Consulta a Ontologias utilizando Linguagem Natural Controlada

Fabiano Ferreira Luz

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para a
obtenção do título
de
Mestre em Ciências

Programa: Pós-Graduação em Ciência da Computação

Orientadora: Profa. Dra. Renata Wassermann

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES

São Paulo, Outubro de 2013

Resumo

A presente pesquisa explora áreas de Processamento de Linguagem Natural (PLN), tais como, analisadores, gramáticas e ontologias no desenvolvimento de um modelo para o mapeamento de consulta em língua portuguesa controlada para consultas SPARQL. O SPARQL é uma linguagem de consulta capaz de recuperar e manipular dados armazenados em RDF, que é a base para a construção de Ontologias. Este projeto pretende investigar utilização das técnicas supracitadas na mitigação do problema de consulta à Ontologias utilizando linguagem natural controlada. A principal motivação para o desenvolvimento deste trabalho é pesquisar técnicas e modelos que possam proporcionar uma melhor interação do homem com o computador. Facilidade na interação homem-computador é convergida em produtividade, eficiência, comodidade dentre outros benefícios implícitos. Nós nos concentramos em medir a eficiência do modelo proposto e procurar uma boa combinação entre todas as técnicas em questão.
Palavras-chave: PLN, SPARQL, RDF, Ontologias, Gramáticas.

Abstract

This research explores areas of Natural Language Processing (NLP), such as parsers, grammars and ontologies in the development of a model for mapping queries in controlled Portuguese into SPARQL queries. The SPARQL query language allows for manipulation and retrieval of data stored as RDF, which forms the basis for building ontologies. This project aims to investigate the use of the above techniques to help curb the problem of querying ontologies using controlled natural language. The main motivation for the development of this work is to research techniques and models that could provide a better interaction between man and computer. Ease in human-computer interaction is converted into productivity, efficiency, convenience, among other implicit benefits. We focus on measuring the effectiveness of the proposed model and look for a good combination of all the techniques in question.
Keywords: NLP, SPARQL, RDF, Ontology, Grammars.

Sumário

Lista de Abreviaturas

PLN	Processamento de linguagem natural (*Natural Language Processing*)
OWL	Linguagem Web de Ontologias (*Ontology Web Language*)
RDF	Framework de Descrição de Ontologia (*Resource Description Framework*)
PCFG	Gramática Livre de Contexto Probabilística (*Probabilistic Context Free Grammar*)
LNC	Linguagem Natural Controlada (*Controlled Natural Language*)
GLC	Gramática Livre de Contexto (*Context Free Grammar*)
CNF	Forma Normal de Chomsky (*Chomsky Normal Form*)
SPARQL	Protocolo SPARQL para consulta RDF (*SPARQL Protocol and RDF Query Language*)
LN	Linguagem Natural

Capítulo 1

Introdução

O enorme volume de informação disponível na *Web* faz com que seja difícil encontrar exatamente o que se procura [Vanti, 2002]. Neste contexto, as áreas de recuperação de informação e Processamento de Linguagem Natural (PLN), assumem papéis importantes no desenvolvimento de métodos e técnicas que podem ser aplicados a sistemas com objetivo de facilitar a vida dos usuários. Mais especificamente, consultas em linguagem natural são capazes de refinar tais buscas, trazendo mais eficiência e facilidade.

A tarefa de processar uma linguagem natural permite aos seres humanos a comunicação com os computadores de maneira mais espontânea, eliminando-se a necessidade de adaptação a formas inusitadas de interação, ou mesmo o aprendizado de uma linguagem artificial, cuja sintaxe costuma ser de difícil aprendizado, a exemplo das linguagens de consulta a bancos de dados [Russell et al., 1995].

As ontologias representam o ponto mais elevado já atingido em termos de representação, compartilhamento e reutilização do conhecimento [de Freitas and Vieira, 2008]. Nós aproveitamos o poder da representação que a ontologia pode proporcionar, no desenvolvimento de um modelo de busca que possa usufruir desta expressividade.

Nesse sentido, nosso trabalho procura combinar técnicas na mitigação do problema de consulta em linguagem natural. Este trabalho propõe a utilização de uma linguagem natural controlada para a realização de consultas em ontologias através da linguagem de consulta SPARQL [Prud'Hommeaux et al., 2008] que é gerada pelo modelo. Assim, utilizamos uma gramática controlada cujas regras possam ser mapeadas para regras de geração de uma consulta SPARQL. Estamos trabalhando com gramática e vocabulário restritos, que permite eliminar ambiguidade e reduzir complexidade [Chechev et al., 2004].

Para verificar a viabilidade do método proposto, apresentamos dois estudos de caso, isto é, duas ontologias em português de domínio público. A primeira que será submetida as buscas do modelo é sobre o curriculum Lattes [da Costa and Yamate, 2009]. A segunda é sobre conceitos relacionados a smartphones [Martins, 2007].

1.1 Motivação

Explorar as possibilidades oferecidas pelos recursos computacionais ao processamento da linguagem humana tem sido um dos grandes desafios em nossos dias [Brito, 2008]. Atualmente, a informática vem contribuindo com novas perspectivas capazes de revolucionar o pensamento sobre a linguagem [Brito, 2008].

O constante avanço no campo da Web Semântica (WS) proporciona a utilização de novas formas de buscas na Web. Uma das ferramentas que utiliza os recursos da WS é o LINKED DATA[Yu, 2011] que dentre os principais objetivos estão: estruturar informações facilitando a recuperação por SPARQL e conectar diferentes domínios. Neste sentido, assim como LINKED DATA, nosso trabalho pretende aproveitar as vantagens desta estrutura.

É importante ressaltar que nosso trabalho é focado em língua portuguesa. Dada a carência na área, este trabalho vem contribuir para pesquisas relacionadas a ontologias e PLN em português.

1.1.1 Por que Ontologias?

Do ponto de vista de representação do conhecimento, as ontologias constituem em um arsenal mais poderoso que os bancos de dados convencionais, levando em consideração a necessidade de codificar o conhecimento para que o torne acessível a quem precisa, categorizando-o, descrevendo-o, modelando-o, estimulando e inserindo regras e padrões definidos. As ontologias são atualmente o estado da arte para esta representação.

Como fator de motivação podemos também citar o interesse atual da comunidade científica pela web semântica e as inúmeras áreas de aplicação previstas para o uso intensivo de ontologias em sistemas de informação [de Freitas and Vieira, 2008]. A vasta gama de aplicações é uma das razões do grande interesse por métodos de construção, manutenção, mapeamento e aprendizado de ontologias [Ribeiro and Vieira, 2008].

1.2 Objetivos e justificativa

Ao longo de décadas pesquisadores de Inteligência Artificial buscam em várias outras ciências descobrir como modelar a inteligência [Hardesty, 2010]. Neste sentido os estudos de Processamento de Linguagem Natural (PLN) procuram investigar a capacidade humana de comunicação. O corrente projeto se justifica pela importância de se explorar e compreender novos campos da Computação; integrar tecnologias em busca de superar atuais desafios proposto pela Inteligência Artificial.

Os objetivos propostos são:

- Desenvolver uma linguagem controlada que possibilite a consulta em ontologias.
- Desenvolver um modelo de mapeamento da gramática do português controlado para SPARQL.
- Medir a qualidade da transformação (tradução) de consultas em português controlado para SPARQL através de métricas tradicionais.

1.3 Contribuições

A tarefa de transformação de consultas em linguagem natural para linguagem de consulta não é uma tarefa trivial [Androutsopoulos et al., 1995], [Tablan et al., 2008], [Nantes, 2008] e [Capetta, 2012] . O modelo que propomos neste trabalho baseia-se na utilização de gramáticas e parsers. Assim, as principais contribuições deste trabalho são as seguintes:

- Item 1. Desenvolvimento de uma linguagem controlada que possibilite a consulta em ontologias.
- Item 2. Desenvolvimento de um modelo de mapeamento de consultas em língua portuguesa controlada para SPARQL.
- Item 3. Implementação de um sistema que permite realizar consultas através do português controlado.

1.4 Organização do Trabalho

O Capítulo 2 é dedicado a introduzir os conceitos necessários à compreensão do trabalho. O Capítulo 3 apresenta trabalhos relacionados. No Capítulo 3.2 apresentamos algumas ferramentas utilizadas no modelo. No Capítulo 4, descrevemos todos os processos do modelo. O Capítulo 5 é dedicado aos testes e discussões. No Capítulo 6 apresentamos nossas conclusões e sugestões para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo é dedicado a uma breve revisão bibliográfica dos conceitos utilizados em nosso trabalho. Apresentaremos uma breve descrição das técnicas e tecnologias utilizadas procurando ao final de cada descrição deixar clara a importância do conceito no contexto do trabalho.

2.1 Linguagem Natural Controlada

Linguagem natural é qualquer linguagem desenvolvida naturalmente pelo ser humano, de forma não premeditada, como resultado da facilidade inata para a linguagem possuída pelo intelecto humano [Chomsky, 1975]. As línguas faladas, escritas e de sinais são exemplos de linguagens naturais.

Atualmente é aceito pela academia que línguas são sistemas. Todos os elementos de uma língua estão ligados entre si a partir de uma variedade de relações [Saussure, 1978].

Apesar da assimilação da linguagem humana acontecer de maneira natural, ela é um sistema de notável complexidade [Chomsky, 1975]. Embora exista uma grande variedade de línguas naturais, qualquer criança humana é capaz de aprender qualquer língua natural. Apesar de estarmos utilizando ideias de Chomsky e de Saussure é importante ressaltar que os dois pertencem a frentes distintas dentro da linguística sendo um gerativista e outro estruturalista.

Processamento de linguagem natural (PLN) é o estudo da modelagem matemática e computacional de vários aspectos da linguagem e o desenvolvimento de uma ampla gama de sistemas [Joshi et al., 1991]. PLN tem o objetivo geral de processar a linguagem humana para que seja compreensível pelo computador. Dentre as áreas importantes de PLN podemos destacar: o reconhecimento da voz, o reconhecimento da escrita e a reprodução da voz a partir do texto [Schneider, 2001].

PLN abrange várias e complexas áreas do conhecimento [da Silva et al., 2007]. Pesquisas em PLN são interdisciplinares, envolvendo conceitos de ciência da computação, linguística, lógica e psicologia. PLN tem um papel especial em ciência da computação, pois muitos aspectos do campo de negócios procuram modelar características linguísticas de computação [Joshi et al., 1991].

Linguagens naturais controladas (LNC) são subconjuntos de linguagem natural, cuja gramáticas e dicionários foram restringidos de modo a reduzir ou eliminar a ambiguidade e complexidade [Schwitter, 2007]. O conceito foi originalmente proposto em 1930 por linguistas e estudiosos que procuraram estabelecer uma variedade "mínima" de Inglês, a fim de torná-lo acessível e utilizável internacionalmente por um maior número de pessoas (especialmente falantes não-nativos) [Schwitter, 2007].

O estudo do processamento de linguagem em domínio limitado é importante por algumas razões. Primeiro, a construção de uma aplicação de PLN de sucesso para uma sub-linguagem requer explorar suas restrições gramaticais e lexicais, permitindo "desvios" que podem ser perfeitamente aceitáveis para especialistas de domínio. Uma segunda razão, por causa das limitações de vocabulário, sintaxe e semântica de uma sub-linguagem, pode ser possível construir uma descrição relativamente completa linguística, algo muito além do estado da arte para as línguas como um todo [Mitkov et al., 2003].

De fato linguagens controladas facilitam o trabalho de modelagem dos sistemas de PLN. Durante a última década, diversas linguagens naturais controladas foram projetadas e usadas para escrever

as especificações de software, apoiar o processo de aquisição e representação do conhecimento, entre elas a inglesa *AttemptoControlled* [Schwitter et al., 2008]. Para este trabalho desenvolvemos uma linguagem controlada que descreve alguns padrões de consultas em língua portuguesa.

Em nosso projeto estamos propondo um português controlado, que é um subconjunto da língua portuguesa tradicional. Procuramos tangenciar alguns padrões do conjunto das frases interrogativas diretas que são aquelas que começam por palavra interrogativa e terminam com um ponto de interrogação [Alvarez, 2008], por exemplo: "Quantos filhos João tem?", "Qual é o número do telefone de Pedro". Outro conjunto de perguntas que abordamos são as frases declarativas interrogativas que não começam por palavra interrogativa porém terminam por ponto de interrogação, por exemplo: "Ana tem um carro novo?", "Paulo é professor de matemática?". O controle da nossa linguagem tem como objetivo reduzir a complexidade e a ambiguidade da linguagem natural.

2.1.1 Gramática Livre de Contexto

Gramáticas livres de contexto (GLC) foram utilizadas pela primeira vez no estudo da linguagem humana. Um caminho para entender o relacionamento entre substantivos, verbos e preposições. GLC's podem capturar importantes aspectos destes relacionamentos [Sipser, 2006].

As linguagens geradas por gramáticas livres de contexto são conhecidas como Linguagens livres de contexto (LLC).

2.1.2 Definição

Uma gramática livre de contexto é uma 4-upla (V, Σ, R, S), onde:

1. V é um conjunto finito chamado de variáveis ou não terminais,
2. Σ é um conjunto disjunto de V, chamado de terminais,
3. R é um conjunto finito de regras, os membros de R, são chamados regras de substituição ou produções da gramática,
4. $S \in V$ é a variável inicial.

2.1.3 Análise Sintática

[Jurafsky and Martin, 2000] define *parsing* como sendo a combinação do reconhecimento de uma *string* de entrada e gerando uma estrutura de saída para ela. Para o parsing sintático em nosso modelo utilizamos o conhecido algoritmo CKY [Younger, 1967] e [Aho and Ullman, 1972]. Este algoritmo recebe dois parâmetros: o primeiro parâmetro é um sentença s; e o segundo parâmetro é a gramática G que é uma gramática livre de contexto (GLC), o algoritmo faz o parser *bottom-up* de acordo com as regras de G e devolve a arvore gerada para a sentença. É necessário que a G esteja na forma normal de Chomsky para que seja possível a realização do parser.

Quando se trabalha com gramáticas livres de contexto (GLC), é conveniente tê-las em forma simplificada. Uma das formas mais simples e mais úteis é a chamada **forma normal de Chomsky** (CNF) [Sipser, 2006] . Em geral a CNF simplifica uma gramática binárizando-a, sendo útil para a construção de algoritmos para fazer *parsers* nas arvores geradas por estas gramáticas. Uma gramática livre de contexto está na forma normal de Chomsky se toda regra está na forma:

$$S \to AB$$
$$A \to \alpha$$

Onde α é qualquer terminal e S,A e B, são quaisquer variáveis; exceto que A e B não podem ser variável inicial. Adicionalmente permitimos a regra $S \to \epsilon$, onde S é a variável inicial. Toda gramática na forma normal de Chomsky é uma GLC, e inversamente, toda GLC pode ser transformada em uma equivalente que está na forma normal de Chomsky.

2.2 Ontologias

O termo ontologia surgiu na filosofia como a área de estudo da natureza do ser, das coisas existentes no mundo. Foi adaptado e hoje é empregado nas áreas de Inteligência Artificial, *Web* Semântica e Arquitetura da Informação como uma técnica de formalização da representação do conhecimento [de Freitas and Vieira, 2008].

Na computação, as ontologias são utilizadas para capturar conhecimento de um determinado domínio de interesse. Para [Gruber et al., 1993] ontologia é "uma especificação formal de uma conceitualização". [Borst, 1997] amplia a definição quando coloca que essa conceitualização também pode ser compartilhada.

A Web Ontology Language (OWL) é uma linguagem formal para a descrição de ontologias [Bechhofer et al., 2004], tornando-as compreensíveis tanto por humanos quanto por computadores. Conceitos simples ou complexos podem ser desenvolvidos utilizando uma vasta gama de operadores que a linguagem OWL pode oferecer.

As ontologias contêm um vocabulário para representação do conhecimento. Esse vocabulário é sustentado por uma conceitualização pré-definida, evitando assim interpretações ambíguas. Dentre os diversos elementos que as ontologias OWL possuem, aqui iremos apresentar apenas os mais relevantes para este trabalho:

- Indivíduos:

  Indivíduos representam objetos no domínio de interesse (ou domínio do discurso). OWL não usa o *Unique Name Assumption (UNA)*, isto significa que dois nomes diferentes podem remeter ao mesmo indivíduo. Por exemplo, *Queen Elizabeth* (Rainha Elizabeth) e *Elizabeth Windsor* podem ser referências ao mesmo indivíduo. Em OWL deve-se declarar explicitamente que os indivíduos são os mesmos, ou diferentes uns dos outros [Horridge et al., 2004].

- Propriedades:

  Propriedades são relações binárias entre indivíduos, podendo ser esta relação com um atributo de dados, denominada propriedade primitiva (Datatype Property), ou outro indivíduo, denominada propriedade objeto (Object Property). Por exemplo, a propriedade *hasSibling* (temIrmão) pode ligar o indivíduo Matthew ao indivíduo Gemma; ou a propriedade *livesIn* (viveEm) pode ligar o indivíduo Matthew ao indivíduo England. A Figura 2.1 mostra exemplos de propriedades que conectam indivíduos [Horridge et al., 2004].

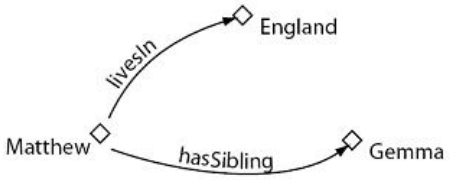


Figura 2.1: *Relação entre propriedades e individuos [Horridge et al., 2004].*

As propriedades podem também ser inversas. Por exemplo, a propriedade inversa de *hasParent* (temPai) é *hasChild* (temFilho) [Horridge et al., 2004].

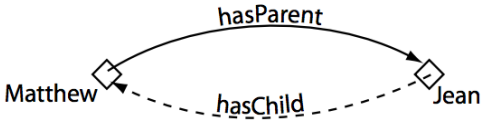


Figura 2.2: *Relação inversa [Horridge et al., 2004].*

As propriedades podem limitar-se a um valor único: são as propriedades funcionais. No exemplo abaixo implica que Jean só pode ter apenas uma mãe biológica (hasBirthMother).

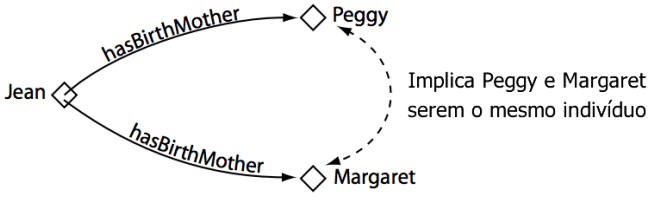


Figura 2.3: *Relação funcional [Horridge et al., 2004].*

Transitiva: se a propriedade **p** é transitiva, e esta propriedade relaciona o indivíduo **a** com o indivíduo **b**, e a mesma propriedade relaciona o indivíduo **b** com o indivíduo **c**, então podemos inferir que o indivíduo **a** está relacionado com o indivíduo **c** através da propriedade **p**. A Figura 2.4 ilustra um exemplo de transitividade.

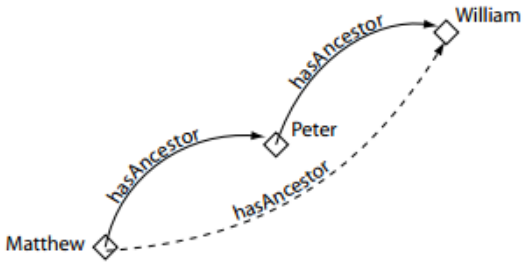


Figura 2.4: *Relação transitiva [Horridge et al., 2004].*

Simétrica: se a propriedade **p** é simétrica, e esta propriedade relaciona o indivíduo **a** com o indivíduo **b** então o indivíduo **b** também está relacionado com o indivíduo **a** pela propriedade **p**. A Figura 2.5 mostra um exemplo de propriedade simétrica.

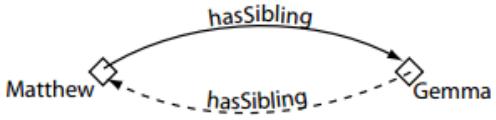


Figura 2.5: *Relação simétrica [Horridge et al., 2004].*

- Classes:

  As classes são conjuntos que contêm os indivíduos, por exemplo, a classe `Gato` pode conter todos os indivíduos que são gatos, no domínio de interesse. Classes são descritas formalmente definindo os requisitos para a participação nas mesmas. As classes também podem ser organizadas em hierarquias superclasse-subclasse, conhecidas como taxonomias. Subclasses são especializações de suas superclasses [Horridge et al., 2004].

  Considere-se as classes `Animal` e `Gato`: `Gato` pode ser subclasse de `Animal`, e assim `Animal` é superclasse de `Gato`. Isso quer dizer que: todos os `Gatos` são `Animais`, ou seja, todos os membros da classe `Gatos` são membros da classe `Animal`; Ser um `Gato` implica ser um `Animal`. Uma característica do OWL é que o relacionamento superclasse-subclasse pode ser computado automaticamente por um mecanismo de inferência (MI). A Figura 2.6 mostra uma representação de classes e seus indivíduos: as classes são representadas como círculos [Horridge et al., 2004].

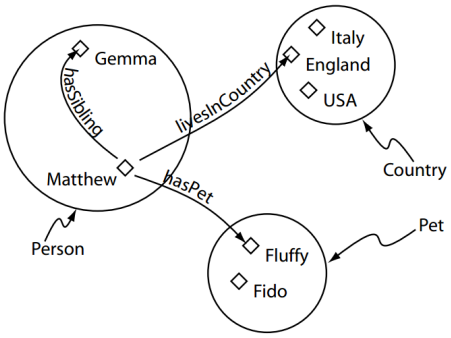


Figura 2.6: *Relações entre classes [Horridge et al., 2004].*

Atualmente existe uma dificuldade em encontrar ontologias na língua portuguesa. Existem ainda poucas propostas na área para nosso idioma. Além disso, existe a dificuldade de compartilhá-las [de Freitas and Vieira, 2008]. A maioria dos trabalhos utilizando ontologias, mesmo que desenvolvida por pesquisadores brasileiros, tomam como base ontologias em língua inglesa, pela simples razão da carência de ontologias na língua portuguesa [Ribeiro and Vieira, 2008].

No contexto do nosso trabalho iremos procurar uma correlação entre os termos que se encontram na ontologia e os termos utilizados durante a consulta. Essa correlação é dependente da interoperabilidade entre essa ontologia e a linguagem natural controlada. O que chamamos de interoperabilidade entre ontologia e linguagem natural é o fato dos elementos da ontologia serem escritos de maneira "clara", com termos bem definidos em linguagem natural. Com isso não estamos incluindo ontologias que utilizam a prática de abreviações e concatenações. Isso gera uma nova palavra que não pertence a língua portuguesa. Por exemplo, o termo "**Conferência Internacional**" não poderia estar grafado como **ConfInter**, pois esta não é uma palavra da língua portuguesa. Optamos por aceitar elementos compostos separados por *underline*.

Mais detalhes sobre as ontologias aceitas pelo nosso modelo estão da Seção 4.1.

2.3 SPARQL

SPARQL (acrônimo recursivo: SPARQL Protocol and RDF Query Language) é uma linguagem de consulta capaz de recuperar e manipular dados armazenados no formato RDF (Resource Description Framework) que é a base da linguagem OWL. RDF é um formato de dados rotulado e dirigido utilizado para representar as informações na *Web* [Prud'Hommeaux et al., 2008]. Como uma linguagem de consulta, SPARQL é orientada a dados, isto é, não existe inferência na própria linguagem.

Uma consulta SPARQL consiste em três partes [Pérez et al., 2006]:

- **Pattern matching**: inclui várias características interessantes de combinação de padrões de gráficos, a união de padrões, aninhamento, filtragem e a possibilidade de escolher a fonte de dados.
- **Solution modifiers**: permite modificar os valores da saída padrão aplicando os operadores clássicos, como: **PROJECTION**, **LIMIT**, **DISTINCT**, **ORDER** e **OFFSET**.
- **Output**: podem ser de diferentes tipos, como: booleanas, seleções, gráficos e descrições de recursos.

Em nosso modelo, iremos trabalhar com uma pequena parte do que a linguagem SPARQL é capaz de fornecer. Trabalharemos na geração de triplas RDF's concatenadas com modificadores de resultados como, por exemplo, seleção, contagem e booleanos.

