* com poder de raciocínio para conversar com usuário. A medida que o diálogo vai avançando, o conhecimento adquirido pelo *chatterbot* é modelado em forma de ontologias.

Um dos primeiros *chatterbot*, ou simplesmente *bot*, a imitar a capacidade humana de conversação que teve grande repercussão foi o ALICE[3] (Artificial Linguistic Internet Computer Entity, Entidade de Computação Linguística Artificial na Internet). O ALICE utiliza AIML (*Artificial Inteligence Markup Language*, Linguagem de Marcação de Inteligência Artificial) como base de conhecimento e se tornou bastante popular por possibilitar simular conversação entre humanos e sistemas computacionais.

Os *bots* que consultam bases AIML geralmente não possuem a capacidade de aprendizado, isto é, não aprendem, apenas respondem questões previstas em suas bases de conhecimento. O entendimento da linguagem natural por sistemas computacionais é um processo minucioso integrado de várias etapas para identificação dos elementos textuais, como verbos, substantivos, sujeito, etc. Além disso é crucial a existência de uma linguagem que represente conhecimento processável por computador.

Quando Alan Turing propôs o Teste de Turing havia um grande desejo em dar à um sistema computacional a habilidade de conversação. Assim um humano poderia interagir com um computador como faz com outros humanos.

O Teste de Turing consiste em um entrevistador realizando perguntas aleatoriamente selecionadas e direcionadas para um computador ou uma pessoa (Vide Figura 1), em espaços separados uns dos outros.

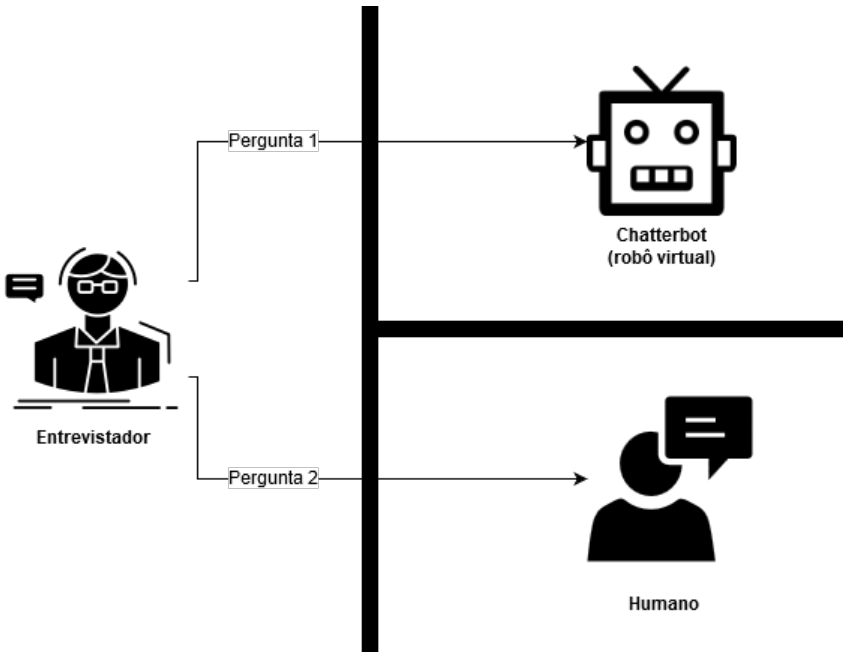

Figura 1 – Representação do Teste de Turing

Com base em suas respostas, o entrevistador deve distinguir entre uma resposta dada por um humano e uma resposta dada por um computador. Se o entrevistador não conseguir distinguir tais respostas, então o computador é dito inteligente (Krol, 1999). Desde a

[3] https://www.pandorabots.com/pandora/talk?botid=a847934aae3456cb

formulação do Teste de Turing a IA tem buscado por um agente computacional com habilidades de dialogar e aprender sobre o que se está dialogando.

Chatterbots, *bots* e/ou sistemas de diálogos são agentes conversacionais, programas computacionais utilizados para simular uma conversa com um ou mais humanos através de métodos textuais (AL-ZUBAIDE; ISSA, 2011). Um *Chatterbot* pode entender o que você está dizendo, analisar, e dar uma resposta plausível. A maioria dos *bots* buscam por palavras chave, frases, e padrões que estão previamente definidos em sua base de conhecimento, porém outros podem utilizar técnicas mais avançadas (RESHMI; BALAKRISHNAN, 2016).

Um *Chatterbot* inteligente deve realizar raciocínio, deduzindo novos fatos e perceber fatos contraditórios. Consideremos o diálogo:

```
Humano: todo carnívoro come apenas carne.
Bot: ok, entendi. O que é carne?
Humano: carne não é planta
Bot: ótimo, algo mais?
Humano: Sim
Bot: diga-me
Humano: cão é um carnívoro que come planta
Bot: desculpe-me, mas acho que você se enganou, pois cão é um carnívoro e
todo carnívoro come apenas carne.
```

Notamos a partir do diálogo acima que o *chatterbot* se manteve "curioso"durante o diálogo. Essa curiosidade permite ao *chatterbot* aprender através do diálogo com o humano. No exemplo, o humano que dialoga com o *bot* informa fatos contraditórios quando diz que *cão come planta e é um carnívoro*, sendo que *carnívoro come apenas carne* e *carne não é planta*. O *bot* de forma inteligente e com capacidade de raciocínio é capaz de identificar e informar ao humano essa contradição.

Aprender com o usuário, modelar conhecimento e realizar raciocínio são desafios que essa dissertação investiga e desenvolve uma solução. Para superar esses desafios utilizamos diferentes áreas e técnicas de pesquisa, como: Processamento de Linguagem Natural (PLN), Representação de Conhecimento, Ontologias, Aprendizado de Ontologias e Aprendizado de Máquina. Assim pretendemos alcançar um *chatterbot* capaz de aprender e raciocinar durante o diálogo com o usuário.

Neste trabalho desenvolvemos um *chatterbot*, nomeado PUPO, para modelagem de conhecimento em OWL a partir de textos em linguagem natural processados através do CoreNLP[4] (Também conhecido como *Stanford Parser*). O *CoreNLP* é um parser sintático e estatístico capaz de identificar os elementos textuais (verbo, adjetivo, substantivo, etc). Para extrair informação relevante a ser traduzida e modelada em OWL é utilizada

[4] https://stanfordnlp.github.io/CoreNLP/

uma abordagem híbrida entre Aprendizagem de Máquina (AM) e o CoreNLP. AM é um campo da computação que envolve o estudo de reconhecimento de padrões e teoria de Aprendizado Computacional em Inteligência Artificial(FAN et al., 2016).

Assim um *chatterbot* com capacidade de, através de uma abordagem híbrida com AM e técnicas simbólicas de aprendizado, extrair informações precisas e consistentes pode se tornar uma ferramenta de suporte ao especialista de domínio. Para fornecer esse suporte ao especialista de domínio, o PUPO à primeira vista faz o papel do engenheiro de conhecimento e realiza a tradução da linguagem natural para OWL. Contudo, todo algoritmo de aprendizagem de máquina possui uma probabilidade de erro associada ao seu resultado.

Para tentar mitigar o erro inerente dos algoritmos de AM na aplicação proposta, existem bases de conhecimento (ontologias escritas em OWL) prontas que modelam conhecimento geral válido para todas as áreas de conhecimento. Nesse cenário o *chatterbot* proposto faz uso de uma Ontologia de representação de conhecimento geral, também conhecida como Ontologia de topo. Pela sua popularidade e reputação do grupo de engenheiros de conhecimento envolvidos em sua construção foi escolhida a SUMO[5](PEASE; NILES; LI, 2002) (*Suggested Upper Merged Ontology*, Ontologia Mesclada Superior Sugerida).

A utilização da SUMO provê um *background* de conhecimento no qual as novas informações serão confrontadas, assim como faz o ser humano ao dialogar. A SUMO atua como um meio de checagem para garantir mais consistência em novas informações extraídas. Além da SUMO o *chatterbot* conta com a utilização do dicionário de dados WordNet[6](MILLER, 1995) para expandir o vocabulário da SUMO através da busca de termos sinônimos. Por fim o PUPO apresenta um mecanismo de raciocínio sobre as ontologias modeladas, podendo assim inferir novos fatos sobre o conhecimento modelado enquanto conversa com o usuário.

Como resultado o PUPO é capaz de modelar conhecimento a partir de diálogos em linguagem natural. Essa modelagem em ontologias alcança expressividade $\mathcal{ALCOHQ}$ (*Attributive Language with Complements, Role hierarchy, Nominals and Qualified cardinality restrictions*, Linguagem Atributiva com Complemento, Hierarquia de Papeis, Nominais e Restrições Qualificadas de Cardinalidade), além de permitir, através dos termos da SUMO e mapeamento com o WordNet, extensão do vocabulário da Ontologia de Topo. Assim os recursos gastos para modelagem de conhecimento podem ser melhor aproveitados durante o desenvolvimento de ontologias consistentes e expressivas.

Assim pretendemos, por meio do PUPO, provar a seguinte hipótese:

"O PUPO é capaz de modelar conhecimento e realizar raciocínio através de diálogos"

[5] www.adampease.org/OP/

[6] https://wordnet.princeton.edu/

1.1 OBJETIVOS

O objetivo geral deste trabalho é *desenvolver um chatterbot para criação e refinamento de ontologias expressivas com lógica de descrições com capacidade de realizar raciocínio durante a construção dessas ontologias.*

1.1.1 Objetivos Específicos

Para contemplar nosso objetivo geral, precisamos alcançar os seguintes objetivos específicos.

- Demonstrar que o PUPO é capaz de:
  - Construir ontologias em OWL 2 DL com expressividade máxima $\mathcal{ALCOHQ}$ a partir de textos;
  - Realizar raciocínio de subsunção/definição deduzindo que classes são subclasses de outras, a partir de suas respectivas descrições;
  - Detectar e verificar inconsistências em tempo de desenvolvimento nas ontologias construídas;
  - Realizar extração de axiomas, automaticamente, utilizando aprendizagem de máquina e técnicas linguísticas;
- Suprir ao usuário um processo de construção de Ontologias – a partir de dados advindos da interação em linguagem natural – buscando prover eficiência para a intervenção humana em sua construção.
- Avançar no estado da arte construindo um *chatterbot* capaz de aprender através de diálogos.

1.2 ESCOPO DESTE TRABALHO

A utilização de Ontologias em projetos para enriquecimento das mesmas ou até criação de novas Ontologias baseadas em outras necessitam em sua maioria um mapeamento semântico entre seus termos. O Mapeamento de Ontologias (Liu; Cao; Dai, 2011) (do inglês, *Ontology Mapping (OM)*) tem papel fundamental em relacionar novos termos com os termos conhecidos de uma certa Ontologia. Esse mapeamento é muitas vezes realizado calculando a similaridade sintática entre os termos ou através de dicionários eletrônicos. Na `SUMO` existe uma relação nomeada *Part*, que representa que algo é parte de outro algo, por exemplo o axioma `Arm SubClassOf Part some Body` (Braço é subclasse de alguma parte do corpo). Porém ao processar linguagem natural essa relação é muitas vezes extraída como `PartOf`. O *OM* tem essa função de mapear `PartOf` como sendo `Part`

e assim o axioma `Arm SubClassOf Part some Body` tem o mesmo valor de `Arm SubClassOf PartOf some Body`.

Nesse contexto é essencial salientar que a abordagem desse projeto utiliza o `WordNet` como dicionário semântico afim de validar axiomas a partir de termos sinônimos. Assim não tem como foco avaliar ou utilizar quaisquer outros tipos de mapeamentos semânticos.

1.3 ESTRUTURA DESTE TRABALHO

A dissertação está organizada da seguinte forma:

- O Capítulo 2 (Fundamentação Teórica) fornece o conhecimento necessário para o entendimento das técnicas desenvolvidas nesse trabalho. Apresentamos alguns *chatterbots* e a linguagem a qual utilizam para conversar.
- No Capítulo 3 fornecemos a arquitetura geral do PUPO e seus componentes bem como os algoritmo utilizados.
- No Capítulo 4 é apresentada a interface do PUPO, onde o usuário pode manipular ontologias a partir de diálogos, além de exemplos de uso para cada funcionalidade.
- No Capítulo 5 apresentamos e comparamos alguns trabalhos voltados à extração de informação e representação de conhecimento, sendo esses trabalhos ferramentas e/ou métodos com ou sem mecanismo de diálogo.
- No Capítulo 6, mostramos a composição das bases de teste, objetivos do experimento e sua execução.
- No Capítulo 7 apresentamos como este trabalho cumpriu seu objetivo geral através dos específicos além de fornecer pontos nos quais o trabalho pode ser aperfeiçoado ou estendido.

2 FUNDAMENTAÇÃO TEÓRICA

Apresentamos neste capítulo a fundamentação teórica a respeito dos principais componentes e tecnologias utilizados no desenvolvimento do nosso *chatterbot*. As subseções seguintes descrevem tecnologias como *chatterbots* e AIML, Processamento de Linguagem Natural (PLN), Ontologias, Representação de Conhecimento e Aprendizagem de Máquina.

2.1 CHATTERBOTS E AIML

Os *chatterbots* ou simplesmente *bots* são robôs virtuais com a função de imitar a capacidade humana de conversação, o *chatterbot* `ALICE` (*Artificial Linguistic Internet Computer Entity*)(MIKIC et al., 2008) é um exemplo. Os *bots* tem a tarefa de conversar e responder questionamentos dos seus usuários. Com o desenvolvimento do *chatterbot* `ALICE` os pesquisadores desenvolveram a linguagem de marcação AIML (*Artificial Intelligence Markup Language*, Linguagem de Marcação para Inteligência Artificial). Outros *chatterbots* baseados no ALICE foram desenvolvidos para suprir algumas necessidades educacionais como o TutorBot, ELEKTRA, BonoBot e o MEARA(MIKIC et al., 2008).

AIML foi Desenvolvida pelo Dr.Richard Wallace e a comunidade *open source* Alicebot entre 1995 e 2000. Essa linguagem de marcação é baseada em XML. Segundo Neves, Barros e Hodges (2006) AIML é a tecnologia mais fácil para ser utilizada na construção de *Chatterbots*. A AIML consiste em categorizar padrões de perguntas e respostas que serão utilizadas durante o diálogo. Tais perguntas e respostas podem ser formadas por frases inteiras ou expressões regulares que a linguagem oferece, assim um mesmo padrão de frase pode servir para diferentes perguntas do usuário. Como derivada do XML, a sintaxe padrão para cabeçalho do documento é visto na Figura 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml version="2.0">
(...)
</aiml>
```

Figura 2 – Cabeçalho AIML

Cada conjunto formado por pergunta e resposta é chamado de *Category*, a qual possui uma *tag* correspondente *<category>*. Dentro de cada *Category* existem dois elementos, o primeiro é chamado de *<pattern>*, que é o padrão de entrada para o *bot* processar, e o segundo elemento é o *<template>* contendo a resposta a ser retornada ao usuário.

2.1.1 A *tag pattern* e *Wildcards*

A *tag <pattern>*, que representa o padrão de entrada para o *bot*, pode conter caracteres especiais que funcionam como expressões regulares auxiliando na sua construção, são chamados de *Wildcards*. O texto pode ser inserido integralmente e no momento que for dado como entrada ao *bot* ele retornará a resposta imediata, também é possível a existência de *Wildcards* em meio ao texto.

O *Wildcard* * funciona de forma a permitir que venham a aparecer na frase *n*-palavras em seu lugar. Na Tabela 1 é evidenciado seu uso.

Padrão	Tag AIML
* cat is * animal	<pattern>* cat is * animal</pattern>
* dog has *	<pattern>* dog has * </pattern>

Tabela 1 – Utilização de *Wildcard* para composição do *pattern*

No padrão "* cat is * animal" é possível alcançar diferentes frases com a mesma base de palavras, por exemplo: *"A cat is an animal"*, *"A cat is a good animal"*, *"Dog has eyes"*, *"A dog has a good nose"*. Tal mecanismo possibilita alcançar uma gama maior de frases sem necessariamente escrevê-las.

Em AIML existem 4 *Wildcards*: #, _, ∧, *. O * e o _ representam uma palavra ou mais, enquanto o # e o ∧ representam zero ou mais palavras. A documentação completa pode ser acessada em https://www.pandorabots.com/docs/.

Considerando o _, que pode ser utilizado de uma forma semelhante ao *, tal símbolo tem como objetivo indicar prefixos e sufixos do padrão, porém quando o *bot* processar a informação, o _ tem precedência em relação ao *. Alguns exemplos são mostrados na Tabela 2.

Padrão	Tag AIML
_ ALICE	<pattern>_ ALICE</pattern>
MOTHER _	<pattern>MOTHER _</pattern>

Tabela 2 – Utilização de _ para composição do *pattern*

Assim a primeira frase da Tabela 2 pode representar *"Who is ALICE"*, *"Where is ALICE"*, e a segunda como sendo *"MOTHER is a woman"*, *"MOTHER has children"*. A utilização destes caracteres especiais (que são intitulados de *Wildcard*) simplifica a escrita dos padrões propostos e suas respostas em diálogos com *chatterbots*.

2.1.2 *Tag template*

Assim como a *tag pattern*, a *tag template* utiliza de mecanismos para simplificação de seu conteúdo, porém ao invés de caracteres especiais são inseridos outras *tags* que introduzem

condições e variáveis ao diálogo(Vide Apêndice A.1).

Na Tabela 3 apresentamos como podemos escrever um padrão e uma resposta em AIML. Esses padrões são previamente definidos pelo programador do *chatterbot*, assim quanto mais padrões forem gerados mais o diálogo se tornará próximo a de um humano.

Padrão	Resposta
* cat is * animal	<template> that's so cool</template>
* dog has *	<template>oh! that is nice, i don't knew</template>

Tabela 3 – Padrões de resposta do PUPO

2.1.3 *Chatterbots* e suas aplicações

Trabalhos como o CHARLIE de Mikic et al. (2008) e T-BOT, Q-BOT Mikic et al. (2009) utilizam-se de *chatterbots* no ambiente educacional. O CHARLIE, por exemplo, serve como uma interface de comunicação entre o Sistema Educacional Inteligente (INES) e o aluno através da internet, como mostrado na Figura 3. Onde *BUI* é a interface gráfica do *bot*, o *interpreter* um interpratador que acessa as bases AIML.

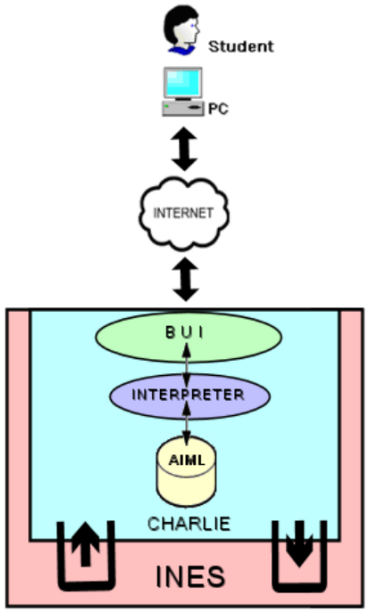

Figura 3 – Arquitetura do CHARLIE dentro do INES

Chatterbots tem sido propostos para os mais diferentes propósitos, o *ClimeBot*(TONIUC; GROZA, 2017), por exemplo, é integrado com o *Slack*[1] para diálogo sobre clima com equipes de projetos. Alguns *bots* saem do escopo de "apenas"dialogar sobre o que sabe e parte para o "aprendizado". O aprendizado nesse sentido, compreende-se do processo de modelar ou modificar a base de conhecimento do *bot*.

[1] https://slack.com/

No ramo de engenharia do conhecimento, com o intuito de facilitar a tarefa dos profissionais envolvidos na construção de ontologias podem ser citados os *chatterbots Renan*(De Azevedo et al., 2014), X-UDeKAM(ALI et al., 2016) e o *Curious Cat*(BRADESKO et al., 2016).

O *Curious Cat* (Gato Curioso) foi desenvolvido para *Android*[2] e dialoga com os usuários sobre os os lugares que os mesmos visitam. Na Figura 4 observamos o *bot* discutindo com o usuário sobre o preço do *cappuccino*. A medida que o mesmo dialoga essa informação é modelada em ontologias.

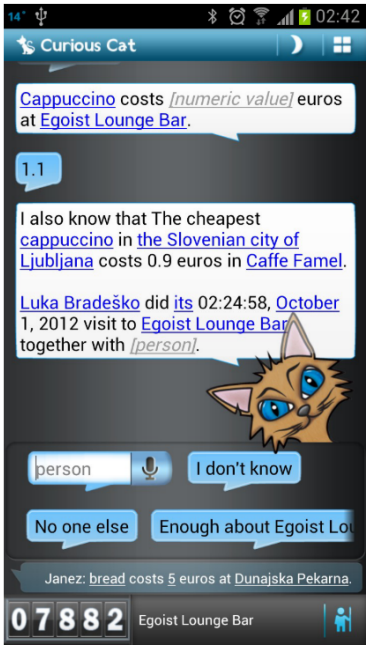

Figura 4 – Captura de Tela do protótipo do Gato Curioso

No que lhe concerne o Renan e o X-UDeKAM dialogam em linguagem natural com o usuário. Durante o diálogo são realizadas diversas tarefas como processamento de linguagem natural e raciocínio em algum formalismo lógico afim de identificar inconsistências e inferências. Mais detalhes serão discutidos no Capítulo 5.

2.2 PROCESSAMENTO DE LINGUAGEM NATURAL

A linguagem é o princípio da comunicação utilizada pelo ser humano. Essa ferramenta é utilizada para expressar a maioria de suas emoções. A linguagem apresenta uma estrutura de significado. O Processamento de Linguagem Natural (PLN) é o estudo de classificação e identificação dos significados da linguagem humana, (TAYAL; RAGHUWANSHI; MALIK, 2014).

O PLN consiste na identificação das partículas que compõem a linguagem, como verbos, adjetivos, substantivos, entre outros. Uma ferramenta computacional para PLN deve fazer uma classificação sintática conforme uma linguagem dada. Tais ferramentas são chamadas de *Parsers*, que tem a função de dado a linguagem natural, retornar uma linguagem

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

classificada sintaticamente, um exemplo de parser é o *Stanford Parser (CoreNLP)*[3]. Em PLN diferentes tarefas e com objetivos distintos são realizadas, ver Figura 5.

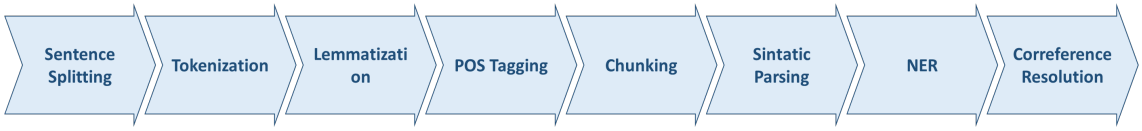

Figura 5 – Principais tarefas para Processamento de Linguagem Natural

Sentence Splitting *(Quebra de Sentença)* - Consiste em determinar o fim das sentenças que compõem um documento.

Tokenization *(Tokenização)* - É o processo de dividir as sentenças nas partículas mínimas que compõem o texto. Cada partícula, também conhecida como *token*, pode ser uma palavra ou símbolo que apresenta um significado.

Lemmatization *(Lematização)* - É a tarefa de se obter a forma básica de uma palavra, independentemente de sua flexão. Por exemplo o verbo *tried* após este processo terá sua forma *try* (infinitivo do verbo). Para esse processo é realizado um processo que envolve conhecimento e contexto, diferentemente de remoção de sufixos utilizados no *stemming*.

POS Tagging *(Etiquetagem POS)* - Todo *Parser* apresenta uma estrutura de símbolos de classificação de partículas da linguagem, como pontuação e palavras. Essa estrutura é denominada de *Part-Of-Speech* ou apenas *POS*. Com isso o *Parser* realiza uma etiquetagem dos itens da linguagem, retornando o texto sintaticamente identificado pelo *POS-Tagger*, que é uma camada do *Parser*, como pode ser visto na Figura 6.

Figura 6 – POS - pelo Stanford Parser

Na Tabela 4 é possível observar algumas das etiquetas/*tags*[4] adotadas pelos *parsers* para classificação gramatical. Para tabela completa vide Apêndice B.1.

Número	Tag	Descrição
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential there

Tabela 4 – POS - Nível de palavra (parcial)

Chunking *(Agrupamento)* - As *tags* da Tabela 4, indicam a função de cada palavra, porém o nível de interpretação textual pode ir além de tal identificação. Conjuntos de

[3] http://nlp.stanford.edu/software/lex-parser.shtml
[4] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

determinadas palavras podem compor um sentido diferente, como é o caso de expressões, ou o uso de artigos antes de um verbo, fazendo do mesmo um substantivo. Essa etiquetagem, é realizada a nível de frases e conhecida como *chunking*. As *Tags* de nível de frase dão uma melhor compreensão do texto a ser processado, como pode ser visto na Figura 7.

```
[NP dog][ADVP also][VP likes][NP saussage]
```

Figura 7 – Chunking

Na Tabela 5 são apresentadas algumas das *tags* a nível de frases. Para tabela completa vide Apêndic B.2.

Número	Tag	Descrição
1	ADJP	Adjective Phrase.
2	ADVP	Adverb Phrase.
3	CONJP	Conjunction Phrase.
4	FRAG	Fragment.

Tabela 5 – POS - Nível de frase (parcial)

Sintatic Parsing *(Parser Sintático)* - Outra camada conhecida dos *Parsers* retorna do texto em linguagem natural um texto hierárquico chamada de árvore de dependência (*Dependency Tree* ou *PCFG Structure*)(KLEIN; MANNING, 2003). Como o nome sugere o texto é representado em forma de árvore e utiliza-se de uma combinação dos métodos de *POS-Tagging* e *Chunking*. Apresentamos na Figura 8 essa estrutura de árvore para a sentença *"dog also likes saussage"*.

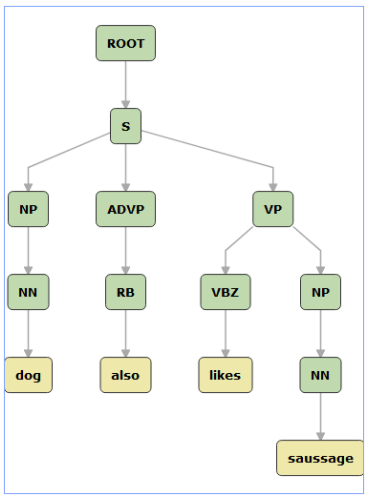

Figura 8 – PCFG Structure (Árvore de dependência) fornecida pelo Stanford Parser através do endereço http://corenlp.run/

Além do processamento do texto e a identificação de sua estrutura existem outras tarefas que o Processamento de Linguagem possibilita. Essas outras tarefas são classificadas como Extração/Recuperação de Informação (EI), das quais podem ser citadas Reconhecimento de Entidades Nomeadas (NER), Extração de Relação e Resolução de Correferência(MAYNARD; BONTCHEVA; AUGENSTEIN, 2017).

NER *(Reconhecimento de Entidade Nomeada)* - Consiste na identificação de entidades com características únicas, por exemplo pessoas (*Person*), organizações (*Organization*), locais (*Location*), datas (*Date*) e expressões temporais. Por exemplo, *Barack Obama* (Person), *Microsoft* (Organization), *New York* (Location), *4th July 2015* (Date/Temporal Expression), etc.

Na Figura 9 é possível observar a classificação de Entidades Nomeadas como *Joe Smith, Jane, France*, etc.

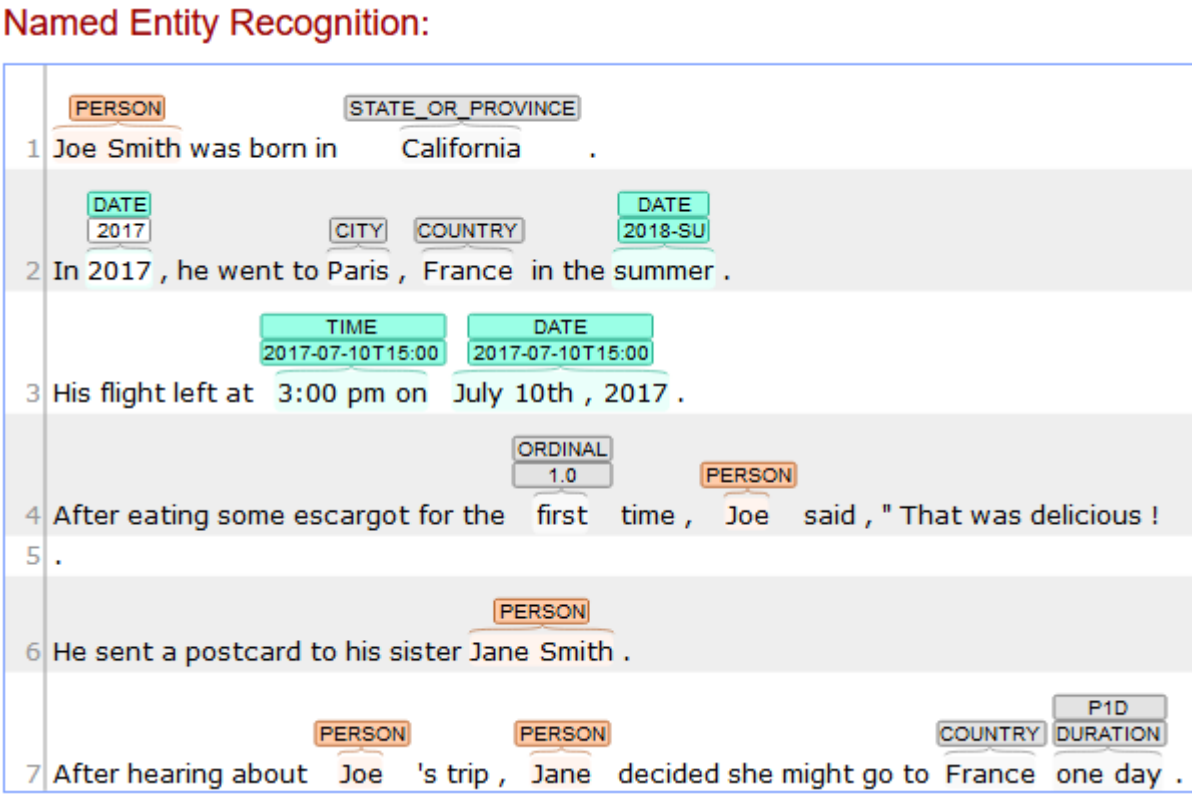

Figura 9 – NER através do CoreNLP (http://corenlp.run/)

Sistemas de NER são geralmente utilizados na primeira etapa de recuperação de informação e resolução de correferência(YADAV; BETHARD, 2018). Yadav e Bethard (2018) afirmam que várias arquiteturas neurais tem sido propostas, a maioria baseadas em alguma forma de redes neurais recorrentes (Recurrent Neural Network - RNN) sobre caracteres, sub-palavras e/ou *Words Embbedding* (palavras embarcadas).

Correference Resolution **(Resolução de Correferência)** - Uma outra tarefa correlacionada a NER é a Resolução de Correferência (RC). A RC geralmente tem o papel de identificar a quais entidades um pronome pertence, por exemplo:

```
Joe Smith was born in California.
In 2017, he went to Paris, France in the summer.
```

O pronome *he* refere-se a *Joe Smith*. Na Figura 10 pode-se observar que o CoreNLP classificou como *he* sendo uma correferência para *Joe Smith*.

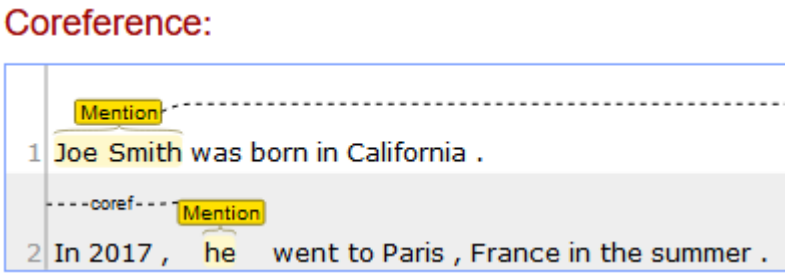

Figura 10 – Resolução de Correferência através do CoreNLP

2.2.1 Extração de Relação

Um texto processado através das tarefas previamente citadas contém metadados que facilitam sistemas computacionais extraírem informação de forma inteligente. Uma das mais importantes tarefas para recuperação/extração de informação consiste na extração de relações.

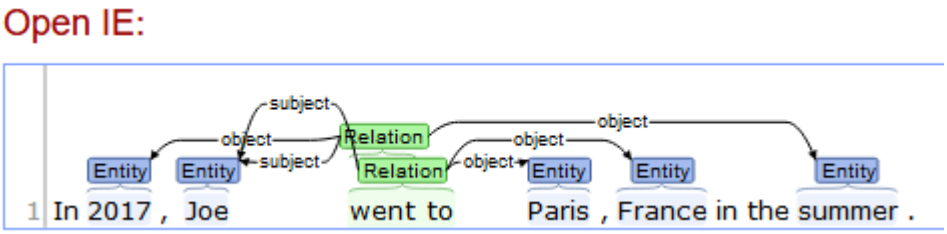

Figura 11 – Extração de relações através do CoreNLP

Na Figura 11 é apresentada a extração de relações para a sentença *"In 2017, Joe went to Paris, France in the summer"*. Nessa sentença foram extraídas as relações:

```
2017 went to summer
Joe went to summer
Joe went to Paris
Joe went to France
```

Das quais *"Joe went to Paris"* e *Joe went to France* estão semanticamente corretas.

Essas tarefas (NER, resolução de correferência e extração de relações) tem sido as mais utilizadas na área de PLN para modelagem de conhecimento, uma vez que possuem a finalidade de identificar e extrair informações dentro da Linguagem Natural.

2.3 WORDNET

O `WordNet` (MILLER, 1995) é um dicionário semântico contendo substantivos, adjetivos, verbos e advérbios organizados em conjuntos de sinônimos. O `WordNet` define seu vocabulário como um conjunto W de pares (f, s), onde f representa uma palavra e s um elemento representante de um significado de f. Cada elemento dotado de significado é chamado de *word* e o dicionário é composto por esse conjunto de *word*s.

O `WordNet` se encontra na versão 3.1 e contém mais de 166.000 pares (f, s) consistindo em mais de 118.000 *word*s com mais de 90.000 significados. Cerca de 40% de seus termos possuem pelo menos um sinônimo. Os conjuntos de sinônimos são chamados *synsets*, exemplos desses *synsets* são vistos na Tabela 6.

Em Inglês	Em Português
{car, automobile}	{carro, automóvel}
{hit, strike}	{acertar, chocar}
{big, large}	{grande, amplo}

Tabela 6 – Exemplos de *synset*

Existem alguns tipos de relacionamentos entre os termos do `WordNet`, dos quais podem ser citados:

Synonymy (sinônimo) - Representa a relação de termos sinônimos, ou seja que apresentam significados equivalentes.

Antonymy (antônimo) - Representa relação inversa do sinônimo, informando conjunto de termos contrários entre si.

Hyponymy (hipônimo) - Representa um conceito mais específico em relação a outro, pode ser considerado uma subclasse se tratando de *OWL*.

Hypernymy (hiperônimo) - Representa um conceito mais geral em relação a outro. Em *OWL* pode ser comparado ao relacionamento hierárquico de superclasse.

Aproveitando os tipos de relacionamento, o dicionário é formado por hierarquias entre conjuntos de significados. Na Figura 12, por exemplo, os termos: *organismo* e *ser* (*synset*[5] *{organismo,ser}*) são termos mais específicos (*hyponymy*) do *synset {coisa viva, coisa animada}*. Assim é possível observar que sua estruturação fornece relacionamentos entre conjuntos de *synsets* e não apenas hierarquia entre termos.

[5] *synset* é uma coleção (*set*) de sinônimos

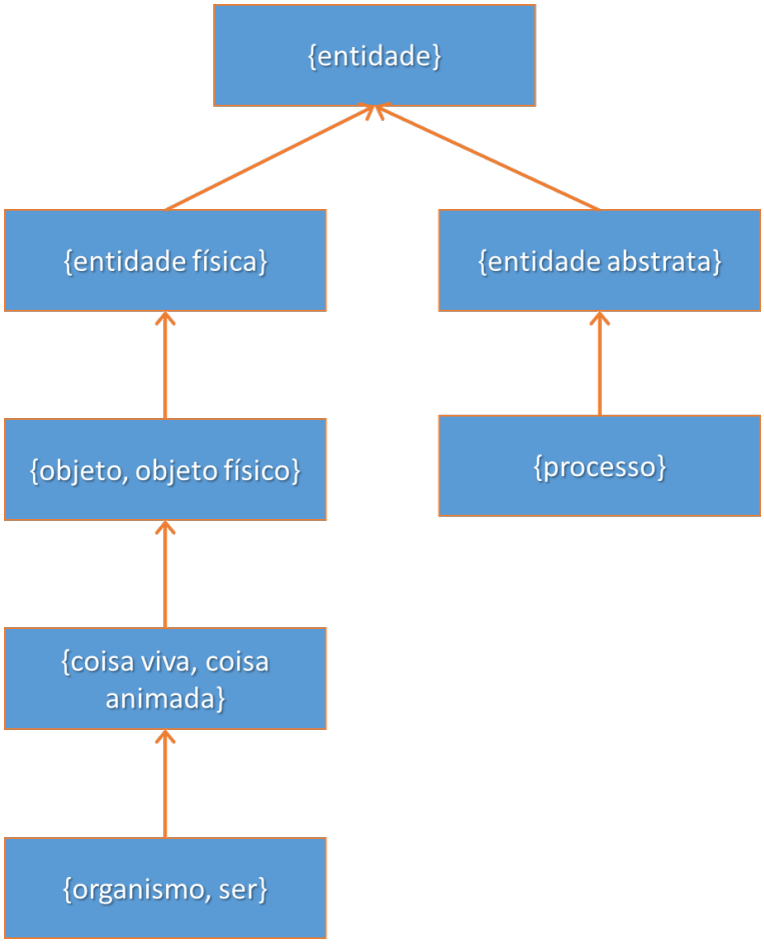

Figura 12 – Hierarquia do `WordNet`

2.4 WEB SEMÂNTICA - ONTOLOGIAS E REPRESENTAÇÃO DE CONHECIMENTO

A representação de conhecimento segundo Cahyani, Manurung e Mahendra (2015) é a tecnologia que pode ser utilizado na Web Semântica para representar conhecimento semântico, compreensível por ferramentas computacionais, referente ao conhecimento humano. Tal representação de conhecimento faz com que os raciocinadores (*Reasoners*), que são motores de inferência que utilizam da representação de conhecimento para verificar a consistência, coesão e possíveis inferências da informação representada computacionalmente.

Ontologias são como um dicionário de termos com definições a partir de relações sintáticas que representa o conhecimento e podem ser compartilhadas e interpretadas por meios computacionais (CAHYANI; MANURUNG; MAHENDRA, 2015).

As Ontologias podem representar conhecimento de diversas áreas e em diferentes níveis, desde conhecimento mais geral ao mais específico. Segundo Guarino (1998), as Ontologias podem ser classificadas em 4 (quatro) tipos principais:

Ontologias de topo - Descrevem conceitos gerais que independem de um problema em particular, como espaço, tempo, eventos, entre outros. Como exemplo de ontologia de topo podemos citar a SUMO(PEASE; NILES; LI, 2002).

Ontologias de domínio e de tarefas - Correspondem a Ontologias que modelam um domínio genérico ou uma tarefa genérica, respectivamente.

Ontologias de aplicação - Modelam conceitos que descrevem de forma especialista as entidades de um domínio ou tarefa.

Algumas das tecnologias mais utilizadas, no quesito de linguagem de representação de conhecimento, são *Resource Description Framework (RDF)*, *Resource Description Framework Schema(RDFS)*, e *Web Ontology Language (OWL)*.

As informações na linguagem RDF são descritas por meio de triplas (sujeito, predicado, objeto). Representando assim uma relação binária entre o sujeito e o objeto, semelhante as relações extraídas em PLN. Por sua vez o *RDFS* define o esquema de representação e hierarquias entre classes. A OWL[6], estende ainda mais o *RDFS* permitindo inserir elementos baseados em Lógica de Descrições(BAADER et al., 2003) (do inglês *Description Logics* ou somente *DL*), assim vindo a ser chamada de OWL DL.

A OWL se encontra em sua segunda versão (OWL 2). Esta é o padrão recomendado pela *W3C*[7] para representação semântica de Ontologias(BAYOUDHI; SASSI; JAZIRI, 2018). A OWL 2 contém duas variantes a OWL 2 DL e OWL 2 *FULL*. A primeira mantém um equilíbrio entre decidibilidade e expressividade dos axiomas que podem ser modelados. Esse fato possibilita que a OWL 2 DL seja compatível com os mais recentes raciocinadores além de constar com expressividade $\mathcal{SROIQ}(\mathcal{D})$(BOBILLO; STRACCIA, 2011).

2.4.1 Expressividade Semântica e Raciocínio

Linguagens de descrição, também conhecidas como DLs, são fragmentos da Lógica de Descrições, categorizadas de acordo sua complexidade. A linguagem básica é conhecida como $\mathcal{AL}$ (BAADER et al., 2003). Assim podemos definir essa linguagem base como:

$$C, D \rightarrow A \quad (Conceito\ Atômico) \tag{2.1}$$
$$\top \quad (Conceito\ Superior) \tag{2.2}$$
$$\bot \quad (Conceito\ Inferior) \tag{2.3}$$
$$\neg A \quad (Negação\ Atômica) \tag{2.4}$$
$$C \sqcap D \quad (Intersecção) \tag{2.5}$$
$$\forall R.C \quad (Restrição\ Universal) \tag{2.6}$$
$$\exists R.\top (Restrição\ Existencial) \tag{2.7}$$

Onde A, C e D são conceitos e R uma relação qualquer. $\top$ representa um conceito o qual todos os outros são filhos dele. Já o $\bot$ é um conceito o qual todos os outros estão hierarquicamente acima dele. Quando adicionamos negação em propriedades, além da

[6] https://www.w3.org/TR/owl2-overview/
[7] https://www.w3.org/

negação de conceitos, acrescentamos o $\mathcal{C}$ (complemento) a expressividade, gerando assim a linguagem $\mathcal{ALC}$.

Os quantificadores universais e existenciais ($\forall$ e $\exists$) não expressam possibilidade de contagem. Por exemplo se quisermos dizer que *um ser unicelular possui apenas 1 célula* teríamos que escrever um axioma do tipo:

```
SerUnicelular SubClasseDe possui exatamente 1 Celula
```

Ao adicionarmos essa possibilidade na linguagem $\mathcal{ALC}$ definimos a linguagem $\mathcal{ALCQ}$.

Além da hierarquia de conceitos, existem as hierarquias entre propriedades, por exemplo:

```
Pedir
    |_ Ordenar
```

Nessa hierarquia, observamos que a propriedade Ordenar é pode ser compreendido como uma subpropriedade de Pedir. Acrescentando esse nível de expressividade na linguagem $\mathcal{ALCQ}$ alcançamos a $\mathcal{ALCHQ}$. Até esse momento a linguagem $\mathcal{ALCHQ}$ definem classes, propriedades e restrições sobre terminologias. A esse conjunto de terminologias damos o nome de *TBox* (*Terminological Box, Caixa Terminológica*).

Além de terminologias podemos acrescentar indivíduos em nossos axiomas em linguagem de descrição. Indivíduos representam seres únicos com identidade própria, por exemplo Israel, Brasil, AppleInc. Ao conjunto desses indivíduos e seus relacionamentos com a *TBox* damos o nome de *ABox* (*Assertion Box*, Caixa de Asserção). Como exemplo temos os seguintes axiomas:

```
{Israel} TypeOf Humano
{Ted} TypeOf Cao
```

Esses axiomas definem que *Israel* é um *Humano* e *Ted* é um *Cão*. Além desses tipos de definição, podemos acrescentar indivíduos nas definições de classes, por exemplo:

```
Brasileiro SubClassOf naturalDe algum {Brasil}
```

Definimos, acima, que um *Brasileiro* é algo que é *naturalDe* algum indivíduo *Brasil*. Assim, todo brasileiro é aquele que é natural do brasil. Essa possibilidade nos permite aumentar mais uma vez a expressividade adicionando o $\mathcal{O}$ a linguagem, gerando assim a linguagem $\mathcal{ALCHOQ}$. Assim a linguagem $\mathcal{ALCHOQ}$ possibilita definições de classes, indivíduos, hierarquias entre conceitos e propriedades, negação, união, intersecção e quantificação universal e existencial.

Cada linguagem apresenta uma complexidade diferente e um processo de raciocínio igualmente complexo. Os *softwares* que raciocinam logicamente sobre uma linguagem são conhecidos como raciocinadores. Os raciocinadores ou motores de inferência podem ser descritos como uma máquinas de estados finito que consiste em três estados: casamento, seleção e execução de regras(SINGH; KARWAYUN, 2010). No casamento de regras o motor de inferência encontra todas as regras que são satisfeitas. Durante a seleção de regras o raciocinador seleciona, segundo alguma(s) estratégia(s), a melhor regra para ser executada. E por fim, executa a regra selecionada. Sendo que, a execução de uma regra pode disparar o casamento de novas regras e assim desencadear o processo de raciocínio, até que todas as regras relevantes sejam executadas e se obtenha uma conclusão. Singh e Karwayun (2010) apresentam diversos raciocinadores, dos quais podem ser citados o `Jess, Hoolet, Pellet, SHER, RacerPro, FaCT, FaCT++, F-OWL, BaseVISor`. Os únicos, desses citados, a suportarem $\mathcal{SROIQ}(\mathcal{D})$ são o `Pellet` e o `FaCT++`.

2.5 APRENDIZAGEM DE MÁQUINA (AM)

Aprendizagem de Máquina é um subcampo do aprendizado computacional e uma grande ferramenta para extração de informação. Esse campo envolve o estudo de reconhecimento de padrões e teoria de Aprendizado Computacional em Inteligência Artificial(FAN et al., 2016). Geralmente Aprendizagem de Máquina pode ser dividida em duas categorias (DUDA; HART; STORK, 2000): Aprendizagem Supervisionada (AS) e Aprendizagem Não Supervisionada (ANS).

ANS - Contém algoritmos que inferem uma função que descreve os dados de treino e assim podem realizar predições. Como exemplo de ANS podem ser citados o *K-Means* e *Expectation-Maximization* (EM) (FAN et al., 2016). O *K-Means*, por exemplo, geralmente é utilizado no processo de *clustering* (clusterização). *Clustering* consiste em agrupar elementos de forma que os mais semelhantes entre si fiquem juntos.

AS - Contém algoritmos que inferem um modelo, e então esse modelo é utilizado para fazer predição. Na categoria de AS podem ser citados o Naive Bayes (NB), Support Vector Machine (SVM), *K-Nearest Neighbors* (KNN) e *Decision Tree*(YANG; CHEN, 2017).

Segundo Ghorpade e Ragha (2012), os algoritmos de AM mais modernos para PLN contém alguma base, principalmente, de aprendizagem de máquina estatística para suas principais tarefas. Essas tarefas consistem em Sumarização Automática, Reconhecimento de Entidades Nomeadas (NER), Anotação Sintática (*POS-Tagging*), Parseamento, Análise de sentimento.

Assim essa área de pesquisa necessita de um grande campo de conhecimento, dentre os quais podem ser citados linguística, ciência da computação, estatística, entre outros. Abordagens baseadas em AM e técnicas de PLN para extrair conceitos e relações ontológicas de dados estruturados e não estruturados como texto ou banco de dados tem sido explorados com o objetivo de construir Ontologias (El Idrissi Esserhrouchni; FRIKH; OUHBI,

2014). Notam-se esforços, em aprendizado de ontologia, para utilização de algoritmos como SVM, Árvores de Decisão (SANTOSO et al., 2016), FCA (DRYMONAS; ZERVANOU; PETRAKIS, 2010) e RCA (ABDELBASSET; OKBA; SOFIANE, 2013), LSI & SVD e Mr.LDA (Variação do LDA) (RANI; DHAR; VYAS, 2017), Naive Bayes (SANTOSO; YUNIARNO; HARIADI, 2015), TF-IDF (YUAN; ZHUANG; LI, 2010).

Além disso AM e algoritmos bem conhecidos como SVM, *Decision Tree*, são bastante explorados e têm ótimo resultado para classificação de documentos textuais(Wei Zhu et al., 2016).

Considerando AS, os problemas podem ser classificados em dois tipos: Problemas de Classificação e Problemas de Regressão.

Problemas de Regressão

Considerando a definição de Jordan, Kleinberg e Schölkopf (2008), o objetivo na regressão é estimar uma relação funcional entre uma variável aleatória de entrada X e uma variável aleatória de saída Y.

Problemas de Classificação

Segundo Gunn (1998) o problema de classificação pode ser restrito à consideração do problema de duas classes sem perda de generalidade. Neste problema, o objetivo é separar as duas classes por uma função que é induzida a partir dos exemplos disponíveis. O objetivo é produzir um classificador que funcione bem em exemplos não vistos, isto é, que seja bem generalizado.

2.5.1 Máquina de Vetor Suporte (SVM)

Na maioria dos casos, SVM é um modelo para classificação em 2 (duas) classes, salvo suas variações para classificação acima de 2 (duas) classes(FRANC; HLAVAC, 2002). Dados textuais são ideais para classificar com essa abordagem(AYDOGAN; AKCAYOL, 2016) devido a sua natureza ampla onde poucas características são irrelevantes, mas correlacionadas (YANG; CHEN, 2017).

A SVM opera sobre um *kernel*. A idéia da função *kernel* é permitir que as operações sejam executadas no espaço de entrada, em vez do espaço de recurso, levando os dados a um novo espaço. Isso fornece uma maneira de abordar o problema em um outro espaço dimensional. A função de classificação da SVM tem a forma:

$$dx = \sum_{i=1}^{m} a_i y_i K(x_i, x) + b \tag{2.8}$$

Onde a e b são parâmetros estimados para serem utilizados na SVM, $K(x_i, x)$ é o *kernel* da função. A função de *kernel* é definida de acordo a natureza do problema. Por exemplo um *kernel* linear possibilita classificação de problemas linearmente separáveis enquanto RBF (*Radial Basis Function*)(GUNN, 1998) problemas não lineares.

Como entrada a SVM recebe um vetor de números reais para ser classificado e retorna sua classificação como resultado. Esse resultado por sua vez pode ser um número real, ou

um nominal (classe), conforme configuração da SVM. Como entrada uma SVM pode receber um vetor de números reais que representa um texto. O método mais utilizado na literatura para representar textos em vetores numéricos em PLN é o word2vec(WOHLGENANNT; MINIC, 2016). Esse processamento pode contar com TF-IDF (*Term Frequency - Inverse Document Frequency*, Frequência de Termo - Frequência inversa de Documento)(PAN et al., 2015; DADGAR; ARAGHI; FARAHANI, 2016). Assim através de uma representação numérica real algoritmos de aprendizado de máquina podem classificar ou predizerem valores através de regressão ou classificação.

2.6 EXTRAÇÃO/RECUPERAÇÃO DE INFORMAÇÃO (IE)

O problema de extração de informação (IE), gerando dados relacionais a partir de texto em linguagem natural, tem recebido atenção crescente nos últimos anos. Um grande repositório de alta qualidade de tuplas extraídas pode potencialmente beneficiar uma ampla gama de tarefas de PLN, como como perguntas e respostas (Q&A), aprendizado de ontologias (AO) e sumarização, afirma Weld (2010).

Um extrator de informação – Sistema responsável por extrair informação semântica de textos em linguagem natural – é uma função de um documento, *d*, para um conjunto de triplas, [arg1, rel, arg2](WELD, 2010). Onde os *args* são sintagmas/frases nominais e *rel* é um fragmento textual que indica uma relação semântica implícita entre as duas frases nominais.

2.6.1 Métricas de avaliação para *IE*

Para que seja possível avaliar o quanto um algoritmo é eficiente em extração de informação (IE) foram definidas as medidas de *Precision* e *Recall*(BAEZA-YATES; RIBEIRO-NETO, 1999). Essas medidas vêm da matriz de confusão (Vide Tabela 7), que indica quais valores foram corretamente ou incorretamente classificados.

Predito	Verdadeiro	Falso
Verdadeiro	TP	FP
Falso	FN	TN

Tabela 7 – Matriz de confução para classificação binária

Os valores de TP (verdadeiro positivo, do inglês *true positive*) indicam os valores positivos classificados de forma correta pelo sistema. O FP (falso positivo, do inglês *false positive*) indica os valores que foram classificados como positivos, mas deveriam ser classificados como falso. FN (falso negativo, do inglês *false negative*) indica os valores classificados como negativos, mas deveriam ser classificados como positivos. Por fim, TN (verdadeiro positivo, do inglês *true negative*) indicam os valores classificados como negativos e eram, de fato.

Através dessa tabela é possível extrair os valores de *Recall* (R) e de *Precision* (P), Equações 2.9 e 2.10.

$$P = \frac{tp}{(tp + fp)} \tag{2.9}$$

$$R = \frac{tp}{(tp + fn)} \tag{2.10}$$

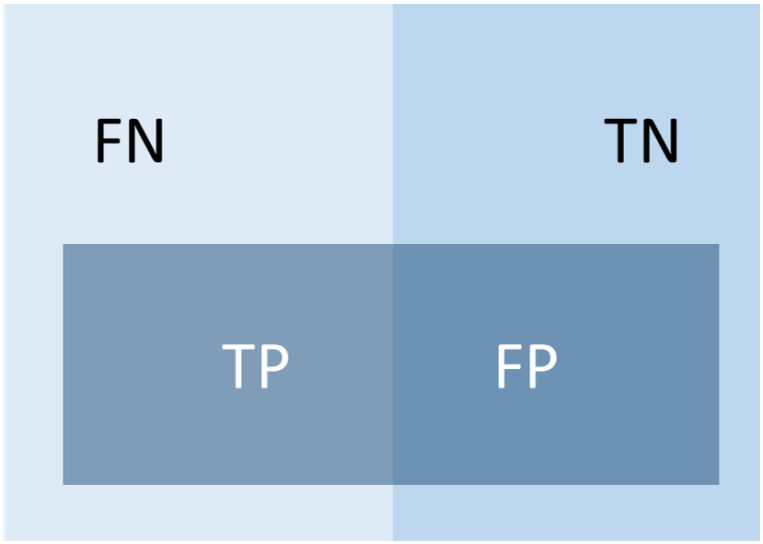

Figura 13 – Representação gráfica da matriz de confusão

Considerando o valor de P, os dois retângulos mais escuros da Figura 13 representam todas as classificações feitas pelo algoritmo. Quando tp é divido por $(tp + fp)$, está sendo calculada a percentagem de acertos dado o que o algoritmo foi capaz de classificar.

O retângulo formado por FN e TP indicam os termos relevantes, ou seja, os elementos que deveriam ser classificados como positivos. Quando se divide tp por $(tp + fn)$ está sendo calculado uma percentagem de acerto em relação a tudo que deveria ser classificado como verdadeiro pelo sistema, sendo esse o valor R.

Além dessas duas medidas, é de fundamental importância citar a *F-measure* (F). Essa medida realiza uma média harmônica entre os valores de R e P, representando assim uma relação entre os valores de tp, fp e fn. O valor de F pode ser calculado com a seguinte expressão:

$$F = 2 * \frac{R * P}{R + P} \tag{2.11}$$

No Capítulo que se segue apresentamos o PUPO e sua arquitetura. Para cada componente da arquitetura exibimos seu funcionamento através de algoritmos e/ou exemplos.

3 O CHATTERBOT PUPO

O PUPO oferece possibilidade de construção automática de Ontologias em DL a partir de interação em LN com seus usuários, além disso o PUPO possui capacidade de realizar raciocínio automático (subsunção/definição e raciocínio de inconsistência), o que o diferencia da maioria dos *chatterbots* "à la Searle"passivos e sem inteligência (capacidade de aprender, raciocinar e representar conhecimento durante os diálogos). Para cumprir os objetivos propostos (Vide Capítulo 1), as tecnologias e os algoritmos desenvolvidos e aplicados no seu desenvolvimento foram fundamentados em áreas e técnicas como AO, PLN, AM e Raciocínio Automático. Essas técnicas e ferramentas foram selecionadas e desenvolvidas com base nos trabalhos relacionados (Vide Capítulo 5). Na sequência é exposta a arquitetura do *chatterbot* desenvolvido, exibindo como o mesmo processa e compreende LN, aprende, representa conhecimento e realiza raciocínio automático a partir de textos.

3.1 ARQUITETURA DO BOT: UMA VISÃO GERAL

Um *chatterbot* funciona como uma interface de diálogo entre usuário e um sistema computacional. Esse diálogo, na maioria dos casos, apresenta dois momentos fundamentais **Interpretação** e **Resposta**. A Interpretação consiste no momento em que o sistema entende a informação do seu usuário. Posteriormente a **Resposta** consiste na fase de, dado a **Interpretação**, gerar informação para o usuário, exemplificando temos:

```
João: Bianca, você sabe onde está minha meia azul?
Bianca:   Você deixou no banheiro.
```

No diálogo acima, João pergunta a Bianca onde está sua meia azul. Bianca durante o processo de Interpretação busca entender o que João deseja. No exemplo, Bianca identifica que é a localização da meia dele (João). Bianca, gera informação para João, durante o processo de resposta. Por fim, Bianca com sua resposta formada, responde a João que ele a deixou no banheiro.

Nesse contexto, *um chatterbot* também provê interpretação e resposta, vinda e provida ao usuário respectivamente.

São exibidos na Figura 14 quatro pacotes que compõem a arquitetura do PUPO. O pacote (1) *Dialog Manager*, responsável por gerenciar o diálogo do PUPO com o usuário, fazendo uso do `ProgramAB`[1]. Por sua vez, o pacote (2) *Morphosyntatic Parsing* tem como objetivo realizar o pré-processamento do texto fornecido pelo usuário. Para realizar esse pré-processamento o componente *NLP Tasks* utiliza o Stanford Parser[2] para gerar a árvore

[1] http://alicebot.blogspot.com.br/2013/01/program-ab-aiml-20-reference.html
[2] https://nlp.stanford.edu/software/lex-parser.shtml

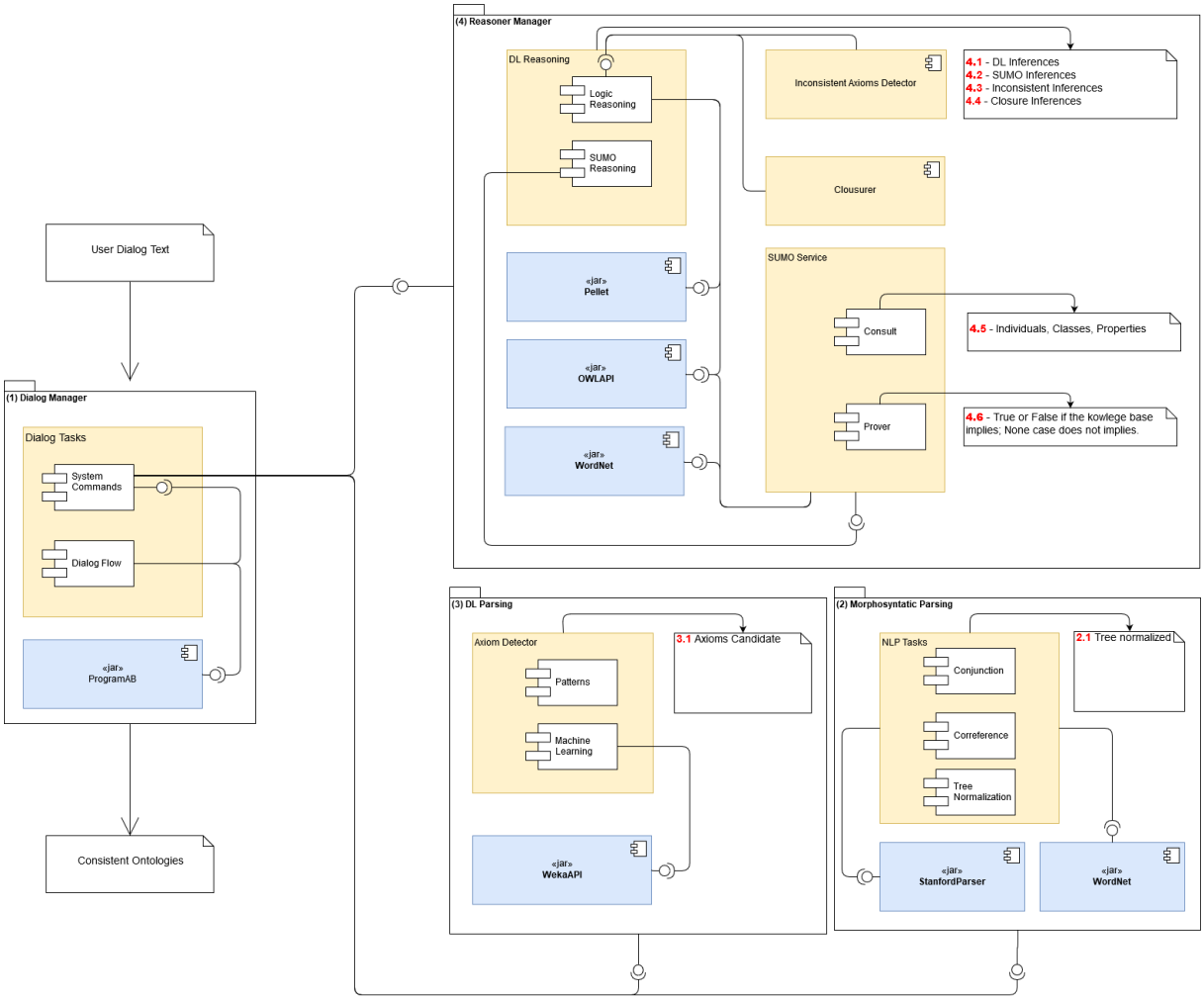

Figura 14 – Diagrama de componentes

de dependência do texto, além do `WordNet`[3] substituindo o verbo *to be* para sua forma no infinitivo e os substantivos para seu singular. Após a realização do pré-processamento dos textos advindos dos usuários uma estrutura de árvore é concedida ao pacote (3) *DL Parsing*. Nesse pacote o componente *Axiom Detector* utiliza padrões sintáticos e semânticos, para gerar axiomas a partir da árvore criada. Após essa fase o componente *Machine Learning*, composto por uma SVM (*Support Vector Machine*, Máquina de Vetor Suporte), realiza a tentativa de encontrar axiomas não identificados através de padrões sintáticos. O *Axiom Detector*, por fim, retorna um conjunto de axiomas extraídos da árvore de dependência.

O último pacote (4) *Reasoner Manager* tem o papel de validar a consistência dos axiomas extraídos. Para isso faz uso da máquina de inferência *Pellet*[4], que fornece um mecanismos para verificação de consistência da base de conhecimento, além de inferência de novos axiomas. Todos os axiomas consistentes e inferidos são persistidos em Ontologias

[3] https://wordnet.princeton.edu/
[4] https://github.com/stardog-union/pellet

por meio da `OWL-API`[5]. Assim, os componentes do pacote (4) *Reasoner Manager* retornam um conjunto de axiomas consistentes juntamente com os axiomas inferidos compondo as bases de conhecimento criadas durante o diálogo com os usuários.

Cada pacote é formado por um conjunto de componentes com funcionalidades bem definidas e que garantem que seu objetivo seja contemplado. A seguir apresentamos em detalhes cada pacote do PUPO.

3.1.1 Pacote (1) Dialog Manager

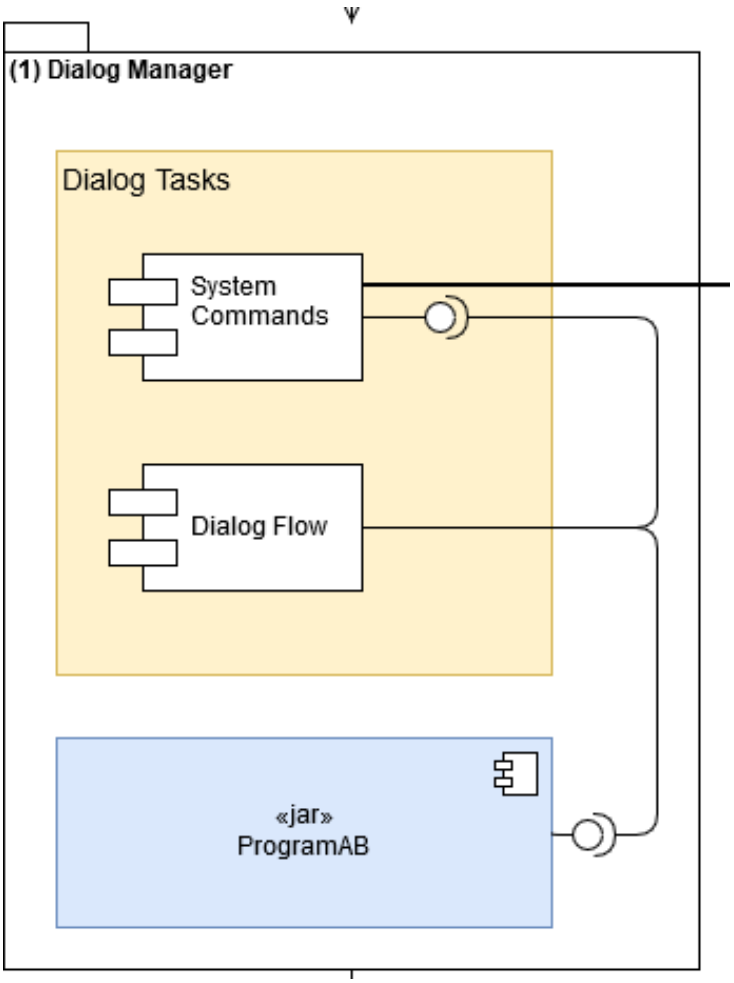

Figura 15 – Dialog Manager

Os componentes deste pacote , Figura 15, são responsáveis por intermediar o diálogo dos usuários com o PUPO e comandos que manipulam as bases de conhecimento, de forma transparente para os usuários.

Utilizando-se do `ProgramAB` o subcomponente *Dialog Flow* (Fluxo de Diálogo) é responsável por interpretar (Fase de Interpretação) a informação que o usuário fornece de entrada para o PUPO. Essa interpretação decide se esse subcomponente fará utilização

[5] http://owlapi.sourceforge.net/

de algum comando de sistema[6], criando uma Ontologia por exemplo ou não. Todo comando de sistema é tratado através do subcomponente *System Commands* (Comandos de Sistema). Apresentamos um exemplo de um início de diálogo entre o usuário e o PUPO:

```
PUPO: Hello i am your Ontology building assistant, do you want to create a
new Ontology?
```

Para que a sentença acima seja dada como resposta, existe uma categoria AIML no PUPO que contém o padrão *hello* e como template *Hello i am your Ontology building assistant, do you want to create a new Ontology?*[7]

Nesse momento, caso o usuário responda NO (não), o componente (1) exibe uma lista de comandos que podem ser executadas no sistema.

```
PUPO:
Say "hello" and let's create an Ontology
Say "load Ontology http://validiri" to load an Ontology
Say "enable youriri" or "disable iri" to active or desactive your
Ontologies
```

Caso o usuário responda YES (sim), o sistema pede uma *IRI* (*Internationalized Resource Identifiers*, Identificador de Recursos Internacionalizado)[8] válida, como segue, para construção da Ontologia.

```
PUPO: Tell me a valid iri to create a new Ontology.
```

O *IRI* é utilizado para identificar as Ontologias e são acessíveis através desse endereço via internet. Nessa fase é criada uma Ontologia com a IRI informada. O sistema pergunta se o usuário deseja informar alguns fatos para inserir na Ontologia.

```
PUPO: The ontology was successful created, do you want tell me some facts
about the domain?
```

Caso responda NO (não), o sistema apresenta a lista de comandos que podem ser executadas no sistema. Respondendo *YES* (sim), o sistema espera definições do usuário a respeito do domínio de conhecimento que o usuário deseja modelar(Vide Apêndice C.2). Assim, ativa seu modo de processamento de linguagem natural (pln)(Vide Apêndice C.3), o qual espera definições a serem processadas.

[6] Para a lista completa de comando de sistema vide Apêndice F
[7] Para detalhes de implementação vide Apêndice C.1
[8] https://www.w3.org/International/O-URL-and-ident.html

Ao fornecer a definição "dog is an animal and has eyes"(cão é um animal e tem olhos) é enviado um comando de sistema que fará uso do pacote (2) *Morphosyntatic Parsing* para adicionar a definição nas Ontologias. O pacote (2) e seus componentes são apresentados na próxima seção.

Com todos os componentes da arquitetura trabalhando em conjunto, o sistema necessita manter o diálogo com o usuário enquanto manipula as diversas ontologias (Conforme Algoritmo 6, Apêndice C.4). Para isso foram desenvolvidas *tags* AIML customizadas.

Cada *tag* AIML *customizada* realiza chamada de métodos para manipulação da base de conhecimento, bem como recebe as respostas do componente que manipula as bases de conhecimento para retornar uma resposta condizente com a vontade do usuário. Para isso, foi desenvolvido um *AIMLProcessor* customizado que avalia *tags* (assim como as *tags Category, Pattern, Template*, etc) específicas. Esse *AIMLProcessor* é uma classe em Java que pode ser implementada para estender a linguagem *AIML*[9]. Essa classe é fornecida pela API do *Program-AB*. Por exemplo, o usuário insere um texto contendo o padrão *classify the text ** o Algoritmo 6 processa o *template* $"< classifyText >< starindex = "1"/ >< /classifyText >"$, como segue.

```
1 <category>
2     <pattern>classify the text *</pattern>
3     <template>
4         <classifyText><star index="1" /></classifyText>
5     </template>
6 </category>
```

A *tag* utilizada no *template* é a *classifyText*, que segundo o algoritmo, dispara a função *classify*.

O comando de "classificação de texto" (*classify*) realiza as tarefas descritas pelos componentes de (1) à (4) adicionando os axiomas nas bases de conhecimento. Após a classificação, axiomatização e adição na base de conhecimento é realizada uma checagem de consistência e inferência. Essa checagem consiste em "chamar" a função *nextInference*, que retorna uma pergunta se um axioma inferido é verdadeiro ou indaga ao usuário se o mesmo deseja informar algum outro texto para ser processado. Em relação ao comando de "próxima inferência" (*nextInference*), é realizada a validação dos axiomas extraídos pelo algoritmo de aprendizagem de máquina, checados pela SUMO ou inconsistências. Considerar inconsistências como inferências permite que o sistema valide com o usuário um conjunto de axiomas contraditórios e remova inconsistências em tempo real, garantindo a qualidade das Ontologias modeladas.

[9] Para detalhes de implementação vide https://code.google.com/archive/p/program-ab/wikis/ExtendingAIML.wiki

As sentenças processáveis pelo PUPO tem caráter de linguagem natural. Por outro lado, a linguagem natural de interação entre o usuário e o PUPO tem características de linguagem natural controlada. Pois a utilização de AIML nos possibilita interagir com o usuário através de padrões detectados em tempo de diálogo, porém previamente modelados. Assim a utilizamos linguagem natural irrestrita na modelagem de conhecimento e linguagem natural controlada para iteração com o usuário.

3.1.2 (2) MorphoSyntatic Parsing

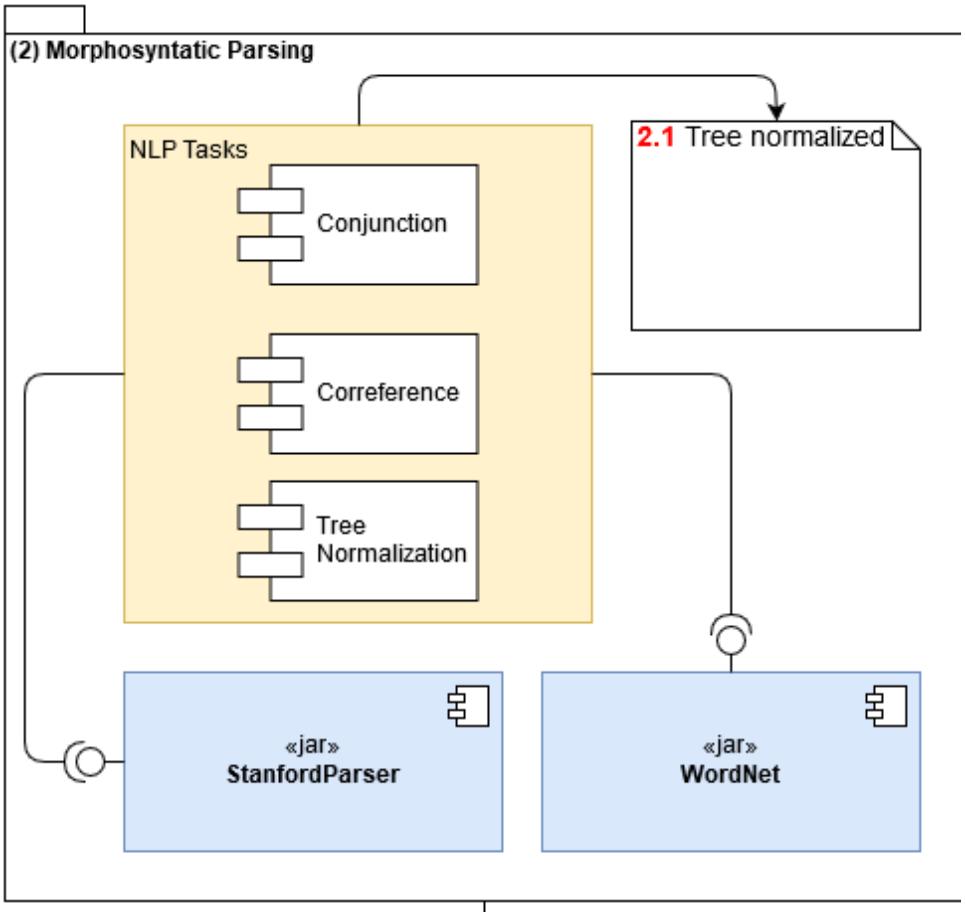

Figura 16 – MorphoSyntatic Parsing

Na etapa em que o texto sai do módulo (1), ocorre o pré-processamento pelos componentes do pacote (2) *MorphoSyntatic Parsing*. Assim o intuito do pacote (2) (Vide Figura 16) é:

- Resolver:
  - Conjunções;
  - Uniões;
  - Correferência.

- Modificar a árvore de dependência para fornecer uma melhor estrutura de axiomatização, por meio do componente *Tree Normalization*

O texto nessa etapa, advindo dos usuários se encontra em linguagem natural, sem anotação ou quaisquer identificações de componentes sintáticos ou semânticos. Esse pacote contém um componente chamado *NLP Tasks* (Tarefas de PLN), e este por sua vez contém três subcomponentes, *Conjunction*, *Correferences* e *Tree Normalization*

O tratamento das conjunções e uniões se dão através do subcomponente *Cunjunction* conforme Algoritmo 1[10].

Algoritmo 1: Union and Conjunction Resolution

Input : Text in natural language
Output: Text in natural language with conjunction and union resolution

```
1  activeConjunction ← null;
2  words ← words in text;
3  idx ← last word index in text;
4  for each word in text do
5      if words[idx] == ('AND' ou 'OR') then
6          activeConjunction ← words [idx ];
7      if words[idx] ==',' then
8          if activeConjunction! = null then
9              words [idx ] ← activeConjunction ;
10     idx ← idx − 1;
11 return text (words);
```

3.1.2.1 Tratamento de conjunção e união

O Algoritmo 1 utiliza uma variável *conjuncaoAtiva* que guarda a última palavra do texto igual a 'AND' ou 'OR'. Percorrendo da última palavra até a primeira todas as vírgulas imediatamente anteriores são substituídas pela variável *conjuncaoAtiva*. Ao encontrar uma conjunção a palavra *conjuncaoAtiva* anterior é substituída pela nova. Exemplificando temos as seguintes situações:

```
Entrada: Dog is an animal, mammal or canivore that has eyes, ears and lags.
Saída: Dog is an animal, mammal or canivore that has eyes and ears and lags.

conjuncaoAtiva = and
```

Percorrendo do final da sentença a variável *conjuncaoAtiva* recebe seu primeiro valor, nesse caso é um *and*.

[10] Para tradução vide Apêdice C.5

```
Entrada: Dog is an animal, mammal or canivore that has eyes and ears and
lags.
Saída: Dog is an animal or mammal or canivore that has eyes and ears and
lags.

conjuncaoAtiva = or
```

Agora ao encontrar a palavra *or* a variável *conjuncaoAtiva* recebe o valor *or*, então todas as vírgulas anteriores são substituídas por *or*.

Apresentamos agora exemplo mais complexo, vejamos a seguinte sentença:

Dog is an animal, and mammal, canine or canivore that has eyes, head or ears and lags. (Cão é um animal, e mamífero, canino ou carnívoro que tem olhos, cabeça ou orelhes and pernas).

O algoritmo realizará as seguintes etapas[11]:

Etapa 1

```
conjuncaoAtiva = and

Entrada: Dog is an animal, and mammal, canine or canivore that has eyes,
head or ears and lags.

Saída: Dog is an animal, and mammal, canine or canivore that has eyes,
head or ears ->and lags.
```

Etapa 2

```
conjuncaoAtiva = or

Entrada: Dog is an animal, and mammal, canine or canivore that has eyes,
head or ears and lags.

Saída: Dog is an animal, and mammal, canine or canivore that has eyes,
head ->or ears and lags.
```

Nota-se que nessa etapa a conjunção ativa que possuía o valor *and* agora passa a ter o valor *or*.

Etapa 3

[11] O símbolo -> representa a conjunção armazenada na variável *conjuncaoAtiva*

```
conjuncaoAtiva = or

Entrada: Dog is an animal, and mammal, canine or canivore that has eyes
or head or ears and lags.

Saída: Dog is an animal, and mammal, canine or canivore that has eyes
or head or ears and lags.
```

A última vírgula foi substituída por *or*.

Etapa 4

```
conjuncaoAtiva = or

Entrada: Dog is an animal, and mammal, canine or canivore that has eyes
or head or ears and lags.

Saída: Dog is an animal, and mammal, canine ->or canivore that has eyes
or head or ears and lags.
```

Nesse momento outro *or* foi encontrado, fazendo com que a *conjuncaoAtiva* fosse substituída novamente por *or*.

Etapa 5

```
conjuncaoAtiva = or

Entrada: Dog is an animal, and mammal, canine or canivore that has eyes
or head or ears and lags.

Saída: Dog is an animal, and mammal or canine or canivore that has eyes
or head or ears and lags.
```

Nesse momento a última vírgula da frase foi encontrada e substituída pelo valor da *conjuncaoAtiva*, nesse caso continha o valor *or*.

Etapa 6

```
conjuncaoAtiva = and

Entrada: Dog is an animal, and mammal or canine or canivore that has eyes
or head or ears and lags.
```

```
Saída: Dog is an animal, ->and mammal or canine or canivore that has
eyes or head or ears and lags.
```

Agora, a *conjucaoAtiva* é substituída por *and*.

Etapa 7

```
conjuncaoAtiva = and

Entrada: Dog is an animal, and mammal or canine or canivore that has eyes
or head or ears and lags.

Saída: Dog is an animal and and mammal or canine or canivore that has
eyes or head or ears and lags.
```

Por fim, a última vírgula da frase é substituída por *and*. Nota-se que após a aplicação do algoritmo duas conjunções *and* ficaram juntas. O componente *conjunction* trata esses casos de conjunções repetidas substituindo-as por seu valor atômico, ou seja apenas uma conjunção.

Essa substituição é realizada na última etapa de tratamento do texto por este componente. Posteriormente fornece a saída para o componente de tratamento de correferência.

Essa substituição é realizada a través de duas expressões regulares:

```
  (\\s+and\\s+and\\s+|\\s+and\\s+and\\s+and\\s+) subsituído por "and"
  (\\s+or\\s+or\\s+|\\s+or\\s+or\\s+or\\s+) subsituído por "or"
```

A primeira expressão regular substitui duas ou mais ocorrências do *and* por apenas uma. De forma semelhante a segunda substitui duas ou mais ocorrências do *or* por apenas uma.

3.1.2.2 Tratamento de correferência

Outro componente desenvolvido foi o *Correference*, que lida com o problema de correferência utilizando o *Stanford Parser*. Este provê um mapeamento dos termos referenciados em um documento textual e suas menções. O *Stanford Parser* provê um mecanismo pronto para identificação de correferência. Porém, no contexto deste trabalho, o algoritmo nativo retorna apenas um mapeamento dos termos de um texto. Por exemplo[12]:

```
Texto:
        Joe Smith was born in California.
        In 2017, he went to Paris, France in the summer.
```

[12] Os exemplos a seguir foram extraídos de https://stanfordnlp.github.io/CoreNLP/api.html

```
Mapeamento:
        1 [1,2] -> 2 [4,4]
```

O mapeamento é no formato $s_0\ [b_0, e_0] \rightarrow s_1[b_1, e_1]$ onde 's' é o índice da sentença e 'b' e 'e' são início e fim do *token* referenciado. A parte à esquerda de $\rightarrow$ representa a sentença de origem e a parte a direita representa o termo (geralmente pronome) a qual se faz referência. Assim, o subcomponente *correferences* utiliza essa mapeamento e substitui o termo referenciado, por exemplo:

```
Texto Original:
        Joe Smith was born in California.
        In 2017, he went to Paris, France in the summer.

Texto Resultado:
        Joe Smith was born in California.
        In 2017, Joe Smith went to Paris, France in the summer.
```

O Algoritmo 2[13] exibe a resolução de correferência do sistema com suporte do *Stanford Parser*. A tarefa da variável *referencia* do algoritmo é armazenar os nomes na sequencia que aparecem no texto, quaisquer termos do tipo pronome possessivo ou pessoal é substituído por esta *referencia*.

No caso de o pronome ser um pronome possessivo a variável *referencia* recebe o complemento "'s", indicando posse. Por exemplo:

Entrada:

```
Joe Smith was born in California.
In 2017, he went to Paris, France in the summer.
His flight left at 3:00pm on July 10th, 2017.
```

Saída:

```
Joe Smith was born in California.
In 2017, Joe Smith went to Paris, France in the summer.
Joe Smith's flight left at 3:00pm on July 10th, 2017.
```

Assim a expressão "his flight" foi substituída por "Joe Smith's flight". Até esse momento as saídas dos algoritmos fornecem linguagem natural não anotada, mas que foram modificadas para um melhor processamento posterior.

[13] Vide tradução no Apêndice C.5

Algoritmo 2: Coreference Resolution using Stanford Parser Support

Input : Text in natural language
Output: Text in natural language with coreference resolution

```
1  doc ← stanfordParser.getdocument(text);
2  corefChains ← doc.corefChains();
3  for correference em corefChains do
4      reference ← null;
5      sentence ← doc.sentences().get(correference.indexSentence − 1);
6      size ← correference.endIndex − correference.startIndex;
7      listWords ← words in sentence;
8      if correference is a name then
9          reference ← correference;
10     if correference is a pronoum then
11         if correference does not start with "'s"and matches
            ("mine/my/your/her/our/yours/his/him/her/its/their/theirs/ours/them") then
12             possessive ← "'s";
13         else
14             possessive ← null;
15         for word ← correference.startIndex − 1 até
            correference.endIndex − 1 + size do
16             if word == correference.startIndex − 1 e reference!= null then
17                 listWords [word] ← reference + possessive;
18             else
19                 listWords [word] ← null;
20     newSentence ← text (listWords);
21     doc = stanfordParser.getDocument(doc.text() replacing sentence by
        newSentence);
22 return doc.text();
```

3.1.2.3 Normalização da árvore de dependência - *Tree Normalization*

Finalmente, o texto já pré-processado com resolução de conjunção, união e correferência é transformado em uma árvore de dependência através do *Stanfor Parser*. O subcomponente *Tree Normalization* é responsável por esse processo de transformação e fornecer essa árvore como entrada para o próximo componente 3) *DL Parsing*. Assim a Figura 17 exibe a árvore de dependência gerada a partir do texto:

```
Joe Smith's flight left at 3:00pm on July 10th, 2017.
```

Na Figura 17 o texto é anotado conforme sua função sintática, onde cada nó da árvore representa uma função sintática. Por exemplo o trecho *(NP (NNP Joe) (NNP Smith) (POS 's))* começa com um NP que representa um sintagma nominal, ou frase nominal. Esse nó possui três filhos. O primeiro *(NNP Joe)* indica a função sintática de

```
(ROOT
  (S
    (NP
      (NP (NNP Joe) (NNP Smith) (POS 's))
      (NN flight))
    (VP (VBD left)
      (PP (IN at)
        (NP (CD 3:00) (NN pm)))
      (PP (IN on)
        (NP (NNP July) (CD 10th) (, ,) (CD 2017))))
    (. .)))
```

Figura 17 – Árvore normalizada com resolução de correferência

```
(ROOT
  (S
    (NP
      (NP (NNP Joe) (NNP Smith) (POS 's))
      (NN flight))
    (VP (VBD leave)
      (PP (IN at)
        (NP (CD 3:00) (NN pm)))
      (PP (IN on)
        (NP (NNP July) (CD 10th) (, ,) (CD 2017))))
    (. .)))
```

Figura 18 – Árvore normalizada utilizando o `WordNet`

um substantivo próprio, bem como o seguindo filho (NNP Smith). O terceiro filho (POS 's) representa a partícula de posse *'s* indicando que os *Joe Smith* é possuidor de algo. A lista completa das anotações sintáticas podem ser encontradas no Apêndice B.1 e B.2.

Apos a resolução de correferência, é utilizado o `WordNet` para transformar os verbos em suas respectivas formas no infinitivo e os substantivos para o singular (Vide Figura 18). Esse processo permite melhor identificar relações de subsunção e/ou de propriedades. Uma relação de subsunção é aquela que utiliza o verbo *to be* para compor a relação, por exemplo *dog is an animal*, verbo *is*. Relações de propriedade são aquelas formadas por verbos diferentes do *to be*, por exemplo *the dog has eyes*, verbo *has*.

Apresentamos na Figura 18 o verbo que antes era *left* se tornou *leave*, ou seja, a forma de *left* no infinitivo.

Por fim a última tarefa de normalização da árvore de dependência sintática é a realização de uma reestruturação dos nós que representam conjunção ou união, para possibilitar sua conversão em axiomas de uma melhor forma. Essa reestruturação consiste, basicamente, em transformar as conjunções (Nós da árvore sintática anotada com a *tag* CC) em

nós principais em relação aos verbos e objetos correspondentes. Consideremos a frase:

```
dog is an animal and has eyes
```

Após o processamento utilizando o `WordNet`, teremos:

```
(ROOT
  (S
    (NP (NN dog))
    (VP
      (VP (VBZ be)
        (NP (DT an) (NN animal)))
      (CC and)
      (VP (VBZ have)
        (NP (NNS eye))))))
```

Note que o verbo *has* foi substituído por *have*, o *is* por *be* e *eyes* posto no singular (*eye*). Agora observemos a árvore normalizada com reestruturação dos nós de conjunção (CC), abaixo.

```
(ROOT
  (S
    (NP (NN dog))
    (VP
      (AND
        (VP (VBZ be)
          (NP (DT an) (NN animal)))
        (VP (VBZ have)
          (NP (NNS eye)))))))
```

A seguir é exibido a extração de um axioma (realizada pelo componente *DL Parsing*, apresentado na seção seguinte) a partir da árvore apresentada na acima.

$$Dog\ SubClassOf\ Animal\ AND\ (\ has\ some\ (\ Eye\ )\ ) \tag{3.1}$$

Transformar a conjunção *AND* em pai de *(VP (VBZ be) (NP (DT an) (NN animal)))* e de *(VP (VBZ have) (NP (NNS eye)))* facilita o processo de axiomatização, pois todos os seus filhos formarão elementos da conjunção.

Ainda se tratando dos verbos, quando um verbo é identificado e sua forma no infinitivo é capturada o PUPO gera uma hierarquia entre esses verbos, que virão a ser tornar propriedades OWL. Por exemplo:

```
Frase:
dog is an animal and has eyes

Hierarquia de propriedades extraídas:
have
   |_ has
```

Desse forma o seguinte axioma é gerado:

```
has SubPropertyOf: have
```

Por fim, essa nova estrutura de árvore é repassada para o pacote (3) *DL Parsing*.

3.1.3 (3) DL Parsing

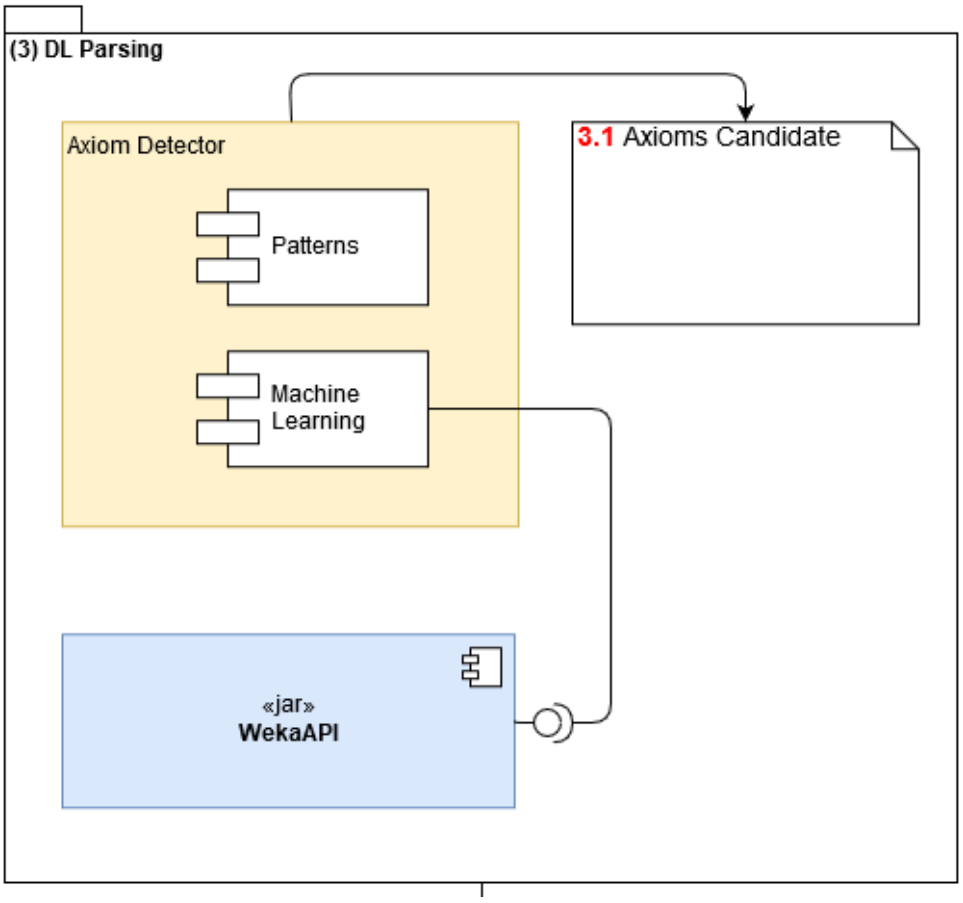

Figura 19 – DL Parsing

O pacote *DL Parsing*, Figura 19, através do componente *Axiom Detector* (Detector de Axiomas), Figura 14, realiza a conversão da árvore de dependência normalizada para axiomas em DL.

```
Frase: Dog is an animal and has eyes
Árvore:
(ROOT
  (S
    (NP (NN dog))
    (VP
      (VP (VBZ is)
        (NP (DT an) (NN animal)))
      (CC and)
      (VP (VBZ has)
        (NP (NNS eyes))))))

Candidatos:
(NP (NN dog))
(NP (DT an) (NN animal))
(NP (NNS eyes))
```

Figura 20 – Captura de frases nominais

Além do processo de axiomatização o componente *Machine Learning*, através de Aprendizagem de Máquina, busca por axiomas que não puderam ser identificados anteriormente. Por exemplo:

$$Dog\ SubClassOf\ has\ some\ (\ Eye\ ) \tag{3.2}$$

Esse processo consiste em duas abordagens, a sintática e a semântica, veremos cada uma em subseções a seguir.

3.1.3.1 Abordagem Sintática

A abordagem sintática desenvolvida tem como finalidade extrair sujeito, predicado e objeto de uma frase em linguagem natural. Para isso, fazemos uso da estrutura de árvore fornecida pelo componente *Morphosyntatic Parsing*.

Nessa árvore os nós anotados com NP representam uma frase nominal, ou seja, são candidatos a sujeito ou objeto, por exemplo.

Assim, conforme Figura 20, um bloco anotado por PP ou NP pode representar um sujeito ou objeto.

Através do subcomponente *Patterns*, no qual os elementos do sujeito se uma sentença qualquer são associados aos elementos do predicado da mesma sentença em forma de tripla (sujeito,verbo,objeto). Todas as *tags* NP ou PP são associadas entre si através de verbos anotados por VB*[14], conforme Algoritmo 3[15].

[14] O * representa que a *tag* pode apresentar quaisquer sufixos
[15] Vide tradução em C.5

Algoritmo 3: Sintatic triples selection

Input : Dependency Tree List from a text
Output: Triple List: [subject; verb; object]

```
1 triples ← [];
2 for each tree in List Tree do
3     for each subject s_i from tree do
4         for each verb v_j from tree do
5             for each object o_k from tree do
6                 triples.add([s_i; v_j; o_k]);
7 return triples;
```

Então, a partir da sentença *Dog SubClassOf Animal AND (has some (Eye))*, são extraídas as seguintes triplas:

$$(NP\ (NN\ dog)) \mid (VBZ\ be) \mid (NP\ (DT\ a)\ (NN\ animal)) \tag{3.3}$$

$$(NP\ (NN\ dog)) \mid (VBZ\ be) \mid (NP\ (NNS\ eyes)) \tag{3.4}$$

$$(NP\ (NN\ dog)) \mid (VBZ\ have) \mid (NP\ (DT\ a)\ (NN\ animal)) \tag{3.5}$$

$$(NP\ (NN\ dog)) \mid (VBZ\ have) \mid (NP\ (NNS\ eyes)) \tag{3.6}$$

Essas triplas serão através de aprendizagem de máquinas, posteriormente, classificadas como válidas ou inválidas.

3.1.3.2 Abordagem Semântica

Com a utilização de padrões é possível alcançar uma maior precisão semântica na axiomatização de sentenças em linguagem natural(RILOFF, 1996; De Azevedo et al., 2014; VÖLKER; HITZLER; CIMIANO, 2007).

Na Figura 21 exibimos um exemplo de extração.

Além de uma abordagem sintática, descrita na subseção anterior, utilizamos também uma abordagem baseada em padrões.

Os padrões utilizados pelo PUPO foram apresentados por Riloff (1996), e adaptados para serem operados sobre a árvore sintática, conforme Tabela 8.

PADRÃO	PADRÃO GERADO
((NP SUJEITO) VBD + VBN)	((NP SUJEITO) SUBCLASSOF VBN+BY SOME)
((NP SUJEITO) VBN)	((NP SUJEITO) SUBCLASSOF VBN SOME)

((NP SUJEITO) VBD + TO + VB)	((NP SUJEITO) VBD + TO + VB SOME)
(VBN + (NP OBJETO))	((NP OBJETO) SUBCLASSOF VBN + BY SOME)
(TO + VB + (NP OBJETO))	((NP OBJETO) SUBCLASSOF VB + BY SOME)
(VBN + TO + VB (NP OBJETO))	((NP OBJETO) SUBCLASSOF VBN + TO + BE + VB +BY SOME)
(VBG + (NP OBJETO))	((NP OBJETO) SUBCLASSOF VBG + BY + SOME)
((NP SUJEITO) + VBD + (NP OBJETO))	((NP SUJEITO) + SUBCLASSOF + (NP OBJETO))
(VB + IN + (NP OBJETO))	((NP OBJETO) SUBCLASSOF VB + IN SOME)
(VBN + IN + (NP OBJETO))	((NP OBJETO) SUBCLASSOF VBN + SOME)
(VBD + VBN + IN + (NP OBJETO))	((NP OBJETO) SUBCLASSOF VBN + BY SOME)

Tabela 8 – Padrões para seleção de triplas semânticas

Analisamos os trabalhos relacionados e dois trabalhos também utilizam padrões para extração(De Azevedo et al., 2014; VÖLKER; HITZLER; CIMIANO, 2007). Contudo os padrões apresentados por De Azevedo et al. (2014) e também por Völker, Hitzler e Cimiano (2007) são alcançados a partir da abordagem sintática, e por isso não foram utilizados na abordagem por padrões, por exemplo:

Em De Azevedo et al. (2014):

$$SN_1\ (are\ a|are\ an|are|is\ a|is\ an|is)\ SN_2\ (or|and)\ SN_3 \tag{3.7}$$

No padrão 3.7, onde *SN* é um sintagma nominal, a sentença *dog is a mammal or a carnivore animal* será analisada como:

$$SN_1 = dog \tag{3.8}$$

$$(is\ a) \tag{3.9}$$

$$SN_2 = mammal \tag{3.10}$$

$$(or) \tag{3.11}$$

$$SN_3 = carnivore\ animal \tag{3.12}$$

```
Padrão: (NP *) (VP (VBZ *) (NP *))
Árvore:
(ROOT
  (S
    (NP (NN dog))
    (VP
      (VP (VBZ is)
        (NP (DT an) (NN animal)))
      (CC and)
      (VP (VBZ has)
        (NP (NNS eyes))))))
Extração:
1 - (NP (NN dog)) (VP (VBZ is) (NP (DT an) (NN animal)))
2 - (NP (NN dog)) (VP (VBZ has) (NP (NNS eyes)))
```

Figura 21 – Exemplo de extração utilizando o padrão (NP *) (VP (VBZ *) (NP *))

A abordagem sintática é capaz de substituir esse padrão, gerando:

$$(sujeito = dog, verbo = is, objeto = a\ mammal\ or\ a\ carnivore\ animal) \tag{3.13}$$

Em Völker, Hitzler e Cimiano (2007):

Esse fato (da abordagem sintática ser capaz de cobrir alguns padrões) também ocorre em Völker, Hitzler e Cimiano (2007), por exemplo, no padrão como:

$$V_0\ Num\ NP(obj)_0 \tag{3.14}$$

Onde V_0 representa um verbo, Num um número qualquer e $NP(obj)_0$ o objeto. Na sentença *dog has 2 eyes* o padrão 3.14 extrairia:

$$V_0 = has \tag{3.15}$$

$$Num = 2 \tag{3.16}$$

$$NP(obj)_0 = eyes \tag{3.17}$$

Enquanto isso a abordagem sintática extrairia:

$$(sujeito = dog, verbo = has, objeto = 2\ eyes) \tag{3.18}$$

Observado esse fato, os padrões de De Azevedo et al. (2014) e de Völker, Hitzler e Cimiano (2007) não foram adaptados para a abordagem semântica, pois a abordagem sintática já cobre, no mínimo, sua maioria.

Cada padrão é responsável por gerar um tipo de axioma específico, como pode ser visto na Tabela 9 (Vide tradução na Tabela 31, Apêndice C.6).

Sentença	Axioma (DL em *Manchester Syntax*)
the victim was murdered.	Victim SubClassOf MurderedBy some
the perpetrator bombed.	Perpetrator SubClassOf bomb some
the perpetrator attempted to kill.	Perpetrator SubClassOf AttemptToKill some
killed victim.	Victim SubClassOf KilledBy some
to kill victim.	Victim SubClassOf KilledBy some
threatened to attack the target.	Target SubClassOf ThreatenedToBeAttackBy some
killing the victim.	Victim SubClassOf KilledBy some
fatality was victim.	Fatality SubClassOf Victim
bomb against the target.	Target SubClassOf BombedBy some
killed with instrument.	Instrument SubClassOf Kill some
was aimed at target.	Target SubClassOf AimedBy some

Tabela 9 – Exemplos de axiomas gerados pelos padrões semânticos

Para extrair informação do texto em LN através desses padrões foi proposto o Algoritmo 4(Vide tradução no Apêndice C.5), o qual verifica se a definição dada pelo usuário contém algum padrão conhecido, onde $s_i; v_j, o_k$ são sujeito, verbo e objeto respectivamente.

Algoritmo 4: Semantic triples selection

Input : Dependency Tree List of a text
Output: Triple List: [subject; verb; object]

```
1 triples ← [];
2 for each tree from List Tree do
3     for each semantic pattern p_i do
4         while tree contains p_i do
5             triple ← [s_i; v_j; o_k]_{p_i};
6             triples.add(triple);
7             remove triple from tree;
8 return triples;
```

Diferente dos padrões apresentados por De Azevedo et al. (2014) e Völker, Hitzler e Cimiano (2007), os apresentados por Riloff (1996) contém uma maior complexidade em relação aos verbos. Na Figura 22 exibimos um exemplo de extração.

Nota-se que o verbo *murdered* por estar no particípio apresenta um valor semântico diferente do seu estado no infinitivo. Assim foi criada uma propriedade (*murderedBy*)

```
Sentença: the victim was murdered
Axioma extraído: Victim SubClassOf murderedBy some Thing
```

Figura 22 – Exemplo de extração por meio de padrões apresentados por Riloff (1996)

indicando adicionando a partícula *By* ao verbo *murdered*. Informação essa que não está explícita no texto, mas que semanticamente está presente e correta. Essa complexidade não seria completamente capturada pela abordagem sintática.

Algumas triplas extraídas pelo PUPO, a princípio, podem apresentar semântica equivocada, pois esses padrões estão preocupados em sua maioria em identificar a estrutura de um trecho do texto dado sua função gramatical. Por exemplo:

```
ARTIGO/Um SUBSTANTIVO/papagaio VERBO/voa ./.
ARTIGO/Um SUBSTANTIVO/pão VERBO/voa ./.
```

Um padrão do tipo *ARTIGO+SUBSTANTIVO+VERBO* extrairia *Um pão voa* como sendo uma relação válida, mas semanticamente equivocada. Para sanar esse problema, e induzir semântica, foi construída uma `SVM` (*Support Vector Machine*, Máquina de Vetor Suporte) para classificação dos padrões sintáticos e outra para os padrões semânticos com um modelo treinado através do WEKA[16] – O `WEKA` é uma ferramenta que permite realizar mineração de dados e aprendizagem de máquina utilizando interface gráfica, isso facilita o processo de construção de modelos utilizando os mais diferentes *algoritmos de aprendizagem de máquina*. Esse modelo é utilizado pelo subcomponente *Machine Learning* através de uma interface para a *API* do `WEKA`.

Como exemplo a `SVM` classifica as triplas 3.3 e 3.6 como válidas e 3.4 e 3.5 como inválidas. Resultando nas triplas 3.19 e 3.20.

$$(NP\ (NN\ dog)) \mid (VBZ\ is) \mid (NP\ (DT\ an)\ (NN\ animal)) \tag{3.19}$$

$$(NP\ (NN\ dog)) \mid (VBZ\ has) \mid (NP\ (NNS\ eye)) \tag{3.20}$$

Após a escolha das triplas novas árvores sintáticas são formadas a partir das mesmas, utilizando o sujeito, verbo e objeto detectados, Figura 23 e 24.

Posteriormente transformando em DL:

$$Dog\ SubClassOf\ Animal \tag{3.21}$$

$$Dog\ SubClassOf\ has\ some\ (Eye) \tag{3.22}$$

Como saída o *Axiom Detector* fornece um conjunto de axiomas extraídos da árvore de dependência normalizada.

[16] https://www.cs.waikato.ac.nz/ml/weka/

```
(ROOT
  (S
    (NP (NN dog))
    (VP (VBZ is)
      (NP (DT an) (NN animal)))))
```

Figura 23 – Árvore de dependência sintática da primeira relação identificada

```
(ROOT
  (S
    (NP (NN dog))
    (VP (VBZ has)
      (NP (NNS eyes)))))
```

Figura 24 – Árvore de dependência sintática da segunda relação identificada

3.1.3.3 Utilização de Aprendizado de Máquina para seleção de relações

Utilizamos AM com o intuito de classificar relações do tipo sujeito-predicado-objeto entre válidas e inválidas. As relações candidatas são extraídas através dos métodos sintáticos e semânticos discutidos nas duas subseções anteriores. As instâncias que serão avaliadas por AM tem a forma $s_i, r_k^i, c0, 1$ onde s_i é um sentença, r_k^i a k-ésima relação candidata da sentença e $c0, 1$ é a classificação entre valida e inválida. c assume valor 1 quando a SVM classifica a relação como válida, ou 0 caso contrário. Por exemplo:

```
"acre is a unit of area used in English-speaking countries"
"acre be unit of area"        "1"
```

Onde:

```
Sentença: "acre is a unit of area used in English-speaking countries"
Relação: "acre be unit of area"
Classificação: "1"
```

Dessa forma interpretamos o problema de extrair relações válidas como um problema de classificação.

Foi utilizado como pré-processamento, largamente difundido na área de aprendizagem de máquina, *StringWord2Vector*, TF-IDF e 2-gram(BALTRUSAITIS; AHUJA; MORENCY, 2018; GUO; YANG, 2016; WOHLGENANNT; MINIC, 2016).

Para a SVM poder operar sobre as instâncias é realizado um filtro de transformação na sentença e na relação candidata. O filtro utilizado possui a finalidade de transformar

o texto em um vetor numérico que representa o texto. Esse filtro foi gerado a partir do *Weka* com as seguintes configurações:

```
TF = TRUE
IDF = TRUE
NGRAM = 1-2gram
WORDSTOKEEP = 100000
```

TF e IDF marcados como TRUE representam a utilização desses algoritmos durante a realização do filtro. NGRAM = 1-2gram indicam a utilização de unigramas e bigramas para composição das sentenças. Consideremos a seguinte frase e sua representação em 1-2gram:

```
sentença: o gato é bonito
representação em 1-2gram: [o, gato, é, bonito, o gato, gato é, é bonito]
```

A configuração WORDSTOKEEP = 100000 aponta que o algoritmo deve manter até 100000 (cem mil) palavras e bigramas para representar o vetor final que será utilizado pela SVM.

Esse filtro de *Word2Vec* é aplicado na sentença e na relação candidata presente na instância a ser classificada.

A `SVM` foi composta por *kernel RBF*(DADGAR; ARAGHI; FARAHANI, 2016) com valores 100 para o C (*Cost*, custo) e 0.01 de gamma (Vide Capítulo 2)[17]. A base de treino foi classificada manualmente e o modelo gerado é utilizado pelo subcomponente *Machine Learning* através da *WekaAPI*. Mais detalhes serão visto na configuração experimental deste trabalho. Veremos agora detalhes do pacote de raciocínio, o pacote *(4) Reasoner Manager*.

3.1.4 (4) Reasoner Manager

Esse pacote, Figura 25, é responsável por acoplar as funcionalidades relacionadas ao processo de raciocínio automático de nosso *chatterbot*. Dado as ontologias modeladas pelo usuário, o componente *Inconsistent Axioms Detector* (Detector de Axiomas Inconsistentes) verifica se existe algum axioma inconsistente ou inferido pelo motor de raciocínio (Pellet) nas bases de conhecimento.

O `Pellet` fornece recursos para o subcomponente *Logic Reasoning* (Raciocínio Lógico), responsável pelo raciocínio em DL. Por sua vez, o *Inconsistent Axioms Detector* requisita recursos do *Logic Reasoning*. Por exemplo ao inserir o seguinte axioma nas bases de conhecimento:

[17] Para mais detalhes sobre como foram estimados esses parâmetros Vide Seção 6.1.2

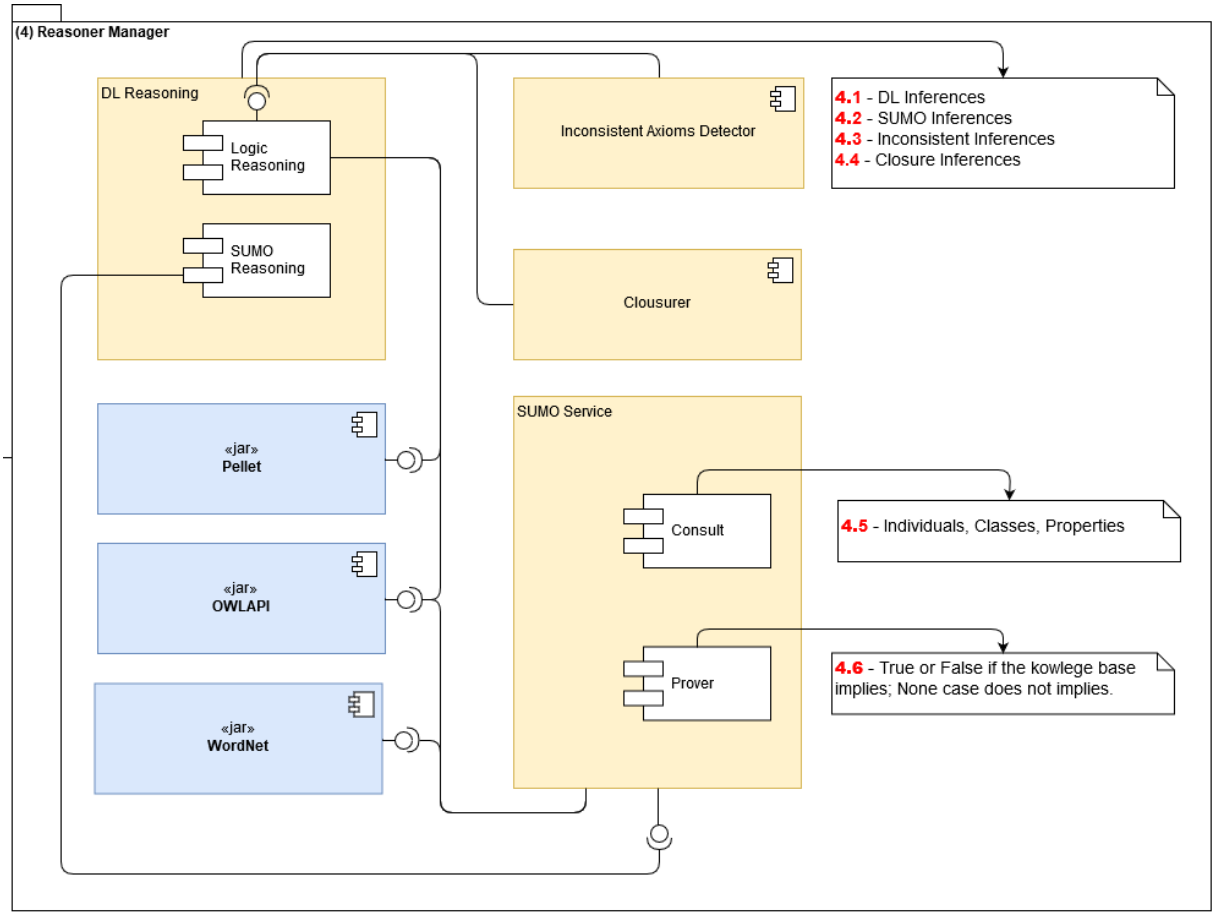

Figura 25 – Reasoner Manager

```
Dog SubClassOf ( not ( has some ( Eye ) ) )
```

Contradizemos o axioma 3.22, assim o sistema pergunta qual dos dois axiomas é válido, pois o mecanismo de raciocínio `Pellet` identificou que os dois axiomas juntos deixam as bases de conhecimento inconsistentes. Finalmente, através da resposta do usuário o axioma problemático é removido e as bases de conhecimento voltam ao seu estado consistente.

O subcomponente *SUMO Reasoning* (Raciocínio SUMO) tem um papel fundamental no aprendizado de Ontologias, o qual consiste em validar os axiomas extraídos através de aprendizagem de máquina. Essa validação se torna necessária, pois algoritmos de aprendizagem de máquina apresentam em taxa de erro que deve ser contornada. A solução desenvolvida consiste em verificar se o axioma em questão é válido semanticamente segundo a `SUMO`.

O *SUMO Reasoning* requisita ao componente *SUMO Service* (Serviço SUMO) a validação de axiomas extraídos por aprendizagem de máquina. O subcomponente *Prover* (Provador) verifica se, segundo a SUMO, o axioma requisitado é semanticamente correto, segundo o Algoritmo 5(Vide Apêndice C.6).

A validação dos axiomas sob a `SUMO` é realizada de forma assíncrona se tratando da

Algoritmo 5: Axioms validation supported by SUMO

Input : Dependency Tree List from a text
Output: Valid axioms supported by SUMO

```
1 approvedAxioms ← [];
2 axioms ← [];
3 axioms.add(extractSintaticAxioms(text));
4 axioms.add(extractSemanticAxioms(text));
5 for each axiom a_i ∈ axioms do
6     if SUMO ⊨ a_i then
7         approvedAxioms.add(a_i);
8 return approvedAxioms;
```

relação com o usuário. As provas de axiomas podem ser custosas em relação ao tempo e processamento, dependendo do tamanho e expressividade envolvida na base de conhecimento. Assim para permitir que o usuário não espere todo o tempo de prova o *Reasoner Manager* requisita a prova do axioma a um *Servidor Rest*[18] desenvolvido em Java para esse propósito e com o `Pellet` como raciocinador. Posteriormente caso o axioma seja válido segundo a `SUMO`, o mesmo é inserido como inferência. Deste modo tais axiomas serão avaliados pelo usuário quando o sistema questionar o usuário sobre a veracidade dos axiomas inferidos.

Como entrada o servidor (componente *SUMO Service*) recebe axiomas escritos em DL (Manchester Syntax). Utilizando-se da OWL-API e o Pellet Reasoner o serviço realiza dois tipos de consulta:

Através do subcomponente *Prover*: Consultas de resultado *True* (Verdadeiro) ou *False* (Falso) que demonstra se um axioma é válido para a base de conhecimento. Por exemplo, ao se consultarmos o seguinte axioma:

```
Canine SubClassOf Mammal
```

Receberemos como resultado da consulta o valor *true* pelo serviço, indicando que o axioma é válido segundo a `SUMO`.

Através do subcomponente *Consult*: Consultas de SuperClasses, EquivalentClasses, SubClasses e Indivíduos, que são formadas por consultas em DL.

Segundo Figura 26 por exemplo[19], devido aos axiomas requisitados pelo *Reasoner Manager* para validação na `SUMO` serem extraídos de linguagem natural e herdarem a mesma variância o `WordNet` é utilizado no processo de prova. O `WordNet` tem como papel fundamental realizar o mapeamento de termos que não estão presentes na `SUMO` por sinônimos. Assim quando uma palavra que não estiver presente na `SUMO`, mas possuir um sinônimo presente, então o axioma é validado com esse novo termo. Por exemplo:

[18] Informações sobre desenvolvimento de serviços *Rest* podem ser encontrados em https://bit.ly/2Vee2Px
[19] Para tradução vide Apêndice C.7

```
QUERY:
    Canine and Mammal

RESPONSE:
    SuperClasses: Carnivore, Thing, WarmBloodedVertebrate,
                  CorpuscularObject, Organism, Entity, Animal,
                  Mammal, Agent, Vertebrate, OrganicObject,
                  Object, OrganicThing, SelfConnectedObject,
                  Physical

    EquivalentClasses: Canine

    SubClasses: Nothing, DomesticDog, Fox

    Instances: [NONE]
```

Figura 26 – Exemplo de consulta utilizando manchester sintax

```
Sentença em linguagem natural: Agency is Business or Formation
Axioma de Consulta: Agency SubClassOf Business or Formation
Resposta: true
```

Figura 27 – Exemplo de utilização de sinônimos para prova de axiomas

Conforme Figura 27, mesmo que a palavra *Formation* não esteja presente na SUMO, através do WordNet é possível obter *Organization* como seu sinônimo. Assim a consulta terá como resultado o mesmo valor que o axioma "*Agency SubClassOf Business or Organization*".

Além disso, todos os termos da SUMO, para validação da proposta desenvolvida, são transformados para caixa baixa, isto é, todos os caracteres em sua forma minúscula e o caractere "_" (underline) removido. Esse método básico de mapeamento para a SUMO foi nomeado como Weak WordNet Ontology Mapping (em tradução livre, Mapeamento Fraco de Ontologias pelo WordNet).

O subcomponente *closurer* (Vide Figura 25) fornece um mecanismo para construção de axiomas de fecho. O axioma de fecho é uma axioma que possui como objetivo realizar restrições nas Ontologias de forma que o raciocinador consegue provar afirmações conforme a base de conhecimento.

Por exemplo, considere os seguintes axiomas:

1. Dog SubClassOf: Carnivore
2. Carnivore SubClassOf: eats some Meat
3. Meat SubClassOf: not Plant

Sem o axioma de fecho seria possível inserir na ontologia que *Dog SubClassOf: eats some Plant* (Cão SubClasseDe: come alguma planta). Esse fato não seria verdadeiro, porém ao criar o axioma de fecho o PUPO pergunta ao usuário:

- PUPO: Is it right to say: In ontology ontology.iri Carnivore SubClassOf: eats only meat ?

E dessa forma, a única coisa a ser comida por um Cão (Dog) é algum tipo de Carne (Meat).

Esses axiomas de fecho são fornecidos ao componente *DL Reasoning* com inferências que serão validadas com o usuário através de diálogo. Ainda com esse exemplo, o *Logic Reasoning* (Raciocinador Lógico, componente que faz uso direto do `Pellet`) insere na ontologia:

- Plant DisjointWith: Meat

O PUPO nesse momento através do *Pellet* obteve a informação (axioma derivado de *Carne SubClasseDe: não Planta*) Planta é um classe disjunta de, ou seja não pode ser deduzido como, Carne.

Neste capítulo apresentamos a arquitetura do PUPO, seus componente e suas funcionalidades. No Capítulo 4 apresentamos o PUPO em uso.

4 CHATTERBOT EM USO

Neste capítulo apresentamos a interface gráfica do PUPO, acessível através de um navegador web[1]. Veremos ainda como o usuário pode interagir com o PUPO, modelar conhecimento e obter informações sobre as ontologias modeladas.

4.1 INTERFACE DO PUPO

O sistema PUPO é acessado por de um navegador web. Para acessá-lo é necessário realizar um cadastro, que pode ser feito pelo próprio usuário na tela de login (Vide Figura 28).

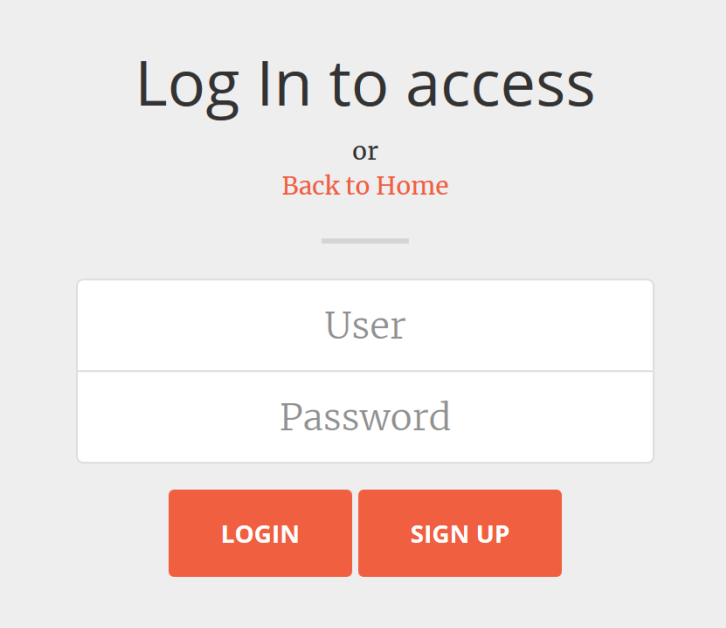

Figura 28 – Login

A tela de login, como na maioria dos sistemas, necessita de dados de *E-mail* e Senha (*Password*) para poder acessar as funcionalidades do sistema. Além disso possui o botão *Signup* (Inscriver-se) que redireciona o usuário para o cadastro.

Na tela de cadastro, Figura 29, é necessário informar:

- E-mail;
- Name (Nome);
- Password (Senha);
- Comfirm Password (Comfirmar Senha);
- Profile (Perfil)

[1] Para acessá-lo é necessário executar o projeto em um servidor ou computador local

– Expert (Especialista);

– Engineer (Engenheiro);

Os dados de perfil caracterizam o usuário, que pode ser um especialista de domínio, um engenheiro de conhecimento, ou ambos, etc.

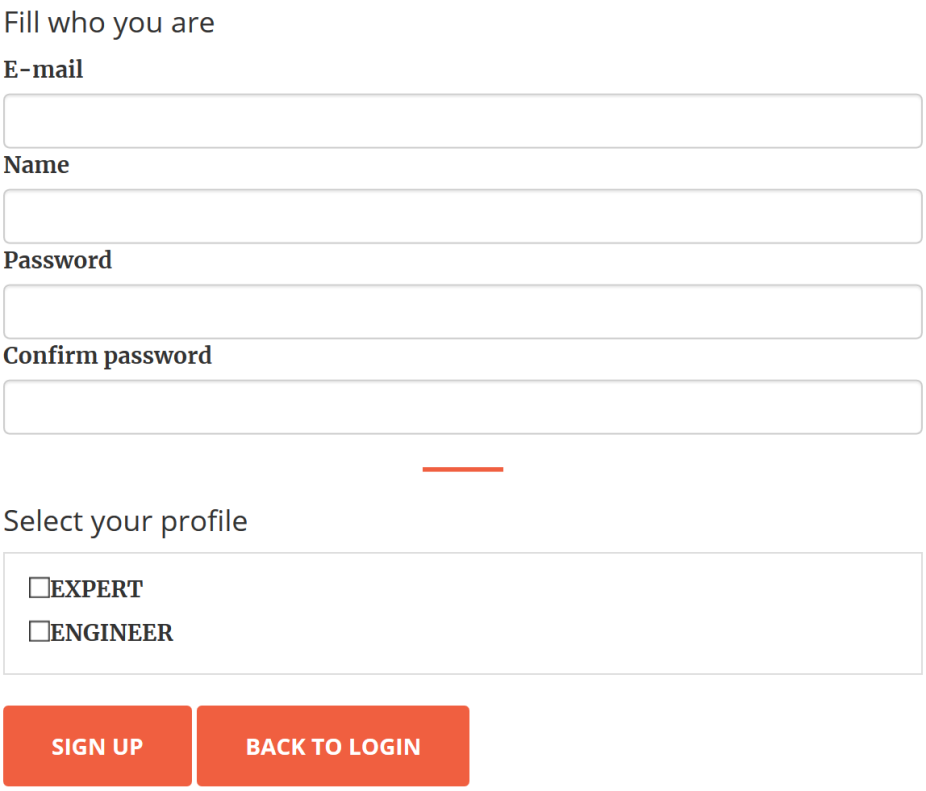

Figura 29 – Cadastro

4.1.1 Interface do usuário para criação e refinamento de Ontologias

Após a realização do login, o usuário acessa a tela principal da qual tem acesso as funcionalidades do PUPO.

Apresentamos na Figura 30 a tela principal e seus componentes enumerados. Cada número representa um componente visual representativo de informações das bases de conhecimento ou do fluxo de diálogo.

1. Campo de consistência - Apresenta dois estados: *Consistent* quando a Ontologia está consistente e *Inconsistent* quando inconsistente;

2. IRI - Exibe a IRI da Ontologia modelada;

3. Information (Informação) - Exibe informações de número de axiomas, classes, propriedades e indivíduos, nessa ordem.

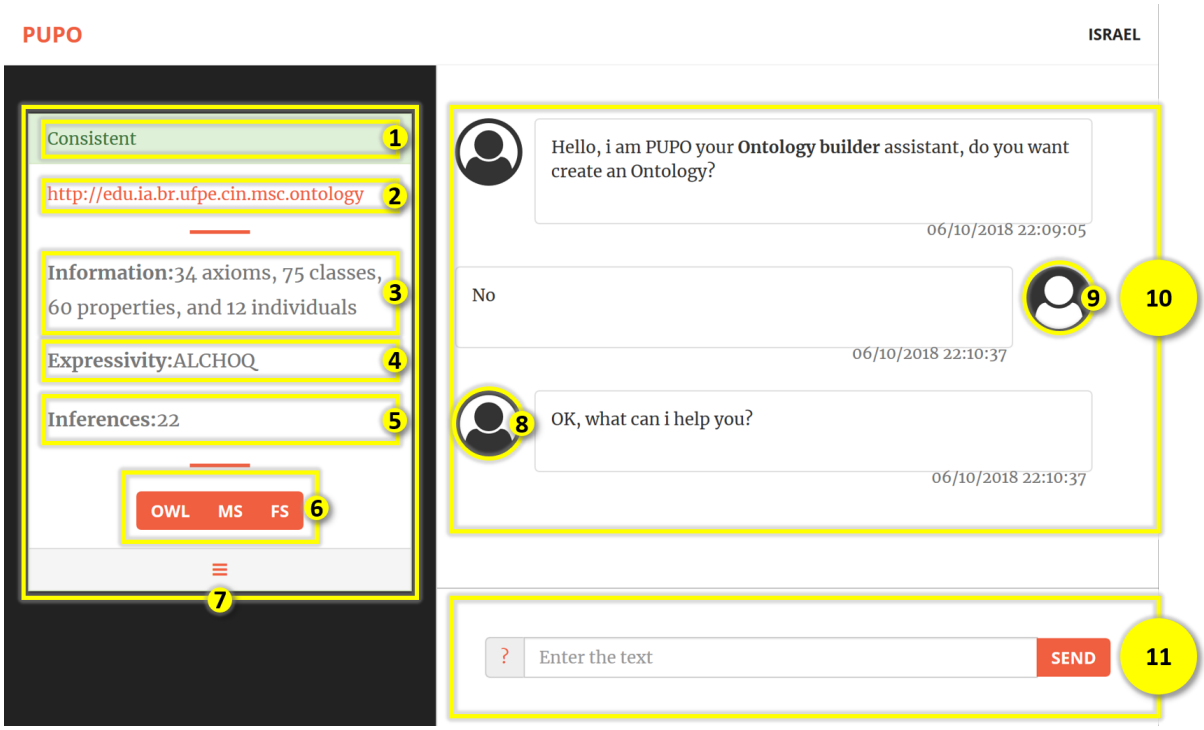

Figura 30 – Tela principal do PUPO

4. Expressivity (Expressividade) - Mostra a expressividade da Ontologia modelada;
5. Inferences (Inferências) - Expõe o número de inferências ocorridas na base do conhecimento;
6. Botões de formato - Esses 3 (três) botões exibem os formatos de visualização da Ontologia, são eles: OWL, Ontology Web Language (Linguagem Web para Ontologias), MS (Manchester Syntax, Sintaxe de Manchester) e FS (Functional Syntax, Sintaxe Funcional). Mais detalhes serão discutidos na Seção 4.2 deste Capítulo;
7. Quadro de Ontologia - Esse quadro representa uma Ontologia modelada e nele estão inclusos todos os itens citados anteriormente;
8. Bot Icon (Ícone do Bot) - Ícone que representa a figura imaginária do PUPO;
9. User Icon (Ícone do Usuário) - Ícone que representa a imagem do usuário;
10. Histórico de Diálogo - Exibe o histórico da conversa do PUPO com o usuário;
11. Caixa de Diálogo - Campo no qual o usuário insere o texto para interação com o PUPO.

4.2 EXEMPLOS DE USO

Nesta seção apresentamos alguns exemplos de uso com algumas das sentenças utilizadas simulando diálogos, algumas sentenças utilizadas nos exemplos foram extraídos de Völker, Hitzler e Cimiano (2007).

4.2.1 Criando Ontologias

A primeira tarefa do *chatterbot* é realizar uma apresentação ao usuário, e posteriormente indicar a possibilidade de criar sua Ontologia (Vide Figura 31).

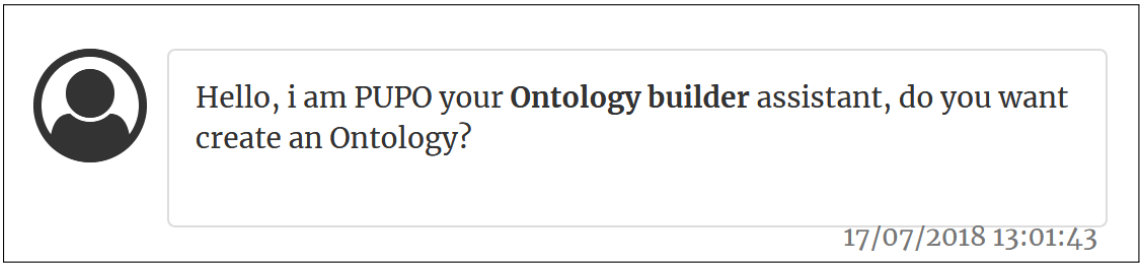

Figura 31 – Diálogo de apresentação

Ao clicar no botão *YES* é requisitado ao usuário a IRI ou nome da Ontologia (Vide Figura 32). No exemplo foi informada a IRI *http://edu.ia.br.ufpe.cin.msc.ontology*.

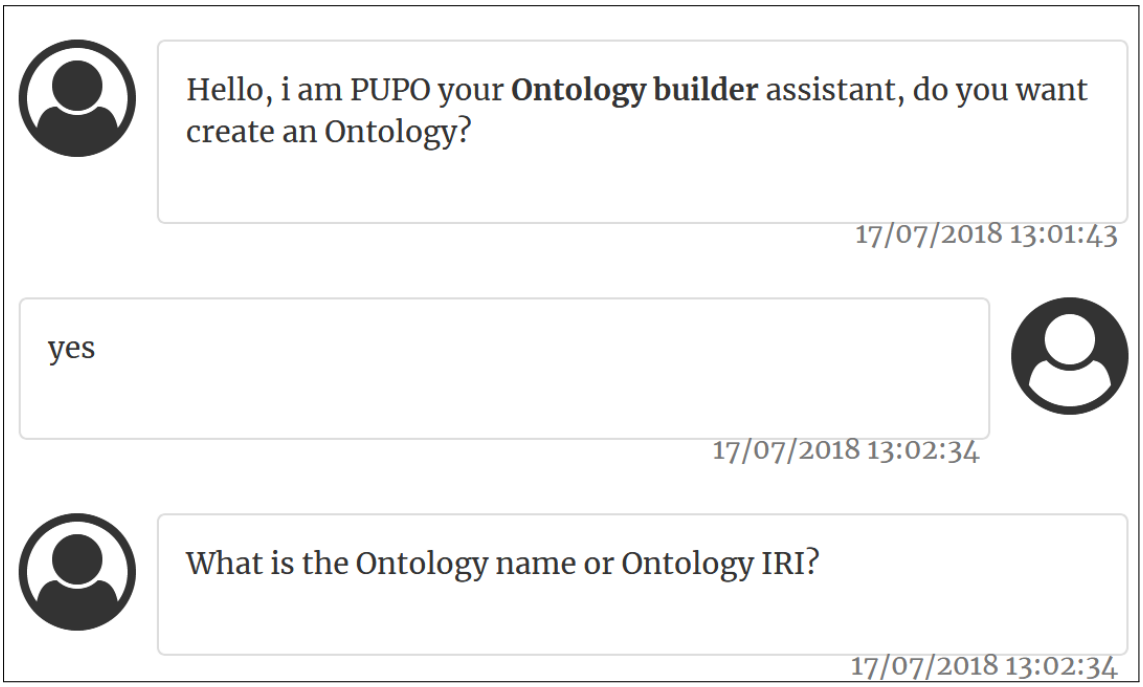

Figura 32 – Diálogo para requisição de IRI na criação de Ontologia

Após a criação da Ontologia, o sistema pergunta se o usuário deseja criar outra (Vide Figura 33). No caso do exemplo a resposta dada pelo usuário foi *NO* (Não).

Após o processo de criação da(s) Ontologia(s) o sistema pergunta ao usuário se o mesmo deseja informar fatos sobre o domínio de conhecimento a ser modelado (Vide

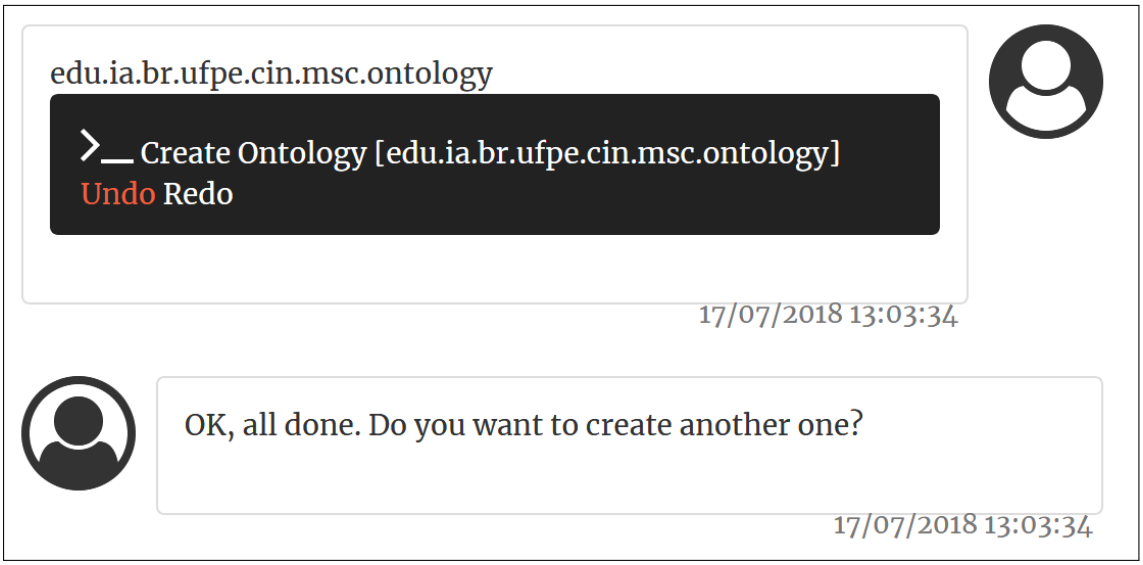

Figura 33 – Diálogo para continuar criando Ontologia a partir de nova IRI

Figura 34). Caso o usuário responda *YES* (Sim) o PUPO responde ao usuário *Tell me, write in natural language*, esperando por uma definição em linguagem antural. O texto em linguagem natural passa por todos os processos descritos nas seções discutidas no capitulo anterior e assim é transformado em Ontologia (OWL DL).

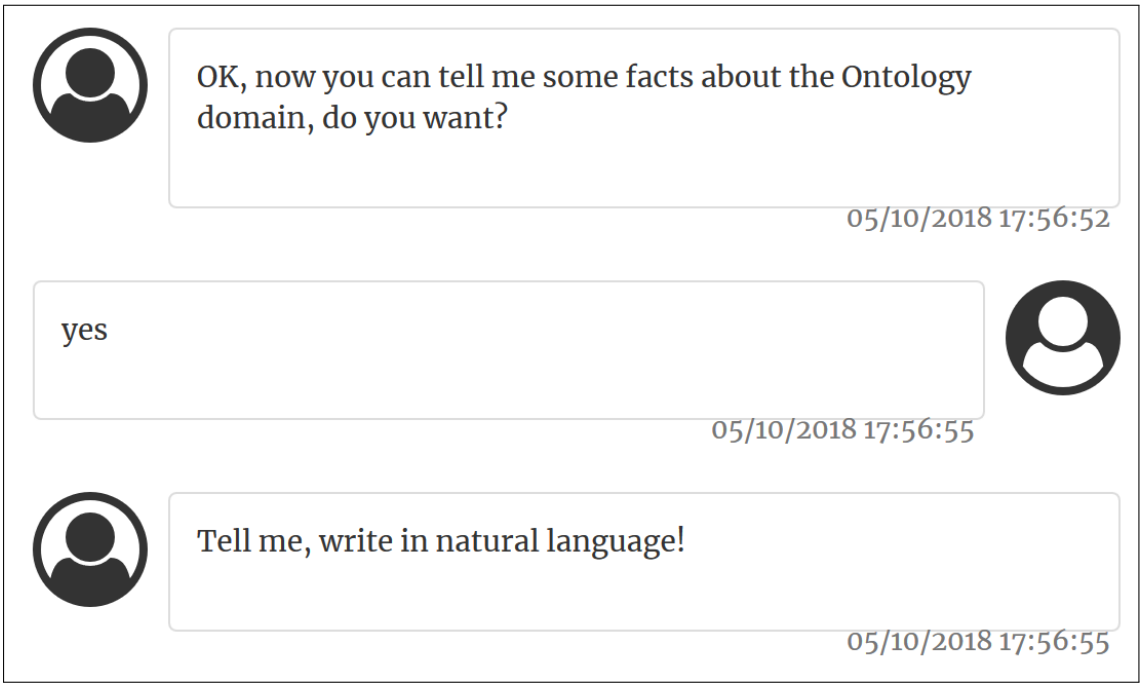

Figura 34 – Diálogo para inserir fatos na(s) Ontologia(s)

Nesse caso foi informado pelo usuário a frase *sustained use is a continuing use without severe or permanent deterioration in the resources* (uso sustentado é um uso contínuo sem deterioração grave ou permanente dos recursos). Além dessa, outra forma de inserir fatos a partir de linguagem natural é informando como entrada *"classify 'text'"*, onde *'text'* é

um texto em linguagem natural.

Tanto a inserção de sentenças na apresentação do PUPO quanto utilizando *classify* possuem o mesmo resultado. O *classify* é fornecido ao usuário para que venha a ser mais rápido solicitar ao PUPO que insira uma nova informação, fornecida em LN, na ontologia.

Na Figura 35 é possível observar a resposta do sistema ao informar a entrada *classify sustained use is a continuing use without severe or permanent deterioration in the resources* (classificar uso sustentado é um uso contínuo sem deterioração grave ou permanente dos recursos). Após o processo de axiomatização o PUPO pergunta se deseja inserir mais alguma coisa, caso positivo o usuário informa outra sentença a ser processada, e assim por diante até não haver mais fatos que o usuário deseja modelar.

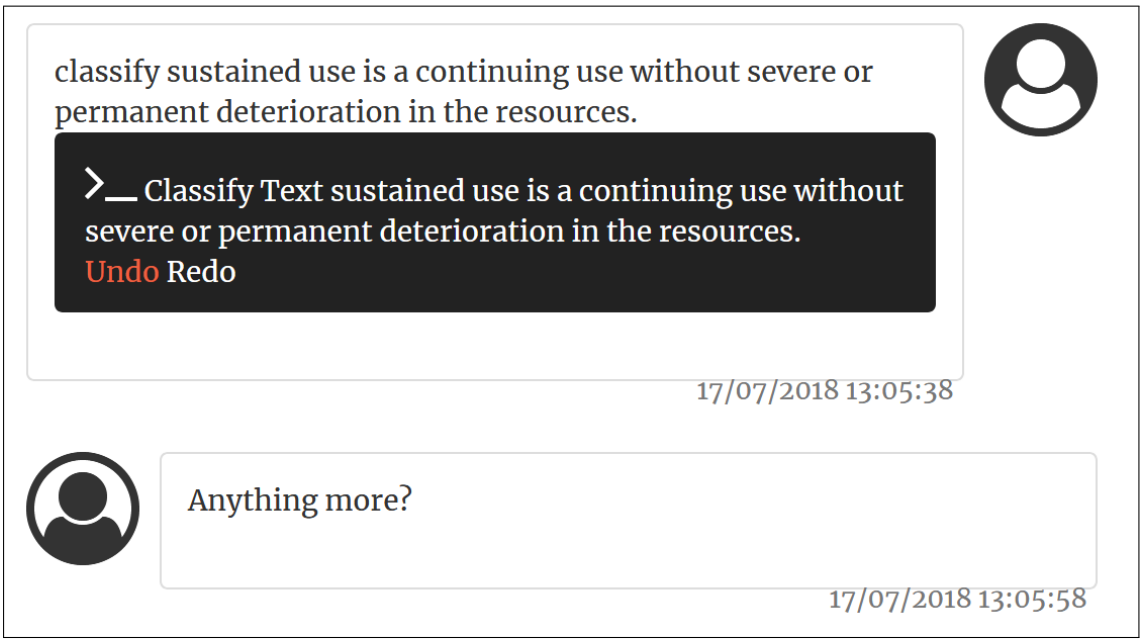

Figura 35 – Diálogo para fatos da Ontologia, usando *Classify*

Após a inserção da sentença anterior, no quadro de Ontologias à esquerda da tela de diálogo, a Ontologia apresentou expressividade ALCH[2], 2 (dois) axiomas, 4 (quatro) classes e 5 (cinco) propriedades (Vide Figura 36).

Exibimos na Figura 37 (Para o código em OWL vide Apêndice G.1) a codificação da Ontologia na sintaxe de *Manchester*. O sistema fornece a possibilidade de salvar a(s) Ontologia(s) modelada(s) em 3 (três) formatos. O primeiro é em OWL, largamente difundido na comunidade. O segundo é em *Manchester Syntax* (MS), que provê uma codificação mais enxuta. O último é o formato de sintaxe funcional (FS), semelhante a MS porém tem como objetivo modelar os axiomas em forma de funções.

Foi inserida uma nova Ontologia com a IRI *http://edu.ia.br.ufpe.cin.msc.ontology2* através do diálogo:

[2] Essa expressividade é calculada automaticamente utilizando o Pellet, que realiza cálculo de lógica e a fornece para o PUPO por meio do componente *Logic Reasoning* (Vide Seção 3.1)

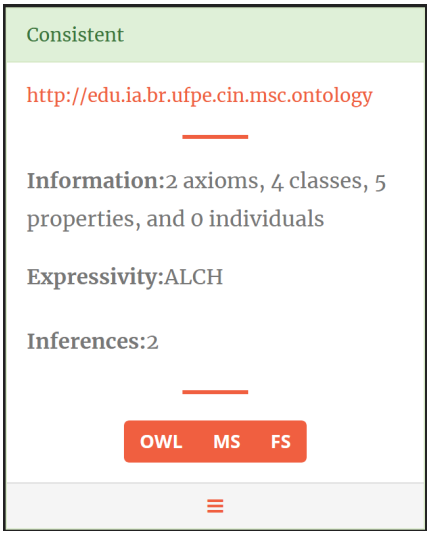

Figura 36 – Ontologia modelada, informações da base de conhecimento