2.3.1 Sintaxe

SPARQL possui centenas de elementos que compõem sua sintaxe [Beckett, 2006], porém vamos nos ater apenas aos elementos mais importantes no contexto deste trabalho. O propósito é transmitir uma noção básica sobre a construção e o funcionamento das consultas SPARQL.

Triplas RDF

Uma tripla RDF contém três componentes [Klyne et al., 2004]:

- **Sujeito** : podem ser indivíduos ou classes da ontologia.
- **Predicado** : são as relações e podem ser propriedades objetos, propriedades primitivas, ou algum tipo de recurso definido pela versão do RDF.
- **Objeto** : são valores das relações, podem ser indivíduos, classes ou dados primitivos.

Uma tripla RDF é convencionalmente escrita na sequência: sujeito, predicado, objeto. O predicado também é conhecido como propriedade da tripla [Klyne et al., 2004]. O sujeito e o objeto são nomes para duas "coisas" no mundo, e o predicado é o nome de uma relação entre os dois.

Sintaxe para triplas

As consultas SPARQL são baseadas em triplas RDF, compostas por sujeito, predicado e objeto.

```
PREFIX p:<our_URI>
ASK {
        p:joao p:tem_filho "joaozinho"
     }
```

Na consulta SPARQL acima `<our_URI>` é a `URI` (em inglês, *Uniform Resource Identifier*), o identificador de recurso da ontologia. PREFIX é a palavra chave que declara `p` como um *alias* da URI.`joao` é o sujeito, `tem_filho` é o predicado e `joaozinho` é o objeto.

Tipos de consulta

Os tipos estruturais de consulta são [Beckett, 2006]:

- `SELECT`: Usado para extrair valores brutos de uma base RDF, os resultados são apresentados em uma tabela.
- `CONSTRUCT`: Usado para extrair informações de uma base RDF e transformar o resultado em um RDF válido.
- `DESCRIBE`: Utilizado para extrair um gráfico de uma base RDF.
- `ASK`: Usado para fornecer simples resultado booleano para uma consulta em uma base RDF.

A estrutura `SELECT` pode ser utilizada em combinação com `DISTINCT`, que elimina soluções duplicadas [Prud'Hommeaux et al., 2008]. Outra função comum em SPARQL é a `count` que é combinada com o `SELECT` para contar resultados.

2.4 Métricas utilizadas

Precisão e **cobertura** são medidas comuns na recuperação da informação. Elas baseiam-se na comparação de um resultado esperado e o resultado do sistema de avaliação. Esses resultados são considerados como um conjunto de elementos, por exemplo, os documentos a serem recuperados [Euzenat, 2007].

A precisão e a cobertura padrão são dadas a seguir, onde A representa um conjunto de respostas da consulta transformada e R é o conjunto de respostas esperado.

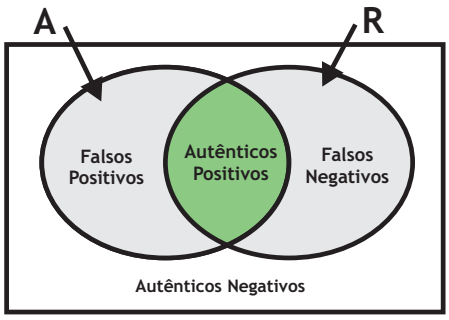


Figura 2.7: *Precisão e cobertura - visão de conjuntos [Bispo Jr, 2011].*

Dado um conjunto de Referencia R a precisão de um conjunto A é dada por:

$$P(A,R) = \frac{|A \cap R|}{|A|} \tag{2.1}$$

e a cobertura é dada por:

$$C(A,R) = \frac{|A \cap R|}{|R|} \tag{2.2}$$

No contexto do nosso trabalho iremos utilizar precisão e cobertura para medir a qualidade das consultas. Elaboramos o conjunto R com 143 perguntas em linguagem controlada e as respostas esperadas, o conjunto R é o nosso conjunto de referência. Submetemos as mesmas 143 perguntas em nosso modelo e as respostas geradas fazem parte do nosso conjunto A.

Uma medida de qualidade comumente utilizada em tarefas de PLN é a **acurácia** por palavra [Duarte, 2009]. A acurácia é definida formalmente como:

$$\text{acurácia} = \frac{|\{\text{palavras corretamente classificadas}\}|}{|\{\text{palavras}\}|} \tag{2.3}$$

Usada na classificação binária a acurácia é muito importante pois é sensível ao número de erros de classificação [Manning and Schütze, 1999]. Neste trabalho utilizamos um conceito mais simples de acurácia, em particular o mesmo principio aplicado à tarefas de *tagging* [Manning and Schütze, 1999]. A reformulação da medida adequando-a ao contexto do nosso trabalho resulta em:

$$\text{acurácia} = \frac{|\{\text{perguntas corretamente respondidas}\}|}{|\{\text{perguntas}\}|} \tag{2.4}$$

Capítulo 3

Trabalhos Relacionados e Ferramentas

Neste capítulo, descrevemos brevemente alguns trabalhos relacionados encontrados na literatura, bem como as ferramentas utilizadas no desenvolvimento do nosso trabalho.

3.1 Trabalhos relacionados

Destacamos dois trabalhos correlacionados. O primeiro é o Querix [Kaufmann et al., 2006], que possui uma abordagem estatística. O sistema consiste de alguns procedimentos para realizar as consultas, que basicamente são: receber a pergunta em linguagem natural do usuário direcionada a determinada ontologia; aumentar os rótulos dos recursos da ontologia através da obtenção dos sinônimos pela WordNet; realizar a análise da sentença; combinar estruturas da consulta com triplas avançadas da ontologia e gerar consulta SPARQL a partir destas triplas.

O Querix funciona como um componente e pode ser adaptado a qualquer aplicação, ele é baseado em clarificação de diálogos[1], então quando existe ambiguidade o sistema pede que o usuário decida. Uma implementação em Java foi desenvolvida para avaliar o Querix. Foi utilizada uma base de conhecimento sobre informação geográfica dos EUA e foram submetidas 215 consultas, obtendo uma precisão total de 77.67% e uma cobertura de 78.6%.

Embora estejamos abordando o mesmo problema, o cenário de testes que o Querix utiliza é bastante distinto do cenário que utilizamos para testar o nosso modelo. Essas diferenças devem-se principalmente às restrições definidas pelo nosso modelo. Por outro lado, no Querix a resolução de ambiguidades é feita pelo usuário tirando do sistema a responsabilidade da mitigação da mesma que é um dos maiores problemas quando se trata de linguagem natural. Na seção 5.2 temos um comparativo entre os resultados do nosso modelo com o Querix.

O segundo trabalho é o SQUALL [Ferré, 2012], que propõe uma linguagem de alto nível que permite a consulta semântica e atualização de RDF, fornece uma linguagem natural controlada em cima de SPARQL 1.1, preservando a adequação, interoperabilidade e expressividade. Sua sintaxe e semântica são formalmente definidas como uma gramática de Montague [Partee, 1976]. A semântica de sentenças SQUALL é formulada em uma linguagem lógica intermediária. O trabalho esboça uma tradução a partir desta linguagem intermediária para SPARQL, proporcionando assim uma semântica operacional e uma possível implementação para SQUALL.

Aplicando as definições, transforma-se a linguagem intermediária em um SPARQL. O trabalho é bastante sólido porem utiliza de algumas abstrações matemáticas para completar o modelo. A linguagem possui boa expressividade e cobertura, entretanto em virtude das abstrações usadas no desenvolvimento do trabalho surgem algumas dificuldades na implementação da ideia, por exemplo, o fato de o modelo não se preocupar com erros léxicos.

Existe uma implementação para o SQUALL que pode ser encontrada no endereço eletrônico http://lisfs2008.irisa.fr/ocsigen/squall/ . A implementação é interessante, porém trata apenas de frases estáticas, onde é possível somente mudar o valor das palavras chaves. Por exemplo, “What is

[1] O termo clarificação de diálogos é utilizado no Querix denominando o ato do usuário poder esclarecer ambiguidades durante as consultas

the name of the author of X?", é uma consulta SQUALL onde **X** é a palavra a ser substituída pelo usuário. O SPARQL gerado leva em consideração a estrutura RDF da DBPedia[2].

3.2 Ferramentas e Métodos Utilizados

O CoGrOO [Kinoshita et al., 2005] é um corretor gramatical acoplável ao LibreOffice. Um corretor gramatical detecta erros nas relações entre as palavras, por exemplo, na sentença "Os menino estudam demais.", o corretor gramatical detecta o erro na concordância entre o artigo "os" e o substantivo "menino". E também sugere uma medida corretiva, como a substituição da palavra "menino" pela mesma palavra flexionada no plural, que é "meninos". Assim, o CoGrOO é capaz de detectar diversos tipos de erros gramaticais, como, por exemplo: colocação pronominal, concordância nominal, concordância entre sujeito e verbo, concordância verbal, uso de crase, regência nominal, regência verbal, erros comuns da língua portuguesa escrita[3].

Estamos utilizando a versão do CoGrOO 4.0 [Finger and Colen, 2012] a qual foi melhorada reduzindo a quantidade de omissões e intervenções indevidas e foram adicionadas novas regras para detecção de erros [Colen, 2013]. Dentre as diversas funcionalidades desta ferramenta, a mais importante no contexto do nosso trabalho é o reconhecimento de *tokens*. Mais especificamente, o CoGrOO nos auxilia na identificação dos substantivos.

A lematização consiste no registro sintético da unidade, a partir de uma forma de realização tomada como referência, normalmente indicada na forma singular e no masculino quando temos substantivos, ou no infinitivo, quando se tratar de verbos [Bevilacqua and Finatto, 2009]. A lematização em nosso contexto resolve problemas de número, gênero e grau dos substantivos durante a comparação das palavras da nossa LNC com nossa ontologia.

Estamos usando uma API java chamada ptStemmer[4]. Essa API nos permite extrair o lema da nossa palavra e assim aplicamos o lematizador nas duas palavras que estamos comparando e simplificamos esta comparação.

O TEP 2.0 [Maziero et al., 2008] é a nova versão do TeP [Dias-Da-Silva and de Moraes, 2003], um dicionário eletrônico de sinônimos e antônimos para o português do Brasil, disponibilizado por meio da base de dados lexical Diadorim1 [Greghi et al., 2002]. Do ponto de vista linguístico, além da ampliação do número de entradas lexicais e dos conjuntos de sinônimos e antônimos em relação à versão inicial, o TeP 2.0 armazena determinadas informações provenientes da base da WordNet.Br [Di Felippo and Dias-da Silva, 2007]. Do ponto de vista computacional, o TeP 2.0 está disponível para consulta e download em http://www.nilc.icmc.usp.br/tep2/.

Sinônimos são palavras que têm o mesmo ou aproximadamente o mesmo sentido que outras. As palavras que serão comparadas podem ser sinônimos uma das outras. Neste caso estamos usando base de dados da ferramenta TEP 2.0. A ferramenta é útil ao usuário que deseja, na produção ou análise de textos, ter opções de escolha de sinônimos ou antônimos, por diversos motivos, dentre eles adequação comunicativa, precisão, correção ou aprendizagem [Maziero et al., 2008].

3.2.1 CKY

Cocke-Younger-Kasami (CKY) [Younger, 1967] e [Aho and Ullman, 1972] é um algoritmo utilizado para fazer parsing em gramáticas livres de contexto. Ele usa programação dinâmica e emprega a abordagem *bottom-up*. A versão tradicional do CKY exige que a gramática a ser utilizada esteja na Forma Normal de Chomsky. A complexidade de tempo desse algoritmo é $\Theta(n^3)$, isso significa que para o caso médio, o número de operações que o algoritmo faz é n ao cubo onde n é o tamanho da entrada (quantidade de palavras da sentença).

Tradicionalmente o algoritmo CKY utiliza como parâmetro de entrada uma GLCP pois constrói a árvore mais provável para cada sentença. Uma GLCP é uma tupla (N, Σ, S, R, q), onde:

[2] http://dbpedia.org/About
[3] http://cogroo.sourceforge.net/#overview
[4] https://code.google.com/p/ptstemmer/

1. N é um conjunto finito chamado de variáveis ou não terminais,
2. Σ é um conjunto disjunto de V, chamado de terminais,
3. R é um conjunto finito de regras, os membros de R, são chamados regras de substituição ou produções da gramática,
4. $S \in V$ é a variável inicial.
5. q é uma função de probabilidade. Como já foi mencionado aqui usamos o valor 1 para qualquer regra existente. Portanto q é desnecessário para nosso algoritmo.

Como veremos na seção 4.7 as ambiguidades sintáticas não são um problema para nosso modelo. Portanto adaptamos o CKY para trabalhar sem probabilidades, ou seja, realizando parser de uma GLC simples na forma normal de Chomsky. De maneira determinística ele irá atribuir probabilidade 1 para toda regra existente na gramática e zero caso contrário.

Entrada: uma sentença $s = x_1...x_n$, uma $GLCP = (N, \Sigma, S, R)$.
Inicialização:
Para todo $i \in \{1...n\}$, para todo $X \in N$,

$$\pi(i, i, X) = \begin{cases} 1 & \text{se } X \to x_i \in R \\ 0 & \text{para outros casos} \end{cases}$$

Algoritmo:

- Para $l = 1...(n-1)$
  - Para $i = 1...(n-l)$
    - $j = i + l$
    - Para todo $X \in N$, calcule

$$\pi(i, j, X) = \max_{\substack{X \to YZ \in R, \\ s \in \{i..(j-1)\}}} (\pi(i, s, Y) \times \pi(s+1, j, Z))$$

e

$$bp(i, j, X) = \arg \max_{\substack{X \to YZ \in R, \\ s \in \{i..(j-1)\}}} (\pi(i, s, Y) \times \pi(s+1, j, Z))$$

Saída: Retorna $\pi(1, n, S) = \max_{t \in T(s)} p(t)$, e os backpointers bp os quais permitem a recuperação do $\arg \max_{t \in T(s)} p(t)$.

Figura 3.1: *O algoritmo CKY [Collins,].*

A primeira parte do algoritmo encontra a probabilidade máxima (sempre 1) para cada terminal na sentença s, ou seja aplica o rótulo para cada palavra:

$$\pi(i, i, X) = \begin{cases} 1 & \text{se } X \to x_i \in R \\ 0 & \text{para outros casos} \end{cases}$$

O laços seguintes em l e i buscam combinações de maior probabilidade de palavras que possam ser rotuladas em conjunto. X representa os rótulos e $q(X \to YZ)$ são as regras de derivações que geram Y e Z partindo de X. Enquanto $\pi(i, j, X)$ guarda a probabilidade máxima das palavras de i à j serem "rotuladas" por X o $bp(i, j, X)$ guarda as regras bem como os índices s que foram usados para

a maximização, sendo que s representa a posição da palavra que separa Y de Z. O CKY retorna a probabilidade máxima de S e o bp.

Capítulo 4

Modelo de Transformação de LNC para SPARQL

O propósito deste trabalho é facilitar o acesso a informações em ontologias. O modelo proposto procura mitigar o problema de consulta em linguagem natural. A consulta utilizando o português controlado tem como objetivo proporcionar uma maior facilidade ao usuário, uma vez que aprender a construir sentenças usando um subconjunto da língua portuguesa é mais fácil que aprender uma nova linguagem. O modelo desenvolvido foi inspirado em outros trabalhos, porém todas as regras são originais e foram criadas a partir de experimentação.

Apresentamos um modelo pelo qual transformamos consultas em uma linguagem controlada para consultas SPARQL. O trabalho também propõe a construção de um português controlado que possa ser utilizado para realizar consultas a ontologias em língua portuguesa desde que esta ontologia seja adequada ao modelo. Além da linguagem controlada também utilizamos uma ontologia controlada. A necessidade de controlar a ontologia é esclarecida na seção 4.1.

LNC e LSPARQL são linguagens criadas para o modelo e são geradas pelas GLNC e GLSPARQL respectivamente. O modelo consiste dos seguintes procedimentos: análise da sentença em LNC, Matching e a geração do SPARQL.

De acordo com a Figura 4.1, no precedimento 1 temos uma ontologia $O1$ na qual todos seus elementos são extraídos utilizando o *framework Jena* (em nosso caso). Do outro lado, o usuário faz uma consulta em LNC $C1$, a consulta é analisada e a sentença rotulada (procedimento 2). Em seguida temos uma segunda análise dessa pergunta, orientada à elementos da ontologia (Matching, procedimento 3), em que buscamos uma relação dos substantivos da consulta com os termos da ontologia de busca. O Matching vai gerar uma linguagem intermediária. Então fazemos a análise sintática desta nova sentença utilizando o algoritmo CKY descrito na seção 3.2.1, onde geramos a árvore apropriada para construção do SPARQL (procedimento 4). No último passo usamos um algoritmo recursivo pós-ordem que converte a árvore gerada pelo CKY em uma consulta SPARQL válida.

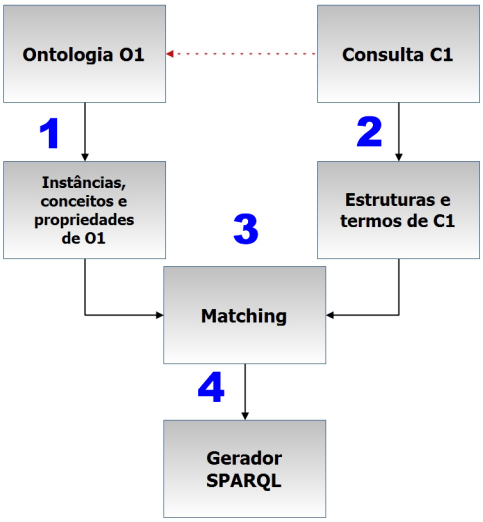


Figura 4.1: *visão do modelo*

4.1 Ontologia controlada

As ontologias podem ter diferentes tipos de estruturas e níveis de detalhamento. A flexibilidade na construção das ontologias dificulta uma interoperabilidade com a linguagem natural. Visto que, para o nosso modelo conversar com ontologias quaisquer escritas em língua portuguesa era uma tarefa difícil, a solução foi estabelecer normas para a construção de uma ontologia apropriada. Denominamos esta ontologia de "ontologia controlada".

As normas utilizadas na construção da nossa ontologia foram escolhidas para ampliar a interoperabilidade entre a ontologia e o nosso modelo. Para o desenvolvimento das regras de construção da ontologia controlada, procuramos investigar algumas ontologias escritas em língua portuguesa no objetivo de encontrar características comuns que contribuíam para interoperabilidade com nosso modelo.

Definimos aqui como ontologia controlada, uma ontologia que obedece as regras para criação de nomes de conceitos e propriedades.

Vamos usar abreviações e expressões regulares para simplificar a escrita das regras. A abreviação para substantivo comum é **Sub**, preposição é **Prep** e substantivo próprio é **Prop**. As regras para denominar os elementos são:

- Para dar nomes a classes e propriedades primitivas utilizamos a regra: (Sub Prep)* Sub. Por exemplo, podemos criar a classe **Tese de doutorado** e a propriedade primitiva **título**.
- Propriedades objetos devem ser criadas toda vez que se cria uma classe. A propriedade deve possuir o mesmo nome da classe, porém precedido da palavra **tem** concatenada com *underline*. Por exemplo, após criada a classe **Tese_de_doutorado**, deve-se criar a propriedade **tem_tese_de_doutorado**.
- Os nomes de indivíduos devem ser criados utilizando a regra: (Sub prep)* Prop. Por exemplo, podemos criar o indivíduo **tese_de_doutorado_do_Paulo** e também o indivíduo **Paulo**.

Na Figura 4.2 temos um exemplo da estrutura obtida utilizando as regras anteriores na construção de uma ontologia[1]. Temos uma ontologia que possui as classes **Pessoa** e **Tese_de_doutorado**, sendo **Paulo** uma instância de **Pessoa** e **tese_de_doutorado_do_Paulo** instância da classe **Tese_de_ Doutorado**. As propriedades neste caso são: **tem_tese_de_doutorado** que é do

[1]Utilizamos o software RDF Gravity v1.0 [Salzburg, 2004] para gerar a Figura 4.2 e outras seguintes

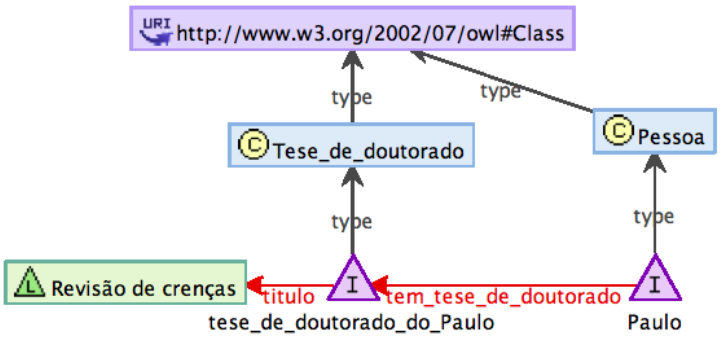


Figura 4.2: *Indivíduos e suas relações*

tipo objeto; e **titulo** que é uma propriedade primitiva. O fato de utilizarmos propriedades que começam com o verbo ter ou possuir, remete a uma limitação de expressividade[2].

É possível utilizar as regras descritas anteriormente para construir uma ontologia adequada a nossa convenção de nomes a partir de uma ontologia já existente. Com essa adaptação podemos ganhar tempo já que a construção de ontologias é um processo custoso. Porém algumas vezes pode ser inviável a adaptação isso vai depender do estado atual da ontologia.

4.2 Linguagem Controlada do Modelo

Estamos propondo consultas limitadas em diversos pontos, que serão discutidos nesta seção. As limitações com relação ao tipo de ontologia foram discutidas na seção 2.2.

4.2.1 Elementos da gramática GLNC

Aqui apresentaremos uma rápida descrição dos elementos terminais da nossa gramática.

- Verbos: limitamos a utilização dos verbos SER, TER e POSSUIR no tempo presente do indicativo da terceira pessoa do plural e do singular.
- Artigos: a, o, as, os, um, uma, uns, umas.
- Preposições: de, da, do, das, dos.
- Pronomes interrogativos: quem, qual, quais.
- Pronomes quantitativos: quantos, quantas.
- Conjunções: aditivas (e), disjuntivas (ou).
- Substantivos: são elementos que fazem parte do domínio de busca do usuário. A consulta terá sucesso caso os substantivos correspondam a elementos da ontologia.

4.2.2 Definição Formal da GLNC

Uma boa forma de limitar a estrutura da nossa linguagem é especificando vocabulário e regras de produção. A utilização de gramáticas permite uma construção infinita utilizando uma quantidade finita de termos [King, 1983]. Construímos uma gramática baseada em uma gramática livre de contexto (GLC). Seja $GLNC$ uma GLC tal que $GLNC = (V, \Sigma, R, S)$, onde $V = \{A, B, QP, VT, X, F,$

[2]Aqui consideramos expressividade de uma linguagem a habilidade da linguagem exprimir ou descrever algo. A "limitação de expressividade" acontece pelo fato de estarmos considerando apenas dois verbos distintos da língua portuguesa, haja visto que o português possui mais de sete mil verbos

$Z, T, U, W, QX, VQ, VS, QP, PQ, Prep, Prop, Sub\}$, Σ =$\{\lambda$, a, o, as, os, um, uma, uns, umas, de, da, do, das, dos, tem, possui, é, são, quantos, quantas, qual, quais, quem, $\{\Theta\}$, $\{\Gamma\}$ $\}$. `Prop` e `Sub` são termos relacionados ao domínio, `Prop` são substantivos próprios e `Sub` substantivos comuns.