```
Usuario: create ontology http://edu.ia.br.ufpe.cin.msc.ontology2
```

Posteriormente, foram inseridas as seguintes sentenças:

1. *data are facts that result from measurements or observations.*
2. *InternalRateOfReturn is a financial or economic indicator of the net benefits expected from a project or enterprise, expressed as a percentage.*
3. *vector is an organism which carries or transmits a pathogen.*
4. *juvenile is a young fish or animal that has not reached sexual maturity.*
5. *tetraploid is a cell or organism having four sets of chromosomes.*
6. *pair Trawling is a bottom or mid-water trawling by two vessels towing the same net.*

Assim as sentenças foram axiomatizadas e inseridas em ambas as Ontologias *http://edu.ia.br.ufpe.cin.msc.ontology* e *http://edu.ia.br.ufpe.cin.msc.ontology2*.

Para demonstrarmos o controle que o usuário possui sobre as ontologias, mostramos na Figura 38 a ontologia *ontology2* desabilitada através do diálogo:

```
Usuario: disable ontology http://edu.ia.br.ufpe.cin.msc.ontology2
```

As Ontologias desabilitadas são exibidas um um quadro de cor vermelha e as habilitadas em cor verde. Quando desabilitadas, o processo de axiomatização e inferência não tem efeito sobre essas bases de conhecimento. Observe que o quadro da ontologia desabilitada fica em um tom avermelhado.

```
Ontology: <http://edu.ia.br.ufpe.cin.msc.ontology>

ObjectProperty: continue
ObjectProperty: continuing
ObjectProperty: continuingWithout
ObjectProperty: permanentDeteriorationIn
ObjectProperty: severeDeteriorationIn

Class: Resource
Class: Sustained

Class: SustainedUse
 EquivalentTo:
  Sustained
  and Use
 SubClassOf:
  Use,
   ((continuingWithout some (permanentDeteriorationIn some Resource))
   or
   (continuingWithout some (severeDeteriorationIn some Resource)))
   and
   (continuing some Use)

Class: Use
```

Figura 37 – Ontologia modelada em *Manchester Syntax*

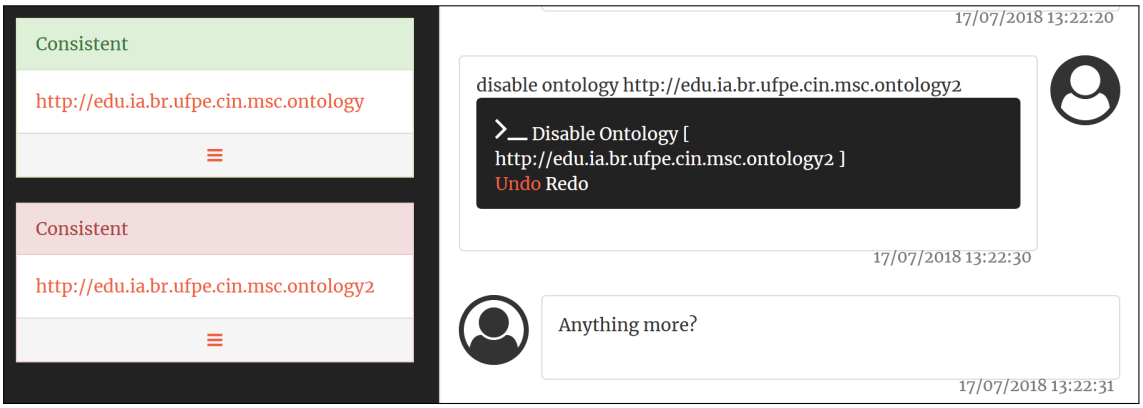

Figura 38 – Ontologia desabilitada

As sentenças a seguir, foram através de diálogo requisitadas pelo usuário para serem inseridas nas bases de conhecimento.

1. *biosphere is the portion of Earth and its atmosphere that can support life.*

2. *vehicles are non-living means of transportation.*

3. *a minister or a secretary is a politician who holds significant public office in a national or regional government.*
4. *a currency is a unit of exchange, facilitating the transfer of goods and services.*
5. *an island or isle is any piece of land that is completely surrounded by water.*
6. *days of the week are Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.*

Notamos nesse passo que o PUPO inseriu as sentenças requisitadas apenas na primeira Ontologia (*ontology*), pois a segunda (*ontology2*) se encontra desabilitada e quaisquer novos axiomas não serão computados na mesma.

Ao processar essas sentenças (total de 12, primeiramente 6 e posteriormente mais 6 após desabilitar a Ontologia *Ontology2*), foi gerado uma Ontologia com expressividade ALCHOQ[3] (Vide Figura 39 para quadro de detalhes da aplicação PUPO).

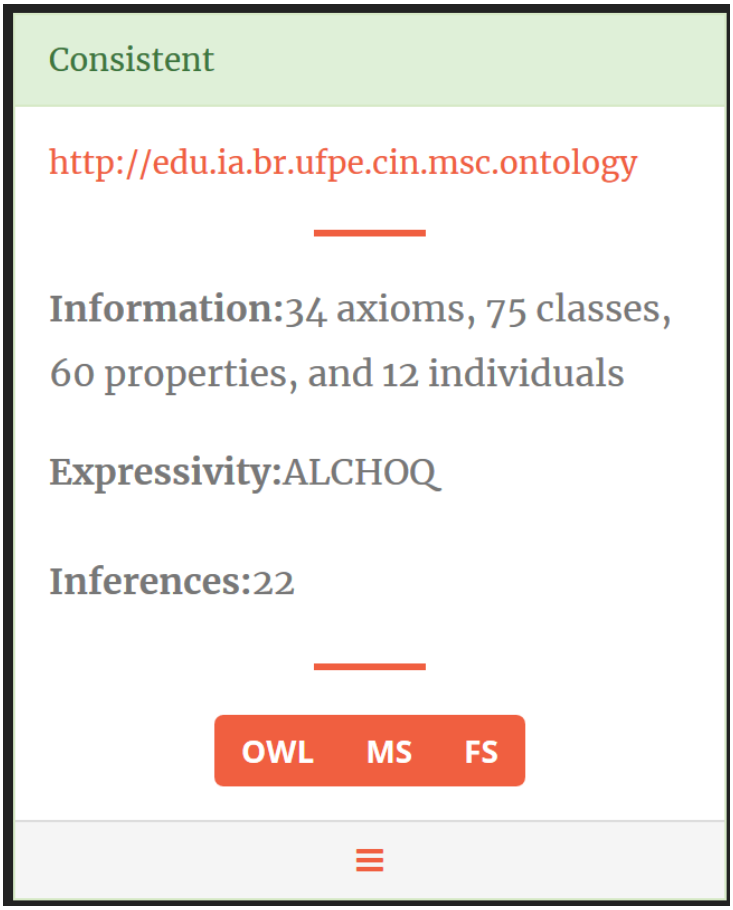

Figura 39 – Detalhes da Ontologia gerada a partir de Völker, Hitzler e Cimiano (2007)

[3] É uma família de linguagem de lógica de descrições denominada Linguagem Atributiva com Complementos, Hierarquias de Papeis, Nominais e Cardinalidade Qualificada

Na Figura 39 consta que foram gerados 12 (doze) indivíduos. Desses 12 (doze), apenas 3 (três) foram extraídos equivocadamente (mais detalhes sobre as limitação da ferramenta serão apresentados no Capítulo 6).

Como exemplo de código OWL dessa Ontologia é exibido na Figura 40, a axiomatização de *tetraploid is a cell or organism having four sets of chromosomes.*

```
<owl:Class rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#Tetraploid">
    <rdfs:subClassOf>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about=
"http://edu.ia.br.ufpe.cin.msc.ontology#Cell"/>
                        <rdf:Description rdf:about=
"http://edu.ia.br.ufpe.cin.msc.ontology#Organism"/>
                    </owl:unionOf>
                </owl:Class>
                <owl:Restriction>
                    <owl:onProperty rdf:resource=
"http://edu.ia.br.ufpe.cin.msc.ontology#having"/>
                    <owl:qualifiedCardinality rdf:datatype=
"http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
4</owl:qualifiedCardinality>
                    <owl:onClass>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource=
"http://edu.ia.br.ufpe.cin.msc.ontology#setOf"/>
                            <owl:someValuesFrom rdf:resource=
"http://edu.ia.br.ufpe.cin.msc.ontology#Chromosome"/>
                        </owl:Restriction>
                    </owl:onClass>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </rdfs:subClassOf>
</owl:Class>
```

Figura 40 – Exemplo de código OWL da sentença *tetraploid is a cell or organism having four sets of chromosomes*

Em *Manchester Syntax* temos:

- Tetraploid SubClassOf: (Cell or Organism) and having exactly 4 (setOf some Chromossome)

4.2.2 Refinamento de Ontologias

Além do processo de modelagem para criação de Ontologias o PUPO provê um mecanismo para refinar Ontologias. O refinamento dar-se-á, a partir do informe de uma IRI válida. Com o propósito de refinamento de Ontologias foi realizado o carregamento da Ontologia *Pizza*[4]. Essa Ontologia em particular apresenta axiomas inconsistentes. O carregamento é realizado através do diálogo apresentado na Figura 41.

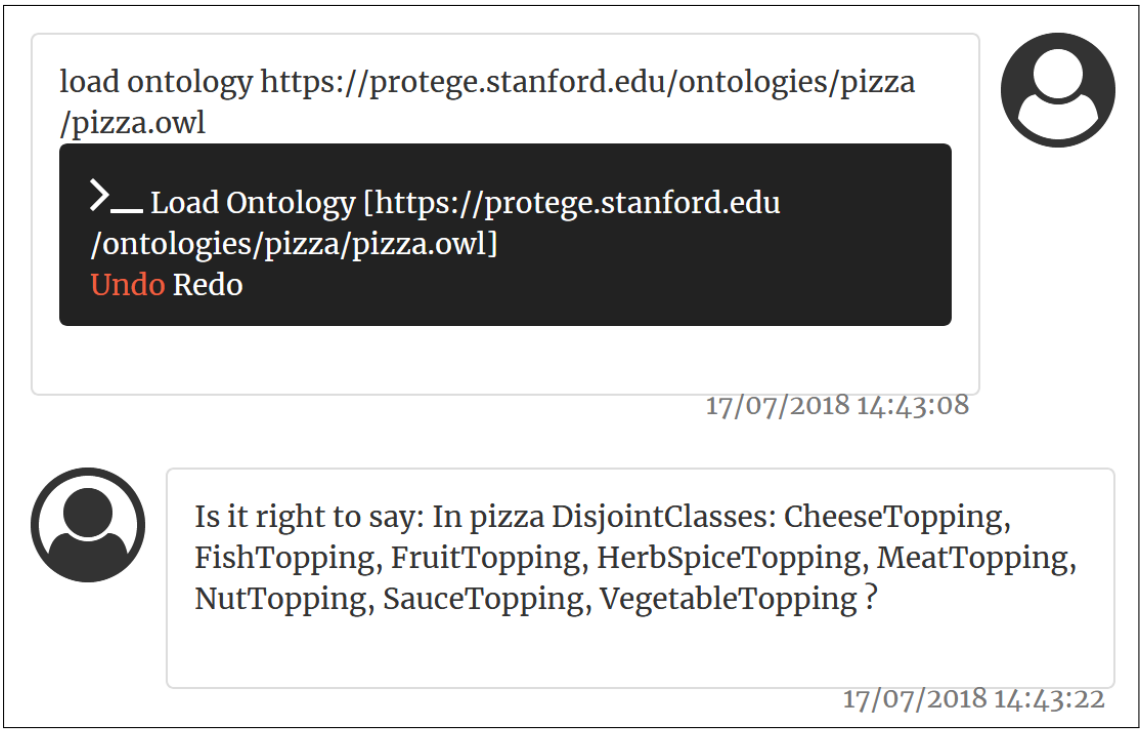

Figura 41 – Carregamento da Ontologia Pizza

Imediatamente após o carregamento é realizada uma checagem de inferências. Desse modo são detectadas possíveis inferências do motor de inferência e também inconsistências. Foi observado que o axioma da Figura 42 é um dos que deixa a Ontologia inconsistente.

```
DisjointClasses: CheeseTopping, FishTopping, FruitTopping
, HerbSpiceTopping, MeatTopping, NutTopping, SauceTopping
, VegetableTopping
```

Figura 42 – Axioma de disjunção que gera insatistatibilidade na ontologia Pizza

Assim se o usuário responder *NO* (Vide Figura 43) esse axioma é removido da base de conhecimento, como foi o caso.

[4] https://protege.stanford.edu/ontologies/pizza/pizza.owl

Figura 43 – Inconsistência da Ontologia Pizza

Em seguida, é identificado que o axioma *hasTopping Domain Pizza* é um outro possível causador dessa inconsistência, pois *IceCream* também é domínio de *hasTopping*. Respondendo *NO* o axioma foi removido e a Ontologia passou a ser consistente, Figura 44.

Figura 44 – Consistência da Ontologia Pizza

Após esse processo, o diálogo continua e com a resposta de *NO* do usuário, como resposta a *Anything more?*, é encerrado o fluxo de diálogo e exibida a lista básica de funcionalidade do sistema. A lista completa de comandos pode ser encontra no apêndice F;

4.2.2.1 Criando axiomas de fecho

Axiomas de fecho tem o papel de realizar um fechamento (restrição) sobre a base de conhecimento. Por exemplo, considere a frase *carnivore eats meat* (Carnívoro come carne) e sua forma axiomatizada *Carnivore SubClassOf eats some meat*. Esse axioma informa que existe pelo menos um *Carnivore* (Carnívoro) que come carne. Um axioma de fecho sobre esse axioma tem a forma *Carnivore eats only meat*. Ou seja, *Carnivore* (Carnívoro) como apenas carne. Assim com esses dois axiomas na base de conhecimento garantimos que todo carnívoro come carne e existe pelo menos um carnívoro.

Para exemplificar na prática, utilizando o PUPO, foi criada uma nova Ontologia *"fish"*, com o diálogo:

User: create ontology fish

Após sua criação foi inserido a seguinte sentença[5] em linguagem natural através do diálogo:

User: classify Fish are gill-bearing aquatic craniate animals that lack limbs with digits.

Essa sentença gerou dois axiomas, conforme apresentamos na Figura 45.

```
1  (1) Fish
2      SubClassOf
3         Gill-bearingAquaticCraniateAnimal
4         and (lack some (limbWith some Digit))
5
6  (2) Gill-bearingAquaticCraniateAnimal
7      EquivalentTo
8         Animal and
9         Aquatic and
10        Craniate and
11        Gill-bearing
```

Figura 45 – Axiomas de definição de Fish

Através do diálogo a seguir, o sistema gera um novo axioma realizando o fecho do primeiro axioma.

User: make closure

O axioma de fecho em *Manchester Syntax* é:

Fish SubClassOf lack only (limbWith some Digit)

Na Figura 46 é exibida a forma com que o sistema pergunta ao usuário se aquele axioma de fecho é verdadeiro, semelhante ao tratamento de inferências.

[5] Retirada de https://en.wikipedia.org/wiki/Fish

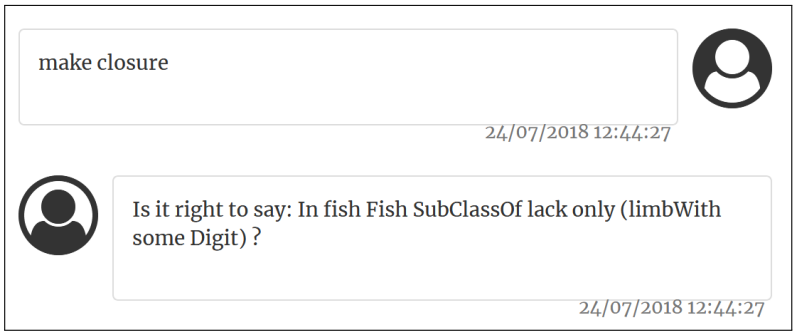

Figura 46 – Axioma de fecho

4.2.3 Consultando fatos nas Ontologias Modeladas

Um recurso fundamental para os sistemas computacionais consiste em consultar o conhecimento modelado e armazenado de forma persistente. Essa consulta permite verificar fatos do mundo real que agora estão representados na forma de conhecimento computacional. O PUPO é capaz de fornecer um mecanismo de consulta que funciona de duas formas: em linguagem natural e/ou *Manchester Syntax*.

Para iniciarmos com o exemplo, considere a criação de um indivíduo (Nemo) na base de conhecimento utilizando a definição do seguinte diálogo:

Usuario: Nemo is a fish.

Agora, considere que a sentença *Fish are gill-bearing aquatic craniate animals that lack limbs with digits* já foi inserida na base de conhecimento. Com esses axiomas já inseridos na base de conhecimento podemos realizar dois tipos de consulta no PUPO: I) Consulta a partir de linguagem natural; II) Consulta a partir de *Manchester Sintax*. A seguir detalhamos cada uma delas.

4.2.3.1 Consulta a partir de linguagem natural

Como recurso, o sistema de diálogo permite consultas na base de conhecimento através de linguagem natural. A Figura 47 exibe a pergunta *Nemo is an animal that lacks limb with digit ?* O PUPO como resposta fornece a sentença original em forma de axioma (*{Nemo} SubClassOf (Animal) AND (lack some (limbWith some (Digit)))*) – Esse axioma é utilizado como consulta em DL e checado na(s) ontologia(s) – além de sua prova, caso o axioma seja verdadeiro em alguma base de conhecimento.

Esse recuso de *query* (consulta) em linguagem natural é acessível através do texto em linguagem natural seguida de um ponto de interrogação, como segue:

- Nemo is an animal that lacks limb with digit ?

O processo de consulta ocorre de forma semelhante ao processo de axiomatização, porém não contamos com Aprendizado de Máquina para inferir novos axiomas, pois nosso

Figura 47 – *Query* em linguagem natural

objetivo nesse momento é verificar se um fato dito em diálogo pelo usuário se confirma na(s) ontologia(s). No momento que a sentença é axiomatizada e essa informação chega ao pacote *(4) Reasoner Manager* (Vide Seção 3.1.4 do Capítulo 3) o componente *DL Reasoning* verifica se o axioma é válido ou não. Por fim se o axioma for válido sua prova é apresentada ao usuário.

4.2.3.2 Consulta a partir de *DL Query* em *Manchester Syntax*

Utilizando *Manchester Syntax*, faz-se necessário fornecer o axioma seguido do ponto de interrogação. Esse ponto de interrogação possibilita ao sistema interpretar a entrada como uma pergunta e gerar uma resposta válida para o usuário. Nesse contexto como o usuário já fornece um axioma em *Manchester Syntax*, então o seu texto não passa pelo processo de axiomatização e vai direto para o componente *DL Reasoning*. Caso o axioma se comprove verdadeiro para alguma ontologia, então o PUPO fornece essa prova para o usuário, assim como segue:

- Usuario: consult Nemo SubClassOf (Animal) AND (lack some (limbWith some (Digit))) ?

Assim se obtém o mesmo resultado da Figura 47. Obrigatoriamente deve haver *consult*

no início e a interrogação ao final da sentença. O *consult* permite ao sistema diferenciar entre um texto em linguagem natural ou *Manchester Syntax*.

No próximo Capítulo apresentamos e comparamos trabalhos relacionados ao nosso trabalho. Assim exibimos alguns diferenciais do PUPO em relação aos demais.

5 TRABALHOS RELACIONADOS

Na literatura são descritas diversas ferramentas com o propósito de auxiliar engenheiros e desenvolvedores na atividade de construção de ontologias a partir de textos em linguagem natural (LN)(BARBUR; BLAGA; GROZA, 2011; VÖLKER; HITZLER; CIMIANO, 2007; DAHAB; HASSAN; RAFEA, 2008; SERRA; GIRARDI; NOVAIS, 2013; De Azevedo et al., 2014; DRYMONAS; ZERVANOU; PETRAKIS, 2010; PETRUCCI; GHIDINI; ROSPOCHER, 2016; ALI et al., 2016). Mesmo com os avanços na área de Processamento de Linguagem Natural (PLN), que pode identificar as funções sintáticas e semânticas de um texto em LN, ainda não é tão fácil converter esse texto, em um formalismo lógico ou linguagem descritiva como a OWL 2 DL, por exemplo. Dentre as principais dificuldades podemos citar:

1. Baixa curva de aprendizado de linguagens ontológicas;
2. Necessidade de engenheiro de conhecimento;
3. Necessidade de especialista de domínio ou fonte de conhecimento especialista de um domínio de conhecimento.
4. Variância da Linguagem Natural - Causando dificuldades do processo de tradução para um formalismo lógico.

Nesse contexto, software e/ou métodos para modelagem de conhecimento podem ser classificados em manuais, semiautomáticos e automáticos (El Idrissi Esserhrouchni; FRIKH; OUHBI, 2014). Vejamos suas definições:

Manuais. Se referem a editores de código onde o engenheiro de conhecimento através dessa ferramenta cria os componentes que deseja, um exemplo de ferramenta manual é o Protégè[1].

Semiautomáticos. Permitem o processamento parcial de informações e apresentam em sua maioria algum mecanismo de interação com o usuário, para que o mesmo venha a refinar a modelagem das bases de conhecimento.

Automáticos. Têm como objetivo processar dados e criar código de linguagem de representação de conhecimento sem necessitar de intervenção humana durante o processo de conversão de linguagem natural para uma linguagem de modelagem de conhecimento.

Diversas ferramentas na literatura utilizam esses métodos manuais, automáticos e semiautomáticos. Na sessão seguinte demonstramos algumas dessas ferramentas, a medida que realizamos uma comparação entre tais ferramentas com o PUPO.

[1] https://protege.stanford.edu/

5.1 ONTORICH

O *Framewordk* OntoRich (BARBUR; BLAGA; GROZA, 2011) utiliza uma versão do `WordNet` chamada de `RiTa WordNet`. Essa versão permite ao programador acessar uma ontologia modelada com as informações do `WordNet`. O *Framework* refina Ontologias a partir de *feed RSS* utilizando o processamento de linguagem natural obtido através da `OpenNLP API`[2].

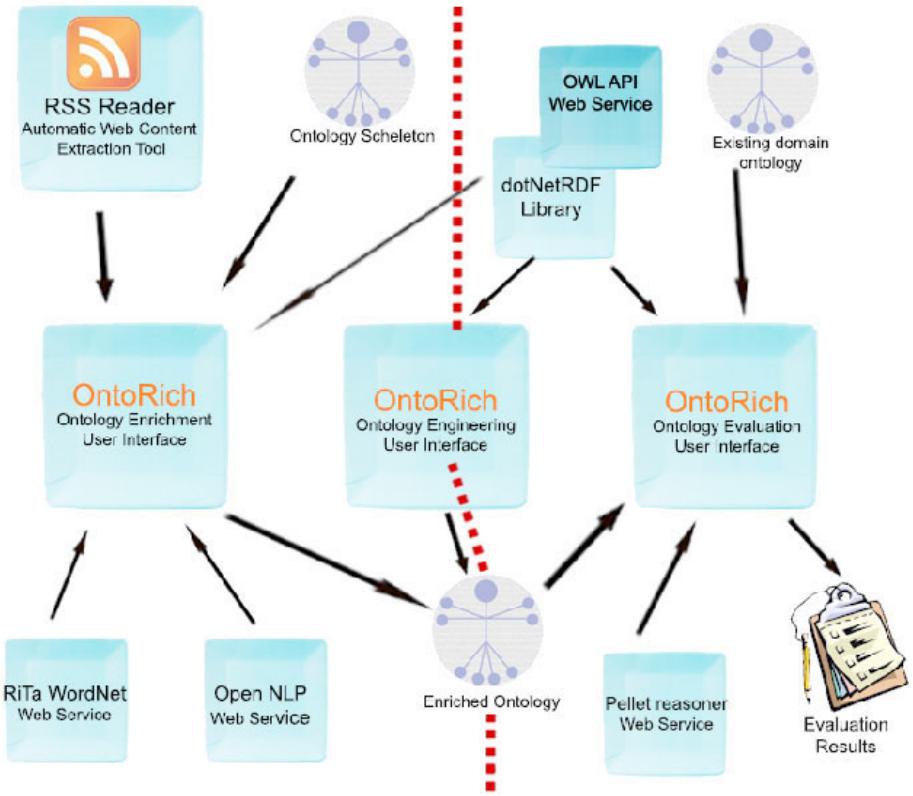

Figura 48 – Processo de modelagem de conhecimento utilizando a OntoRich

Apresentamos na Figura 48 a arquitetura geral do sistema. Nota-se que possui três meios de enriquecimento da Ontologia. O primeiro é o enriquecimento com auxílio do usuário e extração direta dos *feeds*, uso do `OpenNLP` e `RiTa WordNet`. O segundo consiste no engenheiro de conhecimento realizando ajustes nos dados já previamente modelados e o último consiste na avaliação de forma manual além do auxílio de motores de raciocínio e outras Ontologias de domínio.

Na Tabela 10 apresentamos um comparativo entre a forma de modelagem de conhecimento realizada entre o PUPO e o *OntoRich*. As duas ferramentas foram desenvolvidas tando para criação quando para edição de ontologias. Também apresentam métodos de modelagem semiautomático, ou seja, necessitam da intervenção humana para modelar ontologias em OWL. A *OntoRich* utiliza-se de abordagens Linguísticas e estatísticas para extrair informação para as ontologias, enquanto no PUPO utilizamos uma abordagem híbrida entre linguística e aprendizado de máquina.

[2] https://opennlp.apache.org/

	OntoRich	PUPO
Manipulação de Ontologias	Criação e Edição	Criação e Edição
Método	Semiautomático	Semiautomático
Abordagem	Linguística, Estatística, WordNet	Linguística, Aprendizado de Máquina, SUMO, WordNet
Raciocínio Automático	SIM	SIM
Identificação e Tratamento de Inconsistências	NÃO	SIM
Ferramenta de PLN	OpenNLP	CoreNLP (Stanford Parser)
Linguagem Gerada	OWL	OWL

Tabela 10 – Comparativo entre funcionalidades da OntoRich e o PUPO

No que diz respeito a extração de informação o PUPO é capaz de extrair axiomas gerais (em OWL) enquanto a *OntoRich* não o faz. Ainda o PUPO realiza duas tarefas: I) realiza raciocínio automático identificando quais termos são definições e quais termos são subclasses de outros (raciocínio de subsunção); II) Identifica e trata Inconsistências enquanto interage com o usuário através de diálogo (Vide Capítulo 4). Por sua vez o *OntoRich* apenas identifica as inferências das ontologias.

5.2 LEXO

O LExO (VÖLKER; HITZLER; CIMIANO, 2007) é um software o qual apresenta um método que utiliza regras de transformação construídas manualmente para extração de informação advinda de texto em linguagem natural. A utilização de regras permitem, por um lado, que outras sejam adicionadas manualmente afim de melhorar a capacidade de PLN. Por outro lado exige um estudo e validação para cada nova regra inserida, gerando mais custos para transformar LN em OWL. Para processamento de linguagem natural o *LExO* utiliza um processador de linguagem natural (Parser) chamado MINIPAR.

É apresentado na Figura 49 algumas regras utilizadas para transformação de LN em DL.

Consideremos a frase *a good student is a good boy or girl and has good scores.* (Vide Figura 50) e o padrão (NP0 C(rel) VP0) (Vide Figura 49).

A abordagem de Völker, Hitzler e Cimiano (2007) extrai um trecho da frase que compõe a frase nominal *a good boy or girl* (bom menino ou menina) e através da conjunção *and* (e) junta com uma frase verbal *has good scores* (tem boas notas). Esse padrão é capaz de compor o axioma de definição *(good and student) EquivalentTo: (good and (boy or girl)) and (has some (good and scores)).*

Rule	Natural Language Syntax	OWL Axioms
Disjunction	NP_0 `or` NP_1	$X \equiv (NP_0 \sqcup NP_1)$
Conjunction	NP_0 `and` NP_1	$X \equiv (NP_0 \sqcap NP_1)$
Determiner	Det_0 NP_0	$X \equiv NP_0$
Intersective Adjective	Adj_0 NP_0	$X \equiv (Adj_0 \sqcap NP_0)$
Subsective Adjective	Adj_0 NP_0	$X \sqsubseteq NP_0$
Privative Adjective	Adj_0 NP_0	$X \sqsubseteq \neg NP_0$
Copula	NP_0 VBE NP_1	$NP_0 \equiv NP_1$
Relative Clause	NP_0 C(*rel*) VP_0	$X \equiv (NP_0 \sqcap VP_0)$
Number Restriction	V_0 Num $NP(obj)_0$	$X \equiv =\text{Num}\ V_0.NP_0$
Negation (not)	`not` V_0 NP_0	$X \sqsubseteq \neg\exists V_0.NP_0$
Negation (without)	NP_0 `without` $NP(pcomp\text{-}n)_1$	$X \equiv (NP_0 \sqcap \neg\text{with}.NP_1)$
Participle	NP_0 $VP(vrel)_0$	$X \equiv (NP_0 \sqcap VP_0)$
Transitive Verb Phrase	V_0 $NP(obj)_0$	$X \equiv \exists V_0.NP_0$
Verb with Prep. Compl.	V_0 $Prep_0$ $NP(pcomp\text{-}n)_0$	$X \equiv \exists V_0\_Prep_0.NP_0$
Noun with Prep. Compl.	NP_0 $Prep_0$ $NP(pcomp\text{-}n)_1$	$X \equiv (NP_0 \sqcap \exists Prep_0.NP_1)$
Prepositional Phrase	$Prep_0$ NP_0	$X \equiv \exists Prep_0.NP_0$

Figura 49 – Regras de tradução apresentadas por Völker, Hitzler e Cimiano (2007)

```
Frase:
a good student is a good boy or girl and has good scores.

Em português:
um bom estudante é um bom menino ou menina e tem boas notas.

Padrão utilizado:
NP0 C(rel) VP0 -> X EquivalentTo: (NP0 and VP0)

Extração:
X=(good and student) EquivalentTo: NP0=(good and (boy or girl)) and
VP0=(has some (good and scores))

Em português:
X=(bom and estudante) EquivalenteA: NP0=(bom and (menino ou menina)) e
VP0=(tem algum (bom e notas))
```

Figura 50 – Aplicação de padrão apresentado por Völker, Hitzler e Cimiano (2007)

Apresentamos na Tabela 11 um comparativo entre as principais características do PUPO com o LExO.

Quanto a manipulação de ontologias o LExO realiza criação de ontologias, mas não possibilita a manipulação (carregar e editar) ontologias já prontas. O PUPO por sua fez é capaz tanto de criar quanto carregar e editar. O LExO apresenta um método automático de criação de ontologias, ou seja, não necessita da intervenção humana no processo. Sua abordagem é de caráter linguístico, no qual vários padrões (Vide Figura 49) são utilizadas

	LExO	PUPO
Manipulação de Ontologias	Criação	Criação e Edição
Método	Automático	Semiautomático
Abordagem	Linguística	Linguística, Aprendizado de Máquina, SUMO, WordNet
Raciocínio Automático	NÃO	SIM
Identificação e Tratamento de Inconsistências	NÃO	SIM
Ferramenta de PLN	MINIPAR	CoreNLP (Stanford Parser)
Linguagem Gerada	OWL	OWL

Tabela 11 – Comparativo entre funcionalidades do LExO e o PUPO

para extrair informação de linguagem natural. A maioria dos padrões utilizados por Völker, Hitzler e Cimiano (2007) resultam em axiomas de definição. Um axioma de definição representa tudo que uma classe OWL define. Por padrão o PUPO axiomatiza sentenças em OWL através de axiomas de subclasse, permitindo que novos fatos sejam inseridos posteriormente na mesma classe. O PUPO, por interagir com o usuário, apresenta um método semiautomático. Os métodos de extração de informação se equiparam, tanto do LExO quanto do PUPO, pois os dois permitem extração de relações, hierarquias e axiomas em lógica de descrições. O LExO não trata qualquer tipo de raciocínio, como faz o PUPO que trata inconsistências e descobre novos fatos nas ontologias.

5.3 TEXTONTOEX

O **TextOntoEx (DAHAB; HASSAN; RAFEA, 2008)** é um software que utiliza um editor de padrões semânticos que gera padrões de processamento que podem ser utilizados posteriormente para classificar textos. Esses padrões (Vide Figura 51) utilizam uma abordagem mista onde cada termo pode ser uma palavra com etiquetada apresentando *POS-tag* ou um conceito ontológico (classe).

Na primeira linha, Figura 51, Dahab, Hassan e Rafea (2008) apresentam dois conceitos $\langle Color \rangle$ e $\langle Shape \rangle$ seguidos da palavra *on* e outra palavra etiquetada que representa um número ou um ordinal. Por fim a última palavra $\langle PartPlant \rangle$ é uma classe ontológica "parte de planta". Dahab, Hassan e Rafea (2008) explicam que a ferramenta não descobre novas relações, mas sim instâncias de relações já conhecidas.

O processo de classificação automático se torna simples a medida que padrões novos são gerados e armazenados, porém o processo ainda é custoso pelo menos em tempo de criação desses novos padrões.

Semantic Patterns	Natural Text
⟨Color ⟩⟨Shape ⟩on [⟨Ordinal numeral. POS ⟩] ⟨PlantPart ⟩	Yellow striping on second, third, or older leaves
⟨Shspe ⟩⟨Becomes. Verb ⟩⟨Color ⟩	Stripes turn brown
Infected ⟨PlantPart ⟩⟨AbnormalAppearance⟩ as the ⟨Diseases ⟩Progresses	infected leaves die as the disease progresses.
Infected ⟨PlantPart ⟩⟨may.Verb ⟩⟨Appears Verb⟩⟨abnormal Appearance⟩as they ⟨Abnormal Appearance⟩	Infected leaves may look frayed as they die.
Infected ⟨PlantPart⟩⟨be.Verb ⟩ ⟨AbnormalAppearance⟩, and flag ⟨Plant Part⟩⟨May. Verb⟩be a ⟨Help Color POS⟩⟨Color⟩	Infected plants are stunted, and flag leaves may be a light tan

Figura 51 – Padrões semânticos do `TextOntoEx` por Dahab, Hassan e Rafea (2008)

	TextOntoEx	PUPO
Manipulação de ontologias	Criação	Criação e Edição
Método	Semiautomático	Semiautomático
Abordagem	Linguística	Linguística, Aprendizado de Máquina, SUMO, WordNet
Raciocínio Automático	NÃO	SIM
Identificação e Tratamento de Inconsistências	NÃO	SIM
Ferramenta de PLN	*	CoreNLP (Stanford Parser)
Linguagem Gerada	OWL	OWL

Tabela 12 – Comparativo entre funcionalidades do TextOntoEx e o PUPO

Apresentamos na Tabela 12 um comparativo em relação ao PUPO. Diferente do PUPO o *TextOntoEx* não possui mecanismo de raciocínio sobre as ontologias modeladas, isso permite com que fatos contraditórios possam ser inseridos nas ontologias. Enquanto o PUPO permite criação e edição de ontologias já prontas, o *TextOntoEx* permite apenas a criação. As duas ferramentas entretanto fornecem um método semiautomático de extração de informação e axiomatização do conhecimento extraído a partir de textos.

5.4 PARNT

O **PARNT (SERRA; GIRARDI; NOVAIS, 2013)** é uma técnica que a partir de uma fonte de informação e um método estatístico, utiliza processamento de linguagem natural e modelagem de conhecimento; O processo utilizado por Serra, Girardi e Novais (2013) (Vide Figura 52) consiste em realizar uma seleção dos possíveis candidatos a relação contendo uma tripla conceito-relação-conceito, por exemplo "Joao-casado-Maria" representando que "Joao" é "casado" com "Maria". Após isso, as relações que foram elegíveis para serem adicionadas na Ontologia, são de fato persistidas na base de conhecimento.

O método apresenta 4 fases:

- Extração de candidatos a relações, utilizando anotação do texto a ser processado e posteriormente extração de relações a partir de regras;
- Refinamento, verificando a frequência dessas relações no texto, indicando assim sua relevância e forte candidato a relações em potencial;
- Avaliação das relações por especialistas e finalmente,
- atualização na Ontologia final com os relacionamentos taxonômicos ou não taxonômicos.

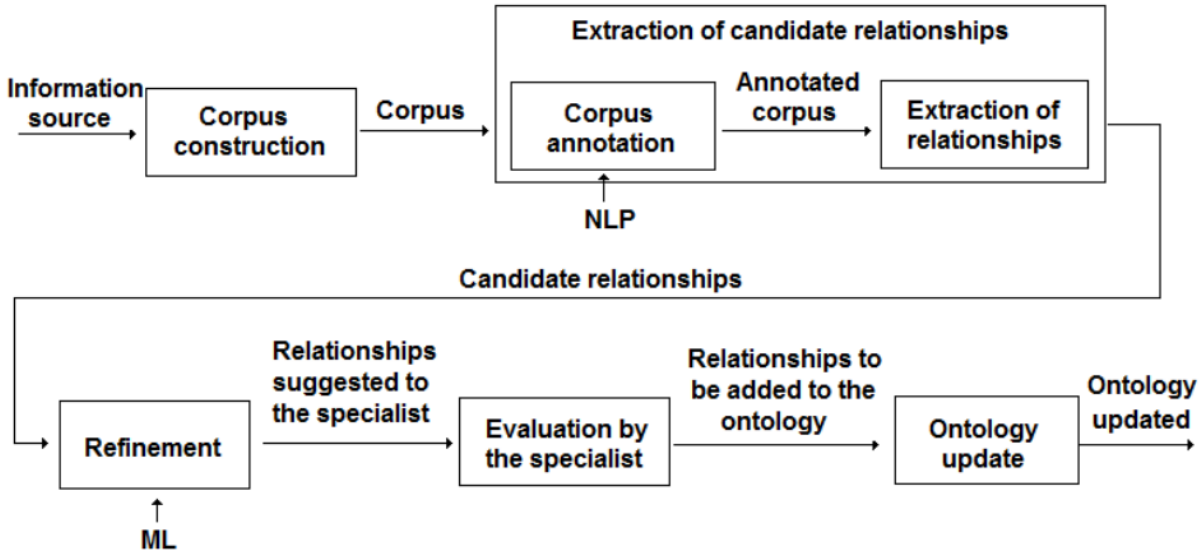

Figura 52 – Fluxo de processamento do PARNT

Na Tabela 13 apresentamos as principais características do PARNT em relação ao PUPO.

O PUPO se destaca no processo de raciocínio automático e utilização de bases semânticas como a SUMO e o WordNet para enriquecimento semântico das ontologias geradas. Além disso conta com possibilidade de criação e edição de ontologias durante o desenvolvimento das mesmas. O PARNT em relação aos demais trabalhos, semelhante ao PUPO,

	PARNT	PUPO
Manipulação de ontologias	Criação	Criação e Edição
Método	Semiautomático	Semiautomático
Abordagem	Linguística, Aprendizado de Máquina	Linguística, Aprendizado de Máquina, SUMO, WordNet
Raciocínio Automático	NÃO	SIM
Identificação e Tratamento de Inconsistências	NÃO	SIM
Ferramenta de PLN	*	CoreNLP (Stanford Parser)
Linguagem Gerada	OWL	OWL

Tabela 13 – Comparativo entre funcionalidades do PARNT e o PUPO

utiliza Aprendizado de Máquina para melhor selecionar os axiomas que serão inseridos na ontologia final.

5.5 RENAN

O sistema de diálogo Renan, desenvolvido por De Azevedo et al. (2014) apresenta uma ferramenta que é dividida em 3 (três) módulos de tradução de linguagem natural para OWL DL, conforme Figura 53.

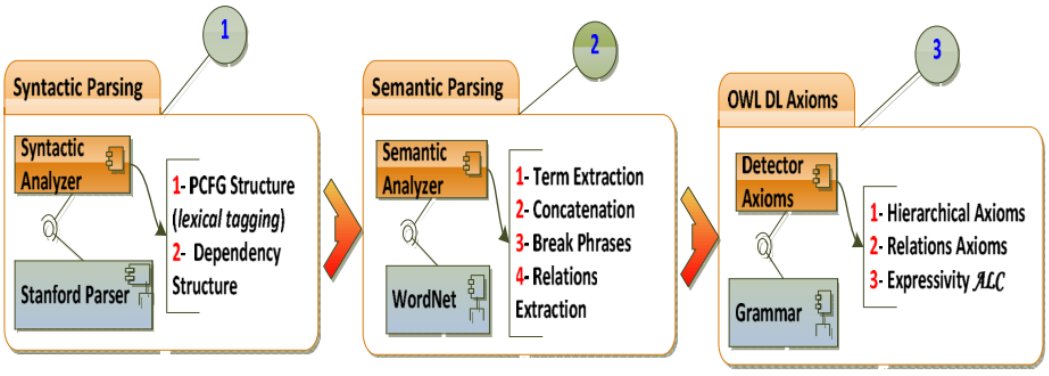

Figura 53 – Arquitetura de De Azevedo et al. (2014)

- *Syntactic parsing Module*, consiste em uma camada de análise sintática da entrada para o sistema. Essa análise é realizada através do `Stranford Parser` e retorna uma árvore sintaticamente anotada.
- *Semantic Parsing Module*, recebe a árvore sintática anotada e realiza uma extração de termos e relações a partir de regras.

- *OWL DL Axioms Module*, cria os axiomas em OWL DL a partir dos conceitos e relações providos pelo módulo anterior.

De Azevedo et al. (2014) utiliza padrões para transformar o texto em OWL. Na Figura 54 apresentamos alguns desses padrões.

Construction Patterns using verbs, intersection (and), conjuction (or) and negation (not)
(1) SN_1 (***are a*** | ***are an*** | ***are*** | ***is a*** | ***is an*** | ***is***) SN ***or/and*** SN (***That/Who/Which***) (***has*** **not**) SN
(2) SN_1 (***are a*** | ***are an*** | ***are*** | ***is a*** | ***is an*** | ***is***) SN ***or/and*** SN (***That/Who/Which***) (***has***) SN
(3) SN_1 (***are a*** | ***are an*** | ***are*** | ***is a*** | ***is an*** | ***is***) SN ***or/and*** SN (***That/Who/Which***) (***Verb***) SN
(4) SN_1 (Verb) SN (***or/and***) SN
(5) SN_1 (Verb) SN (***or/and***) (Verb) SN
(6) SN_1 ***has*** SN_n ***and*** SN_n
(7) SN_1 (***has not***) SN_n
(8) SN_1 (***does not*** | ***doesn't*** | ***is not*** | ***isn't***) SN_n
(9) SN_1 (***does not*** | ***doesn't*** | ***is not*** | ***isn't***) (Verb) SN_n
(10) SN_1 ***has*** SN_n ***and has not*** SN_n
(11) SN_1 (***are a*** | ***are an*** | ***are*** | ***is a*** | ***is an*** | ***is***) SN (***That/Who/Which***) (Verb) (***or***) (Verb) SN
(12) SN_1 (***are a*** | ***are an*** | ***are*** | ***is a*** | ***is an*** | ***is***) SN ***or/and*** SN (***That/Who/Which***) (Verb) (***or/and***) (Verb) SN
(13) SN_1 (Verb) SN (***or/and***) (Verb) SN (***or/and***) (Verb) SN...

Figura 54 – Padrões utilizado por De Azevedo et al. (2014)

Como exemplo, tomemos o padrão *SN1 has SNn and SNn* e a frase *dog has good eyes and ears* (cão tem bons olhos e orelhas). De Azevedo et al. (2014) extrairia:

- dog SubClassOf: has some (good_eyes and ears)

Esse mesmo axioma seria extraído pelo PUPO da seguinte forma:

- Dog SubClassOf: has some (GoodEyes and Ears)

O Renan realiza raciocínio de subsunção e nessa sentença extrairia a seguinte informação:

- good_eyes SubClassOf: eyes

O PUPO por sua vez cria um axioma de definição (equivalência) da seguinte forma:

- GoodEyes EquivaletTo: Good and Eyes

Os dois axiomas estão semanticamente corretos, porém o axioma de equivalência é mais forte que o de subclasse. Pois a equivalência transmite a informação que que para ser um *GoodEyes* (BonsOlhos) é suficiente algo ser *Good* (Bom) e *Eyes* (Olhos) ao mesmo tempo.

Ainda através de raciocínio o Renan é capaz de identificar e solucionar inconsistências na ontologia modelada, bem como o PUPO. Vejamos a seguir um exemplo de como o PUPO e o Renan tratam inconsistências:

Renan:

```
Usuário: vaca come apenas grama
Usuário: herbívoro come apenas planta
Usuário: herbívoro come folhas
Usuário: herbívoro come carne
Usuário: carne é comida e não é uma planta

Sistema: vaca e herbívoro são inconsistentes

Usuário: herbívoro não come carne

[inconsistência resolvida]
```

PUPO:

```
Usuário: vaca come apenas grama
Usuário: herbívoro come apenas planta
Usuário: herbívoro come folhas
Usuário: herbívoro come carne
Usuário: carne é comida e não é uma planta

Sistema: é verdade que herbívoro como carne ?

Usuário: não

[inconsistência resolvida]
```

O PUPO no exemplo acima informa imediatamente o fato para que o usuário digite apenas sim ou não, evitando o trabalho do usuário fornecer uma frase contrária ao axioma inconsistente.

	RENAN	PUPO
Manipulação de ontologias	Criação e Edição	Criação e Edição
Método	Semiautomático	Semiautomático
Abordagem	Linguística	Linguística, Aprendizado de Máquina, SUMO, WordNet
Raciocínio Automático	SIM	SIM
Identificação e Tratamento de Inconsistências	SIM	SIM
Ferramenta de PLN	Stanford Parser	CoreNLP (Stanford Parser)
Linguagem Gerada	OWL	OWL
Expressividade DL	$\mathcal{ALC}$	$\mathcal{ALCHOQ}$
Taxa de sucesso	0.75	0.70

Tabela 14 – Comparativo entre funcionalidades do RENAN e o PUPO

Na Tabela 14 apresentamos as principais características entre o PUPO e o Renan. Primeiramente o PUPO se mostra capaz de editar ontologias posteriormente a sua criação, bem como o Renan. O Renan trata o texto apenas de forma linguística para extrair os axiomas para as ontologias. O PUPO utiliza além de padrões linguísticos algoritmo de aprendizagem de máquina (SVM) e bases semânticas como a SUMO e o WordNet. Por utilizar essas bases semânticas o PUPO consegue alcançar expressividade $\mathcal{ALCHOQ}$, maior que a do Renan ($\mathcal{ALC}$), nas ontologias modeladas em OWL DL, obtendo assim avanços sobre as abordagens do Renan. Porém a taxa de sucesso de PUPO é 0.05 menor que a do Renan.

O desenvolvimento do PUPO é realizado sob os mesmos aspectos do RENAN. Aspectos esses que tem como pontos fundamentais:

1. Modelar conhecimento através de diálogo;
2. Realizar raciocínio em tempo de modelagem;
3. Tratar inconsistências em tempo de modelagem.

O PUPO estende o RENAN em seus aspectos conceituais (citados acima), porém a nível de implementação o PUPO é totalmente independente. Ou seja, sua implementação não utiliza quaisquer partes do RENAN, apenas APIs de terceiros como por exemplo o *CoreNLP*.

5.6 ONTOGAIN

O OntoGain (DRYMONAS; ZERVANOU; PETRAKIS, 2010) tem a capacidade de extrair relações de forma probabilística e regras associativas. As principais tarefas do sistema consistem em preprocessamento, extração de termos, extração de relações taxonômicas e relação não taxonômicas.

Na fase de preprocessamento, através do OpenNLP[3] (*parser* de código aberto semelhante ao *CoreNLP*) e *Java WordNet Library* (JWNL[4]) (Biblioteca Java para acesso ao WordNet), o texto é anotado com *POS-tagging* e realizado *lemmatization*. Para extração de conceito utilizam *C/NC-value*, após essa fase é utilizado *Formal Concept Analysis* (FCA) para extração de relações taxonômicas além de uma abordagem baseada em *clustering*. A extração de relações não taxonômicas é implementada usando o algoritmo preditivo a priori através do Weka. A seguir é exibido um lista de relações extraíveis através dessa abordagem.

```
cause by
need
result
lead to
follow
yield
analyze by
achieve
cause
give
decrease
depend
give
attenuate by
perform by
```

Além disso outra abordagem, baseada em probabilidade é utilizada. Essa abordagem consiste em utilizar a probabilidade de um conceito dado um verbo, através da função de probabilidade:

$$P(c|v) = \frac{f(c,v)}{f(v)} \tag{5.1}$$

Onde c é um conceito e v um verbo, além disso $f(c,v)$ e $f(v)$ representam a função acumulada da relação (c,v) e v, respectivamente. Na Tabela 15 exibimos os valores de *Pre-*

[3] https://opennlp.apache.org
[4] https://sourceforge.net/p/jwordnet/wiki/Home/

cision, *Recall* e *F-measure* para cada domínio de conhecimento (Ciência da Computação e Mediciona), entre as abordagens e cada fase de extração. Nota-se uma boa classificação dos conceitos, mas uma menor eficiência na identificação de quaisquer tipos de relações.

Domínio →	Ciência da Computação		Medicina		Total
Extração ↓	P	R	P	R	F
Conceito	0,8667	0,896	0,897	0,914	0,8933
Taxonomias - FCA	0,442	0,486	0,471	0,416	0,4524
Taxonomias - *Clustering*	0,7133	0,627	0,712	0,673	0,6797
Não Taxonômicas - Regras	0,7285	0,617	0,718	0,677	0,6825
Não Taxonômicas - Probabilístico	0,6167	0,494	0,627	0,559	0,5698
Média	0,67344	0,624	0,685	0,6478	0,6555

Tabela 15 – OntoGain - Avaliação de *Precision, Recall e F-measure* por tipo de extração

Na Tabela 16 mostramos os valores de *Precision*, *Recall* e *F-measure* para os componentes OWL extraídos e dos inferidos através do PUPO. Enquanto a média de *F-measure* do OntoGain é de 0,6555 a do PUPO é 0,82 para os testes aplicados (Veja detalhes do Capítulo 6). Essa melhor pontuação do PUPO pode evidenciar uma maior capacidade do PUPO de modelar conhecimento em relação ao OntoGain.

	Precision	Recall	F-measure
Componentes OWL extraídos	0,7609	0,8974	0,8235
Componentes OWL inferidos	0,7582	0,8846	0,8165
Média	0,7595	0,8910	0,82

Tabela 16 – PUPO - Avaliação de *Precision, Recall e F-measure* por tipo de extração

Na Tabela 17 observamos que o PUPO e o OntoGain possuem características semelhantes, porém oque diferencia nosso trabalho é a habilidade do PUPO realizar raciocínio durante o diálogo (Vide Capítulo 4). Outrossim o PUPO utiliza a SUMO para validar os axiomas inferido pelo processo de aprendizado de máquina. Fazendo assim que a intervenção do usuário para remover axiomas inferidos de forma semanticamente incorreta através de aprendizagem de máquina seja menor.

	OntoGain	PUPO
Manipulação de ontologias	Criação e Edição	Criação e Edição
Método	Semiautomático	Semiautomático
Abordagem	Linguística, Estatística, Aprendizado de Máquina, WordNet	Linguística, Aprendizado de Máquina, SUMO, WordNet
Raciocínio Automático	NÃO	SIM
Identificação e Tratamento de Inconsistências	NÃO	SIM
Ferramenta de PLN	OpenNLP	CoreNLP (Stanford Parser)

Tabela 17 – Comparativo entre funcionalidades do OntoGain e o PUPO

5.7 ONTOLOGY LEARNING IN THE DEEP

Aprendizado Profundo (*Deep Learning*) vem se destacando por suas contribuições em PLN. Aprendizado Profundo, segundo uma das definições encontradas em Deng (2014), é uma classe de técnicas de aprendizado de máquina que exploram muitas camadas de processamento não linear de informações para a extração e transformação de recursos supervisionados ou não supervisionados, e para análise e classificação de padrões.

Petrucci, Ghidini e Rospocher (2016) utilizam Aprendizado Profundo para aprender axiomas em OWL. É composta uma Rede Neural Recorrente (RNN) (Figura 55) para etiquetagem das palavras de uma sentença conforme sua função lógica. Assim dado uma sentença em linguagem natural correspondente a alguma representação formal, Petrucci, Ghidini e Rospocher (2016) aplicam uma etiqueta para cada palavra.

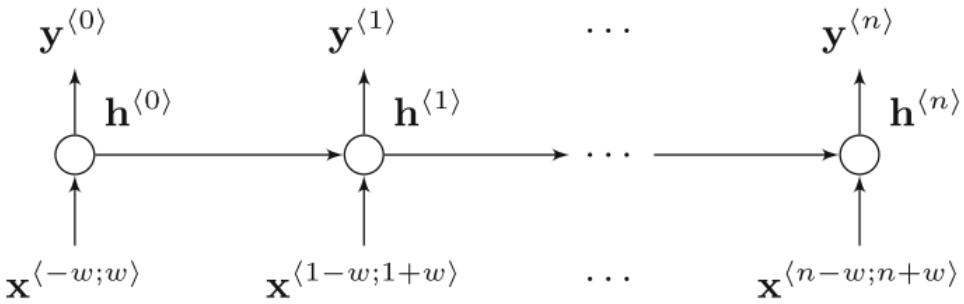

Figura 55 – RNN utilizada para etiquetagem de sentenças

Além do processo de etiquetagem, é realizado um processo de transdução em sentenças. Dado uma sentença em linguagem natural correspondente a uma representação formal, é identificado a estrutura de uma representação forma, a qual seus conceitos e papeis são conectados, e quais conectores são usados. Para isso utilizam de um *RNN Encoder-Decoder*(CHO et al., 2014), conforme Figura 56.

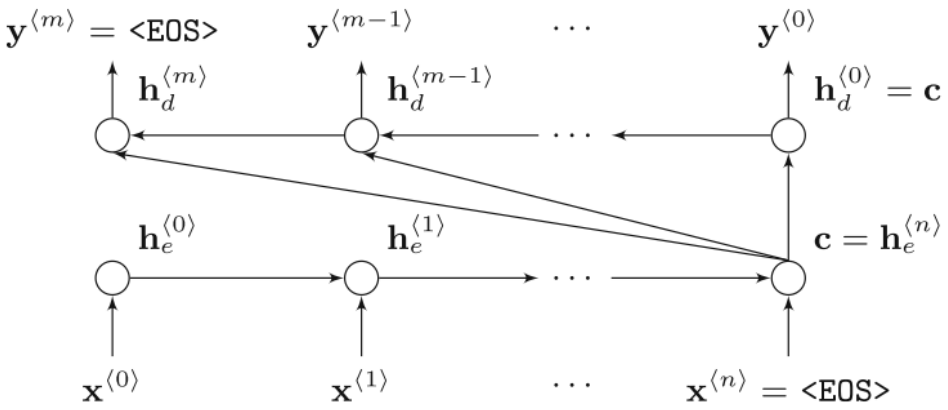

Figura 56 – RNN Encoder-Decoder utilizado para transdução de sentenças

Abaixo vemos um exemplo de como Petrucci, Ghidini e Rospocher (2016) constroi OWL a partir de textos.

Sentença: *A bee is a insect that has exactly* N_0 *legs*

Padrão em LN: *A* C_0 *is a* C_1 *that* R_0 *exactly* N_0 C_2

Sequência etiquetada: $w\ C_0\ w\ w\ C_1\ w\ R_0\ w\ N_0\ C_2$

Padrão Lógico: $C_0 \sqsubseteq C_1 \sqcap = N_0 R_0 . C_2$

Temos um padrão para LN, em seguida uma sequência da frase original etiquetada com a RNN proposta. Nesta etapa vemos a identificação de conceitos e papeis identificados. Por fim um padrão lógico é identificado com o RNN-Encoder-Decoder e por fim construído o axioma lógico correspondente.

Essa abordagem permite uma acurácia de 96%, gerando axiomas lógicos com expressividade $\mathcal{ALCQ}$. Apresentamos na Tabela 18 um comparativo entre as principais características do trabalho de Petrucci, Ghidini e Rospocher (2016) e o PUPO. Quanto a manipulação de ontologias o PUPO permite a criação e edição, enquanto Petrucci, Ghidini e Rospocher (2016) permitem apenas a criação. O método de Petrucci, Ghidini e Rospocher (2016) é automático, sem interação com usuário durante o processo de modelagem de conhecimento. Petrucci, Ghidini e Rospocher (2016) utilizam Aprendizado Profundo, o que é mais avançado tecnologicamente do que a SVM utilizada pelo PUPO. Por esse motivo apresenta um melhor desempenho que este trabalho, com acurácia de 96%.

O PUPO sai na frente a que diz respeito a utilização de raciocínio em tempo de modelagem. Por fim alcança uma expressividade maior que Petrucci, Ghidini e Rospocher (2016) ($\mathcal{ALCQ}$).

	Petrucci, Ghidini e Rospocher (2016)	PUPO
Manipulação de ontologias	Criação	Edição e Criação
Método	Automático	Semiautomático
Abordagem	Linguística, Aprendizado Profundo	Linguística, Aprendizado de Máquina, SUMO, WordNet
Raciocínio Automático	NÃO	SIM
Identificação e Tratamento de Inconsistências	NÃO	SIM
Ferramenta de PLN	-	CoreNLP (Stanford Parser)
Linguagem Gerada	OWL	OWL
Expressividade DL	$\mathcal{ALCQ}$	$\mathcal{ALCHOQ}$

Tabela 18 – Comparativo entre funcionalidades do trabalho de Petrucci, Ghidini e Rospocher (2016) e o PUPO

5.8 X_UDEKAM

O X_UDeKAM(ALI et al., 2016) é um *chatterbot* para aquisição de conhecimento baseado no processo da arquitetura do UDeKAM(ALI; LEE; KANG, 2016). O UDeKAM possui 2 (dois) processos principais (Vide Figura 57): 1 - Classificador de Recursos Desestruturados; 2 - Mecanismo de Linguagem Natural Controlada.

1 - Classificador de Recursos Desestruturados: Esse processo é compreendido da seleção de fontes de dados desestruturados para modelagem de conhecimento através de interface gráfica. Fontes como relatórios clínicos são utilizados como fonte, por exemplo.

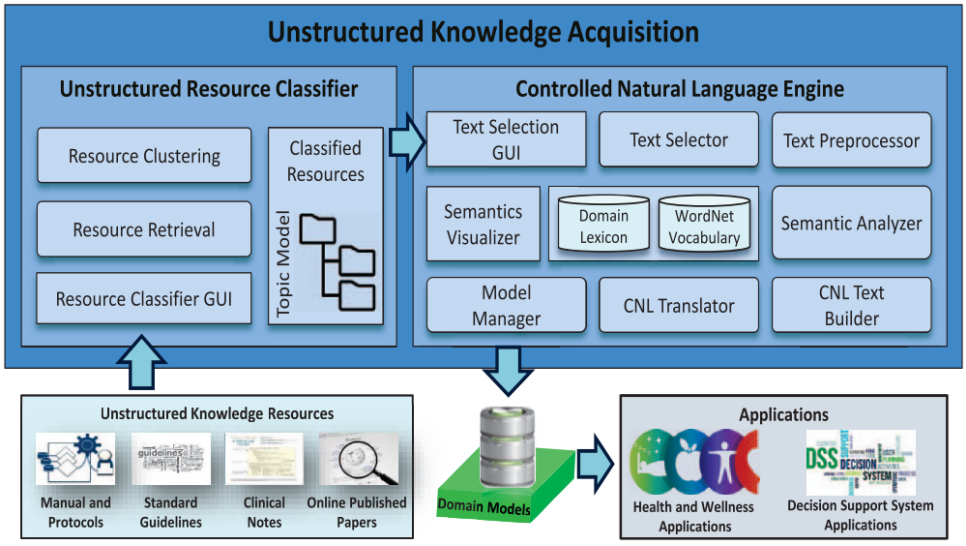

Figura 57 – Arquitetura do UDeKAM

2 - Mecanismo de Linguagem Natural Controlada: Nesse processo é realizada a seleção dos textos pertinentes ao conhecimento que se deseja modelar. Esses textos são pré-processados, analisados de forma semântica e modelados conforme seu domínio de conhecimento.

Segundo Ali et al. (2016) o X_UDeKAM estende o UDeKAM adicionando interação com o usuário através de um *chatterbot* com suas interações modeladas em AIML.

	Ali et al. (2016)	PUPO
Manipulação de ontologias	Criação	Edição e Criação
Método	Semiautomático	Semiautomático
Abordagem	Linguística, Aprendizado de Máquina, WordNet	Linguística, Aprendizado de Máquina, SUMO, WordNet
Raciocínio Automático	NÃO	SIM
Identificação e Tratamento de Inconsistências	NÃO	SIM
Ferramenta de PLN	-	CoreNLP (Stanford Parser)
Linguagem Gerada	OWL	OWL

Tabela 19 – Comparativo entre funcionalidades do trabalho de Ali et al. (2016) e o PUPO

Notamos que, segundo a Tabela 19, o PUPO possui o diferencial de utilizar Raciocínio automático durante o processo de modelagem de conhecimento. Ainda Ali et al. (2016) utilizam aprendizado de máquina para realizar agrupamento (*clustering*) de domínios de conhecimento. Enquanto o PUPO utiliza para selecionar axiomas semanticamente corretos.

5.9 COMPARAÇÃO DOS TRABALHOS RELACIONADOS

Nessa Seção comparamos as funcionalidades e as principais características dos trabalhos relacionados e do PUPO.

Na Tabela 20 apresentamos uma comparação entre as ferramentas do estado da arte para conversão de LN para OWL e/ou sistemas de diálogos. O 'X' representa que o trabalho apresenta a característica descrita na coluna da tabela, o '-' representa sua ausência e o '*' corresponde a um item não aplicável ou o trabalho não deixa explícito.

	Chatterbot/Sistema de Diálogo	Automática	Semiautomática	Linguística	Estatística	Aprendizagem de Máquina	Hierarquias	Relações	Axiomas	Raciocínio Automático	Tecnologias Utilizadas	Domínio
LExO (2007)	-	X	-	X	-	-	X	X	X	-	MINIPAR	Aberto
TextOntoEx (2008)	-	X	-	X	*	*	X	-	-	-	-	Agricultura
OntoGain (2010)	-	X	-	X	X	-	X	X	-	-	OpenNLP, WordNet	Ciência da Computação, Medicina
OntoRich (2011)	-	-	X	X	X	-	X	X	-	X	OpenNLP, RiTa WordNet, Ontologia WordNet, Weka (FCA, C/NC-value, clustering)	Aberto
PARNT (2013)	-	-	X	X	X	X	X	X	-	-	-	Biomedicina
Renan (De Azevedo et al., 2014)	X	-	X	X	-	-	X	X	X	X	Stanford Parser	Aberto
Petrucci, Ghidini e Rospocher (2016)	-	X	-	X	-	X	X	X	X	-	-	Aberto
Ali et al. (2016)	X	-	X	X	-	X	X	*	*	*	WordNet, Aprendizado de Máquina	Medicina

PUPO	X	-	X	X	-	X	X	X	X	X	Stanford Parser (CoreNLP), WordNet, SUMO, Weka (SVM, StringWord2Vec)	Aberto

Tabela 20 – Trabalhos relacionados - Características

Observamos na Tabela 20 que a forma mais utilizada na literatura é uma abordagem linguística. Conforme expomos na Figura 58, todos os oito trabalhos utilizam algum recurso linguístico para extração de informação, uma vez que fornece uma maior precisão. Por sua vez, aprendizagem de máquina é utilizada em 3 (três) trabalhos, indicando um campo ainda a ser bem explorado. Já a abordagem estatística está presente em pelo menos três trabalhos.

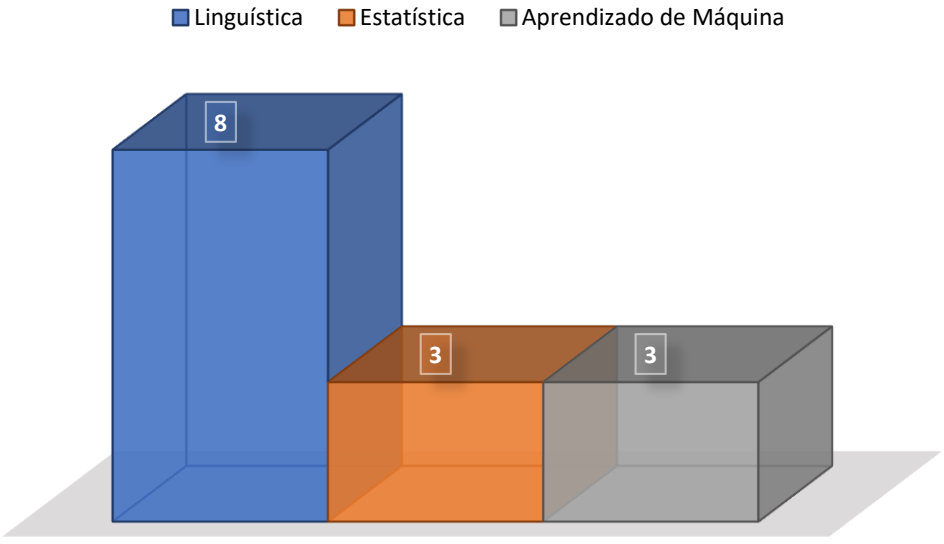

Figura 58 – Comparativo entre técnicas linguísticas, estatísticas e de aprendizado de máquina

Os trabalhos como o LExO, TextOntoEx, OntoGain e Petrucci, Ghidini e Rospocher

(2016) buscam por uma abordagem automática onde o usuário pouco interagia durante o processo de conversão de LN para OWL. Já na maioria dos trabalhos mais recentes (OntoRich, PARNT, Ali et al. (2016), Renan e PUPO) percebeu-se a necessidade do usuário interagir com o sistema afim de modelar ontologias semanticamente mais fiés ao interesse do usuário. O Renan, Ali et al. (2016) e o PUPO nesse contexto permitem uma aproximação maior entre usuário e o processo de modelagem, pois estes utilizam-se de diálogo como meio de interação.

Na Figura 59 demonstramos que todos os trabalhos realizam extração de hierarquias entre conceitos, pois esta é a forma mais comum de extração.

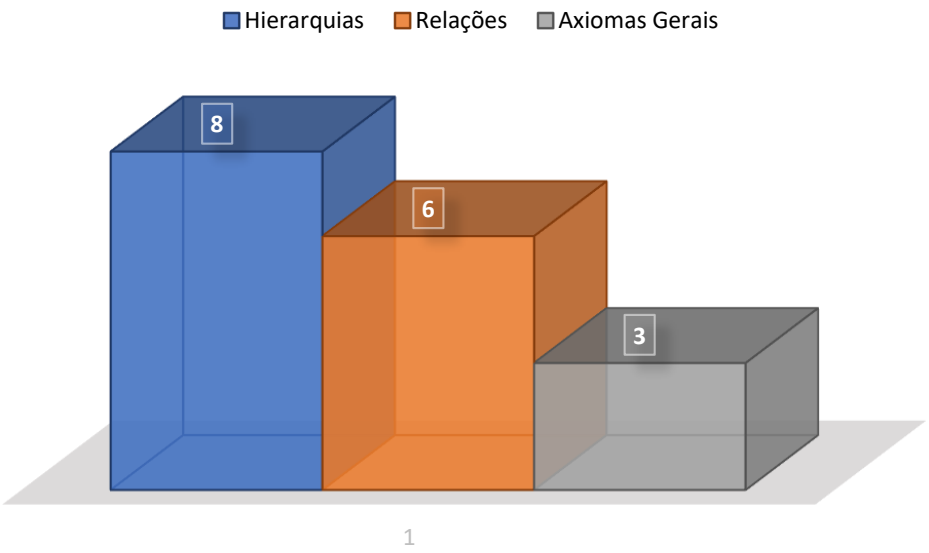

Figura 59 – Comparação entre tipos de extração realizadas nos trabalhos relacionados

Por exemplo, consideremos hierarquia apresentada na Figura 60.

```
Animal
    |_ Mamífero
            |_ Cão
            |_ Gato
```

Figura 60 – Exemplo de hierarquia de conceitos

No exemplo acima, Gato e Cão são Mamíferos, e por sua vez, um Mamífero é um Animal. Esse tipo de extração recupera informação de forma hierárquica. Ainda na Figura 59

```
Juvenil
    |_ PeixeJovem
    |_ AnimalJovem
    |_ não chegou alguma ( MaturidadeSexual)

Propriedades
    chegou
```

Figura 61 – Exemplo de conceitos e propriedades

o tipo de extração de relações está presente em seis dos trabalhos. Esse tipo de relação é um pouco mais completa que as hierarquias, por exemplo na sentença *Juvenile is a young fish or animal that has not reached sexual maturity* (Juvenil é um peixe ou animal jovem que não chegou a maturidade sexual) serão extraídas as relações:

1. Juvenile is a young fish (Juvenil é um peixe jovem);
2. Juvenile is a young animal (Juvenil é um animal jovem);
3. Juvenil has not reached sexual maturity (Juvenil não chegou a maturidade sexual).

Quando transformamos essas relações em um formalismo lógico o verbo *to be* (ser, estar) se torna hierarquia e os outros verbos se tornam propriedades, como segue.

Por fim pelo menos quatro trabalhos, incluindo o PUPO, criam axiomas gerais combinando hierarquias e propriedades, esses tipos de axiomas são, na maioria das vezes, complexos. A seguir veremos a frase *Juvenile is a young fish or animal that has not reached sexual maturity* (Juvenil é um peixe ou animal jovem que não chegou a maturidade sexual) axiomatizada[5].

Segundo Völker, Hitzler e Cimiano (2007)

- Juvenile EquivalentTo: (Young and (Fish or Animal) and not reached some (Sexual and Maturity))

Segundo De Azevedo et al. (2014)

- juvenile SubClassOf: (young_fish or young_animal) and not hasReached only sexual_maturity

Segundo PUPO

- Juvenile SubClassOf: (YoungFish or YoungAnimal) and not (has some (reached some SexualMaturity))

[5] Para traduções dos exemplos seguintes vide Apêndice D

Dos trabalhos pesquisado apenas De Azevedo et al. (2014), Barbur, Blaga e Groza (2011) e o PUPO utilizam de raciocinadores durante o processo de construção das ontologias. Dito isso, sabemos que De Azevedo et al. (2014) e o PUPO interagem com o usuário através de diálogos e raciocinam através de motores de inferência (utilização do *Pellet*). Esse poder de raciocínio pode ser o motivo do PUPO e o Renan conseguirem dialogar em domínios de conhecimento aberto, isto é, sobre quaisquer assuntos.

Na Tabela 21 apresentamos o desempenho dos trabalhos relacionados. As ferramentas `LExO`, `OntoRich` e Ali et al. (2016) não evidenciam em seus trabalhos uma medida de desempenho de suas abordagens, por isso não foram inclusos na análise. Foi analisada a capacidade de cada sistema extrair: I)Hierarquias; II) Relações; e III) Axiomas. Marcamos como 1 (um) a coluna da tabela que o trabalho consegue extrair, e com 0 (zero) caso contrário. A coluna % representa o valor de desempenho descrito no trabalho. Por sua vez a coluna % Ponderado segue a seguinte fórmula:

$$\%\_Ponderado = \frac{\sum_{i=0}^{n} F_i^t * p}{3} \tag{5.2}$$

Onde F_i^t representa a i-ésima característica de um trabalho t. E p representa o desempenho do trabalho. Dividimos por 3, por estarmos analisando 3 características (Hierarquias, Relações e Axiomas). Como exemplo consideremos o OntoGain, então teremos o seguinte cálculo:

$$\%\_Ponderado = \frac{1 * 65,55 + 1 * 65,55 + 0 * 65,55}{3} = \frac{65,55 + 65,55}{3} = \frac{131,10}{3} = 43,70 \tag{5.3}$$

Obtemos assim o valor de desempenho ponderado de acordo o tipo de extração que cada trabalho realiza. Esse cálculo ponderado permite aproximar os trabalhos quanto a suas características de extração.

	Hierarquias	Relações	Axiomas	%	% Ponderado
TextOntoEx	1	0	0	70,00	23,33
OntoGain	1	1	0	65,55	43,70
PARNT	1	1	0	26,47	17,65
Renan	1	1	1	75,00	75,00
Petrucci, Ghidini e Rospocher (2016)	1	1	1	96,00	96,00
MÉDIA				66,60	51,14

Tabela 21 – Desempenho dos trabalhos relacionados

Petrucci, Ghidini e Rospocher (2016) apresenta melhor percentual com 96 de desempenho. Enquanto o pior ficou por conta do `TextOntoEx`. A média de desempenho dos

trabalhos fica em 66,60, enquanto *% Ponderado* apresenta valor 51,14.

A seguir, apresentamos no Capítulo 6, os experimentos realizados que validam o PUPO, compostos por levantamento de questões de pesquisa e hipóteses, por fim concluímos o capítulo e apontamos algumas limitações do PUPO que poderão ser tratadas em trabalhos futuros.

6 EXPERIMENTOS E RESULTADOS

Nessa seção apresentamos nossos experimentos e sua execução. Apresentamos, ainda, as hipóteses, os testes para validá-las bem como os resultados obtidos. A seguir são apresentadas a questão de pesquisa e as hipóteses deste trabalho.

6.1 QUESTÕES DE PESQUISA E DEFINIÇÕES DAS HIPÓTESES

Para demonstrar que o PUPO é capaz de cumprir seus objetivos, formulamos a seguinte questão de pesquisa:

- Um *chatterbot* é capaz de modelar conhecimento, realizar raciocínio e resolver inconsistências a partir da interação em linguagem natural com os usuários?

Dado a questão de pesquisa, foram definidas as hipóteses que seguem:

- **Hipótese Nula** – H_0: O PUPO **não é** capaz de modelar conhecimento e realizar raciocínio através de diálogos.
- **Hipótese de Pesquisa** – H_1: O PUPO **é** capaz de modelar conhecimento e realizar raciocínio através de diálogos.

Assumimos que o PUPO "é capaz"e cumpre seus objetivos quando:

- O PUPO constrói corretamente ontologias com expressividade máxima $\mathcal{ALCOHQ}$;
- Cria axiomas utilizando uma abordagem híbrida entre linguística, aprendizado de máquina e fontes semânticas;
- Realiza raciocínio sobre as ontologias modeladas;
- Identifica e trata inconsistências em tempo de execução.

6.1.1 Formalização das Hipóteses

Afim de simplificar a **Hipótese 1** – H_1, esta foi dividida em hipóteses menores, formalizando assim aquela já definida na Seção 6.1. Assim, foram definidas 4 (quatro) subhipóteses: Hipóteses 1(a), 1(b), 1(c) e 1(d) com uma questão de pesquisa cada.

Como o desempenho ponderado dos trabalhos relacionados foi de 51,14 (Vide Tabela 21), escolheu-se utilizar um limiar de 55 para o desempenho do PUPO. Assim este estaria acima da média do desempenho de seus trabalhos relacionados. Esse valor foi assumido como limiar das sub-hipóteses 1(a) e 1(b).

Essas subhipóteses são detalhadas a seguir.

Hipótese 1(a)

Questão de Pesquisa 1(a)

- O PUPO é capaz de criar ontologias em expressividade máxima $\mathcal{ALCOHQ}$ durante diálogos com os usuário?

Com a finalidade de responder a essa pergunta foi formulada a seguinte subhipótese:

- $H_{0,1}$: PUPO não constrói ontologias corretamente em no mínimo 55% dos casos.
- $H_{a,1}$: PUPO constrói ontologias corretamente em no mínimo 55% dos casos.

Assim, entende-se que o PUPO será capaz de modelar conhecimento advindo de pelo menos 55% das sentenças de definições apresentadas em diálogos.

Hipótese 1(b)

Questão de Pesquisa 1(b)

- O PUPO é capaz de realizar raciocínio durante o diálogo com os usuários?

Para responder essa pergunta, formulou-se a seguinte hipótese:

- $H_{0,2}$: O PUPO não realiza raciocínio em 55% dos casos durante o diálogo com os usuários.
- $H_{a,2}$: O PUPO realiza raciocínio em 55% dos casos durante o diálogo com os usuários.

Dessa forma, o PUPO será apto a realizar raciocínio em 55% das sentenças que apresentem algum axioma de subsunção/definição possível advindos das descrições das classes.

Hipótese 1(c)

Questão de Pesquisa 1(c)

- O PUPO é capaz de identificar fatos contraditórios contido na(s) ontologia(s) enquanto dialoga com os usuários?

Durante o processo de conversação com os usuários o PUPO deve prover estratégias necessárias para identificação de todos os fatos contraditórios ditos pelos usuários. Dessa maneira foi formulada a seguinte hipótese:

- $H_{0,3}$: O PUPO não verifica inconsistências na(s) ontologia(s) em pelo menos 100% dos casos.
- $H_{a,3}$: O PUPO verifica inconsistências na(s) ontologia(s) em pelo menos 100% dos casos.

Assim, compreende-se que o PUPO seja capaz de identificar inconsistências em sentenças contraditórias em 100% das sentenças que de fato apresentem contradições expressas em linguagem natural.

Esse percentual de 100% foi escolhido devido ao fato de que uma base de conhecimento **não deve** apresentar inconsistência em seus axiomas, pois isso compromete o raciocínio dos motores de inferência. O `Pellet` por exemplo **não realiza** raciocínio caso exista alguma inconsistência na ontologia modelada.

Hipótese 1(d)
Questão de Pesquisa 1(d)

- A utilização de aprendizado de máquina e fontes semânticas de dados fornecem ao PUPO uma maior capacidade de gerar axiomas em OWL DL?

Para essa pergunta, foi formalizada a sequinte hipótese:

- $H_{0,4}$ A utilização de aprendizado de máquina, WordNet e SUMO não fornecem melhorias no processo de axiomatização realizado pelo PUPO.
- $H_{a,4}$ A utilização de aprendizado de máquina, WordNet e SUMO fornecem melhorias no processo de axiomatização realizado pelo PUPO.

Essa última hipótese verifica se há algum ganho para o PUPO utilizar a ontologia de topo SUMO, o dicionário de dados *WordNet* e algoritmos de Aprendizado de Máquina para melhor axiomatizar as sentenças em linguagem natural para OWL DL.

6.1.2 Configuração Experimental

Em um primeiro momento a realização dos experimentos definem os melhores parâmetros para geração dos modelos de extração de relação para o *dataset* DSS1(Vide Seção 6.1.2.1). Além de verificar através da `SUMO` que os axiomas extraídos por aprendizagem de máquina estão semanticamente corretos segundo a Ontologia de topo `SUMO` e o `WordNet`.

Escolha do pré-processamento das bases: a técnica de pré-processamento da base (DSS1) escolhida e apoiada pela literatura foi *WordToVec*(WOHLGENANNT; MINIC, 2016) (ou *StringWord2Vec*, ou ainda *Word2Vec*). *WordToVec* é aplicado com a escolha de alguns parâmetros: *TF*, *IDF*, *lowerCaseTokens*, *stopword* e uso de *n-gram*. Com exceção de *n-grams* os parâmetros têm natureza binária, ou seja, o pré-processamento é realizado com a utilização ou não de tal argumento. Já a utilização de *n-grams* consiste em definir um valor de *n*, no caso deste trabalho foi utilizado *2grams* (bigramas). O *WordToVec* gera como resultado um vetor numérico que representa a sentença original. Esse vetor foi submetido a normalização para que sua representação numérica de cada característica (*feature*) ficasse no intervalo [0,1].

A técnica de *WordToVec* embarca as palavras de uma sentença em vetores numéricos. O vetor utilizado pelo PUPO possui dimensão igual a 100000, ou seja 100000 bigramas são

utilizados para representar uma sentença qualquer. A probabilidade de uma palavra não está presente nesse vetor é na maioria das vezes muito baixa. Desse forma a representação vetorial da sentença será a mais fiel possível.

Mesmo com essa alta dimensionalidade, ainda pode existir casos em que uma palavra seja desconhecida. Espera-se com essa técnica que a probabilidade de encontrar uma sentença onde todas as palavras sejam desconhecidas seja baixíssima, ao ponto de não influenciar o desempenho geral do PUPO. Pois com a maioria das palavras conhecidas, o *WordToVec* gera um vetor característico aproximado a sentença completamente conhecida.

Estimando os parâmetros do algoritmo de aprendizagem de máquina: baseado na literatura, a escolha do algoritmo foi `SVM` (*Support Vector Machine*, máquina de vetor suporte) com *kernel* RBF (*Radial Basis Function*, Função de Base Radial). Para construção da `SVM` é necessário estimar 2 (dois) parâmetros, o custo (C) e o *gamma*. O melhor valor de *gamma* e custo (C), para o a base de treino, é obtido através de *GridSearch*[1]. O espaço de busca do custo foi fixado no intervalo [1,1000] através da expressão:

$$pow(BASE, I) \tag{6.1}$$

Onde *pow* representa o cálculo de exponenciação, a *BASE* tem valor fixo 10 (dez) e I varia no intervalo [0,3].

Para o valor de *gamma* utilizou-se o intervalo [10^{-10},10^{-1}] através da expressão:

$$pow(BASE, I) \tag{6.2}$$

Com *BASE* igual a 10 (dez) e I variando no intervalo [-10,-1].

Na composição das bases (*dataset* DS1 e DSS1, vide Seção 6.1.2.1) utilizadas para validar o *chatterbot* durante os experimentos realizados (Vide Seção 6.2) utilizamos a API do *WordNet* denominada `extjwnl`[2], e através dela foram realizadas duas tarefas:

Tarefa 1 Obtenção de todos os substantivos/conceito indexados pelo `WordNet` – A escolha de obtenção dos substantivos deu-se de forma aleatória, das quais cada sentença representa uma definição clara e acessível de um substantivo/conceito através do *WordNet*;

Tarefa 2 Como o *WordNet* pode apresentar várias definições para um mesmo substantivo, realizou-se a captura da primeira definição de cada substantivo/conceito e sua junção através do verbo "is", por exemplo:

Conceito: *dog*

Definição: *a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times*

[1] https://weka.wikispaces.com/Optimizing+parameters?responseToken=03a5ef5741e9c19d5c7c1dae3987a5b72

[2] https://github.com/extjwnl/extjwnl

Sentença gerada: *dog **is** a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times*

6.1.2.1 Dataset (DS1)

O DS1 é um *dataset* composto por um conjunto de **1.242** (mil duzentos e quarenta e duas) sentenças extraídas do `WordNet`, conforme as tarefas 1 e 2 apresentadas na seção anterior. Desse *dataset*, 50% (cinquenta por cento) foi utilizado para treinamento dos algoritmos de aprendizagem de máquina e os outros 50% (cinquenta por cento) para teste.

Preparação do *dataset* experimental DSS1

A partir de DS1 foram extraídas **621** sentenças e dessas sentenças extraídas suas possíveis relações compondo o *dataset* DSS1, na seguinte estrutura.