Tabela 4.1: *Regras de produção R de GLNC*

S	→	QP VS A F X Prop |
		QP VS A X U |
		PQ Sub VT A B Prop |
		QX VS A Sub
		A B Prop VQ A B Sub |
U	→	T | W
T	→	T D Prep Prop | Prop
W	→	W C Prep Prop | Prop
X	→	X Sub Prep | Sub Prep
F	→	Y Prep | Z Prep $\|\lambda$
Z	→	Z Sub C A | Sub C A
Y	→	Y Sub D A | Sub D A
B	→	X $\|\lambda$
VQ	→	VT | VS
QX	→	QP | PQ
A	→	a | o | as | os | um | uma | uns | umas $\|\lambda$
C	→	e
D	→	ou
VS	→	é | são $\|\lambda$
VT	→	tem | possui
QP	→	qual | quais | quem
PQ	→	quantos | quantas
Prep	→	de | da | do | das | dos
Prop	→	$\{\theta \| \theta \in \Theta\}$
Sub	→	$\{\gamma \| \gamma \in \Gamma\}$

Tabela 4.2: *Siglas LNC*

Descrição	Sigla
Qualquer pronome interrogativo	QP
Pronome interrogativo QUANDO	PQ
Verbo SER	VS
Verbo TER ou POSSUIR	VT
Artigo	A
Conjunção	C
Disjunção	D
Substantivo comum	Sub
Substantivo próprio	Prop
Preposição	Prep
Conj. substantivos próprios	Θ
Conj. substantivos comuns	Γ
Palavra vazia	λ

A gramática geradora *GLNC* pode reconhecer diversos padrões diferentes de consultas e embora satisfaça o propósito do nosso trabalho, ainda pode ser ampliada e/ou refinada em trabalhos futuros.

4.2.3 Exemplos de perguntas

Para os nossos exemplos e teste, elaboramos perguntas em LNC dentro do contexto de duas ontologias de domínio público: A ontologia sobre conceitos de *smartphones* [Martins, 2007] que pode ser encontrada no site da ONTOLP[3]; e a ontologia sobre o currículo Lattes que foi utilizada no trabalho [da Costa and Yamate, 2009]. Ambas foram adaptadas para se adequarem ao nosso modelo.

Exemplos de perguntas que podem ser elaboradas com a gramática descrita na Tabela 4.1:

Ontologia sobre *smartphones*:

S1 - Qual é o smartphone do Fernando?
S2 - Qual é o modelo do celular da Maria?
S3 - Quais são os fabricantes de smartphone?
S4 - Qual é a resolução da camera do smartphone do Fernando?
S5 - Quantas câmeras tem o celular do Pedro?
S6 - O iphone da Maria tem conexão de dados wireless?

Ontologia do currículo Lattes:

L1 - Qual é o email do Paulo de Tarso?
L2 - Qual é o fone ou o email do Eduardo Galego?
L3 - Quantos orientandos de mestrado tem a Renata Wassermann?
L4 - Quantos orientandos de doutorado tem a orientadora do Fillipe Resina?
L5 - Paulo de Tarso é um estudante de doutorado?
L6 - Marcelo Finger tem publicações?

A Figura 4.3 e 4.4 ilustram uma visão parcial de como os elementos estão estruturados nas ontologias sobre *smartphones* e currículo Lattes, respectivamente.

4.2.4 Simplificando nossa LNC em uma Linguagem Regular

Para facilitar a visualização das transformações de nossa sentença em cada processo iremos transformar nossa LNC em uma linguagem regular. Conseguimos construir uma expressão regular para cada produção de S.

Primeiro, as derivações dos terminais viram conjunto de palavras: QP = { qual, quais, quem}, PQ = {quantos, quantas}, VS = { é, são, λ}, VT = {tem, possui, possuem}, A = {a,o,as,os, um, uma, uns, umas, λ}, C = {e}, D = {ou}, Prep = {de, da, do, das, dos}.

O segundo passo é construir expressões isoladas para as derivações não terminais:

A $Expr(F)$ é a expressão regular para a derivação `F` $\rightarrow$ `Y Prep | Z Prep` | λ, essa derivação trata o caso do uso das disjunções ou conjunções ou palavra vazia.

$$Expr(F) = (\texttt{Y Prep | Z Prep} \mid \lambda\ )$$

A $Expr(X)$ é a expressão regular para a derivação `X` $\rightarrow$ `X Sub Prep | Sub Prep`, essa derivação trata o caso de recursão de preposições "de".

$$Expr(X) = (\texttt{Sub Prep})^{+}$$

[3] http://www.inf.pucrs.br/ontolp/downloads.php

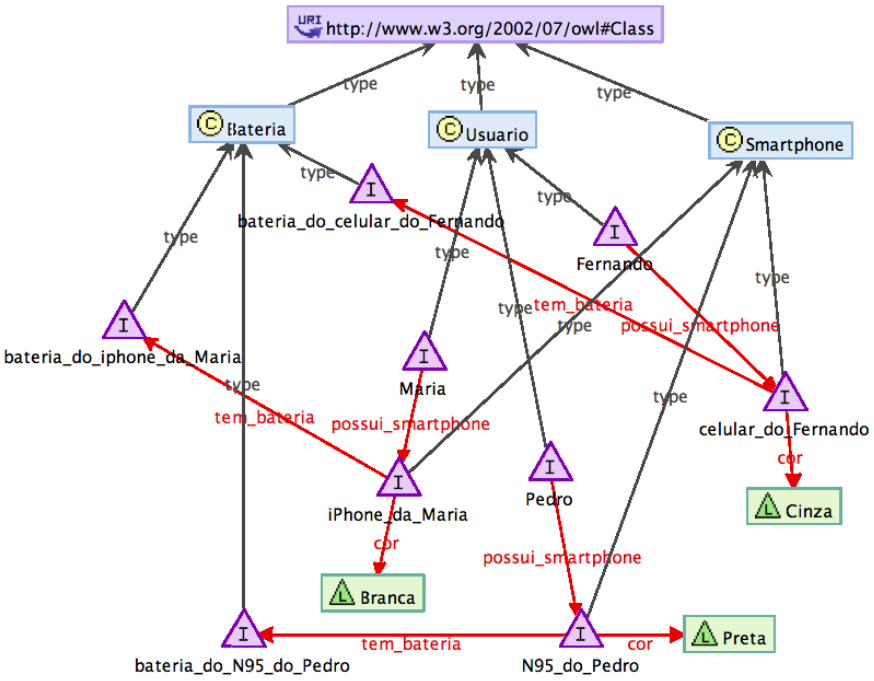


Figura 4.3: *Visão parcial da ontologia de smartphones*

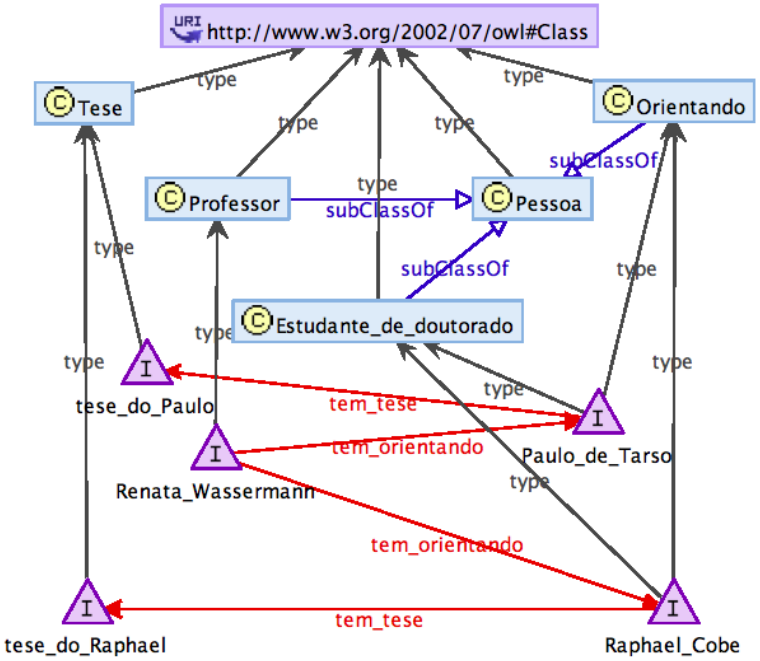


Figura 4.4: *Visão parcial da ontologia de currículo Lattes*

A $Expr(Y)$ é a expressão regular para a derivação `Y` → `Y Sub D A | Sub D A`, essa derivação trata o caso de recursão de disjunções.

$$Expr(Y) = (\texttt{Sub D A})^+$$

A $Expr(Z)$ é a expressão regular para a derivação `Z` → `Z Sub C A | Sub C A`, essa derivação trata o caso de recursão de conjunções.

$$Expr(Z) = (\texttt{Sub C A})^+$$

A $Expr(B)$ é a expressão regular para a derivação `B` → `X` | λ.

$$Expr(B) = (\texttt{X} \mid \lambda)$$

A $Expr(QX)$ é a expressão regular para a derivação `QX` → `QP | PQ`.

$$Expr(QX) = (\texttt{QP} \mid \texttt{PQ})$$

A $Expr(VQ)$ é a expressão regular para a derivação `VQ` → `VS | VT`.

$$Expr(VQ) = (\texttt{VS} \mid \texttt{VT})$$

A $Expr(T)$ é a expressão regular para a derivação `T` → `T D Prep Prop | Prop`.

$$Expr(T) = (\texttt{Prop DJ Prep})^*\texttt{Prop}$$

A $Expr(W)$ é a expressão regular para a derivação `W` → `T C Prep Prop | Prop`.

$$Expr(W) = (\texttt{Prop CJ Prep})^*\texttt{Prop}$$

Usando as expressões anteriores, construímos expressões regulares para cada produção de S. Por exemplo, para derivação `S` → `QP VS A F X Prop`, temos a expressão:

$$r_1 = Expr(\text{QP VS A F X Prop}) = \text{QP VS A } ((\text{Sub D A})^+|(\text{Sub C A})^+|\lambda)(\text{Sub Prep})^+\text{Prop}$$

E assim, analogamente:

$$r_2 = Expr(\text{PQ Sub VT A B Prop}) = \text{PQ Sub VT A } ((\text{Sub Prep})^+|\lambda)\text{Prop}$$
$$r_3 = Expr(\text{QP VS A X (T|W)}) = \text{QP VS A (P Prep)}^+((\text{Prop CJ Prep})^*|(\text{Prop DJ Prep})^*)\text{Prop}$$
$$r_4 = Expr(\text{QX VS A Sub}) = \text{(QP | PQ) VS A Sub}$$
$$r_5 = Expr(\text{A B Prop VQ A B Sub}) = \text{A}((\text{Sub Prep})^+|\lambda)\text{Prop (VT | VS) A}(\text{Sub Prep})^*\text{Sub}$$

Como as linguagens regulares são fechadas pela operação de união [Sipser, 2006] então a nossa LNC também é uma linguagem regular. Esta simplificação nos será útil para esclarecer os processos posteriores.

$$LNC = L(r_1) \cup L(r_2) \cup L(r_3) \cup L(r_4) \cup L(r_5)$$

Onde $L(r_i)$ é a linguagem gerada pela expressão regular i.

4.3 Identificando Termos

A identificação de termos é um procedimento anterior ao Matching, onde são identificados os elementos *(tokens)* que compõe uma consulta e depois verificado se a sentença está em nossa LNC. Para essa identificação utilizamos o corretor gramatical Cogroo 4.0 [Finger and Colen, 2012].

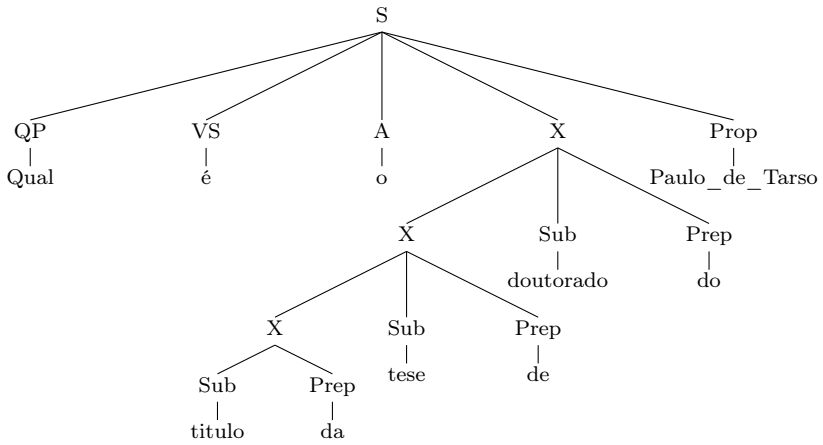
Através da análise sintática do Cogroo podemos por exemplo, identificar se um dado *token* é um substantivo. Posteriormente verificamos se existe uma relação entre os substantivos e os elementos da ontologia.

Na Tabela 4.3, vemos um exemplo da lista de *tokens* geradas pelo Cogroo dada a consulta "**Qual é o titulo da tese de doutorado do Paulo de Tarso?**"

Tabela 4.3: *Lista de tokens geradas pelo Cogroo*

Qual	=	pronome
é	=	verbo ser na 3ª pessoa do singular no presente do indicativo
o	=	artigo
titulo	=	substantivo comum
da	=	preposição de+a
tese	=	substantivo comum
de	=	preposição
doutorado	=	substantivo comum
do	=	preposição de+o
Paulo de Tarso	=	substantivo próprio

Árvore gerada pela LNC:



4.4 Matching

A semelhança semântica entre palavras tem sido alvo de estudo em recuperação e processamento de informação por muitos anos. Similaridade semântica tem importância em uma variedade de aplicações nas áreas de linguística computacional e inteligência artificial, tanto na comunidade acadêmica quanto na indústria [Li et al., 2003]. Como exemplo, podemos incluir: desambiguação, detecção e correção de erros de ortografia, etc.

De fato, encontrar semelhança entre palavras não é uma tarefa trivial [Barnard et al., 2003]. Em nosso trabalho, o matching se resume em correlacionar termos da pergunta em LNC com elementos da ontologia. Temos o auxílio de um dicionário de sinônimos e ferramentas lematizadoras, além de ferramentas que flexionam o gênero e grau dos substantivos. Embora nossa linguagem de consulta e ontologia sejam controladas, obtivemos erros inerentes a dificuldade de realização desta tarefa. Tais erros são descritos na Seção 5.2.1.

Em nosso modelo, o Matching recebe como entrada a pergunta feita em LNC com rótulos encontrados pelo Cogroo. Dentro do Matching temos o dicionário completo da ontologia que será consultada e a partir de então o algoritmo procura pelos termos da consulta dentro da ontologia controlada.

O processo define se a pergunta feita é factível de ser respondida, isto é, se o processo não conseguir encontrar uma correspondência entre os substantivos e os elementos da ontologia, a transformação da sentença em LNC em SPARQL não será possível.

Substantivos encontrados em nossa sentença LNC podem não estar grafados exatamente como na ontologia de busca. Neste caso existem algumas técnicas para mitigar o problema que foram implementadas em nosso protótipo. Um exemplo é a substituição por sinônimo, que pode ser visto na consulta: **"Quantos alunos tem o IME?"**. A palavra "alunos" não está na ontologia, porém identificamos lá seu sinônimo "estudante", assim o termo aluno da sentença é correspondente ao termo estudante da ontologia.

Caso exista a correspondência entre todos os substantivos, o Matching vai garantir uma sentença de saída que sempre pode ser reconhecida pela nossa gramática GLSPARQL. Caso contrário, o Matching para a transformação, isso significa que os termos (pergunta x ontologia) não estão relacionados. Ressalta-se que o mais importante no contexto do nosso trabalho é especificar a função do Matching.

4.4.1 Dicionário da Ontologia

Chamamos de dicionário da ontologia a lista com todos os termos e a classificação deles dentro da nossa ontologia controlada. Assim, sabemos de antemão a classificação e nome de cada elemento presente em nossa ontologia. Por exemplo, na Tabela 4.4 temos uma ilustração parcial da ontologia currículo Lattes.

Descrição	Classificação
Professor	Classe
José_Silva	Indivíduo
tem_professor	propriedade objeto
nome	propriedade primitiva

Tabela 4.4: *Dicionário da ontologia*

4.4.2 Descrição do processo

Dado uma entrada, que trata-se de uma sentença rotulada de acordo com nossa GLNC e uma ontologia controlada, o Matching é um processo que busca correspondência entre termos da ontologia e da sentença em LNC. Quando os termos são encontrados, o Matching classifica cada termo como: classe, substituindo o termo por **_Class_**; propriedade primitiva, substituindo o termo por **_Data_**, propriedade objeto, fazendo a substituição por **_Object_** e indivíduos onde o termo é substituido por **_Individual_**.

Classes e propriedades primitivas podem ser identificadas na sentença em LNC pelo padrão: **(substantivo comum + preposição)*** + **substantivo comum**, de acordo com nossa ontologia controlada[4]. Como existe uma propriedade objeto para cada classe, ignora-se a busca por propriedades objetos. Depois decidimos se vamos utilizar a classe ou propriedade objeto, no caso de utilizar a propriedade objeto, apenas adicionamos a palavra **tem_** na frente do nome da classe. Os indivíduos pertencentes a ontologia controlada são denominados pelo padrão: **(substantivo comum + preposição)*** + **substantivo próprio**, assim podemos localiza-los na sentença em LNC.

O Matching usa a palava **_Term_** quando identifica uma classe, pois, para cada classe existe uma propriedade objeto correspondente. A utilização de uma ou outra vai depender da estrutura da pergunta. Então, o Matching vai substituir a palavra **_Term_** por **_Object_** ou **_Class_** de acordo com o *parsing* que identifica as estruturas. **_Term_** ligados por conjunções ou disjunções são considerado grupos e devem possuir substituição idêntica a substituição do **_Term_** anterior. As regras para a substituição são:

- Regra 1: Quando que um **_Term_** ou um grupo de **_Term_** é ligado a um indivíduo através de uma preposição, **_Term_** é substituido por **_Object_**.
- Regra 2: Quando que a palavra **_Term_** ou um grupo de **_Term_** é ligado a um indivíduo através da palavra **tem** ou **possui**, **_Term_** será substituído por **_Object_**
- Regra 3: Caso as situações anteriores não ocorram, a palavra **_Term_** é substituída pela palavra **_Class_**.

Por exemplo, na consulta "**Qual é o título da tese de doutorado do Paulo**", identificamos durante o Matching, **título** como uma propriedade primitiva, **tese_de_doutorado** uma classe e **Paulo** um indivíduo. Depois da identificação, o Matching gera: "**Qual é o _Data_ do _Term_ do _Individual_**". Aqui temos somente um **_Term_** e este está ligado ao indivíduo, logo, aplicamos a regra 1. A nova frase resulta em: "**Qual é o _Data_ do _Object_ do _Individual_**". É importante saber que o Matching mantém uma associação das novas palavras com os termos encontrados na ontologia, ou seja, **_Data_** está associada com **título**, **_Object_** está associada com a propriedade **tem_tese_de_doutorado** e **_Individual_** está associada à **Paulo**. As palavras originais encontradas na ontologia serão utilizadas na etapa final da geração do SPARQL.

Após esse processo, o Matching gera uma nova sentença, que é reconhecível pela GLSPARQL, como veremos na Seção 4.6.

[4]onde asterisco significa que pode existir zero ou n vezes

4.4.3 Nova linguagem gerada pelo Matching

É importante observar que a linguagem resultante das transformações do Matching é mais restrita, pois ela é resultado de uma reclassificação da LNC. Substantivos podem ser representados por classes, propriedades primitiva ou propriedades objeto da ontologia. No caso de classes e propriedades objeto, o uso de uma ou outra na geração da sentença em LNC_M vai depender de quais palavras antecedem ou sucedem este substantivo. Exemplo:

Quantos celulares tem o Fernando?

A palavra **celular** é uma classe em nossa ontologia sobre *smartphones*, que está acompanhada da propriedade objeto **tem_celular**. Após a aplicação das regras vista na Seção 4.4.2 obtemos:

Quantos _Object_ tem o _Individual_

Então, algumas construções passam a ser impossíveis, como por exemplo:

Quantos _Class_ tem o _Individual_

Portanto, a partir deste ponto não podemos considerar que classes e propriedades pertençam ao mesmo conjunto. Então, pelas regras de transformação do Matching, podemos assumir que `(Sub Prep)* Sub` são transformados em **_Class_**, **_Object_** ou **_Data_** e `(Sub Prep)* Prop` viram **_Individual_**, denominamos as propriedades de **P**, as classes de **CLA** e os indivíduos de **I**.

Após o Matching, a expressão regular r_1 vista na Seção 4.2.4 será transformada na expressão r_{M1} somente substituindo **Sub** por **P** ou **CLA** e Prop por **I**.

$$r_{M1} = \text{QP VS A } ((\text{P D A})^+|(\text{P C A})^+|\lambda)(\text{P Prep})^+ \text{ I}$$

O mesmo acontece com as expressões $r2$, $r3$ e $r4$.

$$r_{M2} = \text{QP VS A (P Prep)}^+((\text{ I CJ Prep})^*|(\text{ I DJ Prep })^*) \text{ I}$$

$$r_{M3} = \text{PQ P VT A (P Prep)}^* \text{ I}$$

$$r_{M4} = (\text{QP} \mid \text{PQ}) \text{ VS A CLA}$$

No Matching acontece a inclusão da palavra _Ask_ nas sentenças nas quais não são encontrados pronomes interrogativos. A expressão r_{M5} irá incluir _Ask_.

$$r_{M5} = \text{\_Ask\_ A(P Prep)}^* \text{ I (VT A P} \mid \text{VS A CLA)}$$

O Matching gera uma nova frase que pode ser reconhecida pela GLSPARQL. Abaixo a composição da nossa LNC_M.

$$LNC_M = L(r_{M1}) \cup L(r_{M2}) \cup L(r_{M3}) \cup L(r_{M4}) \cup L(r_{M5})$$

4.5 GLSPARQL para a geração do SPARQL

Construímos duas gramáticas. A primeira, a GLNC permite a geração das perguntas com mais facilidade. A segunda, GLSPARQL permite a construção do SPARQL a partir da análise sintática. O principal motivo para a construção de duas gramáticas deve-se ao fato de que a primeira gramática tem intenção de ser independente da análise da ontologia. Já na segunda gramática, o *parsing* é

feito usando elementos da ontologia. O que são substantivos na primeira gramática, na segunda gramática são elementos para a construção do SPARQL.

Nas seções seguintes desenvolvemos nossa GLC denominada GLSPARQL que será utilizada nos passos seguintes. Nossa GLSPARQL foi cuidadosamente desenvolvida na tentativa de diminuir a ambiguidade e complexidade, mantendo o propósito de gerar estruturas SPARQL a partir da sua árvore sintática.