```
'classroom is a room in a school where lessons take place','classroom Be room',1
```

A primeira sentença entre aspas simples é composta pela definição do `WordNet`. A segunda sentença é composta por alguma relação extraída da definição. Por fim, adicionado o número 1 para relações válidas semanticamente e 0 para relações inválidas semanticamente. Por exemplo:

Sentença: *sala de aula é uma sala em uma escola onde aulas acontecem*

Possíveis relações:

1. sala de aula é uma sala;
2. sala de aula é uma escola;

Classificação:

1. sala de aula é uma sala; (1 - Válida)
2. sala de aula é uma escola; (0 - Inválida)

As outras **621** (seiscentos e vinte e uma) sentenças foram mantidas como extraídas do `WordNet` para servirem de teste. Dentre as quais, **100** (cem) (Vide Apêndice H) foram utilizadas para testes de construção e raciocínio do PUPO para comprovação das hipóteses *1(a)* e *1(b)*, como apresentamos na Figura 62.

Ao final do processo de avaliação a base contou com 9.550 (nove mil, quinhentos e cinquenta) relações classificadas, sendo 4.775 (quatro mil, setecentos e setenta e cinco) classificadas como 0 (zero, ou seja inválidas) e as outras 4.775 (quatro mil, setecentos e setenta e cinco) como 1 (um, ou seja válidas), conforme Figura 63.

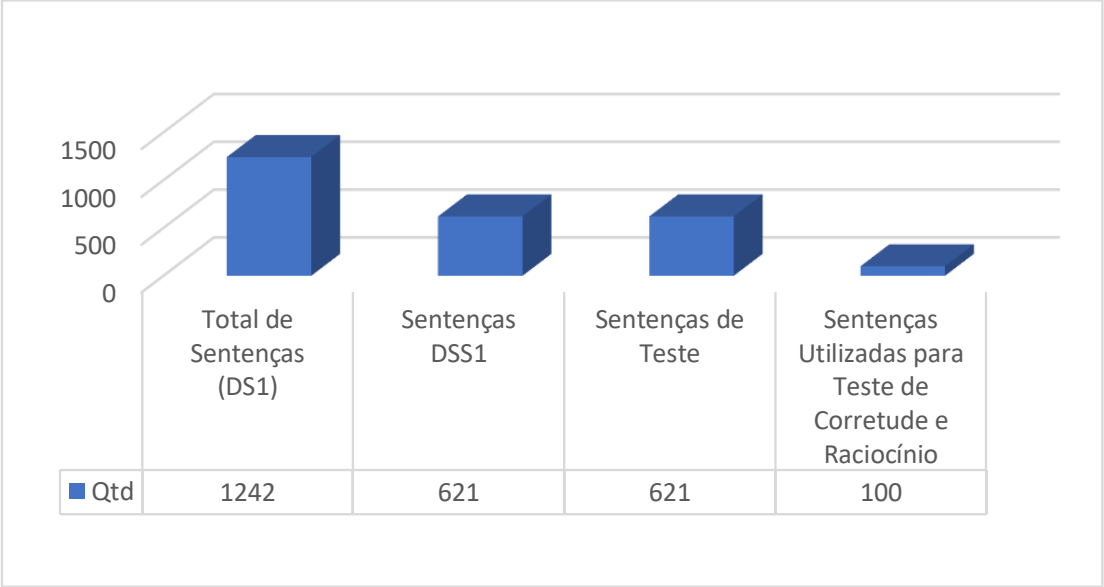

Figura 62 – Quantidade e Utilização das Sentenças Selecionadas

6.1.3 Construção e Raciocínio em Ontologias

As sentenças, abaixo, foram utilizadas para composição da base de exemplos para evidenciar a capacidade do PUPO de modelar conhecimento a partir de textos advindos dos usuários. Essas sentenças foram retiradas de (VÖLKER; HITZLER; CIMIANO, 2007) e (De Azevedo et al., 2014). A escolha dessas sentenças se deu devido a serem utilizadas em trabalhos relacionados, possibilitando uma análise sobre o método proposto, como seguem:

1. Data are facts that result from measurements or observations.
2. InternalRateOfReturn is a financial or economic indicator of the net benefits expected from a project or enterprise, expressed as a percentage.
3. Vector is an organism which carries or transmits a pathogen.
4. Shark is a fish and an aquatic animal. It has only cartilaginous skeleton and is not a marine mammal.
5. Juvenile is a young fish or animal that has not reached sexual maturity.
6. Tetraploid is a cell or organism having four sets of chromosomes.

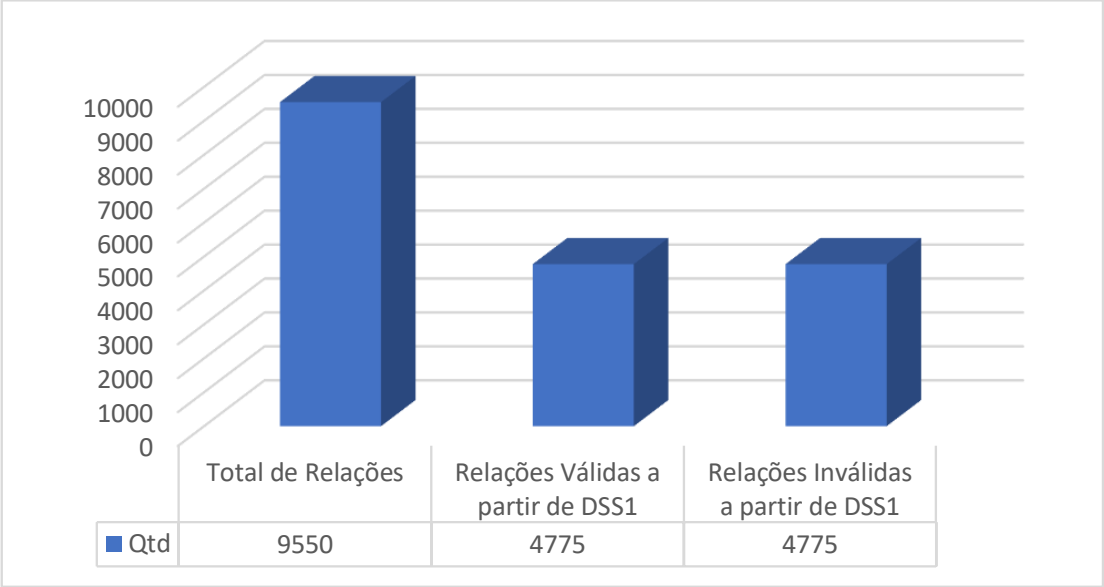

Figura 63 – Quantidade e Utilização das Relações Selecionadas

7. Pair Trawling is a bottom or mid-water trawling by two vessels towing the same net.

8. Biosphere is the portion of Earth and its atmosphere that can support life.

9. Vehicles are non-living means of transportation.

10. A minister or a secretary is a politician who holds significant public office in a national or regional government.

11. An island or isle is any piece of land that is completely surrounded by water.

12. Days of the week are Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.

6.2 EXECUÇÃO DOS EXPERIMENTOS

Nesse seção são apresentados exemplos de funcionamento e validação das Hipóteses levantadas.

6.2.1 Construção de ontologias - Hipótese 1(a)

Sentença (1): data are facts that result from measurements or observations.
Resultado (1):

1. Data SubClassOf: Fact and ((resultFrom some Measurement) or (resultFrom some Observation))

Discussão (1): Essa sentença foi axiomatizada de forma semelhante a Völker, Hitzler e Cimiano (2007). A diferença consistiu apenas na distribuição dos conceitos *Measurement* e *Observation* com apropriedade *resultFrom*. Essa diferença não compromete o valor semânticos do axioma:

```
Data SubClassOf: Fact and ((resultFrom some Measurement) or (resultFrom
some Observation))
```

A sentença foi corretamente axiomatizada, gerando 4 (quatro) classes *Data, Fact, Measurement* e *Observation*. Vemos ainda que *result from* foi transformado na propriedade *resultFrom*. Völker, Hitzler e Cimiano (2007) trata a sentença como sendo:

```
Data EquivalentTo: (Fact and result_from some (Measurement or Observation
))
```

Diferente de Völker, Hitzler e Cimiano (2007), o PUPO trata como uma hierarquia de classes (*SubClassOf*), dado que a equivalência é algo que define tudo que uma classe é. Como o processo de diálogo com o PUPO permite uma modelagem constante assumimos que a utilização de *SubClassOf* constitui uma melhor axiomatização. Evitando assim que uma só classe possuísse várias definições. Por exemplo:

Segundo Völker, Hitzler e Cimiano (2007)

```
Frases:
Gato é um mamífero e possui olhos
Gato é um animal que bebe leite e possui pernas

Axiomas:
Gato EquivalenteA: Mamífero e ( possui algum Olhos )
Gato EquivalenteA: Animal e bebe algum leite e ( possui algum Pernas )
```

Percebemos que ao utilizar a equivalência se alguma outra classe possuir em sua definição *Animal e bebe algum leite e (possui algum Pernas)* então será classificado como Gato. Essa definição é equivocada, pois nem todo animal que bebe leite e possui pernas é um gato (pode ser um cão, por exemplo). Da mesma forma sobre o axioma *Gato EquivalenteA: Mamífero e (possui algum Olhos)*.

Sentença (2): InternalRateOfReturn is a financial or economic indicator of the net benefits expected from a project or enterprise, expressed as a percentage.
Resultado (2):

1. InternalRateOfReturn SubClassOf: ((EconomicIndicator) and (Financial) and ((((of some (expectedFrom some Enterprise)) or (of some (expectedFrom some (expressedAs some Percentage)))) or (of some (exprectedFrom some Project))) and (of some NetBenefit)))

Discussão (2): Völker, Hitzler e Cimiano (2007) axiomatiza a sentença acima como sendo:

```
InternalRateOfReturn EquivalentTo: ((Financial or Economic) and indicator
and of some (Net and Benefit and expected_from some (Project or Enterprise
)) and expressed_as some Percentage)
```

Notamos novamente a distribuição do PUPO em relação a propriedade *expectedFrom*. Notamos nesse exemplo que a expressão *(EconomicIndicator) and (Financial)* deveria ser *(EconomicIndicator) or (FinancialIndicator)*. Dessa forma nota-se um equívoco do PUPO ao tratar sentenças longas com várias propriedades ligadas através de conjunções. Dando espaço assim para melhorias futuras.

Sentença (3): vector is an organism which carries or transmits a pathogen.
Resultado (3):

1. Vector SubClassOf: Organism
2. Vector SubClassOf: Organism and ((carries some owl:Thing) or (transmits some Pathogen))

Discussão (3): O PUPO se comportou, nesse caso, como apresentado por Völker, Hitzler e Cimiano (2007). Uma melhoria futura pode ser o tratamento, desse tipo de caso, realizado como De Azevedo et al. (2014), que trata a sentença axiomatizada como:

```
Vector SubClassOf: Organism and ((carries some Pathogens) or (transmits
some Pathogens))
```

Nesse caso o PUPO poderia distribuir os verbos *carries* e *transmits* com a classe *Pathogen*, assim gerando um axioma com mais detalhes semânticos.

Sentença (4): shark is a fish and an aquatic animal. It has only cartilaginous skeleton and is not a marine mammal.
Resultado (4):

1. Shark SubClassOf: AquaticAnimal
2. Shark SubClassOf: Fish
3. Shark SubClassOf: AquaticAnimal and Fish
4. Shark SubClassOf: (not (MarineMammal)) and (has only CartilaginousSkeleton)
5. CartilaginousSkeleton EquivalentTo: Cartilaginous and Skeleton
6. AquaticAnimal EquivalentTo: Animal and Aquatic
7. MarineMammal EquivalentTo: Mammal and Marine

Discussão (4): Os termos *aquatic* e *animal* foram juntos formando o conceito *AquaticAnimal*. Nesse momento o PUPO infere um novo axioma onde *AquaticAnimal* é equivalente a *Aquatic* e a *Animal*, veja mais detalhes na Seção 6.2.2. Além disso criou uma restrição universal indicando que *Shark* (Tubarão) tem **apenas** *CartilaginousSkeleton* (esqueleto cartilaginoso). Outro fato interessante que não é tratado por Völker, Hitzler e Cimiano (2007), mas que De Azevedo et al. (2014) já trata é a correferência. O tratamento de correferência realizado pelo PUPO substituiu o pronome *It* pela sua referência, *Shark* (Tubarão). Esse tratamento é bastante interessante dado a variância na linguagem natural e seu reaproveitamento de termos. Esse tratamento de correferência permite que as frases descritas acima sejam entendidas antes da axiomatização como:

```
shark is a fish and an aquatic animal.
shark has only cartilaginous skeleton and is  not a marine mammal.
```

Sentença (5): juvenile is a young fish or animal that has not reached sexual maturity.
Resultado (5):

1. Juvenile SubClassOf: (YoungAnimal or YoungFish) and (has some (not (reached some SexualMaturity)))
2. Juvenile SubClassOf: YoungAnimal or YoungFish
3. YoungAnimal EquivalentTo: Animal and Young

4. YoungFish EquivalentTo: Fish and Young

Discussão (5): O PUPO realizou uma axiomatização mais detalhada que Völker, Hitzler e Cimiano (2007) ou De Azevedo et al. (2014), pois a propriedade *hasReached* foi dividude em duas expressando que *Juvenile* é um *YoungAnimal* ou um *YoungFish* de forma que o mesmo possui (propriedade *has*) algo que **não** alcançou (*reached*) *SexualMaturity* (maturidade sexual). Além disso construiu dois axiomas de equivalência:

```
YoungAnimal EquivalentTo: Animal and Young
YoungFish EquivalentTo: Fish and Young
```

Esses axiomas indicam que *YoungAnimal* é um *Animal* e um *Young*. De forma semelhante foi realizado em *YoungFish*. O axioma abaixo foi inserido na ontologia a partir de deduções feitas a partir de aprendizado de máquina, antes mesmo do processo de raciocínio pelo `Pellet`.

```
Juvenile SubClassOf: YoungAnimal or YoungFish
```

Assim o PUPO foi capaz de inserir três axiomas a mais que Völker, Hitzler e Cimiano (2007). Em relação a De Azevedo et al. (2014) inseriu *Juvenile SubClassOf: YoungAnimal or YoungFish* a mais, além dos dois axiomas de definição.

Sentença (6): tetraploid is a cell or organism having four sets of chromosomes.
Resultado (6):

1. Tetraploid SubClassOf: (Cell or Organism) and (having exactly 4 (setOf some Chromosome))
2. Tetraploid SubClassOf: Cell or Organism
3. Tetraploid SubClassOf: having exactly 4 (setOf some Chromosome)

Discussão (6): Nesse exemplo, foram construídas duas propriedades *having* e *setOf*. A extração realizada pelo PUPO conseguiu fornecer um axioma com cardinalidade se saindo melhor que De Azevedo et al. (2014). Além disso a propriedade *setOf* foi mais fiel em relação a apresentada por Völker, Hitzler e Cimiano (2007). O axioma extraído por Völker, Hitzler e Cimiano (2007) apresenta dois termos distintos (o conceito *set* e a propriedade *of*) definindo-o como:

```
Tetraploid EquivalentTo: ((Cell or Organism) or having exactly 4
(Set and of some (Chromosomes)))
```

A capacidade que o PUPO tem de extrair cardinalidade permite modelar conhecimento de forma mais fiel, aumentando assim a expressividade das ontologias modeladas.

Sentença (7): pair Trawling is a bottom or mid-water trawling by two vessels towing the same net.
Resultado (7):

1. PairTrawling EquivalentTo: Pair and Trawling
2. PairTrawling SubClassOf: Bottom or ((mid-waterTrawlingBy some (towing some SameNet)) and (mid-waterTrawlingBy exactly 2 Vessel))

Discussão (7): Essa sentença possui uma relativa complexidade devido a grande quantidade de termos conectados semanticamente. Segundo o PUPO temos que *Pair Trawling* (Pesca de Arrasto em Par) é subclasse de *Bottom* (algo profundo) ou um arrasto em águas médias por (mid-waterTrawlingBy) rebocadores de mesma rede (*towing some SameNet*) e também é um arrasto em águas médias por exatamente 2 navios (*mid-waterTrawlingBy exactly 2 Vessel*). O PUPO ainda conseguiu extrair o axioma *PairTrawling EquivalentTo: Pair and Trawling* de definição de maneira correta.

Völker, Hitzler e Cimiano (2007) extrai o axioma como sendo:

```
PairTrawling EquivaletTo: ((Bottom or MidWater) and Trawling and by
exactly 2 (Vessel and tow some (Same and Net)))
```

Notamos que enquanto Völker, Hitzler e Cimiano (2007) trata *Trawling* e *MidWater* como conceitos o PUPO junta esses conceitos em uma propriedade. Dessa forma a semântica extraída pelo PUPO através da propriedade *mid-waterTrawlingBy* (que tem como significado *pesca de arrasto em águas médias por*) é mais fiel em relação a de Völker, Hitzler e Cimiano (2007).

Sentença (8): biosphere is the portion of Earth and its atmosphere that can support life.
Resultado (8):

1. Biosphere SubClassOf: EarthAtmosphere
2. Biosphere SubClassOf: EarthAtmosphere and (support some Life)
3. Biosphere SubClassOf: (EarthAtmosphere and (portionOf some ({Earth}))) and (support some Life)

4. EarthAtmosphere EquivalentTo: Atmosphere and Earth

Discussão (8): Nesse exemplo, outra habilidade muito importante do PUPO é observada, a habilidade de misturar indivíduos nas definições de conceitos. *Biosphere* é definida entre outras coisas como:

```
Biosphere SubClassOf: portionOf some ({Earth})
```

O termo *{Earth}* entre colchetes identifica um indivíduos, ou seja *Biosphere* (Biosfera) é uma *portionOf {Earth}* (porção da Terra).

Para melhor entendimento, consideremos a sentença:

```
brasileiro vive no Brasil
```

Onde brasileiro que representa uma classe de pessoas é definida como sendo "aquele"que vive no Brasil – Brasil neste contexto tem identidade própria, por isso é tratado como um indivíduo do mundo real e seria axiomatizado como:

```
Brasileiro SubClasseDe: viveNo algum {Brasil}
```

Essa possibilidade que o PUPO confere ao usuário permite a modelagem de ontologias mais expressivas, juntando indivíduos e classes em um mesmo axioma.

Sentença (9): vehicles are non-living means of transportation.
Resultado (9):

1. Vehicle SubClassOf: non-livingMeansOf some Transportation

Discussão (9): Nessa sentença o PUPO inseriu o termo *non* junto da propriedade *livingMeansOf* fornecendo um sentido negativo a propriedade.Völker, Hitzler e Cimiano (2007) extrai esse axioma de forma diferente, como segue.

```
Vehicle EquivalentTo: not Living and Means and of some Transportation
```

O axioma extraído pelo PUPO junta os termos *non-living* e *Means* e a preposição *of* para constituir a propriedade *non-livingMeansOf* fornecendo a semântica que antes pertencia aos conceitos (segundo Völker, Hitzler e Cimiano (2007)) para tal propriedade.

Sentença (10): a minister or a secretary is a politician who holds significant public office in a national or regional government.
Resultado (10):

1. (Minister or Secretary) SubClassOf: (Politician) and ((holds some (significantPublicOfficeIn some National)) or (holds some (significantPublicOfficeIn some RegionalGovernment)))

2. RegionalGovernment EquivalentTo: Government and Regional

Discussão (10): O PUPO poderia ter realizado a distribuição do termo *National* com *Government* como fez Völker, Hitzler e Cimiano (2007), gerando idealmente as propriedade *NationalGovernment* e *RegionalGovernment*. Contudo o axioma modelado pelo PUPO é válido e considerando com semântica e propriedades bem definidas. A maioria das diferenças entre Völker, Hitzler e Cimiano (2007) e o PUPO se tratam de decisão de forma de modelar conhecimento, que é inerente a cada engenheiro de conhecimento. Como o PUPO faz, a princípio, o papel do engenheiro, este tem sua própria forma de modelar conhecimento. Forma essa que não compromete a capacidade lógica de representação de conhecimento nas ontologias resultantes.

Sentença (11): an island or isle is any piece of land that is completely surrounded by water.
Resultado (11):

1. (Island) or (Isle) SubClassOf: (completelySurroundedBy some Water) and (pieceOf some Land)

Discussão (11): Diferente de Völker, Hitzler e Cimiano (2007) o conceito *Piece* não foi detectado pelo PUPO como um conceito (Classe OWL), pois o mesmo juntou o termo com a preposição *Of* e os transformou em uma propriedade (*pieceOf*), dessa forma tornando assim o axioma mais fiel a semântica da linguagem natural. Por fim criou a propriedade *completelySurroundedBy* e a relacionou com *Water*. A propriedade *completelySurroundedBy* juntou um advérbio *completely* (completamente) com o verbo *Surrounded* (Cercado, verbo cercar no particípio) e a preposição *By*. Essa junção de termos embarcou três níveis semânticos (advérbio, verbo, preposição) formando uma propriedade bastante expressiva.

Sentença (12): days of the week are Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.
Resultado (12):

1. daysOf some (Week) SubClassOf: Friday or Thursday or Wednesday or Tuesday or Monday or Saturday or Sunday

Discussão (12): Essa sentença introduz, de maneira bem simples, a possibilidade que o PUPO tem de criar GCI. A expressão *daysOf some (Week)* (dias de alguma semana) representa uma objeto que ainda não é conhecido, mas contém a propriedade *daysOf* e seu alcance é a classe *Week*. Ainda nesse exemplo apresenta uma grande união dos indivíduos *{Friday} or {Thursday} or {Wednesday} or {Tuesday} or {Monday} or {Saturday} or {Sunday}*, fornecendo a relação, previamente citada, entre indivíduos e classes.

Esse mesmo processo foi realizado nas 100 (cem) sentenças (Para listagem das sentenças utilizadas vide Apêndice H) para avaliação do PUPO extraídas a partir de DS1. Os resultados são demonstrados através da Figura 64.

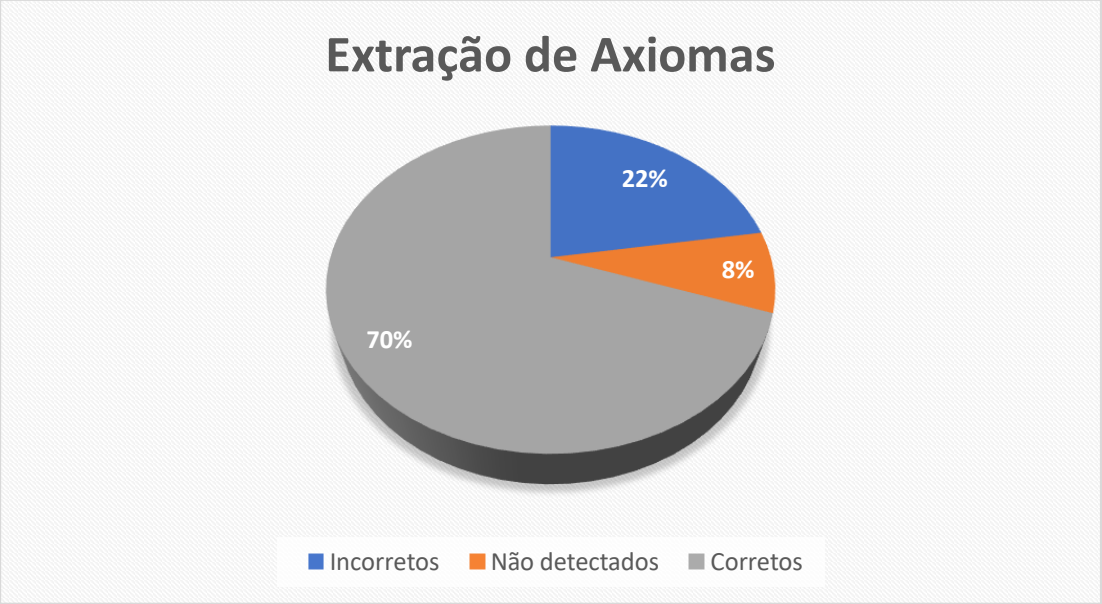

Figura 64 – Percentual de componentes OWL DL extraídos utilizando os algoritmos propostos para a composição do PUPO

As 100 (cem) sentenças de teste produzem 213 axiomas, considerando axiomatização das frases como um todo e axiomas de subsunção/definição extraídos a partir da sentença original. Isso significa que o PUPO idealmente deveria extrair 213 axiomas através de seus algoritmos. Porém, notamos que o PUPO classifica de forma incorreta alguns axiomas, e outros deixa de identificar. Esses 213 axiomas foram avaliados de forma manual como corretos, incorretos e não detectados.

Segundo a Figura 64, 70% (149 axiomas) foram extraídos de forma correta. Do restante dos axiomas, 22% dos axiomas (47 axiomas) foram extraídos de forma errada e 8% (17 axiomas) não foram detectados pela abordagem do PUPO. Os axiomas referentes a cada percentagem pode ser visto na Tabela 22.

	Taxa	nº de axiomas
Total	100%	213
Acerto	70%	149
Erros	22%	47
Não detectados	8%	17

Tabela 22 – Taxa de erros e acertos para avaliação de corretude

O modelo de aprendizado de máquina pode ser melhorado no quesito de utilização de outros parâmetros que melhor se adaptem as sentenças em linguagem natural. Outra abordagem poderia se a utilização de outros algoritmos de AM inclusive Aprendizado Profundo.

O teste de hipótese foi aplicado e os resultados, a seguir obtido.

Teste de hipótese:

Limiar $\rightarrow$ 55%. O limiar consiste na percentagem que se deseja alcançar como resultado, em nosso caso, pretendemos alcançar um limiar de no mínimo 55% de acertos.

Nível de significância $\rightarrow \alpha = 0,05$ (5%). O nível de significância do teste é a probabilidade da hipótese nula ser rejeitada, quando esta é verdadeira, sendo assim a probabilidade de se cometer um erro.

Sustentação:

Com *p-valor* obtido de 2.2e-16, menor que o nível de significância, a hipótese nula $H_{0,1}$ é rejeitada. Ou seja, estatisticamente o PUPO constrói corretamente ontologias em OWL DL em mais da metade dos casos. Com o intervalo de confiança de 95% confirmamos que a taxa de sucesso está entre 64% e 76%.

6.2.2 Construção de Ontologias: Raciocínio de subsunção/definição - Hipótese 1(b)

Nessa seção são avaliados os axiomas inferidos pelo PUPO em tempo de modelagem de conhecimento. Os axiomas inferidos através de Aprendizado de Máquina são avaliados na Seção 6.2.4. As mesmas 100 sentenças (Vide Apêndice H) utilizadas para o teste de

corretude foram utilizadas para este teste de raciocínio. Apenas os axiomas inferidos são apresentados a seguir.

Sentença (1): shark is a fish and an aquatic animal. It has only cartilaginous skeleton and is not a marine mammal.
Resultado (1):

1. CartilaginousSkeleton EquivalentTo: Cartilaginous and Skeleton
2. AquaticAnimal EquivalentTo: Animal and Aquatic
3. MarineMammal EquivalentTo: Mammal and Marine

Discussão (1): Os axiomas apresentados e extraídos em forma de inferências são axiomas de definição de classes, listados acima. Esse tipo de axioma define **tudo** que uma classe é por definição. Em De Azevedo et al. (2014) temos o axioma:

```
CartilaginousSkeleton SubClassOf: Skeleton
```

Vemos que o axioma acima foi corretamente classificado, mas o conceito *Cartilaginous* (cartilaginoso) não foi utilizado na definição desse axioma.

Por outro lado o PUPO aproveita-se ainda do conceito *Cartilaginous* definindo o axioma *CartilaginousSkeleton EquivalentTo: Cartilaginous and Skeleton*. Assim tudo que for *Cartilaginou* e ao mesmo tempo *Skeleton* será classificado como um *CartilaginousSkeleton*, tornando as ontologias resultantes mais expressivas.

Da mesma forma foi feito para os conceitos *AquaticAnimal* e *MarineMammal*.

Sentença (2): juvenile is a young fish or animal that has not reached sexual maturity.
Resultado (2):

1. YoungAnimal EquivalentTo: Animal and Young
2. YoungFish EquivalentTo: Fish and Young

Discussão (2): De Azevedo et al. (2014) captura os substantivos *Animal* e *Fish* para serem superclasses de *YoungAnimal* e *YoungFish*, respectivamente. Essa interpretação, nesse caso, é correta, porém ainda pode ser melhorada semanticamente. Völker, Hitzler e Cimiano (2007) defende a ideia de que cada termo deve compor um conceito distindo do outro. O PUPO interpreta como De Azevedo et al. (2014) e cria as classes *YoungAnimal* e *YoungFish*, porém as define de forma mais completa, semanticamente, com os axiomas de equivalência. Nesse caso, os dois axiomas de definição inferidos definem as classes *YoungFish* e *YoungAnimal*.

Sentença (3): biosphere is the portion of Earth and its atmosphere that can support life.
Resultado (3):

1. EarthAtmosphere EquivalentTo: Atmosphere and Earth

Discussão (3): Esse axioma inferido representa a classe atmosfera terrestre (*EarthAtmosphere*). Essa classe é definida como *Earth* (planeta Terra) e também atmosfera (*Atmosphere*).

Sentença (4): a minister or a secretary is a politician who holds significant public office in a national or regional government.
Resultado (4):

1. RegionalGovernment EquivalentTo: Government and Regional

Discussão (4): O PUPO nesse passo, deixou de realizar a criação de um axioma de definição referente a *NationalGovernment*, mas criou de forma correta a definição de *RegionalGovernment* indicando como a conjunção entre *Government* e *regional*. O PUPO em seu processo de raciocínio extrai esses axiomas de definição para termos compostos (geralmente cadeias de substantivos e/ou adjetivos combinados para representar uma classe) apenas. Para as definições diretas da sentença original são utilizados axiomas de subclasses, isso permite que mais termos sejam adicionados posteriormente sem grandes impactos no raciocínio.

Essa mesma análise foi realizada para as 100 (cem) sentenças de comprovação das hipóteses e somente as definições de classes inferidas, como vistas nos exemplos acima, foram computadas.

Na Figura 65 é exibido o percentual de componentes corretamente inferidos pelos PUPO. Assim, notamos que 69% (95 axiomas) das definições/subsunções de classe foram inferidas de forma correta. Do total de 138 inferências, 22% (30 axiomas) foram inferidas de forma equivocada e 9% (13 axiomas) deixaram de ser inferidas.

Na Tabela 23 exibimos o percentual e o número de acertos, erros e axiomas não detectados.

Para inferência de definição de classes, foi utilizado o seguinte método:

Todos os conceitos compostos por combinações de substantivos e adjetivos são transformados em um axioma de definição composto pelos conceitos individuais. Como exemplo consideremos a situação que apresentamos na Figura 66.

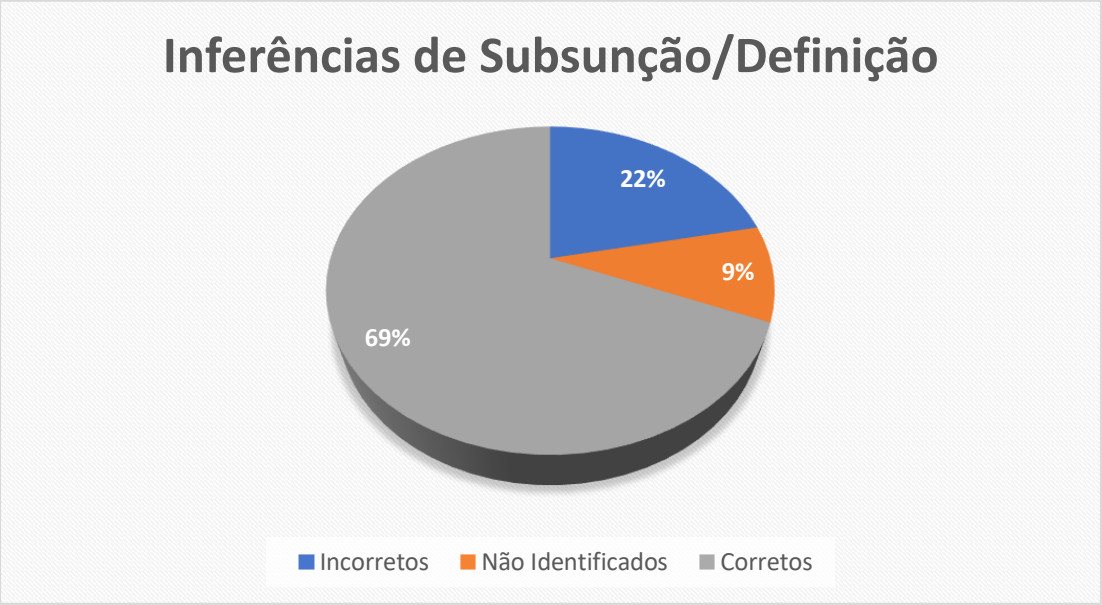

Figura 65 – Percentual de componentes OWL DL inferidos através do processo de raciocínio de definição/subsunção

	Taxa	nº de axiomas
Total	100%	138
Acerto	69%	95
Erros	22%	30
Não detectados	9%	13

Tabela 23 – Taxa de erros e acertos para avaliação de inferências

Notamos nessa situação que o conceito GrandeBoloVerdeMaçã foi definido como sendo algo que também é uma maçã, de forma equivocada. Esse tipo de erro pode ser mitigado investigando a classe gramatical e formulada alguma eurística para composição do axioma de definição. Dessa forma a taxa de acerto de inferências de definição seria melhorada significativamente.

Teste de hipótese:

Limiar $\rightarrow$ (55%)

Nível de significância $\rightarrow \alpha = 0,05$ (5%)

```
Frase: A criança gosta de comer um grande bolo verde de maçã.
Axioma Principal:
Criança SubClasseDe gostaDeComer algum ( GrandeBoloVerdeMaçã)

Axioma Inferido:
GrandeBoloVerdeMaçã EquivalenteA Grande e Bolo e Verde e Maçã
```

Figura 66 – Exemplo de inferência de definição

Sustentação:

O *p-valor* obtido foi de 9.369e-13, sendo menor que o nível de significância, então rejeitamos a hipótese nula $H_{0,2}$ e concluímos que o PUPO realiza raciocínio de subsunção/definição em mais de 55% dos casos. Com intervalo de confiança de 95% a taxa de sucesso fica entre 61% e 76%.

6.2.3 Construção/Refinamento de ontologias: Verificação de Inconsistências - Hipótese 1(c)

Para exemplificar a resolução de inconsistência usaremos as sentenças, retiradas de De Azevedo et al. (2014):

Sentenças:

1. cow eats grass
2. grass is a kind of plant
3. herbivore eats only plant
4. A lion eats meat
5. meat is a kind of food that is not a plant
6. A giraffe eats only plant
7. carnivores are predators that eat only meat
8. herbivore eats leaves
9. herbivore eats meat

Consideremos que todas essas sentenças acima citadas (1-9) já tenham sido axiomatizadas pelo PUPO.

Resultado: Nesse momento a última sentença *herbivore eats meat* tornou a ontologia inconsistente, assim o PUPO afim de resolver os fatos contraditórios pergunta ao usuário:

- PUPO: Is it right to say: In iri.of.ontology Herbivore SubClassOf eats some Meat ?

O usuário ao responder **não**, removerá o axioma *Herbivore SubClassOf eats some Meat* da ontologia, tornando-a consistente novamente. Outro exemplo, dessa vez através de importação de uma ontologia, de inconsistência no refinamento pode ser visto na Seção 4.2.2, ontologia *Pizza*.

Discussão: O motor de raciocínio `Pellet` utilizado pelo PUPO alcança expressividade $\mathcal{SROIQ}(\mathcal{D})$. Dentro dessa expressividade o *Pellet* detecta quaisquer tipos de inconsistências. Como visto na Seção 3.1 todo processo de raciocínio em DL é realizada pelo *Pellet*. Como a expressividade alcançada pelo *Pellet* é maior que a expressividade alcançada pelo PUPO (expressividade $\mathcal{ALCHOQ}$) então o PUPO consegue através do *Pellet* identificar todas as inconsistências que poderiam ser detectadas através do uso apenas do *Pellet*. Concluímos, dessa maneira, que o PUPO identifica as inconsistências em 100% dos casos, descartando a Hipótese nula $H_{0,3}$ - *O PUPO não verifica inconsistências na(s) ontologia(s) em pelo menos 100% dos casos*.

Esse processo de detecção e tratamento de inconsistência além de realização de inferências possibilitam maior qualidade semântica das ontologias geradas ou refinadas pelo PUPO. Lembrando que inconsistência é um tipo de axioma corretamente escrito em linguagem de representação de conhecimento que logicamente deixa a ontologia inconsistente. Assim não é considerado um axioma incorreto.

6.2.4 Aprendizado de Máquina e Fontes Semânticas - Hipótese 1(d)

Nesta seção são apresentados os resultados para avaliação do componente de Aprendizado de Máquina da arquitetura do PUPO (Vide Capítulo 3) que comprovam a Hipótese 1(d). Primeiro, para a base DSS1 (Vide Seção 6.1.2.1), foi realizada uma busca com expressão *pow(BASE,I)* com I no intervalo [0,3] para o C e [-10,-1] para *gamma*. Ao executar *GridSearch*, maximizando a acurácia, encontramos um valor de *gamma* igual a 0.01 e um valor de C igual a 100, segundo Tabela 24.

Parâmetro	Intervalo	Expressão	valor
gamma	10^-1 à 10^{-10}	pow(BASE,I)	0,01
C	1 à 1000	pow(BASE,I)	100

Tabela 24 – GridSearch para o dataset DSS1

Gerando os modelos: Os modelos foram gerados através do Weka 3.8.2[3]. Para a base DSS1 utilizou-se *2gram* e *StringToWord2Vec* com *stemming* e remoção de *stopwords* para cada uma das *features*. A geração dos modelos ocorreu com a utilização de SVM, kernel RBF e contou com validação cruzada "*10-fold cross validation*" e seus valores são apresentados na Tabela 25.

[3] https://www.cs.waikato.ac.nz/ml/weka/

	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
	0,974	0,017	0,983	0,974	0,978	0
	0,983	0,026	0,974	0,983	0,979	1
Weighted Avg.	**0,979**	**0,021**	**0,979**	**0,979**	**0,979**	

Tabela 25 – Pontuação do modelo gerado (10-fold)

A primeira linha da Tabela 25 possui os resultados do treino para classificação de relações inválidas ou seja, classificadas como zero. A pontuação para falsos positivos (*FP Rate*) foi de 0,017, indicando baixo erro. A *F-measure* obtida foi de 0,978 indicando uma boa classificação do modelo.

A segunda linha possui os resultados do treino para classificação de relações válidas, classificadas como 1 (um). A taxa de falsos positivo (*FP Rate*) também foi baixa com valor de 0,026. O valor de *F-measure* obtido foi de 0,979. Isso indica uma boa classificação do modelo para classificar relações válidas. Para os valores de *F-measure* quanto mais próximos de 1 (um) melhor é o modelo.

A última linha, *Weighted Avg*), contém uma média dos valores de classificação. A média obtida para valor de *F-measure* do modelo como um todo foi de 0,979. Esse valor representa que o modelo é capaz de classificar bem as relações conforme seu treino.

Além dos modelos gerados foi utilizado o método de extração de relação do *Stanford CoreNLP*.

Extração de axiomas: A base de validação consistiu em 621 sentenças da base de teste[4] e suas respectivas relações[5].

Existem três métodos de extração utilizados pelo PUPO:

1. ***Semantic* (Semântico)** - A extração dar-se através de padrões (Vide Seção 3.1.3.2, Capítulo 3). Esses padrões são utilizados para extrair relações e essas relações avaliadas através de Aprendizado de Máquina.

2. ***Sintatic* (Sintático)** - A extração dar-se através da distribuição de sujeitos e objetos através dos verbos de uma sentença (Vide Seção 3.1.3.1, Capítulo 3). Cada distribuição torna-se uma relação que será avaliada por Aprendizado de Máquina.

3. **Stanford** - A extração de relações dar-se através do algoritmo nativo do *CoreNLP*.

Classificamos a abordagem de extração do PUPO como híbrida por utilizarmos esses três métodos descritos, *Semantic, Sintatic* e *Stanford*.

Todos os axiomas extraídos através dos métodos *Semantic* e *Sintatic*, e avaliados por Aprendizado de Máquina são confrontados com a SUMO, visando eliminar axiomas extraídos de forma equivocada.

[4] Acessível em https://bit.ly/2SQY0xn
[5] Acessível em https://bit.ly/2SNWigy

Apresentamos na Tabela 26 os valores de *True Positives* (TP), *False Positives* (FP) e *False Negatives* (FN) além dos valores de *Precision*, *Recall* e *f-measure*. A linha *FULL* (COMPLETO) se refere a pontuação geral do método de extração de axiomas.

	TP	FP	FN	Precision	Recall	f-measure
COMPLETO	95	0	12	1	0,9365	0,9649
Semantic	5	0	0	1	1	1
Sintatic	51	0	12	1	0,8095	0,8947
Stanford	39	0	0	1	1	1

Tabela 26 – Extração de Axiomas e checagem na SUMO sem utilização do WordNet

A prova dos axiomas foi realizada sem (mapeamento entre termos sinônimos) a utilização do `WordNet` para mapeamento entre os termos dos axiomas e os termos da `SUMO`. Em termos gerais não houve classificação de FP. Ressalta-se que foram considerados FP os axiomas nos quais a prova resultava em resultado FALSE através da `SUMO`. Por exemplo o axioma "Canine SubClassOf not (Animal)" tem resultado FALSE após a prova. Com o valor de FP igual a 0 (zero) o valor de *Precision* se torna 1 (um). Dentre os axiomas extraídos 12 (doze) deles são verdadeiros, mas foram classificados como falso, gerando assim o valor de FN igual a 12 (doze). O método semântico conseguiu extrair 5 (cinco) relações válidas, sem erros de FP ou FN. O método sintático conseguiu extrair 51 (cinquenta e um) axiomas válidos e 12 (doze) FN. Enquanto o método utilizando extração de relações do `CoreNLP` resultou em 39 (trinta e nove) axiomas válidos e nenhuma classificação para FN. Sendo assim a junção dessas três técnicas possibilita uma melhoria do processo de extração de relações. Essa abordagem híbrida é utilizada em nosso *chatterbot*.

Com o objetivo de elevar a capacidade do motor de raciocínio em termos de vocabulário apresentamos na Tabela 27 os resultados para a utilização do `WordNet` como forma de mapeamento entre termos dos axiomas e termos da `SUMO` através de sinônimos. No geral (primeira linha) nota-se o crescimento dos axiomas que puderam ser provados com a `SUMO` e o mapeamento com o `WordNet`. Enquanto o método anterior foi capaz de extrair 95 axiomas como TP, esse foi capaz de extrair 240. Essa diferença deu-se exclusivamente pela capacidade do raciocinador reconhecer os sinônimos de um termo presente no axioma a se provar, e então fazê-lo. Novamente o método semântico extraiu 5 axiomas TP, devido ao baixo vocabulário dos termos das relações. Por exemplo na sentença *"the perpetrator attempted to kill the officer"* o axioma gerado será *"Perpetrator SubClassOf attemptedToKill some Officer"*. A propriedade *attemptedToKill* não pode ser verificada na `SUMO` pois apresenta uma semântica bem específica não característica de uma Ontologia de topo, como a `SUMO`.

Na Tabela 27, o método sintático continuou com uma alta classificação de TP e o valor de Recall subiu em relação ao anterior, ou seja sua taxa de acerto cresceu com a utilização

	TP	FP	FN	Precision	Recall	f-measure
COMPLETO	240	0	30	1	0,9459	0,9706
Semantic	5	0	0	1	1	1
Sintatic	155	0	30	1	0,8378	0,9118
Stanford	80	0	0	1	1	1

Tabela 27 – Extração de Axiomas e checagem na SUMO com utilização do WordNet

do `WordNet`, extraindo assim 155 axiomas TP. A extração com o `CoreNLP` permaneceu com melhor resultado *Precision* e *Recall* iguais a 1 (um). Além disso o `WordNet` possibilitou que 41 axiomas a mais fossem extraídos. Esse comparativo pode ser visto na Figura 67.

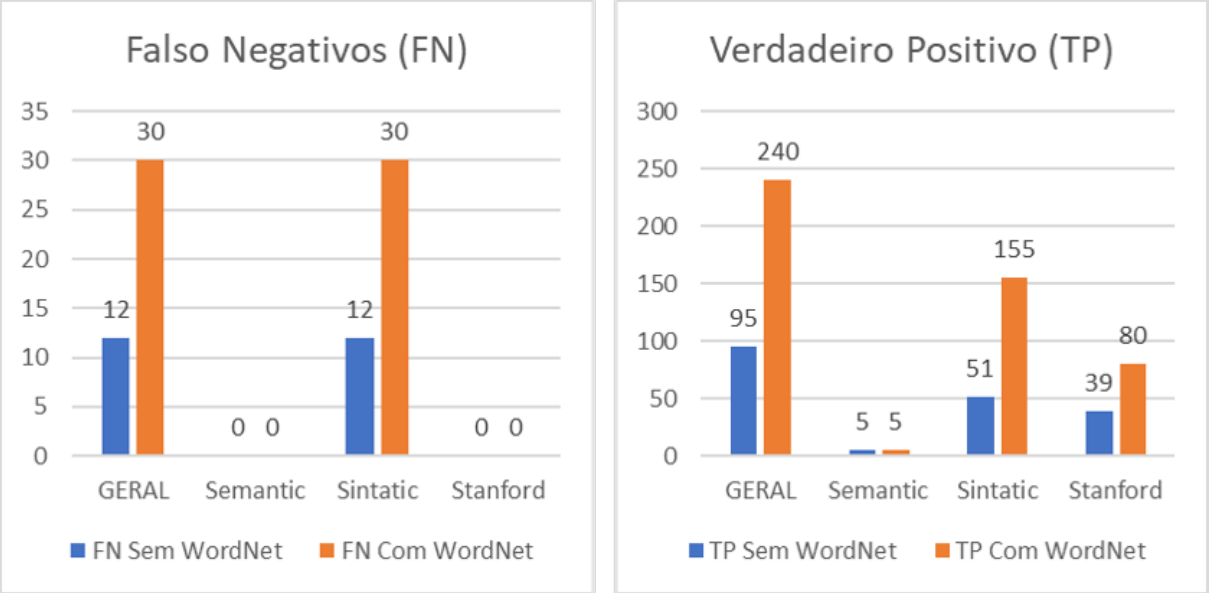

Figura 67 – Comparativo entre extração sem utilização do WordNet vs com utilização do WordNet

É importante salientar que cada método conseguiu extrair pelo menos um axioma diferente dos outros métodos, validando assim a ideia de utilizar os 3 (três) métodos para extração e axiomatização de relações. Em termos gerais, os 145 axiomas extraídos a mais com a utilização do `WordNet` como TP e os 18 como FN indicam o ganho de capacidade de prova do raciocinador. Esses axiomas não puderam ser provados sem o mapeamento com a `SUMO` e `WordNet`.

Avaliação das Ontologias geradas: Os axiomas convertidos a partir de linguagem natural e extraídos por aprendizagem de máquina foram convertidos em Ontologias.

Na Tabela 28 exibimos as métricas das Ontologias geradas (dados extraídos do Protégè). A coluna *Class* representa o número de classes extraídas. A aprendizagem de axioma sem utilização do `WordNet` gerou um ganho de 4 (quatro) classes em relação à conversão direta de linguagem natural para OWL. Por sua vez, o método de aprendizado com utilização do `WordNet` gerou um ganho de 9 (nove) classes em relação ao método anterior, e 13

	Class	Object Property	Individual	Hierarchy	GCI	Expressivity
SUMO	1871	984	58	460	160	$\mathcal{ALCHOQ}$
SUMO + Aprendizado de Máquina	1875 ↑	984	59 ↑	526 ↑	164 ↑	$\mathcal{ALCHOQ}$
SUMO + Aprendizado de Máquina + `WordNet`	1884 ↑	984	60 ↑	644 ↑	169 ↑	$\mathcal{ALCHOQ}$

Tabela 28 – Métricas das Ontologias geradas

(treze) classes em relação a não utilização de aprendizagem de axioma por aprendizagem de máquina. Percebemos assim que a utilização de fontes semânticas como o *WordNet* e a SUMO juntamente com Aprendizado de Máquina favorece o processo de axiomatização de linguagem natural.

Quanto a GCI (*General Concept Inclusion*)(BRANDT, 2005) o método sem utilização do `WordNet` mostra um ganho de 4 (quatro) axiomas, e 9 (nove) axiomas com utilização.

A captura de GCI permite extrair axiomas conforme Figura 68.

```
Frase:
um animal que come carne e anda sobre a terra é um carnívoro terrestre.

Axiomas:
Animal e (come algum Carne) e (andaSobre algum Terra) SubClasseDe:
CarnivoroTerrestre

CarnivoroTerrestre EquivalenteA: Carnivoro e Terrestre
```

Figura 68 – Exemplo de GCI

A parte a esquerda do axioma não representa uma classe, mas sim características que uma classe deve possuir (Animal e (come alum Carne) e (andaSobre algum Terra)) para ser classificado como CarnivoroTerrestre.

Se tratando da hierarquia entre as classes (coluna *Hierarchy*) houve ganho de 66 (sessenta e seis) hierarquias com aprendizado e sem `WordNet`. Considerando aprendizado com `WordNet` houve um ganho de 184 hierarquias.

Na Figura 69 apresentamos as quantidades dos componentes OWL que foram extraídos pelos métodos e checados semanticamente através da SUMO. O componente que apresentou melhor ganho foi *Hierarchy* que representa hierarquias entre classes. Fazendo, assim, com que a Ontologia final gerada tenha uma representação, classificação, mais fiel à informação modelada.

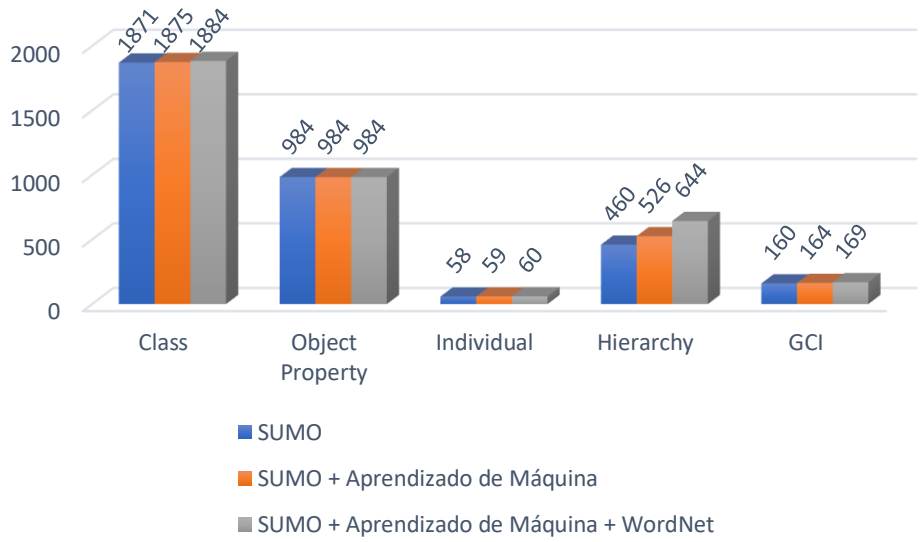

Figura 69 – Componentes OWL extraídos e checagem semântica com a SUMO vs Técnica (Aprendizado de Máquina e/ou Mapeamento com WordNet)

O único componente, avaliado, que não mostrou ganho foi *Object Property*, esse fato pode ter sido derivado das relações serem bastante expressivas e não puderam ser encontradas na *SUMO*. Novamente, afirmamos que as propriedades extraídas pelo PUPO são muitas vezes complexas. Essa complexidade e especificidade não são características de uma ontologia de topo como a SUMO.

Como um todo a utilização de fontes semânticas como a ontologia de topo *SUMO* e o *WordNet* juntamente com Aprendizado de Máquina possibilitaram ganhos no processo de extração e Aprendizado de Ontologia. Caso contrário a Tabela 69 apresentaria os mesmos valores independente da técnica aplicada. Dessa forma consideramos a rejeição da Hipótese nula $H_{0,4}$ - *A utilização de aprendizado de máquina, WordNet e SUMO não fornecem melhorias no processo de axiomatização realizado pelo PUPO*, pois vemos que há de fato melhorias na identificação de classes, indivíduos, hierarquias e GCI comprovando $H_{a,4}$.

Embora a classificação realizada por AM tenha uma alta percentagem, o processo de transformar linguagem natural para OWL ainda é um gargalo observado. Este trabalho utiliza de técnicas de AM e linguística para extrair informações que podem ser modeladas em ontologias, porém transformar uma sentença de linguagem natural para OWL, como o PUPO faz, é uma tarefa que gera muitas vezes axiomas incorretos semanticamente. A pesar de todos esses problemas observados, o PUPO consegue, como mostram os resul-

tados, avançar um pouco mais no processo semi-automático de produção de ontologias expressivas.

6.3 CONCLUSÕES DO CAPÍTULO

Através dos experimentos foi possível verificar que o PUPO é capaz de:

1. Criar ontologias em expressividade máxima $\mathcal{ALCOHQ}$ durante diálogos com os usuários
2. Realizar raciocínio de subsunção/definição durante o diálogo com os usuários
3. Verificar inconsistências na(s) ontologia(s) enquanto dialoga com os usuários
4. Ter maior capacidade de axiomatização na linguagem natural em OWL DL utilizando-se de aprendizado de máquina, `WordNet` e a ontologia de topo SUMO.

Observados os fatos acima citados retornamos a questão de pesquisa principal deste trabalho:

- Um *chatterbot* é capaz de modelar conhecimento, realizar raciocínio e resolver inconsistências advindos da interação com o usuário?

A comprovação das Hipóteses 1(a), 1(b), 1(c) e 1(d) através dos experimentos apoiam a veracidade da Hipótese levantada para este trabalho (H_1). Essa hipótese se comprova verdadeira e assim pode-se concluir que o PUPO é capaz de modelar conhecimento e representá-lo em OWL 2 DL, e realizar raciocínio de subsunção/definição e inconsistência através de diálogos.

6.3.1 Limitações

A abordagem semântica apresenta uma precisão proporcional ao vocabulário da Ontologia de que serve como base para a checagem dos axiomas. Além disso sua cobertura varia de acordo a quantidade de padrões existentes no algoritmo. Quanto maior o número de padrões e o vocabulário da Ontologia para checagem maior será o número de axiomas processados pela abordagem. Quanto ao método sintático a distribuição entre elementos nominais do sujeito e elementos nominais do objeto através do verbo possui um custo muito alto que envolve a combinação desses elementos.

Além disso os axiomas extraídos dessa abordagem trazem menor semântica pois relações formadas por composição dos verbos não é possível. Mantendo o equilíbrio entre as abordagens propostas o `CoreNLP` extrai ótimas relações, porém deixa de extrair as relações que os métodos propostos são capazes. A combinação desses métodos proporciona um ganho juntamente com a utilização da `SUMO` e seus termos mapeados com o `WordNet`, porém o método se torna dependente dos conjuntos de sinônimos presentes no `WordNet`.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo apresentamos as conclusões finais dessa dissertação e as propostas de trabalhos futuros.

7.1 CONCLUSÕES

Este trabalho teve como objetivo implementar um *chatterbot* para criação e refinamento de Ontologias em lógica de descrições. Assim podemos afirmar que foi produzido um mecanismo de diálogo e modelagem de conhecimento mediante várias técnicas como Aprendizado de Máquina, o uso do `WordNet` e uso da ontologia de topo `SUMO`. No que diz respeito a utilização de Aprendizado de Máquina, o PUPO foi capaz de estender sua capacidade de aprendizado em relação a outros sistemas de diálogo. Se tratando do `WordNet`, sua utilização foi de fundamental importância para que o PUPO viesse a expandir seu vocabulário por meio de palavras sinônimas. Por fim, a checagem de conhecimento na SUMO, permitiu ao PUPO modelar conhecimento de uma forma semanticamente correta. Uma vez que todos os axiomas aprendidos por Aprendizado de Máquina são avaliados, checados semanticamente, segundo a SUMO. Posto isso, retomamos nossos objetivos específicos, como segue.

Demonstrar que a ferramenta é capaz de:

1 - Construir ontologias em OWL 2 DL com expressividade máxima $\mathcal{ALCOHQ}$ a partir de textos - O PUPO demonstrou no decorrer dos teste dos testes (Vide Seção 6.2) ser capaz de modelar conhecimento em ontologias de forma correta na maioria dos casos. As ontologias resultantes do diálogo com o PUPO alcançam expressividade máxima $\mathcal{ALCOHQ}$ garantindo bases de conhecimento consistentes e mais expressivas que diversos trabalhos da literatura.

2 - Realizar raciocínio de subsunção/definição deduzindo que classes são subclasses de outras, a partir de suas respectivas descrições - O PUPO é capaz de criar definições e verificar que classes são subclasses de outras mediante suas descrições (Vide Seção 6.2.2) enriquecendo ainda mais as ontologias construídas. Essas definições de classes percebidas automaticamente pelo PUPO facilita o processo de modelagem das ontologias em OWL, tornando assim o PUPO uma alternativa inteligente para modelagem de conhecimento. Outro processo que envolve raciocínio e também um diferencial do PUPO é o processo de produção de axiomas de fecho (Vide Seção 4.2.2.1) provido pelo sistema. Esse se mostra como um valioso recurso para melhor representar conhecimento permitindo maiores restrições e fidelidade das ontologias em relação ao mundo real.

3 - Detectar e verificar inconsistências em tempo de desenvolvimento nas ontologias construídas - Além das contribuições na área de extração de informação o *chatterbot*

PUPO é capaz de resolver as inconsistências encontradas nas diversas bases de conhecimento modeladas em tempo de execução (Vide Seção 6.2.3). Esse método se destaca em relação aos outros sistemas de diálogo por permitir que, ao longo do diálogo, o próprio usuário consiga remover o conhecimento inconsistente presente nas ontologias modeladas. Assim diversos fatos contraditórios logicamente podem ser detectados e tratados pelo PUPO enquanto dialoga com o usuário. Esse feito indica que o PUPO é uma alternativa viável no que diz respeito a modelagem consistente de conhecimento em ontologias expressivas.

4 - Realizar extração de axiomas, automaticamente, utilizando Aprendizagem de Máquina e técnicas linguísticas - A extração de axiomas utilizando técnicas linguísticas (padrões) apoiados por algoritmo de Aprendizado de Máquina (SVM) possibilita ao PUPO extração de fatos das sentenças de forma mais eficaz. A utilização de fontes semânticas como o WordNet e a SUMO possibilitaram melhor qualidade e prevenção de erros nos axiomas extraídos pelos PUPO. Concluímos assim que a utilização de fontes de dados semânticas em conjunto a AM tornam o PUPO mais eficaz no processo de extração de axiomas de linguagem natural para OWL.

5 - Obter um processo de construção de Ontologias – a partir de dados advindos da interação em linguagem natural – buscando prover eficiência para a intervenção humana em sua construção - O PUPO se mostrou capaz de modelar conhecimento em forma de ontologias, provendo assim um meio de intervenção humana eficiente no processo de modelagem valendo-se de diálogos. Constatamos dessa forma que o PUPO como um todo pode ser uma ferramenta de autoria inteligente capaz de ajudar *experts* ou leigos durante o processo de modelagem de conhecimento em ontologias escritas em OWL DL.

6 - Avançar no estado da arte construindo um Chatterbots capaz de aprender através de diálogos - Deduzimos que o PUPO é um *chatterbot* capaz de aprender e dialogar com o usuário se destacando da maioria dos *bots*. Sendo assim tornamos o PUPO mais próximo do que Turing considerava como uma "máquina inteligente"(TURING, 1950). Por conseguinte podemos considerar que o desenvolvimento do PUPO trouxe as seguintes contribuições:

1. **Extração de relação:** a abordagem desenvolvida para extração de relação dispondo de distribuição verbal e padrões semânticos apoiados por classificação utilizando Aprendizagem de Máquina gera uma contribuição na área de extração de relação. Essa abordagem juntamente com as técnicas existentes na literatura permitem uma maior cobertura na extração das possíveis relações.

2. **Axiomatização de Linguagem Natural:** O processo de tradução de linguagem natural (LN) para OWL se mostra promissora como contribuição na área de representação de conhecimento. Com esse processo, atualmente, é possível alcançar

expressividade $\mathcal{ALCOHQ}$, se mostrando assim uma alternativa viável para criação de bases de conhecimento com tradução automática de LN para OWL.

3. **Resolução de inconsistência em tempo de desenvolvimento:** a detecção de inconsistências operando sob um raciocinador em tempo de execução permite ao sistema questionar o usuário sobre a veracidade dos fatos entre os axiomas problemáticos na base de conhecimento. Essa estratégia permite a construção e refinamento de bases de conhecimento consistentes que podem ser reutilizadas em diversos sistemas.

4. ***SUMO Service Query* em Manchester Syntax:** o *SUMO Service Query* é composto por um serviço web que permite ao usuário realizar *queries* na `SUMO` utilizando *Manchester Syntax*. Além disso, suas consultas contam com o *Pellet* como mecanismo de raciocínio retornando assim respostas consistentes e que poderiam não estar explicitamente escritas na `SUMO`, mas que são dedutíveis. Esse serviço pode ser acessado por qualquer aplicação, fornecendo uma interface desacoplada para desenvolvedores.

5. ***Chatterbot* para criação e refinamento de Ontologias a partir de Linguagem Natural:** como principal contribuição o *chatterbot* desenvolvido fornece uma interface com o usuário durante diálogos. Tanto o engenheiro de conhecimento quanto o especialista de domínio são capazes de modelar conhecimento por intermédio de linguagem natural, checar inferências e inconsistências em tempo de execução, além de ter sua nova base de conhecimento apoiada na Ontologia de topo `SUMO`. Ainda a utilização do `WordNet` como dicionário semântico e utilização de seus termos sinônimos para validação dos axiomas permite expansão do vocabulário durante a modelagem de conhecimento. Assim o sistema como um todo se mostra uma ferramenta de autoria capaz de gerar Ontologias expressivas, agilizando o processo de modelagem de conhecimento e economia de recursos – A agilização do processo de modelagem permite um menor tempo gasto durante o processo de desenvolvimento, desencadeando assim menor tempo com o engenheiro de conhecimento e especialista de domínio, os quais apresentam pouca disponibilidade – a partir de LN.

Por fim, podemos concluir que, todas as técnicas desenvolvidas e utilizadas junto com o mecanismo de diálogo fornecem valiosos recursos não apenas para o engenheiro de conhecimento e/ou especialista de domínio, mas todo usuário interessado em modelar conhecimento. O tempo e recursos gastos em projetos que necessitariam da disponibilidade desses atores como o engenheiro de conhecimento e especialista de domínio ao mesmo tempo podem ser minimizados, gerando assim economia em termos financeiros e de tempo.

Na próxima seção, como meio de apontar caminhos para futuras melhorias e avanços científicos provenientes do PUPO ou pelo menos alguma técnica aperfeiçoada por este

trabalho, se encontram os pontos de partida para melhoramento do PUPO observados durante o seu desenvolvimento.

7.2 TRABALHOS FUTUROS

Como objetivo de melhoramento do PUPO, os seguintes aspectos podem ser aperfeiçoados:

Mapeamento Semântico com ontologias de topo – Partindo da utilização da SUMO a abordagem deste trabalho pode ser melhorada com a utilização de alguma técnica bem definida de mapeamento semântico entre uma ou mais ontologias de topo para suportar e reaproveitar conhecimento semântico válido e pronto. Gerando assim um ganho ainda maior no tempo gasto para modelar um conhecimento já presente em uma ontologia de topo.

Explorar outras características do *WordNet* – O PUPO utiliza apenas um dos recursos que o *WordNet* fornece, o *synset* (conjunto de sinônimos). Outros recursos como hiperônimos e hipônimos (*hyperonymy* e *hyponymy*[1]) ou até mesmo antônimos podem ser utilizados para gerarem outras relações semânticas entre os textos advindos do diálogo com os usuário.

Aumento de expressividade – O processo de modelagem de conhecimento a partir de linguagem natural atual apresenta expressividade $\mathcal{ALCHOQ}$. Essa expressividade pode ser melhorada atingindo idealmente $\mathcal{SHOIN}(\mathcal{D})$. Estratégias para identificar *Data Properties* (propriedade de dados) e cardinalidade máxima e mínima são os passos mais evidentes para essa evolução. Esse processo necessita de um estudo para verificar quando um trecho em linguagem natural é uma cardinalidade, uma propriedade de dado ou uma propriedade de objeto (*Object Properties*) independente de como o usuário se expresse.

Melhorias no modelo de Aprendizado de Máquina – Considerando extração de relações, os padrões oferecidos neste trabalho podem ser incrementados, reunindo assim outros diversos padrões que podem ser encontrados no estado da arte, mas que devem ser avaliados para utilização no estado atual deste trabalho. Ainda a classificação de relações utilizando aprendizagem de máquina pode ser melhorado no quesito de formação de um melhor modelo, uma vez que encontrar "o melhor modelo"não foi o objetivo deste trabalho, mas sim um modelo satisfatório que prove o melhoramento na área de extração de relações com o método proposto. Isso implica diretamente na comparação entre diversos algoritmos (árvore de decisão, redes neurais, etc.) de aprendizagem de máquina que podem ser utilizadas e escolha do que apresente melhores resultados. Além disso abordagens como Aprendizado de Máquina Profundo (*Deep Learning*) é uma possibilidade bastante promissora devido sua eficiência no estado da arte.

Integração com a *Ontoclean* – A utilização da *Ontoclean* (GUARINO; WELTY, 2009) pelo PUPO fornecerá ao mesmo uma melhor padronização das relações taxonômicas. Essa

[1] https://wordnet.princeton.edu/

padronização melhora a qualidade geral da ontologia gerada e possibilita melhor poder de raciocínio sobre as bases de conhecimentos desenvolvidas através do PUPO. Fornecendo assim um precioso recurso para manter a qualidade do conhecimento modelado a partir de textos em linguagem natural.

Qualidade das ontologias geradas – O emprego e investigação de alguma técnica que avalie a qualidade do conhecimento gerado pode ser investigado. Essa investigação pode permitir melhores formas de descrever classes e relações (Propriedades) entre essas classes ou indivíduos que formam a ontologia.

Elaboração de padrões para LN – Devido a variância da linguagem natural a extração de informação/relações se torna uma tarefa relativamente complicada. Aconselhamos para contornar esse fato a inserção de padrões genéricos sobre o texto para extração de relações semânticas entre termos e expressões. Esse padrões em comparação à Aprendizado de Máquina, possibilitarão uma precisão semântica maior e melhor qualidade das ontologias resultantes.

REFERÊNCIAS

ABDELBASSET, B.; OKBA, K.; SOFIANE, M. Agent-based approach for building ontology from text. In: *2013 International Conference on Computer Medical Applications (ICCMA)*. IEEE, 2013. p. 1–6. ISBN 978-1-4673-5213-0. Disponível em: <http://ieeexplore.ieee.org/document/6522600/>.

AL-ZUBAIDE, H.; ISSA, A. A. OntBot: Ontology based ChatBot. In: *2011 4th International Symposium on Innovation in Information and Communication Technology, ISIICT'2011*. [S.l.: s.n.], 2011. ISBN 978-1-61284-675-0.

ALI, M.; HUSSAIN, J.; LEE, S.; PREPROCESSING, A. T. X-UDeKAM : An Intelligent Method for Acquiring Declarative Structured Knowledge using Chatterbot. *International Symposium on Perception, Action, and Cognitive Systems*, p. 65–66, 2016.

ALI, M.; LEE, S.; KANG, B. H. UDeKAM: A methodology for acquiring declarative structured knowledge from unstructured knowledge resources. In: *2016 International Conference on Machine Learning and Cybernetics (ICMLC)*. IEEE, 2016. v. 1, p. 177–182. ISBN 978-1-5090-0390-7. ISSN 21601348. Disponível em: <http://ieeexplore.ieee.org/document/7860897/>.

AYDOGAN, E.; AKCAYOL, M. A. A comprehensive survey for sentiment analysis tasks using machine learning techniques. *2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, p. 1–7, 2016. Disponível em: <http://ieeexplore.ieee.org/document/7571856/>.

BAADER, F.; CALVANESE, D.; MCGUINNESS, D. L.; NARDI, D.; PATEL-SCHNEIDER, P. F. (Ed.). *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press, 2003. ISBN 0-521-78176-0.

BAEZA-YATES, R. A.; RIBEIRO-NETO, B. *Modern Information Retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 020139829X.

BALTRUSAITIS, T.; AHUJA, C.; MORENCY, L. P. Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 8828, n. c, p. 1–20, 2018. ISSN 01628828.

BARBUR, G.; BLAGA, B.; GROZA, A. OntoRich - A support tool for semi-automatic ontology enrichment and evaluation. In: *Proceedings - 2011 IEEE 7th International Conference on Intelligent Computer Communication and Processing, ICCP 2011*. [S.l.: s.n.], 2011. p. 129–132. ISBN 9781457714788.

BAYOUDHI, L.; SASSI, N.; JAZIRI, W. How to Repair Inconsistency in OWL 2 DL Ontology Versions? *Data & Knowledge Engineering*, 2018. ISSN 0169-023X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169023X16303172>.

BOBILLO, F.; STRACCIA, U. Reasoning with the finitely many-valued Łukasiewicz fuzzy Description Logic SROIQ. *Information Sciences*, v. 181, n. 4, p. 758–778, 2011. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025510005190>.

BRADESKO, L.; STARC, J.; MLADENIC, D.; GROBELNIK, M.; WITBROCK, M. Curious cat conversational crowd based and context aware knowledge acquisition chat bot. *2016 IEEE 8th International Conference on Intelligent Systems, IS 2016 - Proceedings*, IEEE, p. 239–252, 2016. ISSN 1557-9700.

BRANDT, S. Reasoning in ELH w.r.t. General Concept Inclusion Axioms. p. 30, 2005. Disponível em: <http://lat.inf.tu-dresden.de/research/reports-abs.html#Brandt-LTCS-04-03>.

BUITELAAR, P.; BUITELAAR, P.; CIMIANO, P. *Ontology Learning and Population: Bridging the Gap Between Text and Knowledge - Volume 167 Frontiers in Artificial Intelligence and Applications*. Amsterdam, The Netherlands, The Netherlands: **IOS Press**, 2008.

CAHYANI, D. E.; MANURUNG, R.; MAHENDRA, R. Knowledge representation system for copula sentence in bahasa indonesia based on web ontology language (owl). In: *2015 INTERNATIONAL CONFERENCE ON ADVANCED COMPUTER SCIENCE AND INFORMATION SYSTEMS (ICACSIS)*. [S.l.: s.n.], 2015. p. 137–142.

CHEN, J.; DOSYN, D.; LYTVYN, V. Smart Data Integration by Goal Driven Ontology Learning. In: *Advances in Big Data*. [S.l.: s.n.], 2016. v. 5, p. 286–292. ISBN 9783319478982.

CHO, K.; MERRIENBOER, B. van; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014. p. 1724–1734. ISBN 0021-9258 (Print)\r0021-9258 (Linking). ISSN 00219258. Disponível em: <http://arxiv.org/abs/1406.1078>.

CIMIANO, P. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Secaucus, NJ, USA: **Springer-Verlag New York, Inc.**, 2006.

DADGAR, S. M. H.; ARAGHI, M. S.; FARAHANI, M. M. A novel text mining approach based on TF-IDF and Support Vector Machine for news classification. In: *2016 IEEE International Conference on Engineering and Technology (ICETECH)*. IEEE, 2016. p. 112–116. ISBN 978-1-4673-9916-6. Disponível em: <http://ieeexplore.ieee.org/document/7569223/>.

DAHAB, M. Y.; HASSAN, H. A.; RAFEA, A. TextOntoEx: Automatic ontology construction from natural English text. *Expert Systems with Applications*, v. 34, n. 2, p. 1474–1480, 2008. ISSN 09574174.

De Azevedo, R. R.; FREITAS, F.; ROCHA, R. G. C.; De Menezes, J. A. A.; De Oliveira Rodrigues, C. M.; De F.p. E Silva, G. An approach for learning and construction of expressive ontology from text in natural language. *Proceedings - 2014 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT 2014*, v. 1, p. 16–37, 2014.

DENAUX, R. *Intuitive ontology authoring using controlled natural language*. Tese (Doutorado) — University of Leeds School, 2013. Disponível em: <http://etheses.whiterose.ac.uk/id/eprint/4446>.

DENG, L. Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing*, v. 7, n. 3-4, p. 197–387, 2014. ISSN 1932-8346. Disponível em: <http://nowpublishers.com/articles/foundations-and-trends-in-signal-processing/SIG-039>.

DRYMONAS, E.; ZERVANOU, K.; PETRAKIS, E. G. M. Unsupervised Ontology Acquisition from Plain Texts: The OntoGain System. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [s.n.], 2010. v. 6177 LNCS, p. 277–287. ISBN 3642138802. Disponível em: <http://link.springer.com/10.1007/978-3-642-13881-2_29>.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification (2Nd Edition)*. New York, NY, USA: Wiley-Interscience, 2000. ISBN 0471056693.

El Idrissi Esserhrouchni, O.; FRIKH, B.; OUHBI, B. Building ontologies: A state of the art, and an application to finance domain. In: *International Conference on Next Generation Networks and Services, NGNS*. [S.l.: s.n.], 2014. p. 223–230. ISBN 9781479969371. ISSN 23276533.

FAN, Z.; HU, K.; LI, F.; RONG, Y.; LI, W.; LIN, H. Multi-Objective Evolutionary Algorithms Embedded with Machine Learning - A Survey. *Evolutionary Computation (CEC), 2016 IEEE*, p. 1262–1266, 2016.

FRANC, V.; HLAVAC, V. Multi-class support vector machine. In: *Object recognition supported by user interaction for service robots*. IEEE Comput. Soc, 2002. v. 2, n. October 2014, p. 236–239. ISBN 0-7695-1695-X. ISSN 10514651. Disponível em: <http://ieeexplore.ieee.org/document/1048282/>.

GHORPADE, T.; RAGHA, L. Featured based sentiment classification for hotel reviews using NLP and Bayesian classification. *Proceedings - 2012 International Conference on Communication, Information and Computing Technology, ICCICT 2012*, p. 1–5, 2012.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing. *INTERNATIONAL JOURNAL OF HUMAN-COMPUTER STUDIES*, v. 43, n. 5-6, p. 907–928, 1995.

GUARINO, N. Formal Ontology and Information Systems. *Proceedings of the first international conference*, n. June, p. 3–15, 1998. ISSN 10715819. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.1776&rep=rep1&type=pdf>.

GUARINO, N.; WELTY, C. An overview of ontoclean. In: _____. [S.l.: s.n.], 2009. p. 201–220.

GUNN, S. R. *Support Vector Machines for Classification and Regression*. [s.n.], 1998. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.7171>.

GUO, A.; YANG, T. Research and improvement of feature words weight based on TFIDF algorithm. In: *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*. IEEE, 2016. p. 415–419. ISBN 978-1-4673-9194-8. Disponível em: <http://ieeexplore.ieee.org/document/7560393/>.

HORROCKS, I.; PATEL-SCHNEIDER, P. F. KR and Reasoning on the Semantic Web: OWL. In: *Handbook of Semantic Web Technologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 365–398. Disponível em: <http://link.springer.com/10.1007/978-3-540-92913-0_9>.

JORDAN, M.; KLEINBERG, J.; SCHÖLKOPF, B. *Support Vector Machines*. New York, NY: Springer New York, 2008. (Information Science and Statistics). ISSN 1613-9011. ISBN 978-0-387-77241-7. Disponível em: <http://link.springer.com/10.1007/978-0-387-77242-4>.

KLEIN, D.; MANNING, C. D. Fast Exact Inference with a Factored Model for Natural Language Parsing. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, v. 15, p. 3–10, 2003.

Krol, M. Have we witnessed a real-life turing test? *Computer*, v. 32, n. 3, p. 27–30, March 1999. ISSN 0018-9162.

LEHMANN, J.; VOELKER, J. An Introduction to Ontology Learning. In: LEHMANN, J.; VOELKER, J. (Ed.). *Perspectives on Ontology Learning*. AKA / IOS Press, 2014. p. ix–xvi. Disponível em: <http://jens-lehmann.org/files/2014/pol_introduction.pdf>.

Liu, X.; Cao, L.; Dai, W. Overview on ontology mapping and approach. In: *2011 4th IEEE International Conference on Broadband Network and Multimedia Technology*. [S.l.: s.n.], 2011. p. 592–595.

MAYNARD, D.; BONTCHEVA, K.; AUGENSTEIN, I. Natural Language Processing for the Semantic Web. *Synthesis Lectures on the Semantic Web: Theory and Technology*, Morgan & Claypool, v. 15, n. 2, p. 1–194, dec 2017. ISSN 2160-4711. Disponível em: <http://www.morganclaypool.com/doi/10.2200/S00741ED1V01Y201611WBE015>.

MIKIC, F.; BURGUILLO, J.; RODRIGUEZ, D.; RODRIGUEZ, E.; LLAMAS, M. T-bot and q-bot: A couple of aiml-based bots for tutoring courses and evaluating students. In: *FRONTIERS IN EDUCATION CONFERENCE, 2008. FIE 2008. 38TH ANNUAL*. [S.l.: s.n.], 2008. p. S3A–7–S3A–12.

MIKIC, F. A.; BURGUILLO, J. C.; LLAMAS, M.; RODRIGUEZ, D. A.; RODRIGUEZ, E. CHARLIE: An AIML-based chatterbot which works as an interface among INES and humans. In: *2009 EAEEIE Annual Conference*. IEEE, 2009. p. 1–6. ISBN 978-1-4244-5385-6. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-72849137227&partnerID=tZOtx3y1>.

MIKIC, F. A.; BURGUILLO, J. C.; RODRIGUEZ, D. A.; RODRIGUEZ, E.; LLAMAS, M. T-Bot and Q-Bot: A couple of AIML-based bots for tutoring courses and evaluating students. In: *2008 38th Annual Frontiers in Education Conference*. IEEE, 2008. p. S3A–7–S3A–12. ISBN 978-1-4244-1969-2. ISSN 15394565. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4720469>.

MILLER, G. A. WordNet: A Lexical Database for English. *Commun. ACM*, ACM, New York, NY, USA, v. 38, n. 11, p. 39–41, 1995. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/219717.219748>.

NEVES, A.; BARROS, F.; HODGES, C. iaiml: a mechanism to treat intentionality in aiml chatterbots. In: *TOOLS WITH ARTIFICIAL INTELLIGENCE, 2006. ICTAI '06. 18TH IEEE INTERNATIONAL CONFERENCE ON*. [S.l.: s.n.], 2006. p. 225–231.

NOY, N. F.; FERGERSON, R. W.; MUSEN, M. A. The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. n. 1, p. 17–32, 2007.

PAN, L.; TANG, H.; ZHOU, L.; WANG, L.; ZHU, Q. An Identification Method of News Scientific Intelligence Based on TF-IDF. In: *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*. IEEE, 2015. p. 501–504. ISBN 978-1-4673-6593-2. Disponível em: <http://ieeexplore.ieee.org/document/7429665/>.

PEASE, A.; NILES, I.; LI, J. The suggested upper merged ontology: A large ontology for the semantic web and its applications. In: *In Working Notes of the AAAI-2002 Workshop on Ontologies and the Semantic Web*. [S.l.: s.n.], 2002. p. 2002.

PETRUCCI, G.; GHIDINI, C.; ROSPOCHER, M. Ontology Learning in the Deep. In: *Knowledge Engineering and Knowledge Management*. [S.l.: s.n.], 2016. v. 2, p. 480–495. ISBN 9783319490045.

RANI, M.; DHAR, A. K.; VYAS, O. P. Semi-automatic terminology ontology learning based on topic modeling. *Engineering Applications of Artificial Intelligence*, Elsevier Ltd, v. 63, p. 108–125, 2017. ISSN 09521976. Disponível em: <http://dx.doi.org/10.1016/j.engappai.2017.05.006>.

RESHMI, S.; BALAKRISHNAN, K. Implementation of an inquisitive chatbot for database supported knowledge bases. *Sadhana - Academy Proceedings in Engineering Sciences*, 2016. ISSN 09737677.

RILOFF, E. Automatically Generating Extraction Patterns from Untagged Text. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*. AAAI Press, 1996. (AAAI'96), p. 1044–1049. ISBN 0-262-51091-X. Disponível em: <http://dl.acm.org/citation.cfm?id=1864519.1864542>.

SANTOSO, J.; GUNAWAN, G.; GANI, H. V.; YUNIARNO, E. M.; HARIADI, M.; PURNOMO, M. H. Noun phrases extraction using shallow parsing with C4.5 decision tree algorithm for Indonesian Language ontology building. In: *2015 15th International Symposium on Communications and Information Technologies, ISCIT 2015*. IEEE, 2016. p. 149–152. ISBN 9781467368209. Disponível em: <http://ieeexplore.ieee.org/document/7458329/>.

SANTOSO, J.; YUNIARNO, E. M.; HARIADI, M. Large scale text classification using map reduce and naive bayes algorithm for domain specified ontology building. In: *Proceedings - 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2015*. IEEE, 2015. v. 1, p. 428–432. ISBN 9781479986460. Disponível em: <http://ieeexplore.ieee.org/document/7334739/>.

SERRA, I.; GIRARDI, R.; NOVAIS, P. PARNT: A statistic based approach to extract non-taxonomic relationships of ontologies from text. *Proceedings of the 2013 10th International Conference on Information Technology: New Generations, ITNG 2013*, p. 561–566, 2013.

SINGH, S.; KARWAYUN, R. A comparative study of inference engines. In: *ITNG2010 - 7th International Conference on Information Technology: New Generations*. [S.l.: s.n.], 2010. ISBN 9780769539843.

STUDER, R.; ANGELE, J.; SURE, Y.; STAAB, S.; ERDMANN, M.; WENKE, D. OntoEdit: Collaborative Ontology Development for the Semantic Web. p. 221–235, 2007.

TAYAL, M. A.; RAGHUWANSHI, M. M.; MALIK, L. Syntax Parsing: Implementation Using Grammar-Rules for English Language. In: *ELECTRONIC SYSTEMS, SIGNAL PROCESSING AND COMPUTING TECHNOLOGIES (ICESC), 2014 INTERNATIONAL CONFERENCE ON*. [S.l.: s.n.], 2014. p. 376–381.

TONIUC, D.; GROZA, A. Climebot: An argumentative agent for climate change. In: *Proceedings - 2017 IEEE 13th International Conference on Intelligent Computer Communication and Processing, ICCP 2017*. [S.l.: s.n.], 2017. ISBN 9781538633687. ISSN 2065-9946.

TURING, A. M. Computing Machinery and Intelligence. *Mind*, Oxford University Press, v. 59, n. October, p. 433–460, 1950.

VÖLKER, J.; HITZLER, P.; CIMIANO, P. Acquisition of OWL DL axioms from lexical resources. *The Semantic Web: Research and Applications*, p. 670–685, 2007. ISSN 03029743.

WARWICK, K.; SHAH, H. *Turing's Imitation Game: Conversations with the Unknown*. 1st. ed. New York, NY, USA: Cambridge University Press, 2016. ISBN 1107056381, 9781107056381.

Wei Zhu; Wei Zhang; Guo-Zheng Li; Chong He; Lei Zhang. A study of damp-heat syndrome classification using Word2vec and TF-IDF. In: *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2016. p. 1415–1420. ISBN 978-1-5090-1611-2. Disponível em: <http://ieeexplore.ieee.org/document/7822730/>.

WELD, D. S. Open Information Extraction using Wikipedia. n. July, p. 118–127, 2010.

WOHLGENANNT, G.; MINIC, F. Using word2vec to build a simple ontology learning system. In: *CEUR Workshop Proceedings*. [S.l.: s.n.], 2016. ISSN 16130073.

YADAV, V.; BETHARD, S. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. *Proceedings of the 27th International Conference on Computational Linguistics*, p. 2145–2158, 2018. ISSN 01960644. Disponível em: <http://aclweb.org/anthology/C18-1182>.

YANG, P.; CHEN, Y. A survey on sentiment analysis by using machine learning methods. In: *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. [S.l.: s.n.], 2017. p. 117–121.

YUAN, C.; ZHUANG, Y.; LI, X. Natural Language Processing Based Ontology Learning. *2010 International Conference on Computer Application and System Modeling (ICCASM)*, IEEE, n. Iccasm, p. 700–704, oct 2010. Disponível em: <http://ieeexplore.ieee.org/document/5620325/>.

ZHOU, Z.; QI, G.; SUNTISRIVARAPORN, B. A new method of finding all justifications in owl 2 el. In: *PROCEEDINGS OF THE 2013 IEEE/WIC/ACM INTERNATIONAL JOINT CONFERENCES ON WEB INTELLIGENCE (WI) AND INTELLIGENT AGENT TECHNOLOGIES (IAT)*. Washington, DC, USA: **IEEE Computer Society**, 2013. WI-IAT '13, p. 213–220.

APÊNDICE A – AIML E *CHATTERBOTS*

A.1 UTILIZAÇÃO DE *WILDCARDS*

Quando se utiliza do wildcard *, é possível capturar o valor da expressão substituída pelo * informando o índice *atributo index* do mesmo, caso haja mais de um por meio do trecho de código <star index="1"/>. Um exemplo de categoria contemplando a *tag template* é exibido a seguir.