4.5.1 Definição Formal

Seja $GLSPARQL$ uma GLC tal que $GLSPARQL = (V, \Sigma, R, S)$, onde V ={ E1, E2, E3, T1, T2, T3, T4, T5, T6, T7, VS, VT, QP, PQ, VTP, PO, PD, PF, IND, CLA, VSCLA, Prep, A, PVT, CJ, PFCJ, DJ, PFDJ, L, CJP, CJPI, DJP, DJPI}, Σ ={é, a, o, as, os, um, uma, uns, umas, são, tem, possui, possuem, qual, quais, quem, quantos, quantas, de, da, do, das, dos, e, ou, _Ask_, _Object_, _Data_, _Class_, _Individual_, λ }

Tabela 4.5: *Regras de derivação dos não terminais de GLSPARQL*

S	→	E1	T1
S	→	E1	T2
S	→	E1	T3
S	→	E1	T4
S	→	E1	T5
S	→	E1	T6
S	→	E1	T7
S	→	E2	T1
S	→	E2	T2
S	→	E3	T1
S	→	E3	T2
S	→	E3	T5
E1	→	QP	VS
E1	→	QP	L
T1	→	T1	VTP
T1	→	PF	IND
T1	→	PF	T1
T1	→	PVT	IND
T1	→	PVT	T1
T2	→	IND	VSCLA
T2	→	VS	CLA
T2	→	T1	VSCLA
T3	→	PFCJ	T1
T3	→	PFCJ	T3
T4	→	PFDJ	T1
T4	→	PFDJ	T4
T5	→	IND	VTP
T6	→	T6	DJPI
T6	→	T1	DJPI
T7	→	T7	CJPI
T7	→	T1	CJPI
VTP	→	VT	PO
VTP	→	VT	PD
PVT	→	PD	VT
PVT	→	PO	VT
VSCLA	→	VS	CLA
PFDJ	→	PD	DJ
PFDJ	→	PO	DJ
PFCJ	→	PD	CJ
PFCJ	→	PO	CJ

PF	→	PO	Prep
PF	→	PD	Prep
CJPI	→	CJP	IND
DJPI	→	DJP	IND
CJP	→	CJ	Prep
DJP	→	DJ	Prep
IND	→	A	IND
CLA	→	A	CLA
PO	→	A	PO
PD	→	A	PD

Tabela 4.6: *Regras de derivação dos terminais de GLSPARQL*

VS	→	é | são
VT	→	tem | possui | possuem
QP	→	qual | quais | quem
E2	→	quantos | quantas
E3	→	_Ask_
Prep	→	de |da|do|das|dos
A	→	a|o|as|os|um|uma|uns|umas
CJ	→	e
DJ	→	ou
PO	→	_Object_
PD	→	_Data_
CLA	→	_Class_
IND	→	_Individual_
L	→	λ

4.6 LNC_M é um subconjunto de LSPARQL

Nos passos seguintes, iremos mostrar que a GLSPARQL gera pelo menos a LNC_M. A ideia é que a linguagem gerada por cada expressão regular que compõe LNC_M também pode ser gerada utilizando n passos de derivação de $GLSPARQL$. Para simplificar chamamos _Class_, de **CLA**, _Object_ e _Data_ de P e _Individual_ de I. Faremos um abuso da notação para mostrar que a partir de S podemos gerar $L(r_{M1})$, $L(r_{M2})$, $L(r_{M3})$, $L(r_{M4})$ e $L(r_{M5})$. Então, temos:

- Derivações para gerar r_{M1}:

$$S \Rightarrow E1\ T1$$

E1 pode ser derivado para:

$$E1 \Rightarrow QP\ VS$$
$$E1 \Rightarrow QP\ \lambda$$

T1 pode ser derivado para:

$$T1 \Rightarrow PF\ T1$$
$$T1 \Rightarrow PF\ IND$$

PF pode ser derivado para:

$$PF \Rightarrow PO\ Prep \Rightarrow \text{\_Object\_ Prep}$$
$$PF \Rightarrow PD\ Prep \Rightarrow \text{\_Data\_ Prep}$$

Como T1 é recursivo, podemos gerar:

$$T1 \Rightarrow^* (PO\ Prep)^+\ IND$$
$$T1 \Rightarrow^* (PD\ Prep)^+\ IND$$

$$T1 \Rightarrow^* (P\ Prep)^+\ I \tag{4.1}$$

Então, com as devidas substituições, n derivações partindo de $E1\ T1$ pode gerar:

$$S \Rightarrow E1\ T1 \Rightarrow^* (QP(\lambda|VS))(\text{P Prep})^+\text{I} \tag{4.2}$$

Analogamente, temos:

$$S \Rightarrow E1\ T3 \Rightarrow^* (QP(\lambda|VS))(\text{P C (A}|\lambda))^+((\text{P prep})^+\text{I}) \tag{4.3}$$
(4.4)

$$S \Rightarrow E1\ T4 \Rightarrow^* (QP(\lambda|VS))(\text{P D (A}|\lambda))^+((\text{P prep})^+\text{I}) \tag{4.5}$$

Como **P** pode ser derivado para "P→A P", logo podemos gerar:

$$4.2 + \ 4.3 + \ 4.5 = \text{QP VS A ((P D A)}^+|(\text{P C A})^+|\lambda)(\text{P Prep})^+\text{I} = r_{M1}$$

- Derivações para gerar r_{M2}:

T6 pode ser derivado para:

$$T6 \Rightarrow T1\ DJPI$$
$$T6 \Rightarrow T6\ DJPI$$

T7 pode ser derivado para:

$$T7 \Rightarrow T1\ CJPI$$
$$T7 \Rightarrow T7\ CJPI$$

DJPI pode ser derivado para:

$$DJPI \Rightarrow DJP\ IND$$

CJPI pode ser derivado para:

$$CJPI \Rightarrow CJP\ IND$$

DJP e CJP podem ser derivados para:

$$CJP \Rightarrow CJ\ Prep$$
$$DJP \Rightarrow DJ\ Prep$$

Como T6 e T7 são recursivos, podendo ser derivados para T1 que também é recursivo, podemos derivar para as seguintes construções:

Então, com as devidas substituições, n derivações partindo de $E1\ T6$ pode gerar:

$$S \Rightarrow E1\ T6 \Rightarrow^* \text{QP VS (P Prep)}^+\text{(I DJ Prep)}^+ \text{ I} \tag{4.6}$$

E partindo de $E1\ T7$ podemos gerar:

$$S \Rightarrow E1\ T7 \Rightarrow^* \text{QP VS (P Prep)}^+\text{(I CJ Prep)}^+ \text{ I} \tag{4.7}$$

Podemos gerar:

$$4.6 + \ 4.7 = \text{QP VS A (P Prep)}^+\text{((I CJ Prep)}^+\text{|(I DJ Prep)}^+\text{) I } = r_{M2}$$

- Derivações para gerar r_{M3}:

Para r_{M3}, temos um caso distinto, o E2, que deriva para os terminais "**quantos**" e "**quantas**", portanto a regra E2 é equivalente a regra PQ da Seção 4.2. Aqui, iremos utilizar outras derivações de T1:

T1 também pode ser derivado para:

$$T1 \Rightarrow PVT\ T1$$
$$T1 \Rightarrow PVT\ IND$$

PVT pode ser derivado para:

$$PVT \Rightarrow PO\ VT$$

Então, com as devidas substituições, n derivações partindo de $E1\ T1$ pode gerar:

$$S \Rightarrow E2\ T1 \Rightarrow^* \text{PQ P VT A (P prep)}^* \text{ I } = r_{M3}$$

- Derivações que geram r_{M4}:

Para r_{M4}, temos dois não terminais distintos o E2 e T2, E2 deriva para os terminais "**quantos**" e "**quantas**", assim a regra E2 é equivalente a regra PQ da Seção 4.2. T2 pode ser derivado para:

$$T2 \Rightarrow \text{ VS CLA}$$

O não terminal CLA pode ser derivado para:

$$CLA \Rightarrow \text{ A CLA}$$

Então partindo de T2 podemos gerar:

$$T2 \Rightarrow^* \text{ VS A CLA}$$

Então, com as devidas substituições, n derivações partindo de $E2$ $T2$ pode gerar:

$$S \Rightarrow E2\ T2 \Rightarrow^* \text{PQ VS A CLA}$$

Partindo de $E1$ $T2$ pode gerar:

$$S \Rightarrow E1\ T2 \Rightarrow^* \text{ QP VS A CLA}$$

As duas derivações anteriores são equivalentes à:

$$r_{M5} = \text{(QP | PQ) VS A CLA}$$

- Derivações para gerar r_{M5}:
  Outras derivações para T1 são:

$$T1 \Rightarrow \text{T1 VTP}$$
$$VTP \Rightarrow \text{VT P}$$

Então, com as devidas substituições, n derivações partindo de $E3$ $T1$ pode gerar:

$$S \Rightarrow E3\ T1 \Rightarrow^* \text{ \_Ask\_ A (P Prep)* I VT A P} \tag{4.8}$$

n derivações partindo de $E3$ $T5$ pode gerar:

$$S \Rightarrow E3\ T5 \Rightarrow^* \text{ \_Ask\_ A I VS A CLA} \tag{4.9}$$

n derivações partindo de $E3$ $T2$ pode gerar:

$$S \Rightarrow E3\ T2 \Rightarrow^* \text{ \_Ask\_ A (P Prep)* I VS A CLA} \tag{4.10}$$

As três construções geram:

$$\text{4.8+ 4.9+ 4.10} = \text{\_Ask\_ A(P Prep)* I (VT A P | VS A CLA)} = r_{M5}$$

Como visto nos passos anteriores, nossa GLSPARQL gera r_{M1}, r_{M2}, r_{M3}, r_{M4} e r_{M5}. Portanto ela gera pelo menos a nossa linguagem LNC_M, isto é $LNC_M \subseteq LSPARQL$. O importante neste ponto é compreender que toda sentença em LNC_M é factível de ser interpretada pelo algoritmo CKY descrito na Seção 3.2.1, já que o CKY utiliza as regras da GLSPARQL durante a análise sintática.

4.7 Ambiguidades inerentes ao modelo

4.7.1 Ambiguidade em Gramáticas Livre de Contexto

A ambiguidade em uma linguagem é quando um símbolo ou uma expressão possui mais do que um sentido (por exemplo, a palavra: bota em língua portuguesa), ou quando uma expressão pode ser gramaticalmente analisada de duas maneiras diferentes, ou seja, quando existem duas árvores de derivação diferentes para a mesma sentença [Basten, 2007]. O primeiro caso é chamado ambiguidade léxica, e o segundo caso, ambiguidade sintática.

O problema de detecção de ambiguidades é indecidível para o caso geral [Schmitz, 2007] e [Brabrand et al., 2010]. Não existe um algoritmo genérico que possa identificar uma gramática ambígua. É possível analisar uma GLC simples para decidir se é ambígua ou não, e se for ambígua, convertê-la em uma gramática não ambígua. No entanto, como mencionado anteriormente, a maioria dos problemas relativos à ambiguidade da GLC, em geral, são muito difíceis ou mesmo insolúveis. Sendo a detecção de ambiguidades em GLC para o caso geral uma tarefa impossível, quaisquer tentativas de resolver o problema são baseadas em aproximações [Schmitz, 2007].

Também não é possível encontrar todas as ambiguidades individuais em uma gramática de recursão infinita. Uma abordagem seria simplesmente começar a procurar um número de sentenças possíveis, mas para uma gramática de linguagem infinita isso nunca acabaria. Quando a busca é interrompida após a execução de um determinado período de tempo, ainda podem haver ambiguidades desconhecidas não verificadas [Basten, 2007].

4.7.2 Avaliando ambiguidades no modelo

Com relação a ambiguidades léxicas, nossa GLNC é dependente da ontologia, pois o conjunto de substantivos que podem ser utilizados nas consultas são palavras pertencentes a ontologia. Na GLNC estamos sujeitos a todas ambiguidades presentes nas ontologias. Na segunda gramática, a GLSPARQL, não existem ambiguidades léxicas pois cada palavra possui um único significado.

Dado que o problema de detectar ambiguidade sintática em uma GLC é indecidível para o caso geral, para verificar a ambiguidade sintática de nossa gramática fixamos um tamanho máximo de sentença, em 11 palavras (a maior frase usada nos testes). Com um uso da ferramenta NLTK 2.0[5], geramos todas as sentenças possíveis de até 11 palavras. Após a geração verificamos que nossa GLSPARQL não contem ambiguidade sintática para o conjunto gerado.

O CKY está preparado para o tratamento de ambiguidades sintáticas, ele pode fazer *parsing* de uma Gramática Livre de Contexto Probabilística (GLCP) que é gerada por um Treebank [Wallis, 2008]. Um Treebank é um corpus de texto em que cada frase já foi analisada, ou seja, anotada com uma estrutura sintática. A necessidade de usar uma GLCP está relacionada com a ampliação da capacidade gerativa da gramática. Atestamos que as ambiguidades sintáticas não são um problema para nosso modelo atualmente. Outros tipos de ambiguidades como ambiguidades semânticas são discutidas na seção 5.2.

4.8 Gerando o SPARQL

Caso o processo de Matching obtenha êxito, uma nova sentença em LNC_M será gerada e submetida ao CKY. O algoritmo de análise sintática recebe como parâmetros de entrada a sentença em

[5] http://nltk.org/

LNC_M e as regras da GLSPARQL e gera uma árvore de derivação. Nesta seção, iremos detalhar o processo no qual nosso modelo recebe a árvore gerada pelo nosso algoritmo CKY e a transforma em uma consulta SPARQL.

4.8.1 Relacionando $GLSPARQL$ com SPARQL

Estruturas básicas

As estruturas básicas são denominadas E's em nosso modelo. Elas relacionam partes da regra de produção com elementos básicos do SPARQL, ou seja, estruturas de seleção booleana ou de contagem. No caso da estrutura booleana, a relação deve-se pela ausência do pronome interrogativo. A Tabela 4.7 exibe as estruturas SPARQL para cada entrada LNC_M.

Tabela 4.7: *Tipos de estruturas*

Rótulo	Entrada	Saída	Descrição
E1	qual quem quais	SELECT ?X	Seleção
E2	quantos quantas	SELECT Count(?X)	Contagem
E3	_Ask_	ASK	Booleana

Definindo mapeamento de triplas RDF

Descrevemos alguns tipos de composição de triplas RDFs que serão utilizadas durante o mapeamento. As triplas T1 descrevem as triplas que podem ser criadas quando em LNC nós usamos termos ligados entre si pela preposição "de". T2 são utilizadas quando precisamos checar se um indivíduo pertence a determinada classe ou recuperar indivíduos de uma determinada classe. As triplas T3 e T4 tratam triplas que podem ser criadas quando utilizamos conjunções e/ou disjunções de propriedades. A tripla T5 trata o caso de aparecer T2 à direita de T1. As T6 e T7 são responsáveis pela conjunção e disjunção de indivíduos, respectivamente. Na Tabela 4.8 detalhamos entradas e saídas.

Tabela 4.8: *Tipo de triplas*

Nome	Entrada	Tripla de saída
T1	PO Prep IND PD Prep IND PO Prep T1 PD Prep T1	p:_Individual_ p:_Object_ ?A p:_Individual_ p:_Data_ ?A ?T1 p:_Object_ ?A ?T1 p:_Data_ ?A
T2	IND VS CLA VS CLA	p:_Individual_ r:type p:_Class_ ?A r:type p:_Class_
T3	PD C T1 PO C T1	p:_Individual_ p:_Data_ ?A p:_Individual_ p:_Object_ ?A
T4	PD D T1 PO D T1	p:_Individual_ p:_Data_ ?A p:_Individual_ p:_Object_ ?A
T5	T1 T2	?T1 ?B ?T2
T6	T1 DJP IND T6 DJP IND	p:_Individual_ p:_Property_ ?A p:_Individual_ p:_Property_ ?A
T7	T1 CJP IND T7 CJP IND	p:_Individual_ p:_Property_ ?A p:_Individual_ p:_Property_ ?A

A Palavra **_Property_** é usada para indicar que tanto **_Object_** quanto **_Data_** podem ser utilizadas para gerar a tripla. O uso de uma ou outra vai depender da última tripla que foi gerada. T4 e T6 também produzem a estrutura **{**T_{new}**}UNION{**T_{old}**}**, onde T_{new} significa a tripla que foi gerada pela T6 e T_{old} a tripla anteriormente produzida.

4.8.2 Árvores e interpretações

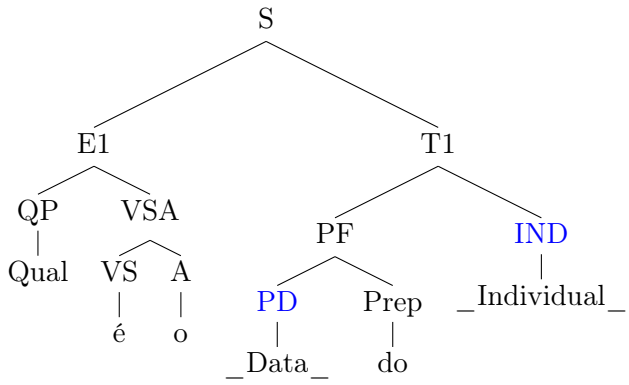
Nesta seção nós discorremos sobre alguns tipos de árvores de derivação possíveis, explicando construções importantes. A presente seção exibe exemplos compostos por uma consulta x em LNC, x em LNC_M, a árvore gerada pelas regras da $GLSPARQL$ a partir de x *(bottom-up)* e a saída dos elementos **E's** e **T's**.

Preposições

As triplas do tipo T1 tratam substantivos (propriedades e Indivíduos) ligados pela preposição "de" e suas flexões. No momento da geração das triplas existem transferência de variáveis entre uma tripla e outra, essa transferência é determinada pelas regras da Tabela 4.8. A ordem de criação de cada tripla é da direita para a esquerda, resolvendo-se a partir das folhas. A produção da tripla está relacionada com quem são os filhos das **T's**, por exemplo, **T1**'s podem dar origem a triplas diferentes.

- Caso 1: T1 depois de PF e IND

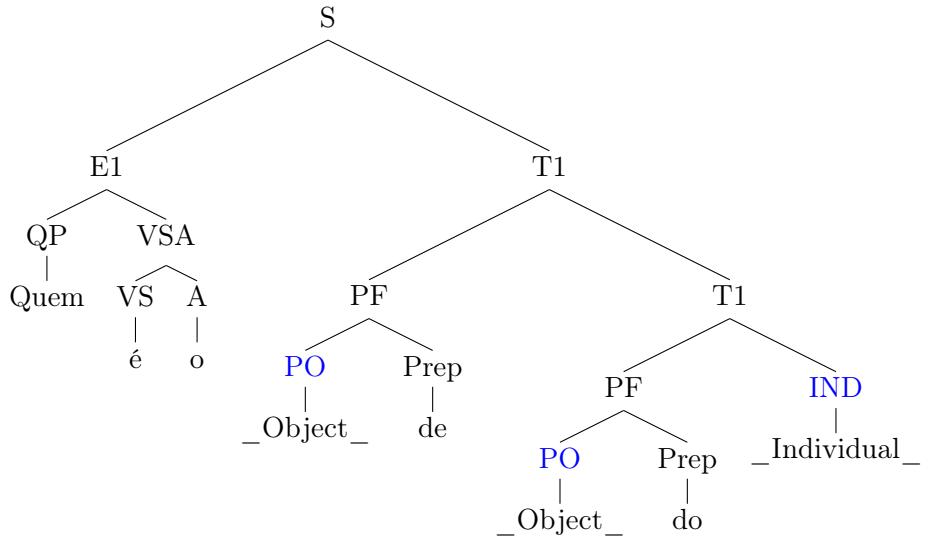
  1. LNC: **qual é o telefone do IME?**
  LNC_M: qual é o _Data_ do _Individual_



  `E1` : estrutura básica SPARQL `SELECT ?A.`
  `T2` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A}

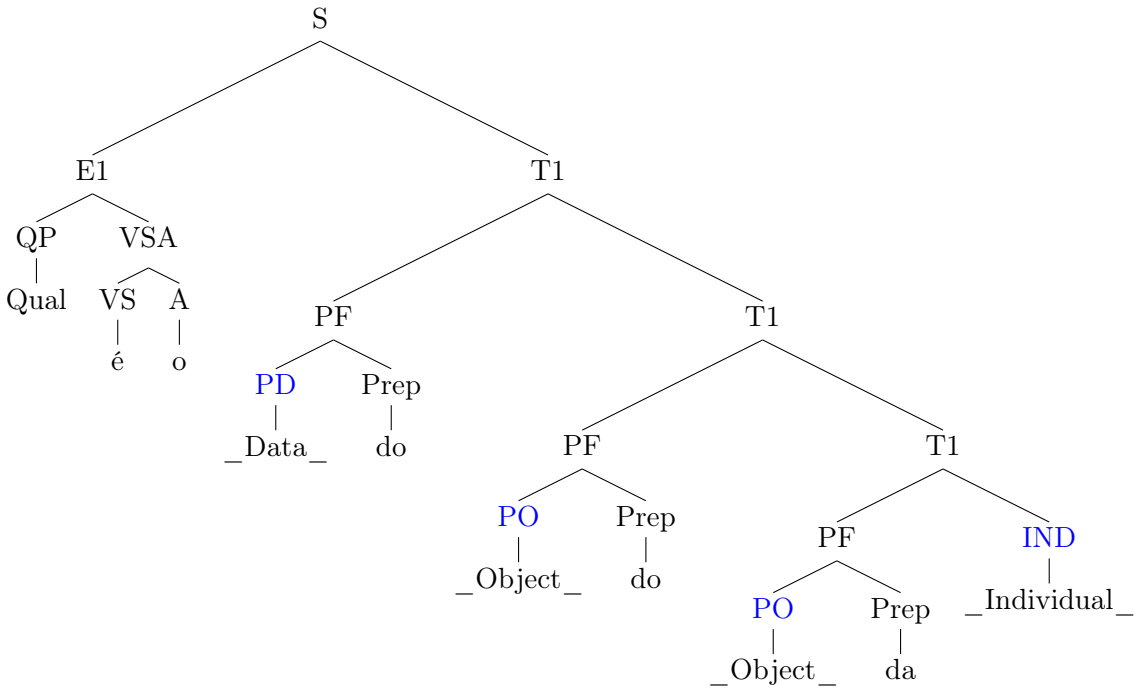
- Caso 2: T1 depois de T1 e PF.

  2. LNC: **quem é o diretor de pós-graduação do IME?**
  LNC_M: quem é o _Object_ de _Object_ do _Individual_



`E1` : estrutura básica SPARQL `SELECT ?B.`
`T1` : é a tripla SPARQL { p:_Individual_ p:_Object_ ?A }
`T1` : é a tripla SPARQL { ?A p:_Object_ ?B }

- Caso 3: Casos recursivos.
  3. LNC: **qual é o email da secretária do reitor da USP?**
  LNC_M: qual é o _Data_ do _Object_ do _Object_ da _Individual_

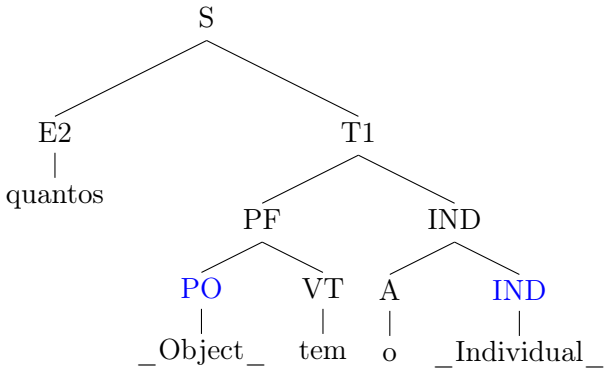


`E1` : estrutura básica SPARQL `SELECT ?E.`
`T1` : é a tripla SPARQL { p:_Individual_ p:_Object_ ?A}
`T1` : é a tripla SPARQL { ?A p:_Object_ ?B }
`T1` : é a tripla SPARQL { ?B p:_Data_ ?C }

Perguntas quantitativas

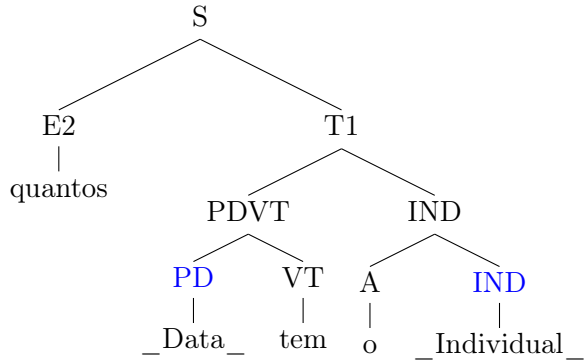
Estas estruturas não se diferenciam muito das estruturas anteriores, exceto pela parte E da sentença que agora possui um contador.

4. LNC: **quantos acessórios tem o E63?**
LNC_M: quantos _Object_ tem o _Individual_



`E2` : estrutura básica SPARQL `SELECT Count(?A).`
`T1` : é a tripla SPARQL { p:_Individual_ p:_Object_ ?A }

5. LNC: **quantos emails tem o Fabiano?**
LNC_M: quantos _Data_ tem o _Individual_

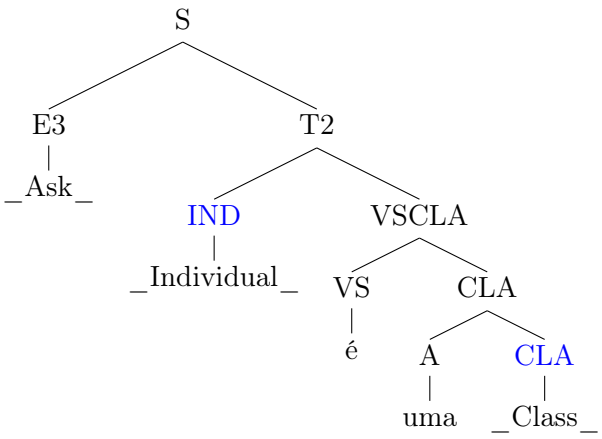


`E2` : estrutura básica SPARQL `SELECT Count(?A).`
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }

Perguntas booleanas

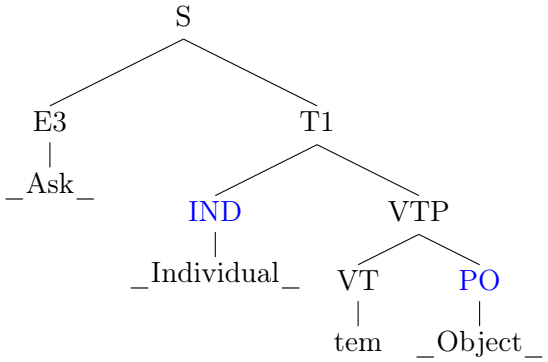
Perguntas booleanas são aquelas que esperam como resposta uma negação ou uma afirmação. Em nossa gramática identificamos estas perguntas como sendo aquelas que não possuem pronome interrogativo.

6. LNC: **A USP é uma universidade ?**
LNC_M: _Ask_ _Individual_ é uma _Class_ ?



`E3` : estrutura básica SPARQL `ASK`.
`T2` : é a tripla SPARQL { p:_Individual_ r:type p:_Class_ }

7. LNC: **Fabiano tem orientador?**
LNC_M: _Ask_ _Individual_ tem _Object_?



`E3` : estrutura básica SPARQL `ASK`.
`T1` : é a tripla SPARQL { p:_Individual_ p:_Object_ ?A }

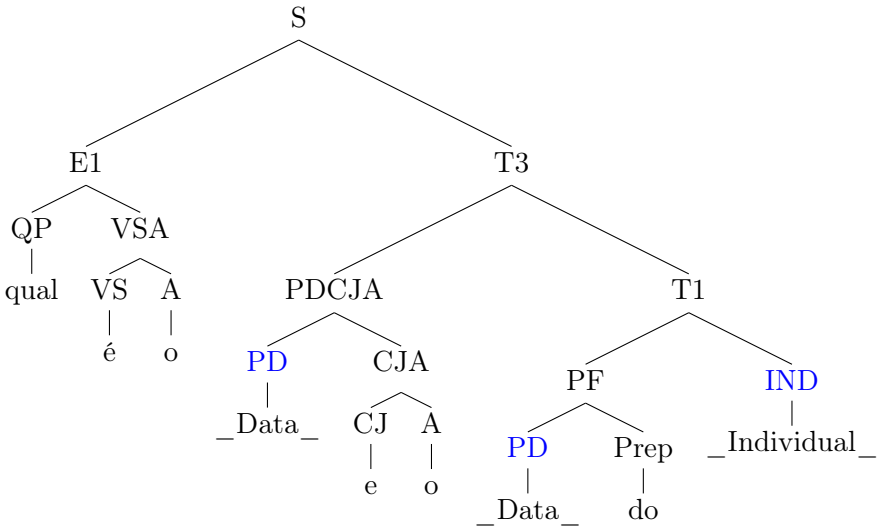
Conjunções e disjunções de Propriedades

Nesta seção abordaremos conjunções, disjunções e padrões mistos, bem como a recursão destes casos. T3 e T4 tratam conjunções e disjunções, respectivamente. Neste caso perceba que não há transferência de valores entre uma tripla e outra e são utilizadas mais que uma variável na resposta global.

Conjunção e disjunção não podem aparecer na mesma sentença, isso gera ambiguidade. T1 e T2 podem aparecer na parte inferior da árvore, ou seja, T1 e T2 devem ser sub-árvores de T3 ou T4.

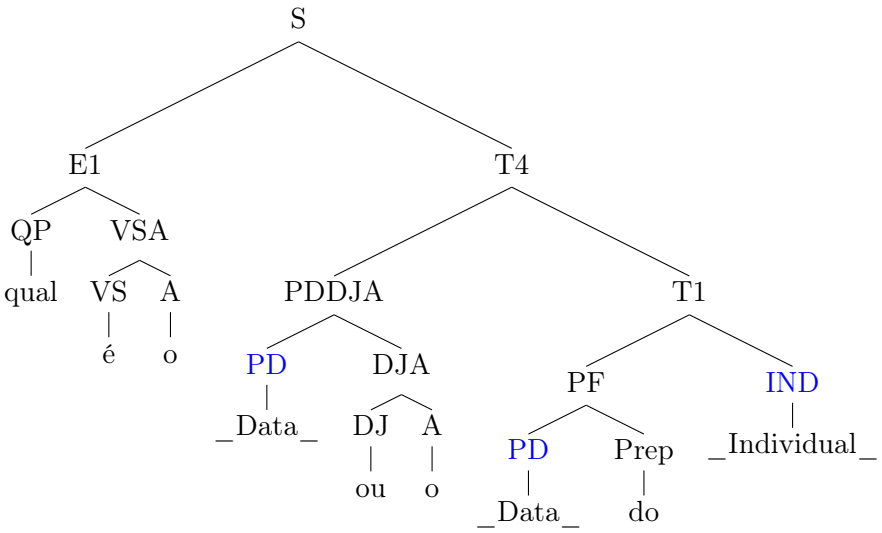
- Caso 1: Conjunção.

  8. LNC: **qual é o email e telefone do Fabiano?**
  LNC_M: qual é o _Data_ e o _Data_ do _Individual_?



`E1` : estrutura básica SPARQL `SELECT *`.
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
`T3` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?B }

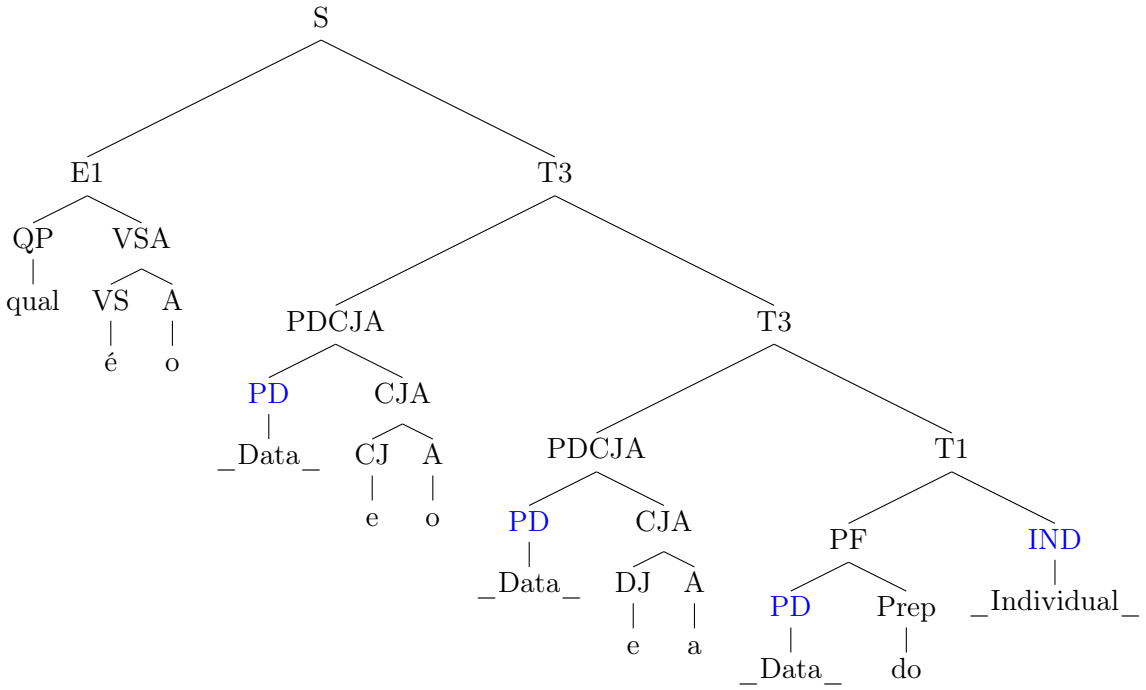
- Caso 2: Disjunção.
  9. LNC: **qual é o endereço ou o telefone do Paulo?**
  LNC_M: qual é o _Data_ ou o _Data_ do _Individual_?



`E1` : estrutura básica SPARQL `SELECT *`.
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
T1 e T4 são unidas por UNION
`T4` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?B }

- Caso 3: Recursão de conjunção.

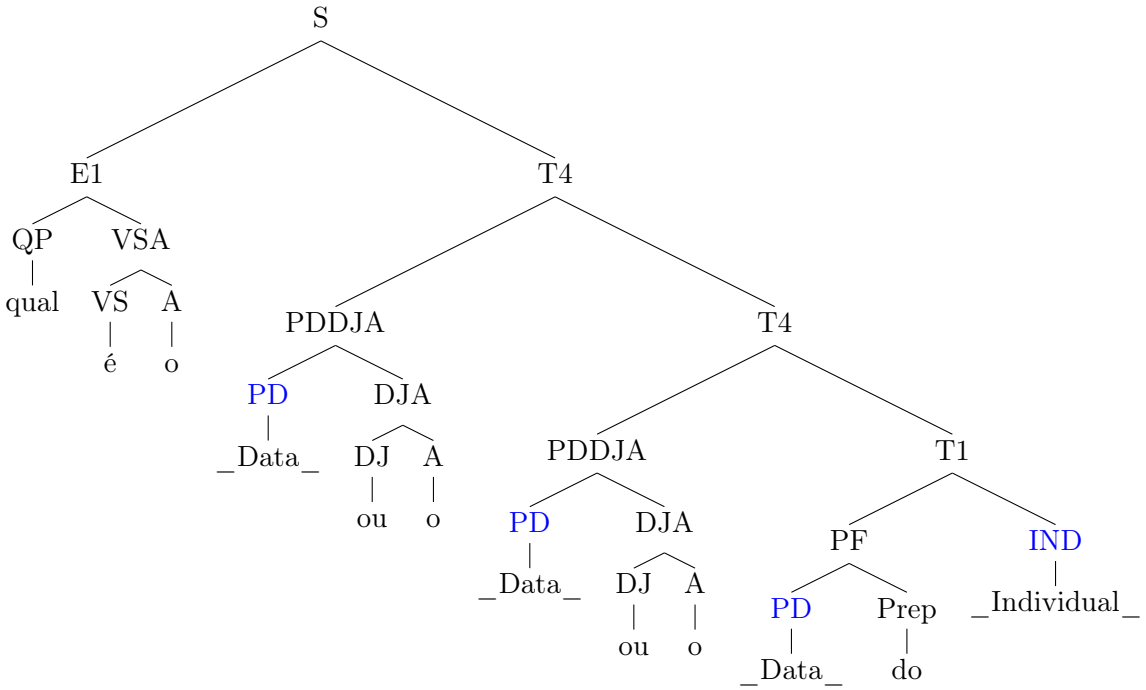
  10. LNC: **qual é o endereço e o telefone e a webpage do IME?**
  LNC_M: qual é o _Data_ e o _Data_ e a _Data_ do _Individual_?



`E1` : estrutura básica SPARQL `SELECT *`.
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
T1 e T4 são unidas por UNION
`T3` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?B }
`T3` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?C }

- Caso 4: Recursão de disjunção.

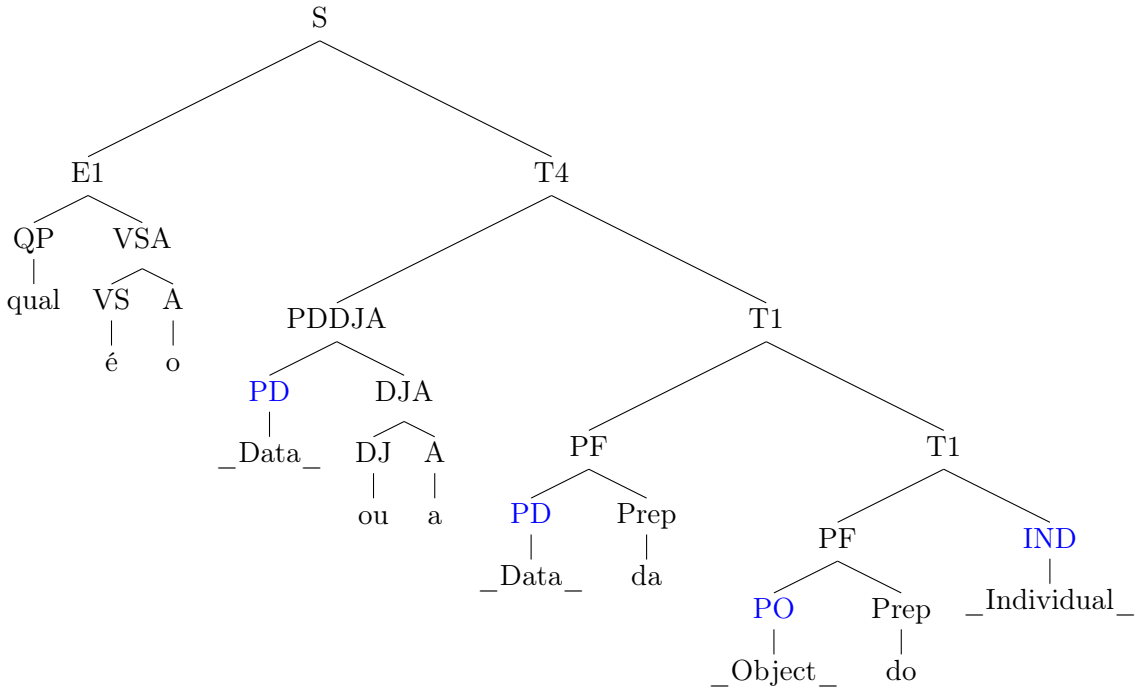
11. LNC: **qual é o material ou o formato ou a cor do iPhone4?**
LNC_M: qual é o _Data_ ou o _Data_ ou o _Data_ do _Individual_?



`E1` : estrutura básica SPARQL `SELECT *`.
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
T1 e T4 são unidas por UNION

`T4` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?B }
`T4` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?C }

- Caso 5: Recursão de T1 com disjunção.
  12. LNC: **qual é o titulo ou a área da tese de doutorado do Paulo de Tarso?**
  LNC_M: qual é o _Data_ ou a _Data_ da _Object_ do _Individual_?

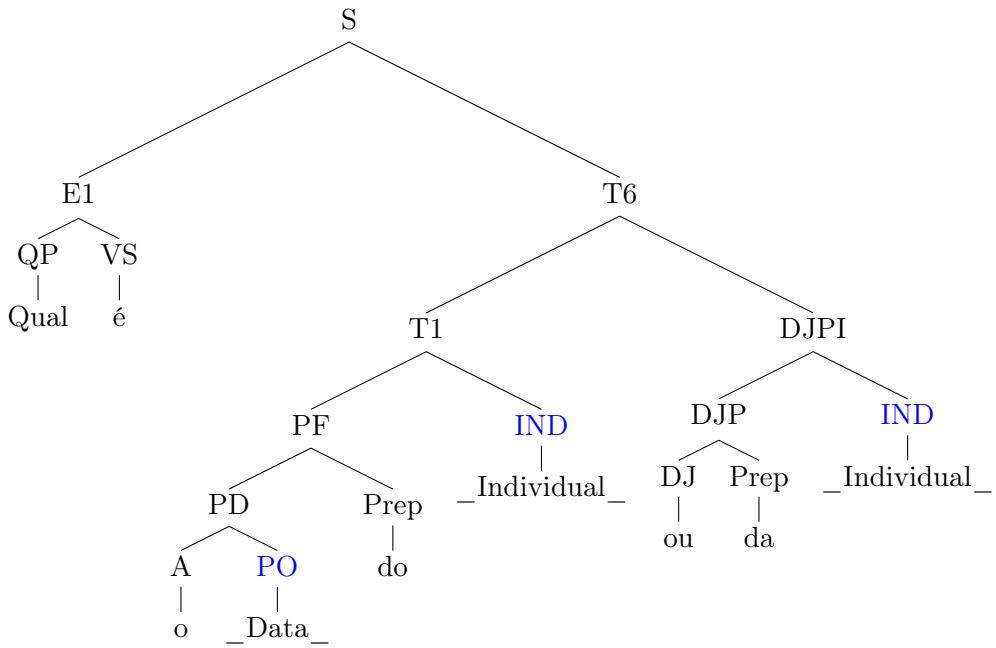


`E1` : estrutura básica SPARQL `SELECT *`.
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
T1 e T4 são unidas por UNION
`T4` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?B }
`T4` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?C }

Conjunções e disjunções de Indivíduos

No nosso modelo, conjunções de propriedades e indivíduos não podem ser utilizadas na mesma consulta. Nosso modelo tenta simplificar o uso das conjunções e disjunções. T6 e T7 são responsáveis por tratar as disjunções e conjunções entre indivíduos, respectivamente. Assim como no caso das conjunções e disjunções entre propriedades, aqui também elas não podem acontecer juntas em uma consulta.

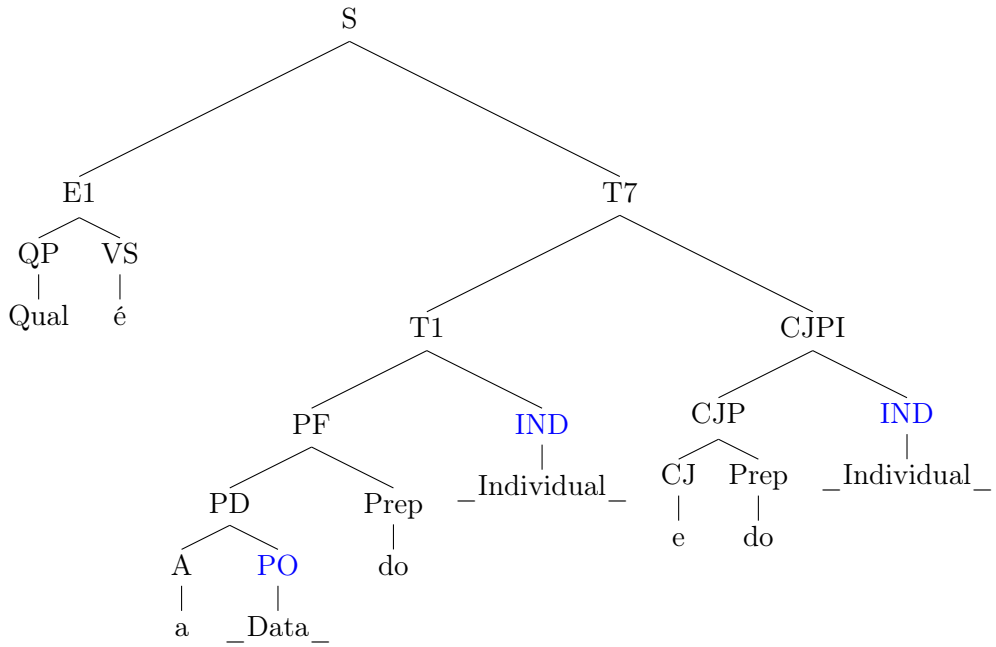
- Caso 1: Disjunção:
  13. LNC: **Qual é endereço profissional do Fabiano ou da Renata?**
  LNC_M Qual é o _Data_ do _Individual_ ou da _Individual_



`E1` : estrutura básica SPARQL `SELECT ?A.`
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
T1 e T6 são unidas por UNION
`T6` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }

- Caso 2: Conjunção:

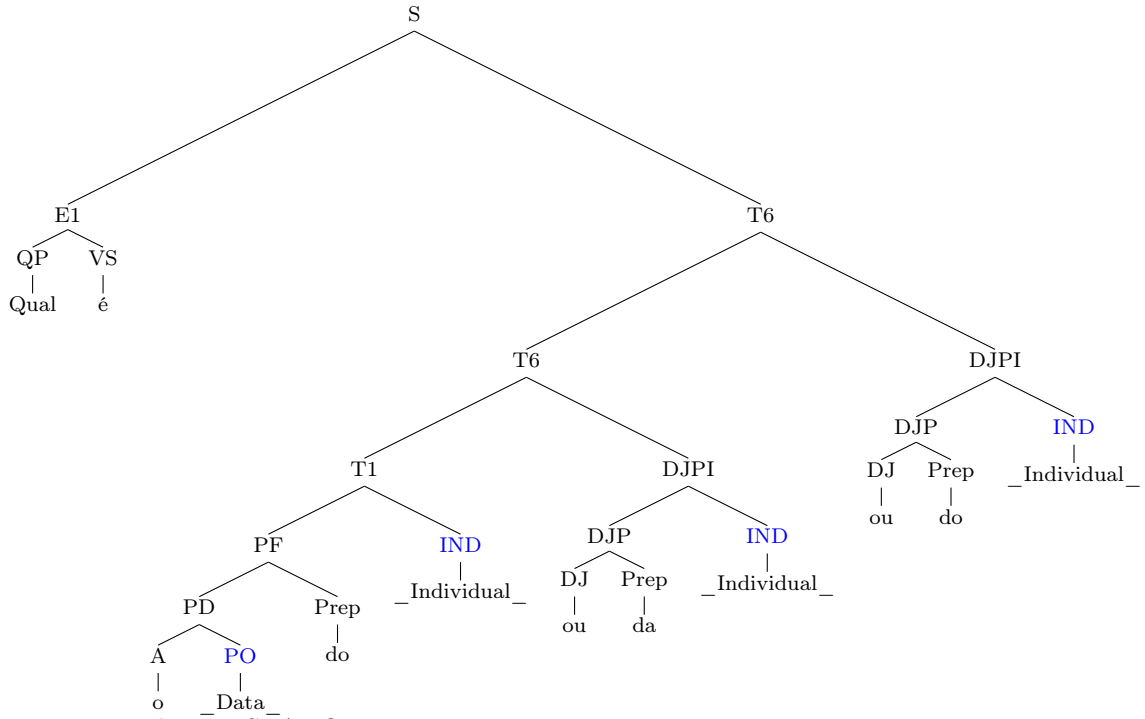
14. LNC: **qual é a formação do Paulo de Tarso e do Fabiano?**
LNC_M: qual é a _Data_ do _Individual_ e do _Individual_



`E1` : estrutura básica SPARQL `SELECT * .`
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
`T6` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?B }

- Caso 3: Recursão de Conjunção:

15. LNC: **qual é o e-mail do Fabiano ou do Paulo ou do Raphael?**
LNC: qual é o _Data_ do _Individual_ ou do _Individual_ ou do _Individual_

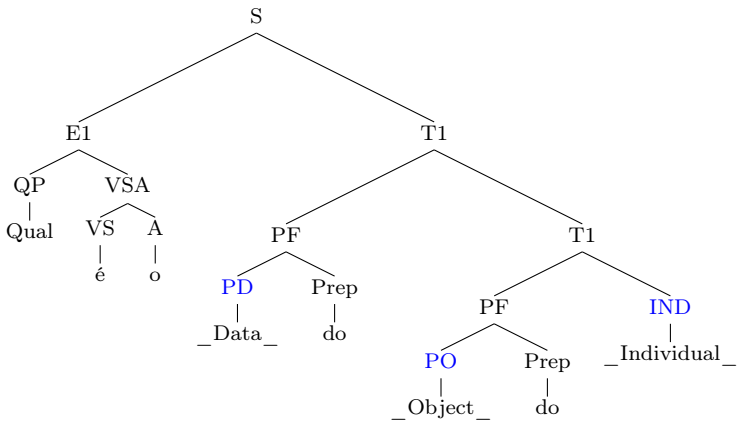


`E1` : estrutura básica SPARQL `SELECT ?A.`
`T1` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
T1 e T6 são unidas por UNION
`T6` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }
A ultima T6 é unida com a T6 anterior
`T6` : é a tripla SPARQL { p:_Individual_ p:_Data_ ?A }

O caso de recursão de disjunção é análogo ao caso anterior. Aqui também é permitida a recursão de T1 com conjunção ou disjunção.

4.8.3 Algoritmo de produção do SPARQL

Voltando ao exemplo da seção 4.4.2. Temos a sentença "**Qual é o _Data_ do _Object_ do _Individual_**" gerada pelo Matching. Primeiramente fazemos uma análise sintática em nossa sentença em LNC_M usando o CKY. O resultado dessa análise é uma árvore de derivação, que será gerada usando as regras da GLSPARQL. Exemplo:



Nosso algoritmo recursivo recebe como parâmetro a árvore gerada pelo CKY e produz o SPARQL utilizando as regras definidas na seção 4.8.1. Na árvore, os nós não terminais E_x e T_y são importantes

na construção do SPARQL, onde E_x é interpretado como estrutura básica e T_y como uma tripla RDF. De acordo com as Tabelas 4.7 e 4.8, x pode assumir valores de 1..3 e y de 1..7.