```
<category>
<pattern>I like the color * to paint the *</pattern>
<template>
<star index="2"/>
is a nice color to paint
<star index="1"/>.
</template>
</category>
```

No caso de uma entrada com o padrão “I like the color blue to paint the sky”, o primeiro * guarda a palavra “blue” e o segundo a palavra “sky”, que podem ser utilizadas na mesma resposta através da tag <star index="índice”/>.

APÊNDICE B – PROCESSAMENTO DE LINGUAGEM NATURAL

B.1 ANOTAÇÃO A NÍVEL DE PALAVRAS

Número	Tag	Descrição
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential there
5	FW	Foreign word
6	IN	Preposition
7	JJ	Adjective
8	JJR	Adjective, comparativo
9	JJS	Adjective, superlativo
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular ou coletivo
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparativo
22	RBS	Adverb, superlativo
23	RP	Particle
24	SYM	Symbol
25	TO	to
26	UH	Interjection
27	VB	Verb,

28	VBD	Verb, passado
29	VBG	Verb, gerúndio ou particípio
30	VBN	Verb, past participle
31	VBP	Verb, singular no presente excluindo a terceira pessoa
32	VBZ	Verb, terceira pessoa do singular no presente
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun
36	WRB	Wh-adverb

Tabela 29 – POS - Nível de palavra

B.2 ANOTAÇÃO A NÍVEL DE FRASE

Número	Tag	Descrição
1	ADJP	Adjective Phrase.
2	ADVP	Adverb Phrase.
3	CONJP	Conjunction Phrase.
4	FRAG	Fragment.
5	INTJ	Interjection.
6	LST	List marker.
7	NAC	Not a Constituent.
8	NP	Noun Phrase.
9	NX	Usado para agrupar NPs mais longos e complexos
10	PP	Prepositional Phrase.
11	PRN	Parenthetical.
12	PRT	Particle.
13	QP	Quantifier Phrase.
14	RRC	Reduced Relative Clause.
15	UCP	Unlike Coordinated Phrase.
16	VP	Vereb Phrase.
17	WHADJP	Wh-adjective Phrase.
18	WHAVP	Wh-adverb Phrase.

19	WHNP	Wh-noun Phrase.
20	WHPP	Wh-prepositional Phrase.
21	X	Unknown, uncertain, our unbracketable.

Tabela 30 – POS - Nível de frase

APÊNDICE C – O *CHATTERBOT* PUPO

C.1 DIÁLOGO DE APRESENTAÇÃO DO PUPO

```
<category>
<pattern>hello</pattern>
<template>
  <think>
      <waitingYesNo>true</waitingYesNo>
  </think>
 Hello, i am PUPO your <b>Ontology builder</b> assistant, do you want
 create an Ontology?
         <think>
                 <set name="mode">talk</set>
        <set name="inference-asks">no</set>
   </think>
</template>
</category>
```

A *tag <waitingYesNo>* indica que o PUPO espera como responda um *yes* (sim) ou um *no* (não). A *tag <set name="mode»talk</set>* guarda no "cérebro" AIML do PUPO que ele está no modo de diálogo e não de processamento de linguagem natural com o intuito de transformar LN em OWL. Já a *tag <set name="inference-asks»no</set>* armazena a informação de que o sistema não deve realizar perguntas de inferência para usuário, evitando quebra do fluxo de diálogo.

C.2 IMPLEMENTAÇÃO DO MODO DE PROCESSAMENTO DE LINGUAGEM NATURAL

```
<category>
<pattern>yes</pattern>
<that>
    OK, now you can tell me some facts about the Ontology domain, do
    you want?
</that>
<template>
    <think><set name="mode">nlp</set></think>
    Tell me, write in natural language!
```

```
</template>
</category>
```

A *tag* < *set name* = "*mode*" > *nlp* < /*set* > permite ao PUPO entrar em modo *nlp* (Modo de Porcessamento de Linguagem Natural), esperando uma sentença para ser axiomatizada e transformada em OWL.

C.3 SELEÇÃO DE MODOS DO PUPO

```
<category>
<pattern>*</pattern>
<template>
<condition name="mode">
        <li value="nlp">
            <srai>classify <star index="1"/></srai>
        </li>
        <li value="axiom">
            <srai>create axiom <star index="1"/></srai>
        </li>
        <li value="loadOntology">
            <srai>load ontology <star index="1"/></srai>
        </li>
        <li value="createOntology">
            <srai>create ontology <star index="1"/></srai>
        </li>
        <li value="removeOntology">
            <srai>remove ontology <star index="1"/></srai>
        </li>
        <li value="disableOntology">
            <srai>disable ontology <star index="1"/></srai>
        </li>
        <li value="enableOntology">
            <srai>enable ontology <star index="1"/></srai>
        </li>
        <li>
            I cannot understood you, sorry.
            I can understand some functionalities, look below.
            <br/><srai>functionalities</srai>
    </li>
</condition>
```

```
32 </template>
33 </category>
```

C.4 ALGORITMO PARA FUNCIONAMENTO DO PUPO ATRAVÉS DE AIML

Algoritmo 6: Main AIML *tags* AIML for user interaction

Input : AIML Code
Output: Text in natural language from user dialog

```
1  nodeName ← node.getNodeName();
2  if nodeName.equalsIgnoreCase("loadOntologyFromIRI") then
3      return loadOntologyFromURI (node, ps);
4  else if nodeName.equalsIgnoreCase("createOntology") then
5      return createOntology (node, ps);
6  else if nodeName.equalsIgnoreCase("disableOntology") then
7      return disableOntology (node, ps);
8  else if nodeName.equalsIgnoreCase("removeOntology") then
9      return removeOntology (node, ps);
10 else if nodeName.equalsIgnoreCase("enableOntology") then
11     return enableOntology (node, ps);
12 else if nodeName.equalsIgnoreCase("createOWLAxiom") then
13     return createOWLAxiom (node, ps);
14 else if nodeName.equalsIgnoreCase("removeOWLAxiom") then
15     return removeOWLAxiom (node, ps);
16 else if nodeName.equalsIgnoreCase("nextInference") then
17     return nextInference ();
18 else if nodeName.equalsIgnoreCase("classifyText") then
19     return classify (node, ps);
20 else
21     return evalTagContent (node, ps, null);
```

C.5 ALGORITMOS

Algoritmo 7: Tradução - Union and Conjunction Resolution (Resolução de União e Conjunção)

Entrada: Texto em linguagem natural
Saída : Texto em linguagem natural com união e conjunção

```
1  conjuncaoAtiva ← null;
2  palavras ← palavras do texto;
3  indice ← indice da ultima palavra do texto;
4  Para cada palavra do texto faça
5      Se palavras[indice] == ('AND' ou 'OR') então
6          conjuncaoAtiva ← palavras [indice ];
7      Se palavras[indice] ==',' então
8          Se conjuncaoAtiva! = null então
9              palavras [indice ] ← conjuncaoAtiva ;
10     indice ← indice − 1;
11 retorne texto (palavras);
```

Algoritmo 8: Tradução - Coreference Resolution using Stanford Parser Support (Resolução de correferência a partir do Stanford Parser)

Entrada: Texto em linguagem natural
Saída : Texto em linguagem natural com resolução de correferência

```
1  doc ← stanfordParser.getdocument(texto);
2  corefChains ← doc.corefChains();
3  Para correferencia em corefChains faça
4      referencia ← null;
5      sentence ← doc.sentences().get(correferencia.indiceSentenca − 1);
6      size ← correferencia.fimIndice − correferencia.inicioIndice;
7      listWords ← palavras da sentenca;
8      Se correferencia é um nome então
9          referencia ← correferencia;
10     Se correferencia é um pronome então
11         Se correferencia não termina com "'s"e corresponde a um elemento de
            ("mine/my/your/her/our/yours/his/him/her/its/their/theirs/ours/them") então
12             possessivo ← "'s";
13         caso contrário
14             possessivo ← null;
15         Para word ← correferencia.inicioIndice − 1 até
            correferencia.fimIndice − 1 + size faça
16             Se word == correferencia.inicioIndice − 1 e referencia!= null então
17                 listWords [word] ← referencia + possessivo;
18             caso contrário
19                 listWords [word] ← null;
20     novaSentenca ← texto (listWords);
21     doc = stanfordParser.getDocument(doc.text() substituindo sentence por
        novaSentenca);
22 retorne doc.text();
```

Algoritmo 9: Tradução - Sintatic triples selection (Seleção de triplas sintáticas)

Entrada: Lista de árvores de dependência de um texto
Saída : Lista de triplas: [sujeito; verbo; objeto]

```
1  triplas ← [];
2  Para cada árvore na lista de árvores faça
3      Para cada sujeito s_i da árvore faça
4          Para cada verbo v_j da árvore faça
5              Para cada objeto o_k da árvore faça
6                  triplas.add([s_i; v_j; o_k]);
7  retorne triplas;
```

Algoritmo 10: Tradução - Semantic triples selection (Seleção de triplas semânticas)

Entrada: Lista de árvores de dependência de um texto
Saída : Lista de triplas: [sujeito; verbo; objeto]

```
1 triplas ← [];
2 Para cada árvore na lista de árvores faça
3     Para cada padrão semântico p_i faça
4         Enquanto árvore contém p_i faça
5             tripla ← [s_i; v_j; o_k]_{p_i};
6             triplas.add(tripla);
7             remove tripla de árvore;
8 retorne triplas;
```

Algoritmo 11: Tradução - Axioms validation supported by SUMO (Validação dos axiomas a partir da SUMO)

Entrada: Lista de árvores de dependência de um texto
Saída : Axiomas válidos segundo a SUMO

```
1 axiomasAprovados ← [];
2 axiomas ← [];
3 axiomas.add(extractSintaticAxioms(texto));
4 axiomas.add(extractSemanticAxioms(texto));
5 Para cada axioma a_i ∈ axiomas faça
6     Se SUMO ⊨ a_i então
7         axiomasAprovados.add(a_i);
8 retorne axiomasAprovados;
```

C.6 OUTRAS TRADUÇÕES

Sentença	Axioma (DL em *Manchester Syntax*)[1]
a vítima foi assassinada.	Vitima SubClassOf AssassinadoPor some
o agressor bombardeado.	Agressor SubClassOf bombardeia some
o agressor tentou matar.	Agressor SubClassOf TentouMatar some
vítima morta.	Vitima SubClassOf MortoPor some
matar a vítima.	Vitima SubClassOf MortoPor some
ameaçou atacar o alvo.	Alvo SubClassOf AmeacadoSerAtacado-Por some
matando a vítima.	Vitima SubClassOf MortoPor some
fatalidade foi vítima.	Fatalidade SubClassOf Vitima
bomba contra o alvo.	Alvo SubClassOf BombardeadoPor some
morto com instrumento.	Instrumento SubClassOf Mata some
foi objetivado ao alvo.	Alvo SubClassOf ObjetivadoPor some

Tabela 31 – (Em Português) Exemplos de axiomas gerados pelos padrões semânticos

C.7 CONSULTA E RESULTADO REALIZADOS E OBTIDOS PELO SERVIÇO SUMOREST

```
CONSULTA:
    Canino e Mamífero

RESULTADO:
    SuperClasses: Carnivoro, Coisa, VertebradoDeCorpoQuente,
                  ObjetoCorpuscular, Organismo, Entidade, Animal,
                  Mamifero, Agente, Vertebrado, ObjetoOrganico,
                  Objeto, CoisaOrganica, ObjetoConectadoEntreSi,
                  Físico

    ClassesEquivalentes: Canino

    SubClasses: Nada, CãoDoméstico, Raposa

    Instâncias: [NADA]
```

[1] Esses padrões foram testados na língua inglesa, não há quaisquer garantias sobre a língua portuguesa

APÊNDICE D – TRABALHOS RELACIONADOS

D.1 TRADUÇÕES DO PROCESSO DE AXIOMATIZAÇÃO SEGUNDO O LEXO, RENAN E O PUPO, RESPECTIVAMENTE

Segundo Völker, Hitzler e Cimiano (2007)

- Juvenil EquivalenteA: (Jovem e (Peixe ou Animal) e não chegou alguma (Sexual e Maturidade))

Segundo De Azevedo et al. (2014)

- juvenil SubClasseDe: (jovem_peixe ou jovem_animal) e não temChegado somente maturidade_sexual

Segundo PUPO

- Juvenil SubClasseDe: (JovemPeixe ou JovemAnimal) e não (tem algo (chegou alguma MaturidadeSexual))

APÊNDICE E – CÓDIGO JAVA UTILIZADO NA TRANSFORMAÇÃO DE LINGUAGEM NATURAL PARA AXIOMAS EM *MANCHESTER SYNTAX*

Listing E.1 – Código em linguagem de programação Java do TreeWrapper

```
1
import edu.stanford.nlp.trees.Tree;
3
public class TreeWrapper {
5

7     public TreeWrapper() {
      }
9
      public IRIITriple convert(final Tree root){
11        IRIITriple sub = new RIITriple(Tree.valueOf("(_)"));
          convert(sub, root);
13        return sub;
      }
15


17
      public IRIITriple createTree(final Tree node){
19        return new RIITriple(node);
      }
21
      public IRIITriple convert(IRIITriple root, Tree node) {
23
          if (node.isPreTerminal()) {
25            IRIITriple leaf = createTree(node);
              root.addChild(leaf);
27            return leaf;
          }
29        else {
              Tree firstTree = null;
31            IRIITriple first = null;
              IRIITriple left = root;
33            Tree[] children = node.children();
              for (int i = 0; i < node.children().length; i++) {
35                Tree child = children[i];
                  String token = child.value();
37
                  if ((i == 0) && (token.matches("(VB(.*)|MD)"))){
39                    IRIITriple subRiiTree = createTree(child);
                      root.addChild(subRiiTree);
41                    left = subRiiTree;
                      root = subRiiTree;
43
                      for (Tree brother : children) {
45                        if (brother.value().equals("ADVP")) {
                              subRiiTree.addValue(brother);
47                        }
                      }
49                }
```

```
                else if (token.matches("(CC|,|WDT|WP)")) {

                    IRIITriple connec = createTree(child);;

                    if(token.matches("WDT|WP")){
                        child.setValue("CC");
                        child.children()[0].setValue("AND");

                        IRIITriple parent = root;
                        while(parent.moveUp().moveUp()!=null){
                            parent = parent.moveUp();
                        }
                        parent.moveUp().swapChildren(parent, connec);
                        connec.addChild(parent);
                    }else{

                        if(token.equalsIgnoreCase(",")){
                            child.setValue("CC");
                            child.children()[0].setValue("OR");
                        }

                        left.moveUp().swapChildren(left, connec);
                        connec.addChild(left);
                    }

                    left = connec;

                    if (i == 1) {
                        firstTree = child;
                        first = connec;
                        root = connec;
                    }
                }
                else if (token.matches("(NN)(.*)")){
                    if ((i > 0)
                            && (children[i-1].value().matches("ADJP(.*)"))
                            && (left.getValue().toUpperCase().matches("\\s?((AND)|(
                                OR))\\s?"))){

                        IRIITriple subst = createTree(child);

                        for (IRIITriple triple : left.getChildren()){
                            triple.addChild(subst);
                        }

                        left = subst;
                    }
                    else{
                        left = convert(left, child);
                    }
                }
                else if ((token.matches("(MD|RB|VB|S|NN|JJ|DT|NP|ADJP|PRT|PP)(.*)"))
                        || ((left.getValue().toUpperCase().matches("(.*)(OF|AS|TO|
                            FROM|BY|IN|WITH(OUT)?)\\s+SOME\\s+\\(\\s*")) && (token.
                            equals("VP")))){
                    left = convert(left, child);
                }
```

```
                else if(!token.matches("(ADVP)")){

                    if (left.getValue().toUpperCase().matches("\\s+(NOT)\\s+\\(\\s+"
                        )) {
                        left = convert(left, child);
                    } else {
                        left = convert(root, child);
                    }
                }

                if ((i == 0)
                        || (firstTree.value().matches("(DT|CD|JJ|ADVP|ADJP)(.*)"))
                        || (firstTree.value().matches("(NN)(.*)") && (child.value().
                            matches("(NN)(.*)")))){
                    firstTree = child;
                    first = left;
                }

            }

            return first;
        }
    }
}
```

Listing E.2 – Código em linguagem de programação Java do RIITriple

```
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import edu.stanford.nlp.trees.Tree;

public class RIITriple implements IRIITriple {

    private List<IRIITriple> children = new ArrayList<>();
    private IRIITriple parent = null;
    private List<Tree> values = new ArrayList<>();

    public RIITriple(final Tree t) {
        this.values.add(t);
    }

    /*
     * (non-Javadoc)
     *
     * @see
     * edu.br.ufpe.cin.msc.rii.wrapper.IRIITriple#swapChildren(edu.br.ufpe.cin.
     * msc.rii.wrapper.IRIITriple, edu.br.ufpe.cin.msc.rii.wrapper.IRIITriple)
     */
    @Override
    public void swapChildren(final IRIITriple c1, final IRIITriple c2) {
        int index = this.children.indexOf(c1);

        this.children.set(index, c2);
```

```
32          c1.setParent(null);
            c2.setParent(this);
34
        }
36
        /*
38       * (non-Javadoc)
         *
40       * @see edu.br.ufpe.cin.msc.rii.wrapper.IRIITriple#moveUp()
         */
42      @Override
        public IRIITriple moveUp() {
44          return this.parent;
        }
46
        @Override
48      public void addChild(IRIITriple child) {
            this.children.add(child);
50          child.setParent(this);
        }
52
        @Override
54      public String getValue() {
            StringBuilder sb = new StringBuilder();
56          for (Tree leaf : this.values) {
                if (leaf.getLeaves().get(0).value().toUpperCase().matches("NOT")) {
58                  sb.append(" not ( ");
                }
60              // if is a IN or TO ends a property
                else if (leaf.value().matches("(IN|TO)")) {
62                  sb.append(capitalize(leaf.getLeaves().get(0).value()));
                    sb.append(" some ( ");
64              } else if (leaf.value().matches("(CD)")) {
                    sb.append(" exactly ");
66                  sb.append(leaf.getLeaves().get(0).value());
//                  sb.append("1");
68                  sb.append(" ( ");
                } else if (leaf.value().matches("(WDT|WP)")) {
70                  sb.append("AND");
                }
72              // if is a DT discart
                else if (leaf.value().matches("(DT|PRP\\$|MD)")) {
74              }
                // if is a vb*
76              else if (leaf.value().matches("VB(.?)")) {

78                  String value = leaf.getLeaves().get(0).value();
                    value = capitalize(value);
80
                    // if the value is not a TO BE verb add some at the end
82                  if (!value.toUpperCase().matches("(BE|DO|DOES)")) {
                        sb.append(value);
84                      sb.append(" some ( ");
                    }
86              } else if (leaf.value().matches("ADVP")) {
                    sb.append(capitalize(leaf.getLeaves().get(0).value()));
```

```
88              } else {

90                  String value = leaf.getLeaves().get(0).value();
                    value = capitalize(value);
92                  // if the value is not a TO BE verb
                    if (!value.toUpperCase().matches("(BE)")) {
94
                        // if is NNP (proper name) insert inside {} indicating a
96                      // individual
                        if (leaf.value().matches("(NNP(.*))")) {
98                          sb.append("{");
                            sb.append(value);
100                         sb.append("}");
                        } else {
102                         sb.append(value);
                        }
104                 }
                }
106         }

108         return sb.toString().replaceAll("_", "");
        }
110
        @Override
112     public void setParent(IRIITriple parent) {
            this.parent = parent;
114     }

116     @Override
        public int level() {
118         if (this.parent != null) {
                return this.parent.level() + 1;
120         }
            return 0;
122     }

124     @Override
        public boolean isLeaf() {
126         return this.children.isEmpty();
        }
128
        @Override
130     public List<IRIITriple> getChildren() {
            return this.children;
132     }

134     @Override
        public void addValue(Tree value) {
136         this.values.add(value);
        }
138
        @Override
140     public String toString() {
            StringBuilder sb = new StringBuilder(tab(level()));
142         for (Tree t : values) {
                for (Tree value : t.getLeaves()) {
144                 sb.append(value.value());
```

```
            }
        }
        if (!this.children.isEmpty()) {
            for (IRIITriple child : children) {
                sb.append("\n");
                sb.append(tab(child.level()));
                sb.append(child.toString());
            }
        }
        return sb.toString();
    }

    @Override
    public String toLogicalString() {
        return postProcessing(toHierarchicalString(this, "", ""));
    }

    private static String tab(int level) {
        String tab = "";
        for (int i = 0; i < level; i++) {
            tab += " ";
        }
        return tab;
    }

    private String toHierarchicalString(IRIITriple root, String prefix, String sufix
        ) {
        String valueStr = root.getValue();
        // if the value is a some value insert a ')' as suffix
        if (prefix.endsWith(" some ( ") && valueStr.startsWith(" exactly ")) {
            prefix = prefix.substring(0, prefix.length() - " some ( ".length());
        } else if (prefix.endsWith(" some ( ")
                && valueStr.toUpperCase().matches("(.*)(OF|AS|TO|FROM|BY|IN|WITH(OUT
                    )?)\\s+SOME\\s+\\(\\s*")) {
            prefix = prefix.substring(0, prefix.length() - " some ( ".length());
        } else if (valueStr.endsWith(" ( ")) {
            sufix += " )";
        }

        if (root.isLeaf()) {
            return String.format("%s%s%s", prefix, valueStr, sufix);
        } else {
            StringBuilder sb = new StringBuilder();

            int i = 0;
            // if is a conjunction
            if (valueStr.toUpperCase().matches("\\s*(AND|OR)\\s*")) {

                for (IRIITriple child : root.getChildren()) {

                    // insert the conjunction between the clauses
                    if (i > 0) {
                        sb.append(String.format(" %s ", valueStr.toUpperCase()));
                    }
                    sb.append(" ( ");
                    sb.append(toHierarchicalString(child, prefix, sufix));
                    sb.append(" ) ");
```

```
200                     i++;
                    }
202
                } else {
204                 prefix = String.format("%s%s", prefix, valueStr);

206                 // if the children is greater than 1 consider as a conjunction
                    if (root.getChildren().size() > 1) {
208                     for (IRIITriple child : root.getChildren()) {
                            // insert the conjunction between the clauses
210                         if (i > 0) {
                                sb.append(" AND ");
212                         }
                            sb.append(String.format("( %s )", toHierarchicalString(child
                                , prefix, sufix)));
214                         i++;
                        }
216                 }
                    // if the children size is one or less go to next
218                 else {
                        for (IRIITriple child : root.getChildren()) {
220                         sb.append(toHierarchicalString(child, prefix, sufix));
                        }
222                 }
                }
224             return sb.toString();
            }
226     }

228     private static String capitalize(final String value) {
            return Character.toUpperCase(value.charAt(0)) + value.substring(1, value.
                length());
230     }

232     private static String postProcessing(String value) {
            Pattern p = Pattern.compile("(\\w|\\w-\\w)+\\{(\\w|\\w-\\w)+\\}|\\{(\\w|\\w
                -\\w)+\\}(\\w|\\w-\\w)+|(\\w|\\w-\\w)+\\{(\\w|\\w-\\w)+\\}(\\w|\\w-\\w)+
                ");
234         Matcher m = p.matcher(value);
            while(m.find()){
236             String result = m.group();
                value = value.replace(result, result.replaceAll("\\{|\\}",""));
238             m = p.matcher(value);
            }
240         return value.replaceAll("\\}\\{", "")
                    .replaceAll("some (O|o)nly", "only")
242                 .replaceAll("some \\(\\s+(O|o)nly", "only ( ")
                    .replaceAll(" some exactly ", " exactly ")
244                 .replaceAll("\\(\\s*\\)", " ");
        }
246 }
```

APÊNDICE F – COMANDOS DO SISTEMA

Comando em linguagem natural (inglês)	Descrição
load Ontology *iri*	Carrega uma Ontologia a partir de uma iri válida
create Ontology *iri*	Cria uma Ontologia a partir de uma iri ou um nome. Caso seja um nome o sistema automaticamente irá inserir *http://* como prefixo.
disable Ontology *iri*	Desabilita uma Ontologia do sistema
remove Ontology *iri*	Remove uma Ontologia do sistema
enable Ontology *iri*	Habilita uma Ontologia do sistema
natural language sentence ?	Realiza uma consulta nas Ontologias do sistema utilizando linguagem natural.
consult *manchester syntax axiom to query* ?	Realiza uma consulta nas Ontologias do sistema utilizando *Manchester Syntax*.
create property *name*	Cria uma propriedade nas Ontologias ativas.
remove property *name*	Remove uma propriedade das Ontologias ativas.
create class *name*	Cria uma classe nas Ontologias ativas.
remove class *name*	Remove uma classe das Ontologias ativas.
create axiom *axiom in manchester syntax*	Cria um axioma utilizando *Manchester Syntax* nas Ontologias ativas.
remove axiom *axiom in manchester syntax*	Remove um axioma utilizando *Manchester Syntax* das Ontologias ativas.
create individual *name*	Cria um indivíduo nas Ontologias ativas.
remove individual *name*	Remove um indivíduo das Ontologias ativas
next inference	Busca por novas inferências nas Ontologias ativas e exibe sequencialmente para o usuário.
classify *text in natural language*	Axiomatiza um texto em linguagem natural e adiciona seus axiomas nas Ontologias ativas.

check consistence	Checa a consistência das Ontologias ativas. Caso inconsistente o sistema mostra dos axiomas problemáticos sequencialmente para que possam ser validados com o usuário
make closure	Realiza o fechamento (axioma de fecho) nas Ontologias ativas. Cada axioma é exibido ao usuário para que o mesmo venha a validá-los.

Tabela 32 – Comandos do sistema

APÊNDICE G – CÓDIGO OWL

Listing G.1 – Código OWL gerado a partir da sentença *sustained use is a continuing use without severe or permanent deterioration in the resources*

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns="http://edu.ia.br.ufpe.cin.msc.ontology#" xml:base="http://edu.ia.br.
    ufpe.cin.msc.ontology" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdf="
    http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:xml="http://www.w3.org/XML
    /1998/namespace" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:rdfs="http:
    //www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology"/>

    <!--
    ///////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////
     -->

    <!-- http://edu.ia.br.ufpe.cin.msc.ontology#continue -->
    <owl:ObjectProperty rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#continue"/
        >

    <!-- http://edu.ia.br.ufpe.cin.msc.ontology#continuing -->
    <owl:ObjectProperty rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#continuing
        "/>

    <!-- http://edu.ia.br.ufpe.cin.msc.ontology#continuingWithout -->
    <owl:ObjectProperty rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#
        continuingWithout"/>

    <!-- http://edu.ia.br.ufpe.cin.msc.ontology#permanentDeteriorationIn -->
    <owl:ObjectProperty rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#
        permanentDeteriorationIn"/>

    <!-- http://edu.ia.br.ufpe.cin.msc.ontology#severeDeteriorationIn -->
    <owl:ObjectProperty rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#
        severeDeteriorationIn"/>

    <!--
    ///////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////
     -->

    <!-- http://edu.ia.br.ufpe.cin.msc.ontology#Resource -->
    <owl:Class rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#Resource"/>

    <!-- http://edu.ia.br.ufpe.cin.msc.ontology#Sustained -->
    <owl:Class rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#Sustained"/>
```

```
41
    <!-- http://edu.ia.br.ufpe.cin.msc.ontology#SustainedUse -->
43

45  <owl:Class rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#SustainedUse">
        <owl:equivalentClass>
47          <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
49                  <rdf:Description rdf:about="http://edu.ia.br.ufpe.cin.msc.
                        ontology#Sustained"/>
                    <rdf:Description rdf:about="http://edu.ia.br.ufpe.cin.msc.
                        ontology#Use"/>
51              </owl:intersectionOf>
            </owl:Class>
53      </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="http://edu.ia.br.ufpe.cin.msc.ontology#Use"/>
55      <rdfs:subClassOf>
            <owl:Class>
57              <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Class>
59                      <owl:unionOf rdf:parseType="Collection">
                            <owl:Restriction>
61                              <owl:onProperty rdf:resource="http://edu.ia.br.ufpe.
                                    cin.msc.ontology#continuingWithout"/>
                                <owl:someValuesFrom>
63                                  <owl:Restriction>
                                        <owl:onProperty rdf:resource="http://edu.ia.
                                            br.ufpe.cin.msc.ontology#
                                            permanentDeteriorationIn"/>
65                                      <owl:someValuesFrom rdf:resource="http://edu
                                            .ia.br.ufpe.cin.msc.ontology#Resource"/>
                                    </owl:Restriction>
67                              </owl:someValuesFrom>
                            </owl:Restriction>
69                          <owl:Restriction>
                                <owl:onProperty rdf:resource="http://edu.ia.br.ufpe.
                                    cin.msc.ontology#continuingWithout"/>
71                              <owl:someValuesFrom>
                                    <owl:Restriction>
73                                      <owl:onProperty rdf:resource="http://edu.ia.
                                            br.ufpe.cin.msc.ontology#
                                            severeDeteriorationIn"/>
                                        <owl:someValuesFrom rdf:resource="http://edu
                                            .ia.br.ufpe.cin.msc.ontology#Resource"/>
75                                  </owl:Restriction>
                                </owl:someValuesFrom>
77                          </owl:Restriction>
                        </owl:unionOf>
79                  </owl:Class>
                    <owl:Restriction>
81                      <owl:onProperty rdf:resource="http://edu.ia.br.ufpe.cin.msc.
                            ontology#continuing"/>
                        <owl:someValuesFrom rdf:resource="http://edu.ia.br.ufpe.cin.
                            msc.ontology#Use"/>
83                  </owl:Restriction>
                </owl:intersectionOf>
85          </owl:Class>
```

```
            </rdfs:subClassOf>
87      </owl:Class>

89      <!-- http://edu.ia.br.ufpe.cin.msc.ontology#Use -->

91      <owl:Class rdf:about="http://edu.ia.br.ufpe.cin.msc.ontology#Use"/>
    </rdf:RDF>
93  <!-- Generated by the OWL API (version 5.1.7.2018-09-02T11:51:43Z) https://github.
        com/owlcs/owlapi/ -->
```

APÊNDICE H – SENTENÇAS PARA TESTES DE CORRETUDE E RACIOCÍNIO DE SUBSUNÇÃO/DEFINIÇÃO

1 acceptor rna is RNA molecules present in the cell (in at least 20 varieties, each variety capable of combining with a specific amino acid) that attach the correct amino acid to the protein chain that is being synthesized at the ribosome of the cell (according to directions coded in the mRNA)
2 acre is a unit of area (4840 square yards) used in English-speaking countries
3 acreage is an area of ground used for some particular purpose (such as building or farming)
4 aeroplane is an aircraft that has a fixed wing and is powered by propellers or jets
5 affirmative action is a policy designed to redress past discrimination against women and minority groups through measures to improve their economic and educational opportunities
6 airstream is the flow of air that is driven backwards by an aircraft propeller
7 al-maunah is a radical insurgent Islamist group consisting of disaffected middle-class professionals in Malaysia who want to overthrow the government by violent means and set up an Islamic state
8 amati is a violin made by Nicolo Amati or a member of his family
9 "ambit is an area in which something acts or operates or has power or control: ""the range of a supersonic jet"""
10 ancestry is inherited properties shared with others of your bloodline
11 annotation is the act of adding notes
12 anthrax bacillus is a species of bacillus that causes anthrax in humans and in animals (cattle and swine and sheep and rabbits and mice and guinea pigs)
13 antibody is any of a large variety of proteins normally present in the body or produced in response to an antigen which it neutralizes, thus producing an immune response
14 arctic moss is an erect greyish branching lichen of Arctic and even some north temperate regions constituting the chief food for reindeer and caribou and sometimes being eaten by humans

```
15 arena theater is a theater arranged with seats around at least three
   ↪ sides of the stage
16 arm is the part of a garment that is attached at the armhole and that
   ↪ provides a cloth covering for the arm
17 arteria circumflexa scapulae is an artery that serves the muscles of
   ↪ the shoulder and scapular area
18 article is one of a class of artifacts
19 asean is an association of nations dedicated to economic and political
   ↪ cooperation in southeastern Asia and who joined with the United
   ↪ States to fight against global terrorism
20 association area is cortical areas that are neither motor or sensory
   ↪ but are thought to be involved in higher processing of information
21 association of southeast asian nations is an association of nations
   ↪ dedicated to economic and political cooperation in southeastern
   ↪ Asia and who joined with the United States to fight against global
   ↪ terrorism
22 assortment is a collection containing a variety of sorts of things
23 assumption is a hypothesis that is taken for granted
24 aster family is plants with heads composed of many florets: aster
25 aubergine is hairy upright herb native to southeastern Asia but widely
   ↪ cultivated for its large glossy edible fruit commonly used as a
   ↪ vegetable
26 auction bridge is a variety of bridge in which tricks made in excess of
   ↪ the contract are scored toward game
27 auditorium is the area of a theater or concert hall where the audience
   ↪ sits
28 austro-asiatic is a family of languages spoken in southern and
   ↪ southeastern Asia
29 austronesian is the family of languages spoken in Australia and Formosa
   ↪ and Malaysia and Polynesia
30 austronesian language is the family of languages spoken in Australia
   ↪ and Formosa and Malaysia and Polynesia
31 autobus is a vehicle carrying many passengers
32 ayatollah is a high-ranking Shiite religious leader who is regarded as
   ↪ an authority on religious law and its interpretation and who has
   ↪ political power as well
33 azolla is a genus of fern sometimes placed in its own family Azollaceae
```

```
34 bacillus anthracis is a species of bacillus that causes anthrax in
   ↪ humans and in animals (cattle and swine and sheep and rabbits and
   ↪ mice and guinea pigs)
35 backfire is a fire that is set intentionally in order to slow an
   ↪ approaching forest fire or grassfire by clearing a burned area in
   ↪ its path
36 backwash is the flow of air that is driven backwards by an aircraft
   ↪ propeller
37 baltic language is a branch of the Indo-European family of languages
   ↪ related to the Slavonic languages
38 bantoid language is a family of languages widely spoken in the southern
   ↪ half of the African continent
39 bantu is a family of languages widely spoken in the southern half of
   ↪ the African continent
40 bas bleu is a woman having literary or intellectual interests
41 basal ganglion is any of several masses of subcortical grey matter at
   ↪ the base of each cerebral hemisphere that seem to be involved in
   ↪ the regulation of voluntary movement
42 basin is the quantity that a basin will hold
43 basis is a relation that provides the foundation for something
44 basket weave is a cloth woven of two or more threads interlaced to
   ↪ suggest the weave of a basket
45 batters box is an area on a baseball diamond (on either side of home
   ↪ plate) marked by lines within which the batter must stand when at
   ↪ bat
46 battle of spotsylvania courthouse is a battle between the armies of
   ↪ Grant and Lee during the Wilderness Campaign
47 battlefield is a region where a battle is being (or has been) fought
48 battleground is a region where a battle is being (or has been) fought
49 beer is a general name for alcoholic beverages made by fermenting a
   ↪ cereal (or mixture of cereals) flavored with hops
50 bilateral descent is line of descent traced through both the maternal
   ↪ and paternal sides of the family
51 bloat is swelling of the rumen or intestinal tract of domestic animals
   ↪ caused by excessive gas
52 block anesthesia is anesthesia of an area supplied by a nerve
53 bloodline is ancestry of a purebred animal
54 borage family is a widely distributed family of plants distinguished by
   ↪ circinate flowers and nutlike fruit
```

```
55 boxcar is a freight car with roof and sliding doors in the sides
56 brain is the brain of certain animals used as meat
57 branchia is respiratory organ of aquatic animals that breathe oxygen
   ↪ dissolved in water
58 brass is a memorial made of brass
59 brass instrument is a wind instrument that consists of a brass tube
   ↪ (usually of variable length) that is blown by means of a cup-shaped
   ↪ or funnel-shaped mouthpiece
60 breast is either of two soft fleshy milk-secreting glandular organs on
   ↪ the chest of a woman
61 bribe is payment made to a person in a position of trust to corrupt his
   ↪ judgment
62 bruges is a city in northwestern Belgium that is connected by canal to
   ↪ the North Sea
63 bureau of diplomatic security is the bureau in the State Department that
   ↪ is responsible for the security of diplomats and embassies overseas
64 bureau of intelligence and research is an agency that is the primary
   ↪ source in the State Department for interpretive analyses of global
   ↪ developments and focal point for policy issues and activities of
   ↪ the Intelligence Community
65 bureau of justice assistance is the bureau in the Department of Justice
   ↪ that assists local criminal justice systems to reduce or prevent
   ↪ crime and violence and drug abuse
66 bureau of justice statistics is the agency in the Department of Justice
   ↪ that is the primary source of criminal justice statistics for
   ↪ federal and local policy makers
67 bureau of the census is the bureau of the Commerce Department
   ↪ responsible for taking the census
68 calves feet is feet of calves used as food
69 canary seed is a mixture of seeds used to feed caged birds
70 cancel is a notation cancelling a previous sharp or flat
71 capitalist economy is an economic system based on private ownership of
   ↪ capital
72 car carrier is a trailer that can be loaded with new cars for delivery
   ↪ to sales agencies
73 cargo ships is conveyance provided by the ships belonging to one
   ↪ country or industry
74 carry nation is United States prohibitionist who raided saloons and
   ↪ destroyed bottles of liquor with a hatchet (1846-1911)
```

```
75 cataloged procedure is a set of control statements that have been placed in a library and can be retrieved by name
76 cereal oat is widely cultivated in temperate regions for its edible grains
77 chafing is soreness or irritation of the skin caused by friction
78 chapterhouse is a house used as a residence by a chapter of a fraternity
79 charabanc is a vehicle carrying many passengers
80 chemical notation is a notation used by chemists to express technical facts in chemistry
81 chemical property is a property used to characterize materials in reactions that change their identity
82 chopper is an aircraft without wings that obtains its lift from the rotation of overhead blades
83 chromatogram is the recording (column or paper strip) on which the constituents of a mixture are adsorbed in chromatography
84 chu kiang is a river in southeast China that flows into the South China Sea
85 city of bridges is a city in northwestern Belgium that is connected by canal to the North Sea
86 civil rights movement is movement in the United States beginning in the 1960s and led primarily by Blacks in an effort to establish the civil rights of individual Black citizens
87 cladonia rangiferina is an erect greyish branching lichen of Arctic and even some north temperate regions constituting the chief food for reindeer and caribou and sometimes being eaten by humans
88 class sarcodina is characterized by the formation of pseudopods for locomotion and taking food: Actinopoda
89 classroom is a room in a school where lessons take place
90 closed universe is (cosmology) a universe that is spatially closed and in which there is sufficient matter to halt the expansion that began with the big bang
91 cloth is artifact made by weaving or felting or knitting or crocheting natural or synthetic fibers
92 cocain is a narcotic (alkaloid) extracted from coca leaves
93 coffee is a beverage consisting of an infusion of ground coffee beans
94 commission on human rights is the commission of the Economic and Social Council of the United Nations that is concerned with human rights
```

```
95  commission on narcotic drugs is the commission of the Economic and
    ↪ Social Council of the United Nations that is concerned with drug
    ↪ traffic
96  commission on the status of women is the commission of the Economic and
    ↪ Social Council of the United Nations that is concerned with the
    ↪ status of women in different societies
97  common wheat is widely cultivated in temperate regions in many
    ↪ varieties for its commercially important grain
98  commonwealth of australia is a nation occupying the whole of the
    ↪ Australian continent
99  commonwealth of nations is an association of nations consisting of the
    ↪ United Kingdom and several former British colonies that are now
    ↪ sovereign states but still pay allegiance to the British Crown
100 communist party is a political party that actively advocates a
    ↪ communist form of government
```