O algoritmo percorre a árvore visitando todas as sub-árvores da raiz da esquerda para a direita e por último a raiz. A raiz da sentença é sempre **S**. As derivações de **S** sempre vão para um tipo **E** e um tipo **T** nesta ordem, vide Tabela 4.5. Desta forma, nosso algoritmo sempre gera uma *string* que é a concatenação de uma estrutura SPARQL com uma ou mais triplas.

Algoritmo 1 - Algoritmo pós-ordem para produção do SPARQL

```
GerarSPARQL(id)
 1. E = {E1, E2, E3} e T = {T1, T2, T3, T4, T5}
 2. Nó n ← Arvore.pegar(id)
 3. se n = Terminal então
 4.     retorne n.derivação
 5. senão
 6.     nf1 ← n.filho1
 7.     nf2 ← n.filho2
 8.     var ← geraVariavelDaTripla()
 9.     se n.rótulo = S então
10.         p2 ← gerarSPARQL(nf2)
11.         retorne gerarSPARQL(nf1)+ var+"{"+p2+"}"
12.     senão
13.         se n.rótulo ∈ E então
14.             Estrutura ← Saida do rótulo, vide Tabela 4.7.
15.             retorne Estrutura+" "+gerarSPARQL(nf1)+" "+gerarSPARQL(nf2)
16.         senão
17.             se n.rótulo ∈ T então
18.                 p2 ← gerarSPARQL(nf2)
19.                 p1 ← gerarSPARQL(nf1)
20.                 retorne ▷ Escreve tripla conforme Tabela 4.8 +"."+as triplas anteriores.
21.             senão
22.                 retorne gerarSPARQL(nf1)+" "+gerarSPARQL(nf2)
23.             fim se
24.         fim se
25.     fim se
26. fim se
```

No algoritmo 1 a identificação do nó que produz a estrutura ocorre na linha 13. Na linha 14 a variável texto **Estrutura** recebe o valor de acordo com a Tabela 4.7. A linha 15 é o retorno da função onde a variável **Estrutura** é posicionada corretamente no texto SPARQL. Na linha 17 ocorre a identificação dos nós que produzem as triplas. As variáveis **p1** e **p2** são responsáveis por explorar os nós filhos. A linha 20 é o retorno da função, onde escreve-se a nova tripla produzida de acordo com a Tabela 4.8 concatenada com as triplas anteriores.

Recordando o exemplo anterior "**Qual é o _Data_ do _Object_ do _Individual_**", o primeiro T que nosso algoritmo encontra é **T1**, composto pelo "**_Individual_**" e o **_Object_** que resulta na tripla **p:_Individual_ p:_Object_ ?A**, onde **?A** é a variável de saída, vide Tabela 4.8. O segundo **T1** encontrado é composto pela propriedade de saída da **T1** anterior, ou seja, a variável **?A** e uma **_Data_**, gerando a tripla **?A p:_Data_ ?C**.

Passos do algoritmo:
1º : 1ª T1, gerando a *string* inicial: { p:_Individual_ p:_Object_ ?A}
2º : 2ª T1, a *string* resultante: { ?A p:_Data_ ?B. p:_Individual_ p:_Object_ ?A}
3º : E1, *string* final: SELECT ?B { ?A p:_Data_ ?B. p:_Individual_ p:_Object_ ?A }

Meta SPARQL resultante:

```
PREFIX p:<our_uri#>
SELECT ?B { ?A p:_Data_ ?B . p:_Individual_ p:_Object_ ?A}
```

Com as devidas substituições pelas palavras originais a consulta resulta:

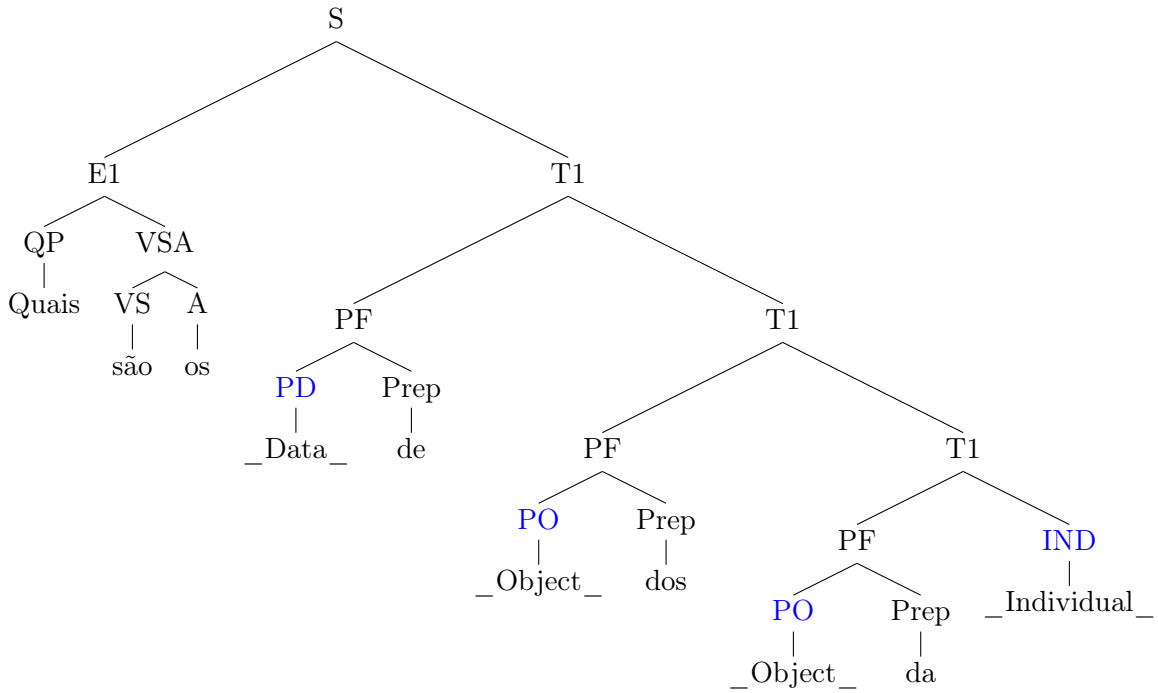
```
PREFIX p:<http://www.semanticlattes.com.br/curriculo#>
SELECT ?B { ?A p:titulo ?B . p:Paulo p:tem_tese_de_doutorado ?A}
```

O **PREFIX** de cada elemento é armazenado durante o processo de Matching. Se os **PREFIX** forem comuns a todos os elementos, geramos apenas um alias.

Outros exemplos

Exemplo 1:

A consulta "**quais são os modelos de teclados dos celulares da Nokia?**" produz a sentença "**qual é o _Data_ do _Object_ do _Object_ do _Individual_**" quando submetida ao Matching. A seguir, a árvore gerada a partir do *parsing* utilizando a GLSPARQL:



Passos do algoritmo na geração da consulta SPARQL:
1º : 1ª T1, *string* resultante: { p:_Individual_ p:_Object_ ?A}
2º : 2ª T1, resultando na *string*: { ?A p:_Object_ ?B. p:_Individual_ p:_Object_ ?A }
3º : 3ª T1, resultando a *string*: { ?B p:_Data_ ?C . ?A p:_Object_ ?B. p:_Individual_ p:_Object_ ?A }
4º : E1, *string* final: SELECT ?C { ?B p:_Data_ ?C . ?A p:_Object_ ?B. p:_Individual_ p:_Object_ ?A } .

Meta SPARQL resultante:

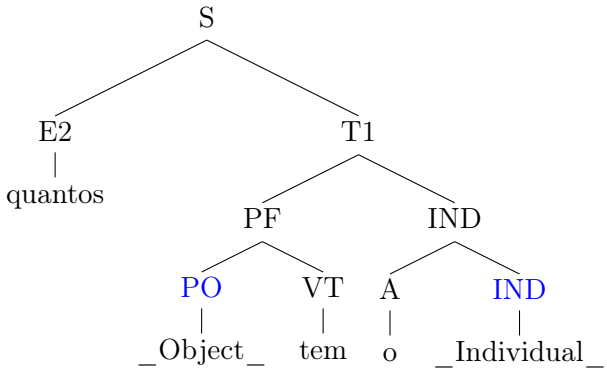
```
PREFIX p:<our_uri#>
SELECT ?C WHERE {?B p:_Data_ ?C . ?A p:_Object_ ?B.
                 p:_Individual_ p:_Object_ ?A
                }
```

Com as devidas substituições pelas palavras originais a consulta resulta:

```
PREFIX p:<http://www.rodrigo.goulart.nom.br/feevale/blogseimagem/Smartphone.owl#>
SELECT ?C WHERE {?B p:modelo ?C . ?A p:tem_teclado ?B.
                 p:Nokia p:tem_celular ?A
                }
```

Exemplo 2:

A consulta "**quantos celulares tem o Fernando?**" produz a sentença "**quantos _Object_ tem o _Individual_**" quando submetida ao Matching. A seguir, a árvore gerada a partir do *parsing* utilizando a GLSPARQL:



Passos do algoritmo, na geração da consulta SPARQL:
1º : T1, a *string* resultante: { p:_Individual_ p:_Object_ ?A }
2º : E2, *string* final SPARQL: SELECT Count(?A) { p:_Individual_ p:_Object_ ?A } .

Meta SPARQL resultante:

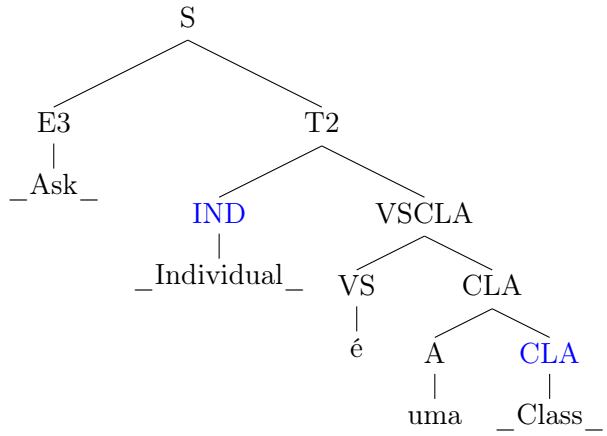
```
PREFIX p:<our_uri#>
SELECT Count(?A) WHERE {p:_Individual_ p:_Object_ ?A}
```

Com as devidas substituições pelas palavras originais a consulta resulta:

```
PREFIX p:<http://www.rodrigo.goulart.nom.br/feevale/blogseimagem/Smartphone.owl#>
SELECT Count(?A) WHERE {p:Fernando p:tem_celular ?A}
```

Exemplo 3:

A consulta "**Renata Wassermann é uma professora?**" produz a sentença "**_Ask_ _Individual_ é uma _Class_**" quando submetida ao Matching. A seguir, a árvore gerada a partir do *parsing* utilizando a GLSPARQL:



Passos do algoritmo, na geração da consulta SPARQL:
1º : T2, *string* inicial: { p:_Individual_ r:type p:_Class_ }
2º : E3, *string* final SPARQL: ASK { p:_Individual_ r:type p:_Class_} .

Meta SPARQL resultante:

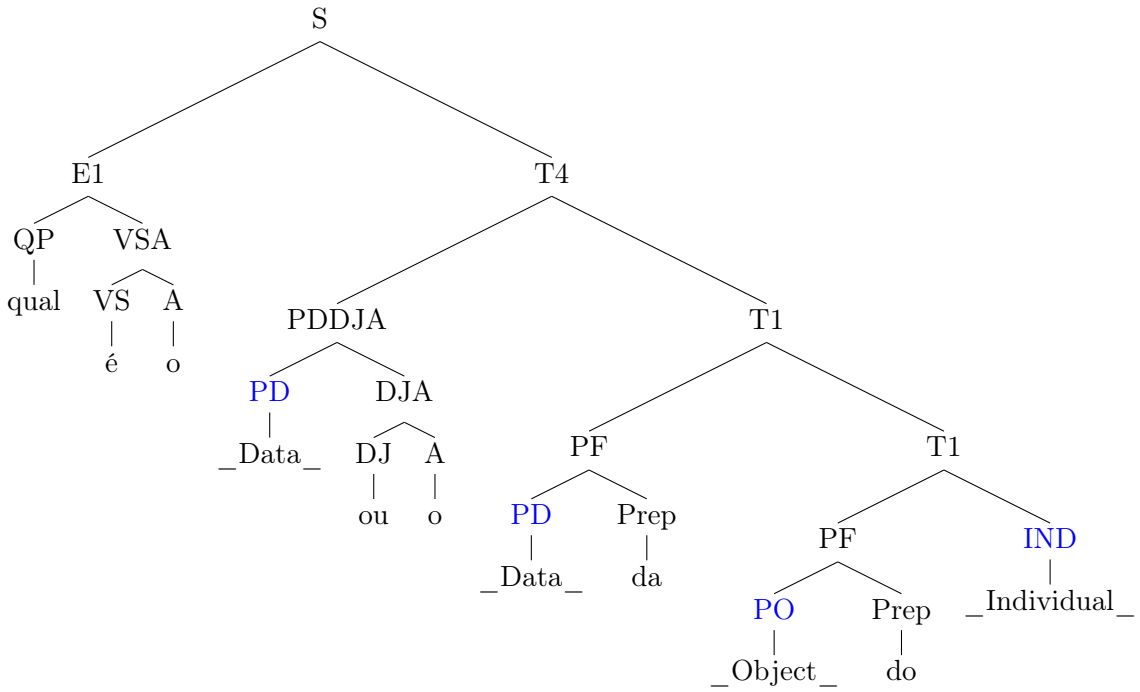
```
PREFIX r: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p:<our_uri#>
ASK {p:_Individual_ r:type p:_Class_ }
```

Com as devidas substituições pelas palavras originais a consulta resulta:

```
PREFIX r: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p:<http://www.semanticlattes.com.br/curriculo#>
ASK {p:Renata_Wassermann r:type p:Professor }
```

Exemplo 4:

A consulta "**qual é o título ou o tema da tese_de_doutorado do Esdras?**" produz a sentença "**qual é o _Data_ ou o _Data_ da _Object_ do _Individual_**" quando submetida ao Matching. A seguir, a árvore gerada a partir do *parsing* utilizando a GLSPARQL:



Passos do algoritmo, na geração da consulta SPARQL:
1º : 1ª T1, *string* inicial: { p:_Individual_ p:_Object_ ?A }
2º : 2ª T1, a *string* resultante é: { ?A p:_Data_ ?B . p:_Individual_ p:_Object_ ?A}
T1 e T4 são unidas por UNION
3º : T4, a *string* resultante:{{?A p:_Data_ ?C } UNION {?A p:_Data_ ?B} . p:_Individual_ p:_Object_ ?A }
4º : E1, a *string* final: SELECT * {{?A p:_Data_ ?C } UNION {?A p:_Data_ ?B} . p:_Individual_ p:_Object_ ?A }

Meta SPARQL resultante:

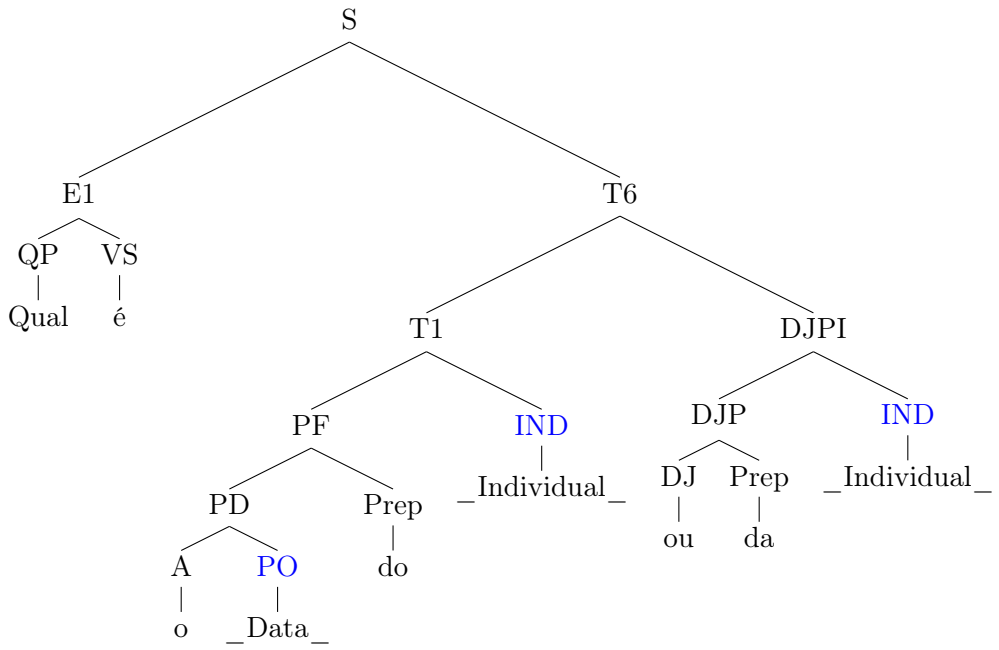
```
PREFIX p:<our_uri#>
SELECT * {{?A p:_Data_ ?C.} UNION {?A p:_Data_ ?B} .p:_Individual_ p:_Object_ ?A.}
```

Com as devidas substituições pelas palavras originais a consulta resulta:

```
PREFIX p:<http://www.semanticlattes.com.br/curriculo#>
SELECT * {{?A p:titulo ?C.} UNION {?A p:tema ?B} .p:Esdras p:tem_teste_de_doutorado ?A.}
```

Exemplo 5:

A pergunta: "**Qual é o email do Fabiano ou da Renata Wassermann**" após o matching é transformada em: **Qual é o _Data_ do _Individual_ ou da _Individual_**. O *parsing* segundo nossa *GLSPARQL* é:



Passos do algoritmo, na geração da consulta SPARQL:
1º : 1ª T1, *string* inicial: { p:_Individual_ p:_Data_ ?A }
T1 e T6 são unidas por UNION
2º : T6, a *string* resultante:{{p:_Individual_ p:_Data_ ?A } UNION {p:_Individual_ p:_Data_ ?A} }
3º : E1, a *string* final: SELECT ?A {{p:_Individual_ p:_Data_ ?A } UNION {p:_Individual_ p:_Data_ ?A}}

Meta SPARQL resultante:

```
PREFIX p:<our_uri#>
SELECT ?A {{p:_Individual_ p:_Data_ ?A.} UNION {p:_Individual_ p:_Data_ ?A}}
```

Com as devidas substituições pelas palavras originais a consulta resulta:

```
PREFIX p:<http://www.semanticlattes.com.br/curriculo#>
SELECT ?A {{p:Renata_Wassermann p:email ?A.} UNION {p:Fabiano p:email ?A}}
```

4.8.4 Considerações finais sobre o modelo

Toda sentença em LNC_M é factível de ser analisada pela $GLSPARQL$, ou seja, uma sentença em LNC_M sempre será transformada em uma consulta SPARQL. Contudo, como veremos na seção 5.2.1, o SPARQL gerado pode não obter a resposta esperada.

Com nosso Matching, não existe uma garantia de que sempre conseguiremos uma correlação satisfatória entre os termos da ontologia e da sentença em LNC. Estamos vulneráveis a limitações que não são uma exclusividade do nosso modelo. Medir semelhança entre termos é um problema difícil e conhecido em processamento de linguagem natural [Richardson et al., 1994] e [Li et al., 2003].

Outro fato interessante é que perguntas não aceitas pelo nosso modelo podem ser transformadas facilmente em consultas aceitas. Por exemplo, a pergunta: "**Quantos professores são orientandos da Renata Wassermann?**" não está na LNC, pois a junção "quantos professores são" é rotulada como "PQ Sub VS" uma construção impossível em nossa LNC. Porém podemos transformar esta consulta em: "**Quantos orientandos da Renata Wassermann são professores?**"

Outro exemplo é: "**Quantos orientandos a orientadora do Ricardo Augusto tem?**". Neste caso estamos usando o verbo no final da frase, outra construção impossível. A transformação para esta pergunta é: "**Quantos orientandos tem a orientadora do Ricardo Augusto?**". Esta propriedade linguística é denominada extraposição e diz respeito ao reposicionamento dos elementos dentro da sentença mantendo a semântica [Machado, 2003] e [Cardoso, 2009].

Capítulo 5

Testes e Resultados

5.1 Escolha do conjunto de testes

Como caso de estudo utilizamos duas ontologias de domínio público. A primeira é a ontologia do currículo Lattes que foi baseada no trabalho Semantic Lattes [da Costa and Yamate, 2009] e a segunda é uma ontologia de conceitos relacionados a aparelhos smartphones [Martins, 2007]. A ontologia do currículo Lattes utilizada em nosso trabalho possui apenas um subconjunto de instâncias (329 instâncias), ou seja, não compreende toda informação da base de dados atual da plataforma Lattes. Na ontologia sobre o currículo Lattes fizemos algumas adaptações adequando-a ao modelo. Estas adaptações consistem em separar os nomes dos recursos que estão concatenados e adicionar preposições quando necessário, para que os mesmos se assemelhem a linguagem natural. A Tabela 5.1 mostra alguns exemplos desta adaptação:

Original	Adaptado
ArtigoConferencia	Artigo_de_Conferencia
ConferenciaInternacional	Conferencia_Internacional
EstudanteDoutorado	Estudante_de_Doutorado
paginafinal	pagina_final
temAutor	tem_Autor

Tabela 5.1: *Alguns exemplos da adaptação*

Outra mudança que fizemos na ontologia foi trocar os nomes das instâncias que na versão original são grandes ID's. Nós trocamos por nomes que fazem mais sentido para o nosso trabalho e nos ajudam no processo de Matching. Por exemplo, a instância **6fbc5870-638f-4dc3-ad96-970aa7cba43d** foi renomeada para **tese de dissertação do Paulo de Tarso**.

Além das adaptações, na ontologia sobre *smartphones* fizemos a exclusão de algumas classes e relações que estavam fora do nosso padrão de ontologia e também criamos novas classes e relações. As ontologias adaptadas podem ser encontradas em http://ime.usp.br/~fluz/curriculo.owl e http://ime.usp.br/~fluz/smartphone.owl. Estamos utilizando duas ontologias de domínios diferentes para atestar que nossa gramática não possui dependência de um contexto específico, o modelo depende da estrutura da ontologia, nos proporcionando assim uma maior abrangência na aplicação.

Dado que temos uma ontologia controlada e uma linguagem controlada, não faz sentido usarmos um corpus de perguntas totalmente aleatórias. Portanto, desenvolvemos formulários, que podem ser encontrados no Apêndice B, para auxiliar colaboradores na elaboração de perguntas para testar o modelo.

Obtivemos um total de 196 perguntas enviadas, das quais escolhemos 144, que o sistema teria potencial para responder. Pelo fato de nossa ontologia estar pouco povoada, escolhemos apenas perguntas que seriam factíveis de serem respondidas do ponto de vista do conteúdo, isto é, perguntas onde o Matching teria uma possibilidade de resposta. Por exemplo, a pergunta: "**quantos artigos**

possui João Ninguém" não seria escolhida pois **João Ninguém** não é um indivíduo em nossa ontologia. As perguntas utilizadas nos testes encontram-se no Apêndice A.

Das 144 perguntas selecionadas, 95 são do contexto da ontologia do currículo Lattes e outras 49 do contexto da ontologia de *smartphones*.

5.2 Resultados e discussão

A avaliação foi feita submetendo as perguntas descritas na seção anterior ao nosso protótipo e comparando as respostas obtidas com as respostas produzidas manualmente. Para avaliar o modelo fizemos uso de duas métricas tradicionais em tarefas de PLN. A primeira métrica utilizada é a medida de acurácia vista na Seção 2.4, onde a interpretação sobre os resultados é basicamente uma verificação de entrada e saída ou seja:

$$\text{acurácia} = \frac{|\{\text{perguntas respondidas corretamente}\}|}{|\{\text{perguntas}\}|}$$

Nesta primeira avaliação admitimos que as perguntas que não passaram no Matching obtiveram resposta errada. Nos testes com o conjunto de perguntas sobre a ontologia do currículo Lattes obtivemos 79 respostas corretas de um total de 95 perguntas, enquanto no teste com o conjunto de perguntas para a ontologia sobre *smartphones* obtivemos 42 respostas corretas de 49 perguntas. Ao total temos 144 perguntas das quais 121 foram respondidas corretamente.

Tabela 5.2: *Testes: acurácia*

Perguntas	Acurácia
curriculo Lattes	0.83157
smartphone	0.85714
Total	0.84027

A segunda métrica utilizada é a precisão e cobertura vistas na Seção 2.4. Neste caso temos uma interpretação diferente para as perguntas que não passaram no Matching: não existem respostas para elas . Apenas as perguntas traduzidas são submetidas para a verificação da resposta gerada. Das 95 perguntas sobre a ontologia do currículo Lattes, 82 foram factíveis de serem traduzidas para SPARQL. Da submissão destas consultas SPARQL's obtivemos 79 respostas corretas, ou seja, $|A_O| = 82$, $|A_O \cap R_O| = 79$ e $|R_O| = 95$. Das 49 perguntas para a ontologia sobre *smartphones* , 44 foram transformadas em SPARQL e 42 geram respostas corretas, ou seja, $|A_S| = 44$, $|A_S \cap R_S| = 42$ e $|R_S| = 49$. Ao total temos 144 perguntas das quais 126 foram transformadas e 121 foram respondidas corretamente.

Tabela 5.3: *Testes: precisão*

Perguntas	Precisão	Cobertura
curriculo Lattes	0.96341	0.83157
smartphone	0.95454	0.85714
Total	0.96031	0.84027

Na Tabela 5.3 temos um total de precisão de 0.96031 e 0.84027 de cobertura, que consideramos um bom resultado. Embora o Querix visto na Seção 3.1, tenha obtido 0.7767 de precisão e 0.7860 de cobertura não podemos comparar os resultados já que o cenário dos testes e as restrições são diferentes.

Quando submetemos uma pergunta ao nosso modelo, existem algumas situações que impossibilitam a geração do SPARQL:

- **Perguntas não aceitas pela gramática**: estas perguntas não serão processadas. Não existia esta situação em nosso conjunto de teste.

- **Perguntas não resolvidas no Matching**: mesmo que as perguntas estejam em LNC, o Matching pode não conseguir correlacionar os termos da LNC com a ontologia de busca. Em nosso conjunto de teste, temos 17 erros que se encaixam nesta situação. Por exemplo, na consulta "**quantas teses possuem a orientação da Renata Wassermann?**", o Matching não conseguiu relacionar a palavra **teses** com o termo **tese_de_doutorado**. Na seção 5.2.1, sugerimos uma abordagem para mitigar este problema.

O SPARQL gerado pelo modelo pode retornar:

- **Vazio**: no caso da relação procurada ser inexistente na ontologia, o retorno vazio não é considerado uma falha ou erro. Algumas informações que o modelo procura podem existir, mas devido a ausência de relações entre as entidades, a informação pode não ser encontrada. Por exemplo, na Figura 5.1.

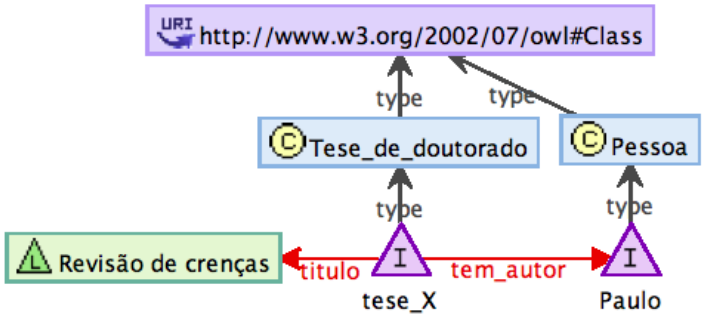


Figura 5.1: *Indivíduos e suas relações*

Dada a consulta "**qual é o titulo da tese_de_doutorado do Paulo**" verifica-se que na ontologia existe uma representação para o trecho "**titulo da tese_de_doutorado**", pois, como esperado pelo nosso modelo a instância de **tese_de_doutorado** possui **titulo**. Porém a relação de que **Paulo** possui uma tese de doutorado não existe. Portanto, o modelo não encontra resposta para esta pergunta.

O retorno vazio também pode acontecer quando o Matching atribui um termo equivocadamente (seja usando o lema ou algum método de similaridade *fuzzy*), este caso é considerado falha do sistema. Por exemplo, a consulta "**quantos orientandos tem a orientadora do Ricardo Augusto?**", a palavra orientadora está no masculino em nossa ontologia, então usamos o lematizador para verificar a correspondência e a palavra **orientadora** resulta em **orient**. o Matching correlaciona com a primeira palavra que começa com **orient**, relacionando **orientadora** com **orientando** ao invés de **orientador**.
SPARQL resultante:

```
PREFIX p: <http://www.semanticlattes.com.br/curriculo#>
SELECT (COUNT(?B) AS ?Resp) {?A p:tem_orientando  ?B .
                 p:Ricardo_Augusto  p:tem_orientando  ?A }
```

Observa-se que um Matching prudente verificaria a correlação entre os termos alterando o gênero e/ou grau do substantivo antes de utilizar o lema. Esta tarefa é um tanto quanto trabalhosa portanto vamos deixar como sugestões para trabalhos futuros.

Em outro caso, o problema de retorno vazio pode estar relacionado com problemas de incompatibilidade entre a estrutura da ontologia e a estrutura que o modelo espera. Nos testes, 3 perguntas geram este tipo de erro (consultar Apêndice A, ontolattes: 73 e 95 e smartphone: 4).

A consulta "**quais professores do IME são autores de livros?**" gera problema de incompatibilidade de estrutura, pois, **autores** e **livros** são duas classes distintas na ontologia do currículo Lattes, no entanto o modelo esperava um único termo para representar autores de livros.

Perguntas ambíguas como: **quais são os autores de livros da USP?** tendem a gerar problemas com relação a incompatibilidade da estrutura da ontologia com o modelo, veremos outros exemplos deste problema bem como abordagens para tratá-los na seção 5.2.1.

- **Não vazias inesperadas**: As respostas não vazias inesperadas podem acontecer em virtude das estruturas inesperadas que serão abordadas na Seção 5.2.1 ou até mesmo por falhas do Matching. O Matching pode associar termos à uma palavra semelhante porém incorreta para o contexto.
- **Resposta esperada**: a situação de sucesso, na qual se incluem 123 perguntas do nosso teste. As consultas "**Qual é o idioma ou pais de publicação da tese de doutorado do Paulo de Tarso?**" e "**Quais orientandos da Renata Wassermann são professores?**" são exemplos de consultas respondidas corretamente em nosso teste, gerando os respectivos SPARQL's.

```
PREFIX p: <http://www.semanticlattes.com.br/curriculo#>
SELECT  ?B ?C {{ ?A  p:idioma  ?C .} UNION {?A p:pais_de_publicacao  ?B }.
            p:Paulo_de_Tarso p:tem_tese_de_doutorado  ?A }
```

```
PREFIX p: <http://www.semanticlattes.com.br/curriculo#>
PREFIX r: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?A {?A r:type  p:Professor . p:Renata_Wassermann p:tem_orientando  ?A .}
```

5.2.1 Falhas do modelo

É conveniente fazer uma distinção entre o problema que pretendemos atacar e os problemas inerentes ao desenvolvimento do modelo. Com relação ao problema foco do trabalho, trata-se basicamente da consulta a ontologias utilizando linguagem natural controlada. Já os problemas oriundos do desenvolvimento do modelo, em sua maioria, estão relacionados a utilização de uma ontologia inapropriada e a encontrar correlação entre os termos. Nesta seção, comentamos sobre os erros relativos ao segundo problema.

Os erros que acontecem em nosso modelo estão relacionados com falhas no processo de Matching e mais raramente podem acontecer diferenças entre a estrutura que a consulta SPARQL exige e a estrutura da ontologia controlada. Erros com relação ao processo de Matching podem ser minimizados desenvolvendo um Matching especifico para um domínio, por meio da previsão de termos equivalentes. Na Tabela 5.4, apresentamos alguns conceitos da ontologia do currículo Lattes e seus equivalentes bem como generalizações.

Tabela 5.4: *Termos equivalentes e generalizações*

Termo ontologia	**Termo consulta**
Estudante de doutorado	Doutorando
Dissertação de mestrado	Dissertação
Orientador	Orientandor de mestrado ou Orientandor de doutorado
Email	Endereço de email
Estudante de mestrado	Estudante, aluno ou discente

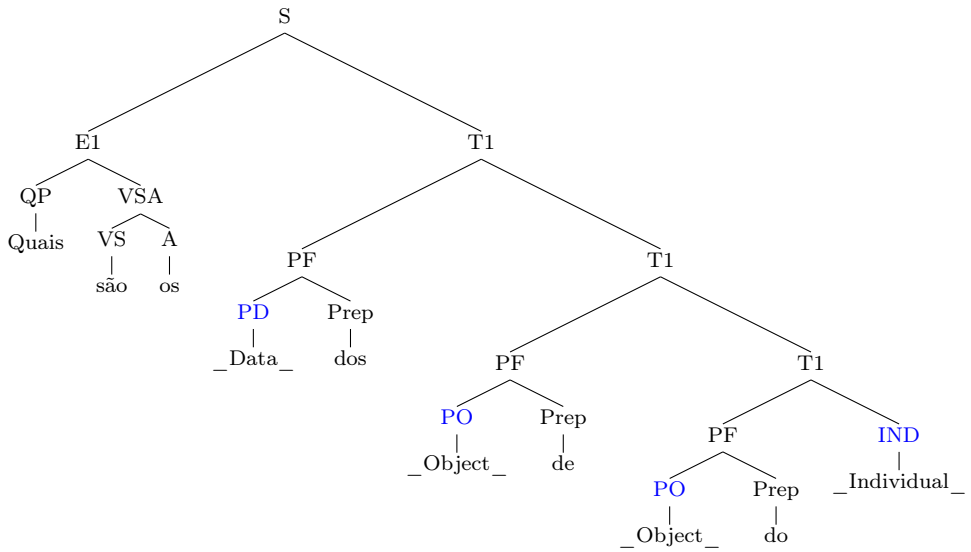
Observando a Tabela 5.4, vemos que alguns termos mantém uma equivalência direta, por exemplo, "**endereço de email**" e "**email**". Porém outros termos representam generalizações e nem sempre podem ser utilizados como termos equivalentes. Um Matching específico deve prever estas situações minimizando erros que podem vir a ocorrer. Para o reconhecimento de indivíduos, outras técnicas como busca por similaridade podem ser aplicadas.

A seguir discutimos um problema relacionado a flexibilidade e ambiguidades inerentes a linguagem natural, que aplica-se tanto na construção das ontologias quanto na elaboração de consultas.

Estruturas inesperadas

É possível construir perguntas ambíguas que contrariam a estratégia utilizada por nosso algoritmo para a geração das triplas. Iremos utilizar como exemplo uma pergunta que respeita a LNC e está dentro do contexto da ontologia de *smartphones*. Mesmo presente dentro da ontologia, a informação não pode ser recuperada pois o SPARQL gerado por esta pergunta espera uma estrutura específica (vide Figura 5.2).

A consulta "**quais são os endereços das fabricantes de celulares do Brasil**" produz a sentença "**quais são os _Data_ das _Class_ de _Class_ do _Individual_**" quando submetida ao Matching. A seguir, a árvore gerada a partir da análise sintática utilizando a GLSPARQL:



Passos do algoritmo na geração da consulta SPARQL:
1º : 1ª T1, string inicial { p:_Individual_ p:_Object_ ?A }
2º : 2ª T1 é a tripla SPARQL { ?A p:_Object_ ?B. p:_Individual_ p:_Object_ ?A }
3º : 3ª T1 é a tripla SPARQL { ?B p:_Data_ ?C. ?A p:_Object_ ?B. p:_Individual_ p:_Object_ ?A }
4º : E1, string final: SELECT ?C { ?B p:_Data_ ?C. ?A p:_Object_ ?B. p:_Individual_ p:_Object_ ?A } .

Meta SPARQL resultante:

```
PREFIX r: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p:<our_uri#>
SELECT ?C WHERE { ?B p:_Data_ ?C. ?A p:_Object_ ?B.
                  p:_Individual_ p:_Object_ ?A }
```

Com as devidas substituições pelas palavras originais, a consulta resulta:

```
PREFIX r: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p:<http://www.rodrigo.goulart.nom.br/feevale/blogseimagem/Smartphone.owl#>
SELECT ?C WHERE { ?B p:endereco ?C. ?A p:tem_fabricante ?B.
                  p:Brasil p:tem_celular ?A }
```

Analisando o SPARQL gerado, concluímos que a consulta esperava que a ontologia estivesse organizada como na Figura 5.2.

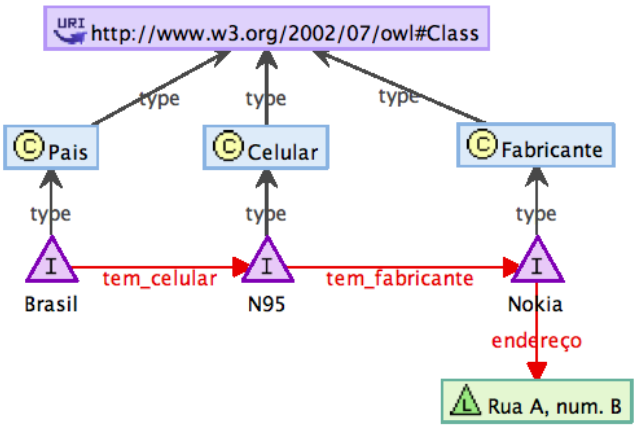


Figura 5.2: *Estrutura esperada pelo SPARQL*

No entanto as informações poderiam estar organizadas como na Figura 5.3. Isto evidencia que a flexibilidade na construção das ontologias podem induzir nosso modelo a falhas.

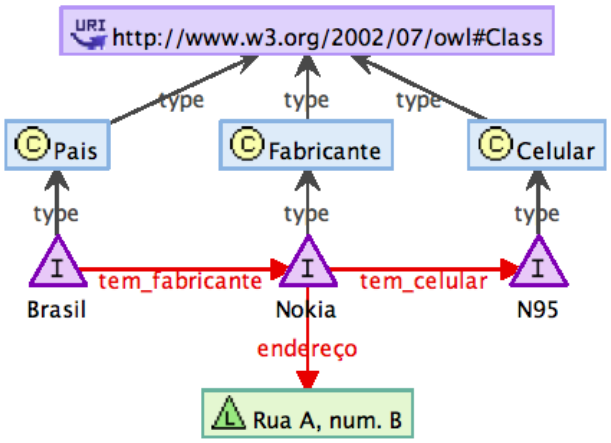


Figura 5.3: *Estrutura apropriada*

Vamos sugerir algumas abordagens com o propósito de mitigar o problema citado:

1. **Regras e inferências:**

   A utilização dos recursos de inferência das ontologias podem amenizar estas falhas. Por exemplo, pode-se criar uma regra de propagação de transitividade onde:

   ```
   rule: (?A tem_fabrica ?B ) (?B tem_celular ?C) -> (?A tem_celular ?C)
   ```

   Assim a ontologia poderia inferir que a relação "**Brasil tem_celular N95**" existe.

2. **Especialização dos conceitos:**

   Outra abordagem para minimizar problemas é adequar os conceitos da ontologia de busca, especializando-os. Por exemplo, no caso da pergunta anterior, confusões ocorrem pelo fato de existir duas classes distintas: a classe **Fabricante** e a classe **Celular**. Para este caso em específico, a criação de um conceito composto pelas duas classes simplificaria o problema, por exemplo **Fabricante de Celular**. A Figura 5.4 exibe a nova estrutura.

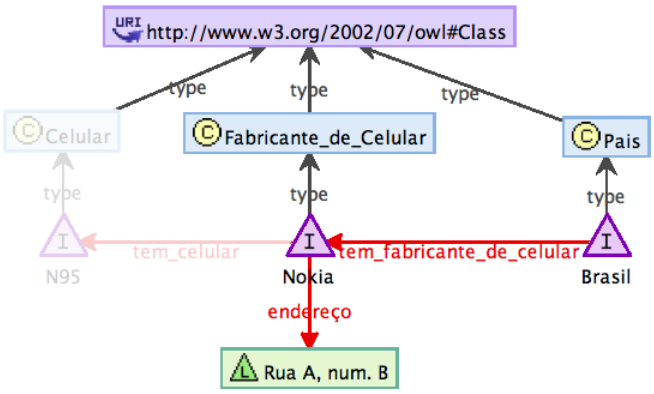


Figura 5.4: *Estrutura após a especialização dos conceitos*

A criação do termo **Fabricante de Celular** auxilia a reduzir a ambiguidade na interpretação da consulta "**quais são os endereços das fabricantes de celulares do Brasil**". Como a análise sintática das consultas é orientada a ontologia, consegue-se eliminar a ambiguidade de interpretação.

3. **Validação de triplas:**

   Para a consulta funcionar com perguntas que usam preposições, a estrutura da ontologia deve obedecer um padrão. Este padrão pode ser observado na Figura 5.2: os indivíduos devem possuir uma relação correspondente ao termo ligado ao lado esquerdo pela preposição. Por exemplo, no trecho "**celulares do Brasil**", se **celular** e **Brasil** são termos distintos, então o modelo espera que o indivíduo **Brasil** possua uma relação "**tem_celular**" ou similar. Nosso modelo então cria a tripla RDF "**Brasil tem_celular ?A**", onde **?A** é a variável objeto da tripla, que retorna vazia se a relação não existir. O problema é que a informação pode existir mas não com a relação esperada.

   Uma abordagem para amenizar o problema da inexistência de relações é testar as triplas que estão sendo criadas, descartando relações inexistentes. Assim, durante a criação de triplas podemos mudar a ordem da utilização dos termos. Por exemplo, durante a criação, a tripla "**p:Brasil p:tem_celular ?A**" deve ser testada para verificar se o resultado é vazio. Sendo vazio guardamos o termo **tem_celular** e testamos a tripla com o próximo termo, por exemplo, **p:Brasil p:tem_fabricante ?A**. Reutilizando os termos que foram guardados para gerar as próximas triplas. Usando esta inversão na geração das triplas chegamos ao seguinte SPARQL:

```
PREFIX r: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX p:<http://www.rodrigo.goulart.nom.br/feevale/blogseimagem/Smartphone.owl#>
SELECT ?C WHERE {?B p:endereco ?C. ?A p:tem_celular ?B.
                 p:Brasil p:tem_fabricante ?A.}
```

Todas as 3 três abordagens apresentadas anteriormente não são soluções definitivas, porém podem amenizar os problemas com relação a incompatibilidade da estrutura da ontologia de busca com o modelo. A interpretação que sugerimos para a criação das triplas, não cobre as possibilidades interpretativas inerentes a linguagem natural. De fato, quando temos o fragmento "**fabricantes de celulares do Brasil**", pelo nosso modelo, não sabemos se **Brasil** está relacionado com **fabricantes** ou **celulares**. Porém, algumas vezes a ontologia pode nos auxiliar nesta desambiguação.

Notadamente, as perguntas propensas a falhas são aquelas que usam um grande número de preposições ligando os substantivos. Dentre as perguntas desenvolvidas para o nosso trabalho apenas 3 perguntas geraram erros relacionados a incompatibilidade de estrutura, representando 15% dos erros. Portanto, embora os problemas descritos anteriormente sejam difíceis de tratar, aqui eles acontecem com menor frequência.

Capítulo 6

Conclusões

O propósito deste trabalho foi explorar métodos tradicionais no desenvolvimento de um modelo que possa mitigar o problema de consulta em ontologias utilizando linguagem natural. Procuramos minimizar o problema propondo uma ontologia controlada e uma linguagem controlada. Nossa abordagem foi a conversão de uma linguagem controlada em linguagem de consulta SPARQL, com o propósito da busca.

Com relação ao fragmento do português utilizado, estamos tratando importantes elementos gramaticais que transformam e ou geram perguntas, por exemplo, preposições, conjunções e disjunções. Estamos trabalhando com dois verbos básicos[1] da língua portuguesa: "ser" e "ter", o primeiro denota a noção de existência e o segundo tem uma conotação existencial ou possessiva [de Avelar, 2004]. O trabalho investiga consultas utilizando o "ser" e o "ter" fazendo a associação de fragmentos da linguagem natural com construções ontológicas que remetem a estes verbos.

Outro aspecto significante do trabalho é que estamos tomando como ponto importante para resolução do problema de consulta em ontologias a tradução da LNC para SPARQL. Porém, a tradução da nossa LNC em uma linguagem de consulta só faz sentido se soubermos como os dados estão estruturados na base. Ou seja mais do que saber o que buscar, é preciso saber como buscar. Entretanto, nosso trabalho não se dedica a encontrar uma estrutura perfeita[2] para a ontologia de busca, mas sim, através de regras para dar nomes aos elementos da ontologia, encontrar uma relação[3] entre ela e o modelo desenvolvido.

Outra importante observação oriunda do desenvolvimento do trabalho é que em algumas situações a especialização de termos da ontologia pode auxiliar na redução de ambiguidade das sentenças em linguagem natural. Um exemplo deste caso pode ser encontrado na Seção 5.2.1 mais especificamente na subseção onde abordamos sobre a especialização dos conceitos da ontologia.

Como pontos positivos podemos destacar o fato de nossa abordagem ser inovadora e apresentar resultados positivos, outro aspecto positivo do trabalho é que ele é facilmente implementável. Como pontos negativos, podemos destacar o fato de que o modelo está sujeito a falhas como foi visto na seção 5.2.1. Outro aspecto negativo é o fato da nossa linguagem ser relativamente limitada permitindo um conjunto restrito de consultas.

De fato, o problema com o qual estamos trabalhando é um problema muito difícil, pois nos deparamos com outros problemas conhecidos nas áreas de representação do conhecimento, processamento de linguagem natural, linguística, dentre outras. Nosso modelo procura reduzir estes problemas.

[1] Básicos no sentido de serem bastante utilizados na linguagem cotidiana.

[2] Neste caso, ontologia perfeita significa uma estrutura na qual sabemos exatamente como os elementos estão organizados.

[3] Usamos a palavra "relação" no sentido de investigarmos os casos em que as consultas obtiveram sucesso, descrevendo os elementos que colaboraram para este sucesso.

6.1 Considerações Finais

Mostramos que existe um subconjunto da linguagem natural que pode ser relacionado através das regras gramaticais com um subconjunto da linguagem SPARQL, possibilitando assim, consultas controladas em uma ontologia controlada.

De acordo com nossos experimentos, o bom funcionamento do modelo é dependente da interoperabilidade entre a ontologia e a LNC. Um aspecto interessante é que, desde que tenhamos uma ontologia apropriada, existe uma consulta que consegue recuperar qualquer informação presente nela.

6.2 Contribuições desse trabalho

As principais contribuições desse trabalho são:

- Levantar parâmetros no auxilio do desenvolvimento de uma ontologia com maior interoperabilidade com a linguagem natural, ressaltando problemas e propondo abordagens para mitigação dos mesmos;
- Relacionar fragmentos da linguagem natural à estruturas ontológicas;
- A principal contribuição do trabalho foi propor e implementar um modelo que possibilite a consulta a ontologias utilizando uma linguagem natural controlada.

6.3 Sugestões para Pesquisas Futuras

Utilizando o modelo atual é possível continuar realizando experimentos com o objetivo de refinar a modelagem, identificando erros e ampliando o conjunto de consultas possíveis. Entre os trabalhos futuros podemos citar:

- Ampliar a capacidade das gramáticas, ou seja, aumentar os modelos de perguntas possíveis;
- Modelar padrões para o desenvolvimento de ontologias apropriadas a trabalhar com Linguagem Natural. Existem alguns esforços no intuito de auxiliar construções de ontologias para se trabalhar com o processamento da linguagem natural [Estival et al., 2004], e outros trabalhos sobre a relação entre ontologias e linguagem natural [Bateman, 1993]. De fato, embora estes trabalhos sejam voltados para a língua inglesa, eles são positivos e ressaltam elementos que são relevantes no contexto do nosso trabalho, por exemplo, capturar relações da linguagem utilizando o conceito de hierarquia e reduzir ambiguidade.
- Desenvolver Matchings mais eficientes, que possam ser efetivos na realização da tarefa de identificação e associação entre termos (ontologia $\times$ linguagem natural) independente de domínio.
- Utilizar um vocabulário controlado. Com a utilização desse vocabulário podemos ter um controle maior sobre o modelo diminuindo os erros ocasionados pelo Matching.
- Minimizar o problema de ambiguidade dando a opção de escolha ao usuário durante a consulta.
- Utilizar mais informações da árvore sintática. Por exemplo, o uso das sub-árvores para identificar núcleos de sintagmas auxilia no "entendimento/interpretação" das perguntas. Isso acrescenta mais informações ao modelo.

Apêndice A

Perguntas

As perguntas a seguir estão relacionadas ao contexto da ontologia sobre o currículo Lattes.

1. Qual é o email do Paulo de Tarso?
2. Qual é o fone ou o email do Eduardo Galego?
3. Qual é o ano e idioma da dissertação de mestrado do Esdras?
4. Qual é o idioma ou pais de publicação da tese de doutorado do Paulo de Tarso?
5. Qual é o título da tese de doutorado do Paulo de Tarso?
6. Qual é o nome do orientador do Fillipe Resina?
7. Quem são os orientandos de mestrado da Renata Wassermann?
8. Quem são os orientandos de doutorado da Renata Wassermann?
9. Quem é o orientador do Christian Paz?
10. Quantos orientandos de mestrado tem a Renata Wassermann?
11. Quantos orientandos de doutorado tem o orientador do Fillipe Resina?
12. Quais são os professores?
13. Quantos são os estudantes de mestrado?
14. Qual é o titulo da tese de doutorado do Christian Paz?
15. Qual o nome do orientador do Raphael Cobe?
16. Raphael Cobe é um professor?
17. Quais são os livros?
18. Quem é o autor do livro Frontiers in Belief Revision?
19. Karina Valdivia tem email?
20. Qual orientando de mestrado da Renata Wassermann tem email?
21. Renata Wassermann tem email?
22. Qual é o fone ou o email do José David?
23. Qual é o ano e idioma da dissertação de mestrado do José David?

24. Qual é o idioma ou pais de publicação da tese de doutorado do Marcio Moretto?
25. Quantos orientandos de doutorado tem Leliane de Barros?
26. Quantos orientandos de mestrado tem o orientador do Christian Paz?
27. Paulo de Tarso tem webpage?
28. Quais os nome dos orientandos de mestrado da orientadora do Ricardo Augusto?
29. Quantos capítulos tem o livro Frontiers in Belief Revision?
30. Quem são os orientandos da Renata Wassermann?
31. Quem são os orientandos do Yoshiharu Kohayakawa?
32. Quem é o orientador do Wonder Luz?
33. Quais as linhas de pesquisa do Roberto Hirata Junior?
34. Quem são os professores do Fabiano?
35. Quem são os orientandos de mestrado do Marcelo Finger?
36. Quem são os orientandos de doutorado do Marcelo Finger?
37. Quem é o orientador do Fabiano?
38. Quais são as instituições?
39. O orientador do Fillipe Resina é um doutor
40. Renata Wassermann é uma professora?
41. IME é uma instituição?
42. Marcelo Finger tem publicações?
43. Qual o endereço profissional do Roberto Hirata Junior ?
44. Qual o titulo da dissertação de mestrado do Fabiano?
45. Quem é o orientador do Luiz Carlos Vieira?
46. Qual é o email do Christian Paz?
47. Quais orientandos da Renata Wassermann são professores?
48. Quais orientandos de mestrado do Marcelo Finger tem email?
49. Paulo de Tarso é um estudante de doutorado?
50. Quais são as publicações dos professores do departamento de ciência da computação
51. Quais professores do departamento de ciência da computação tem publicação?
52. Quais estudantes do IME são professores?
53. O orientando do orientador do Fillipe Resina é um doutor?
54. Quais os professores do IME tem artigos publicados?
55. Quantas titulações tem Luiz Carlos Vieira?

56. Quantos são os doutorandos?
57. Quantos professores do IME tem artigos publicados?
58. Qual é a webpage do departamento de ciência da computação do IME?
59. Qual é a formação do José David?
60. Quantos institutos tem a USP?
61. Quantos orientandos tem a orientadora do Ricardo Augusto?
62. Quais são as instituições ou unidades da USP?
63. Quais são os artigos publicados do Fábio Franco?
64. Quantos professores possui a USP?
65. Quais os orientandos e os artigos da Renata Wassermann?
66. Quais estudantes de mestrado do IME possuem publicação?
67. Quantos professores tem o departamento de ciência da computação?
68. Quem são os professores e os alunos do departamento de ciência da computação?
69. Qual orientando do orientador do Ricardo Augusto é professor?
70. Qual é o email do Yoshiharu Kohayakawa?
71. Qual o ano e os autores dos artigos da orientadora do Ricardo Augusto?
72. Quais são os artigos de Flávio Soares?
73. Quais alunos da Renata Wassermann são autores de livros?
74. Quantas teses possui a orientação da Renata Wassermann?
75. Quantos orientandos da Renata Wassermann são estudante de doutorado?
76. Fabiano é estudante?
77. Quem são os estudantes de inteligencia artificial da USP?
78. Fabiano é estudante de mestrado?
79. Quem são os professores das disciplinas do IME?
80. Quais são os alunos da Renata Wassermann?
81. Quais os professores do IME possuem doutorado?
82. Quais são os projeto de pesquisa do Alfredo Goldman?
83. Marcel é o professor da disciplina de Computação Gráfica?
84. Quais orientandos da Renata Wassermann são orientadores?
85. Quais professores do IME são titulares?
86. Wonder Luz é doutorando de ciência da computação?
87. Quantos artigo publicado tem Fábio Kon?

88. Paulo de Tarso é um Livro?
89. Quantos são os orientadores?
90. Quais são as conferencias nacionais de computação?
91. Qual é o email do Eduardo Galego ou do Paulo de Tarso?
92. Qual é o endereço profissional do Fabiano ou do Esdras ou do Fillipe Resina?
93. Qual é o titulo da tese de doutorado do Esdras e do Paulo de Tarso?
94. Quem são os professores do departamento de ciência da computação?
95. Quais são os autores de livros da USP?

As perguntas a seguir estão relacionadas ao contexto da ontologia sobre smartphones.

1. Qual é o smartphone do Fernando?
2. Qual é o modelo do celular da Maria?
3. Qual é o processador do E63?
4. Quais são os fabricantes de celulares?
5. Qual é a resolução da câmera do smartphone do Fernando?
6. Qual é o tempo de autonomia da bateria do celular da Maria?
7. Qual é a cor ou o material do celular da Sandra?
8. Quantas câmeras tem o celular do Pedro?
9. Quantos usuários possuem smartphone da Apple?
10. Qual é o tamanho da tela do celular do Rodrigo?
11. Qual o código de área do numero do celular do Rodrigo?
12. Qual é o sistema operacional do celular da Sandra?
13. O celular do Fernando tem bluetooth?
14. O iphone da Maria tem conexão de dados wireless?
15. A Motorola é uma fabricante de smartphone?
16. O smartphone do Rodrigo tem fone de ouvido?
17. O iphone da Sandra tem conexão de dados bluetooth?
18. Quantos chips tem o celular da Sandra?
19. Qual é o modelo do celular do Luis?
20. Quantos smartphones tem a Sandra?
21. O smartphone da Nokia tem câmera?
22. Qual smartphone da Nokia possui bluetooth?
23. Quais são as conexão de dados do I617?

24. Qual a capacidade de memória do celular do Luis?
25. Quantos celulares tem a Maria?
26. O celular da Maria é touch screen?
27. Qual é o tamanho da tela do X2-01?
28. Os smartphones da Samsung tem câmera?
29. Qual é o formato e a cor do celular do Fernando?
30. Qual é o tempo de autonomia da bateria do IPhone4?
31. Symbian é um smartphone?
32. Qual é o preço do Iphone3S?
33. Quantas antena tem o celular do Rodrigo?
34. Qual é a resolução da câmera do celular do Luis?
35. Quantos modelo de smartphone tem a Apple?
36. O Iphone da Sandra tem bateria de lítio?
37. Quais são os componentes do X2-01?
38. Quem é a fabricante do IPhone?
39. Quais são os sistema operacional dos smartphones da Apple?
40. Qual é o player do Windows Mobile?
41. Quais são as operadoras de telefonia?
42. O celular do Fernando tem câmera
43. Quem é a fabricante do IPhone?
44. Qual é o processador do E63?
45. O celular do Luis tem caneta?
46. Qual o numero do celular da Maria?
47. Qual é o material da bateria do E63?
48. A câmera do celular do João tem flash?
49. Quantos acessórios tem o Treo do Rodrigo?

Apêndice B

Formulários

Formulário 1: Consulte uma ontologia utilizando a GLNC (contexto currículo Lattes)
Gramática de produção:

Tabela B.1: *Regras de produção R de GLNC*

S	→	QP VS A F X Prop |
		QP VS A X U |
		PQ Sub VT A B Prop |
		QX VS A Sub
		A B Prop VQ A B Sub |
U	→	T | W
T	→	T D Prep Prop | Prop
W	→	W C Prep Prop | Prop
X	→	X Sub Prep | Sub Prep
F	→	Y Prep | Z Prep |λ
Z	→	Z Sub C A | Sub C A
Y	→	Y Sub D A | Sub D A
B	→	X |λ
VQ	→	VT | VS
QX	→	QP | PQ
A	→	a | o | as | os | um | uma | uns | umas |λ
C	→	e
D	→	ou
VS	→	é | são |λ
VT	→	tem | possui
QP	→	qual | quais | quem
PQ	→	quantos | quantas
Prep	→	de | da | do | das | dos
Prop	→	$\{\theta \mid \theta \in \Theta\}$
Sub	→	$\{\gamma \mid \gamma \in \Gamma\}$

Tabela B.2: *Siglas LNC*

Descrição	Sigla
Qualquer pronome interrogativo	QP
Pronome interrogativo QUANDO	PQ

Verbo SER	VS
Verbo TER ou POSSUIR	VT
Artigo	A
Conjunção	C
Disjunção	D
Substantivo comum	Sub
Substantivo próprio	Prop
Preposição	Prep
Conj. substantivos próprios	Θ
Conj. substantivos comuns	Γ
Palavra vazia	λ

Substantivo comum e substantivo próprio são palavras do contexto. O contexto é currículo Lattes. Façam qualquer pergunta relacionada ao currículo Lates utilizando nomes de professores e colegas do IME.
Exemplo:
Fabiano é estudante de computação?
Qual é o título da tese de doutorado do Paulo de Tarso?
Favor elaborar 03 ou mais perguntas utilizando nossa GLNC. Desde já, grato pela colaboração de todos.

Formulário 2: Consulte uma ontologia utilizando a GLNC (contexto smartphones)
Gramática de produção:

Tabela B.3: *Regras de produção R de GLNC*

S	→	QP VS A F X Prop |
		QP VS A X U |
		PQ Sub VT A B Prop |
		QX VS A Sub
		A B Prop VQ A B Sub |
U	→	T | W
T	→	T D Prep Prop | Prop
W	→	W C Prep Prop | Prop
X	→	X Sub Prep | Sub Prep
F	→	Y Prep | Z Prep |λ
Z	→	Z Sub C A | Sub C A
Y	→	Y Sub D A | Sub D A
B	→	X |λ
VQ	→	VT | VS
QX	→	QP | PQ
A	→	a | o | as | os | um | uma | uns | umas |λ
C	→	e
D	→	ou
VS	→	é | são |λ
VT	→	tem | possui
QP	→	qual | quais | quem
PQ	→	quantos | quantas
Prep	→	de | da | do | das | dos
Prop	→	$\{\theta \mid \theta \in \Theta\}$
Sub	→	$\{\gamma \mid \gamma \in \Gamma\}$

Tabela B.4: *Siglas LNC*

Descrição	Sigla
Qualquer pronome interrogativo	QP
Pronome interrogativo QUANDO	PQ
Verbo SER	VS
Verbo TER ou POSSUIR	VT
Artigo	A
Conjunção	C
Disjunção	D
Substantivo comum	Sub
Substantivo próprio	Prop
Preposição	Prep
Conj. substantivos próprios	Θ
Conj. substantivos comuns	Γ
Palavra vazia	λ

Substantivo comum e substantivo próprio são palavras do contexto. O contexto é smartphones. Façam qualquer pergunta relacionada a smartphones, também pode ser sobre o smartphone de algum usuário específico. Os usuários são: Fernando, Luis, Maria, Pedro, Rodrigo e Sandra. Algumas marcas presentes na ontologia são: Samsung, Apple e Nókia.
Exemplo:
O celular do Fernando tem câmera?
Qual é o material da bateria do smartphone da Maria?
Favor elaborar 03 ou mais perguntas utilizando nossa GLNC. Desde já, grato pela colaboração de todos.

Referências Bibliográficas

[Aho and Ullman, 1972] Aho, A. V. and Ullman, J. D. (1972). *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc. 4, 12

[Alvarez, 2008] Alvarez, B. (2008). *Considerações sobre as interrogativas no Amphitrvo de Plauto*. PhD thesis, Universidade Federal do Rio de Janeiro. 4

[Androutsopoulos et al., 1995] Androutsopoulos, I., Ritchie, G., and Thanisch, P. (1995). Natural language interfaces to databases-an introduction. *arXiv preprint cmp-lg/9503016*. 2

[Barnard et al., 2003] Barnard, K., Duygulu, P., Forsyth, D., De Freitas, N., Blei, D. M., and Jordan, M. I. (2003). Matching words and pictures. *The Journal of Machine Learning Research*, 3:1107–1135. 23

[Basten, 2007] Basten, H. (2007). Ambiguity detection methods for context-free grammars. *Master's thesis, Universiteit van Amsterdam, August*. 32

[Bateman, 1993] Bateman, J. A. (1993). Ontology construction and natural language. In *Proceedings of the International Workshop on Formal Ontology*, pages 83–93. 58

[Bechhofer et al., 2004] Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L., et al. (2004). OWL web ontology language reference. *W3C recommendation*, 10:2006–01. 5

[Beckett, 2006] Beckett, D. (2006). SPARQL RDF query language reference v1. 8. *Consultado (08/02/2012) en [http://www. dajobe. org/2005/04-sparql/]*. 8

[Bevilacqua and Finatto, 2009] Bevilacqua, C. R. and Finatto, M. J. B. (2009). Lexicografia e terminografia: alguns contrapontos fundamentais. *ALFA: Revista de Linguística*, 50(2). 12

[Bispo Jr, 2011] Bispo Jr, E. (2011). Métricas de avaliação de alinhamentos de ontologias. Master's thesis, IME/USP - Inst. de Matemática e Estatística da Universidade de São Paulo. 9

[Borst, 1997] Borst, W. (1997). *Construction of engineering ontologies for knowledge sharing and reuse*. PhD thesis, Centre for Telematics and Information Technology University of Twente. 5

[Brabrand et al., 2010] Brabrand, C., Giegerich, R., and Møller, A. (2010). Analyzing ambiguity of context-free grammars. *Science of Computer Programming*, 75(3):176–91. 32

[Brito, 2008] Brito, G. (2008). Linguistas e computadores: Que relação é essa? *Periódicos da UFSC, Working papers em linguistica*. 1

[Capetta, 2012] Capetta, L. (2012). Consulta a banco de dados em linguagem natural. *Revista OMNIA Exatas*, 4(1):72–80. 2

[Cardoso, 2009] Cardoso, A. (2009). Extraposição de orações relativas: uma abordagem comparativa entre o português antigo e o português actual. Centro de Linguística da Universidade de Lisboa. http://194.117.6.240/files/adriana_cardoso/Cardoso_2009.pdf. Último acesso: 2012-09-30. 47

[Chechev et al., 2004] Chechev, M., Ranta, A., Damova, M., and Enache, R. (2004). D4. 3 grammar-ontology interoperability. MOLTO - Multilingual Online Translation. 1

[Chomsky, 1975] Chomsky, N. (1975). *Reflections on language*. Temple Smith London. 3

[Colen, 2013] Colen, W. (2013). Aprimorando o corretor gramatical cogroo. Master's thesis, IME/USP - Inst. de Matemática e Estatística da Universidade de São Paulo. 12

[Collins,] Collins, M. Probabilistc context-free grammars. Comprehensive notes, Springer 2013. http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf. Último acesso: 2013-02-15. 13

[da Costa and Yamate, 2009] da Costa, A. P. and Yamate, F. S. (2009). Semantic lattes: uma ferramenta de consulta de informações acadêmicas da base lattes baseada em ontologias. Universidade de São Paulo - USP. 1, 19, 49

[da Silva et al., 2007] da Silva, B., Montilha, G., Rino, L., Specia, L., Nunes, M., de Oliveira Jr, O., Martins, R., and Pardo, T. (2007). Introdução ao processamento das línguas naturais e algumas aplicações. Série de Relatórios do Núcleo Interinstitucional de Lingüística Computacional. 3

[de Avelar, 2004] de Avelar, J. O. (2004). Dinamicas morfossintaticas como ter, ser e estar em portugues brasileiro. Biblioteca Digital da Unicamp. 57

[de Freitas and Vieira, 2008] de Freitas, L. A. and Vieira, R. (2008). Ontologias e lingua portuguesa. *Anais do CELSUL 2008*. 1, 2, 5, 7

[Di Felippo and Dias-da Silva, 2007] Di Felippo, A. and Dias-da Silva, B. C. (2007). Towards an automatic strategy for acquiring the wordnet. br hierarchical relations. In *Proceedings of the 5th Workshop in Information and Human Language Technology*. 12

[Dias-Da-Silva and de Moraes, 2003] Dias-Da-Silva, B. C. and de Moraes, H. R. (2003). A construçao de um thesaurus eletrônico para o português do brasil. *ALFA: Revista de Linguística*, 47(2). 12

[Duarte, 2009] Duarte, J. (2009). *O Algoritmo Booting at Start e sua aplicações*. PhD thesis, PUC - Rio. 9

[Estival et al., 2004] Estival, D., Nowak, C., and Zschorn, A. (2004). Towards ontology-based natural language processing. In *Proceeedings of the Workshop on NLP and XML (NLPXML-2004): RDF/RDFS and OWL in Language Technology*, pages 59–66. Association for Computational Linguistics. 58

[Euzenat, 2007] Euzenat, J. (2007). Semantic precision and recall for ontology alignment evaluation. In *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 348–353. 8

[Ferré, 2012] Ferré, S. (2012). SQUALL: A controlled natural language for querying and updating RDF graphs. In *Controlled Natural Language*, pages 11–25. Springer. 11

[Finger and Colen, 2012] Finger, M. and Colen, W. (2012). *CoGrOO: Corretor Gramatical acoplável ao LibreOffice e Apache OpenOffice*. CCSL IME/USP, São Paulo, Brasil. 12, 21

[Greghi et al., 2002] Greghi, J. G., Martins, R. T., and Nunes, M. d. G. V. (2002). DIADORIM-A Lexical Database for Brazilian Portuguese. In *LREC*. 12

[Gruber et al., 1993] Gruber, T. et al. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220. 5

[Hardesty, 2010] Hardesty, L. (2010). A grand unified theory of AI. MIT Newspaper, MIT News Office. 2

[Horridge et al., 2004] Horridge, M., Knublauch, H., Rector, A., Stevens, R., and Wroe, C. (2004). A practical guide to building OWL ontologies using the protégé-OWL plugin and CO-ODE tools edition 1.0. *University of Manchester*. 5, 6, 7

[Joshi et al., 1991] Joshi, A. et al. (1991). Natural language processing. *Science (New York, NY)*, 253(5025):1242. 3

[Jurafsky and Martin, 2000] Jurafsky, D. and Martin, J. H. (2000). *Speech & Language Processing*. Pearson Education India. 4

[Kaufmann et al., 2006] Kaufmann, E., Bernstein, A., and Zumstein, R. (2006). Querix: A natural language interface to query ontologies based on clarification dialogs. In *5th International Semantic Web Conference (ISWC 2006)*, pages 980–981. 11

[King, 1983] King, M. (1983). *Parsing natural language*. Academic Press, Inc. 17

[Kinoshita et al., 2005] Kinoshita, J., Salvador, L. d. N., and Menezes, C. E. D. (2005). Cogroo–um corretor gramatical para a língua portuguesa, acoplável ao openoffice. In *CLEI 2005 - XXXI Conferência Latinoamericana de Informática*. 12

[Klyne et al., 2004] Klyne, G., Carroll, J., and McBride, B. (2004). Resource description framework (RDF): Concepts and abstract syntax. *W3C recommendation*, 10. 8

[Li et al., 2003] Li, Y., Bandar, Z. A., and McLean, D. (2003). An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882. 23, 47

[Machado, 2003] Machado, I. (2003). *Escola de semiótica: a experîencia de Tártu-Moscou para o estudio da cultura*. Atelie Editorial. 47

[Manning and Schütze, 1999] Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press. 9

[Martins, 2007] Martins, F. G. R. (2007). Ontologia de domínio para análise de blogs. 1, 19, 49

[Maziero et al., 2008] Maziero, E. G., Pardo, T. A., Di Felippo, A., and Dias-da Silva, B. C. (2008). A base de dados lexical ea interface web do tep 2.0: thesaurus eletrônico para o português do brasil. In *Companion Proceedings of the XIV Brazilian Symposium on Multimedia and the Web*, pages 390–392. ACM. 12

[Mitkov et al., 2003] Mitkov, R. et al. (2003). *The Oxford handbook of computational linguistics*. Oxford University Press Oxford. 3

[Nantes, 2008] Nantes, L. (2008). Desenvolvimento de um sistema baseado em linguagem natural para consulta em banco de dados na web. 2

[Partee, 1976] Partee, B. H. (1976). *Montague grammar*. Academic Pr. 11

[Prud'Hommeaux et al., 2008] Prud'Hommeaux, E., Seaborne, A., et al. (2008). Sparql query language for rdf. *W3C recommendation*, 15. 1, 7, 8

[Pérez et al., 2006] Pérez, J., Arenas, M., and Gutierrez, C. (2006). Semantics and complexity of SPARQL. *The Semantic Web-ISWC 2006*, pages 30–43. 7

[Ribeiro and Vieira, 2008] Ribeiro, L. and Vieira, R. (2008). Ontolp: Engenharia de ontologias em língua portuguesa. In *Anais do XXVIII congresso da SBC-SEMISH-Seminário integrado de software e hardware, Belém do Pará*. 2, 7

[Richardson et al., 1994] Richardson, R., Smeaton, A. F., and Murphy, J. (1994). Using wordnet as a knowledge base for measuring semantic similarity between words. Technical report, Technical Report Working Paper CA-1294, School of Computer Applications, Dublin City University. 47

[Russell et al., 1995] Russell, S., Norvig, P., Canny, J., Malik, J., and Edwards, D. (1995). *Artificial intelligence: a modern approach*, volume 2. Prentice hall. 1

[Salzburg, 2004] Salzburg, R. (2004). RDF gravity v1.0. http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html. Último acesso: 2012-01-18. 16

[Saussure, 1978] Saussure, F. (1978). Curso de linguística geral. Dom Quixote. 3

[Schmitz, 2007] Schmitz, S. (2007). Conservative ambiguity detection in context-free grammars. In *Automata, Languages and Programming*, pages 692–703. Springer. 32

[Schneider, 2001] Schneider, M. (2001). Processamento de linguagem natural (PLN). Master's thesis, PUC - Campinas. 3

[Schwitter, 2007] Schwitter, R. (2007). *Controlled natural languages. Technical report.* Centre for Language Technology, Macquarie University. 3

[Schwitter et al., 2008] Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., and Hart, G. (2008). A comparison of three controlled natural languages for OWL 1.1. In *4th OWL Experiences and Directions Workshop (OWLED 2008 DC), Washington*, pages 1–2. 4

[Sipser, 2006] Sipser, M. (2006). *Introduction to the Theory of Computation*, volume 27. Thomson Course Technology Boston, MA. 4, 21

[Tablan et al., 2008] Tablan, V., Damljanovic, D., and Bontcheva, K. (2008). A natural language query interface to structured information. *The Semantic Web: Research and Applications*, pages 361–375. 2

[Vanti, 2002] Vanti, N. (2002). Da bibliometria à webometria: uma exploração conceitual dos mecanismos utilizados para medir o registro da informação e a difusão do conhecimento. *Ciência da Informação*, 31(2):152–162. 1

[Wallis, 2008] Wallis, S. (2008). *Searching treebanks and other structured corpora.* Mouton de Gruyter. 32

[Younger, 1967] Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and control*, 10(2):189–208. 4, 12

[Yu, 2011] Yu, L. (2011). Linked open data. In *A Developer's Guide to the Semantic Web*, pages 409–466. Springer. 1